VISUAL AND SPATIAL ANALYSIS

Advances in Data Mining, Reasoning, and Problem Solving

Edited by Boris Kovalerchuk and James Schwing

Advanced visual analysis and problem solving has been conducted successfully for millennia. The Pythagorean Theorem was proven using visual means more than 2000 years ago. In the 19th century, John Snow stopped a cholera epidemic in London by proposing that a specific water pump be shut down. He discovered that pump by visually correlating data on a city map. The goal of this book is to present the current trends in visual and spatial analysis for data mining, reasoning, problem solving and decision-making. This is the first book to focus on visual decision making and problem solving in general with specific applications in the geospatial domain – combining theory with real-world practice. The book is unique in its integration of modern symbolic and visual approaches to decision making and problem solving. As such, it ties together much of the monograph and textbook literature in these emerging areas.

This book contains 21 chapters that have been grouped into five parts: (1) visual problem solving and decision making, (2) visual and heterogeneous reasoning, (3) visual correlation, (4) visual and spatial data mining, and (5) visual and spatial problem solving in geospatial domains. Each chapter ends with a summary and exercises.

The book is intended for professionals and graduate students in computer science, applied mathematics, imaging science and Geospatial Information Systems (GIS). In addition to being a state-of-the-art research compilation, this book can be used as a text for advanced courses on the subjects such as modeling, computer graphics, visualization, image processing, data mining, GIS, and algorithm analysis.

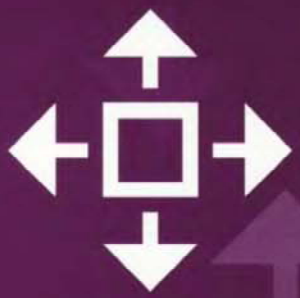
ISBN 1-4020-2939-X
9 781402 029394

springeronline.com

VISUAL AND SPATIAL ANALYSIS

Boris Kovalerchuk and James Schwing (Eds.)

VISUAL AND SPATIAL ANALYSIS



Advances in Data Mining, Reasoning, and Problem Solving

Edited by
Boris Kovalerchuk
and James Schwing

VISUAL AND SPATIAL ANALYSIS

Visual and Spatial Analysis

Advances in Data Mining, Reasoning, and Problem Solving

Edited by

Boris Kovalerchuk
Central Washington University,
Ellensburg, WA,
U.S.A.

and

James Schwing
Central Washington University,
Ellensburg, WA,
U.S.A.

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 1-4020-2939-X (HB)
ISBN 1-4020-2958-6 (e-Book)

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

Sold and distributed in North, Central and South America
by Springer,
101 Philip Drive, Norwell, MA 02061, U.S.A.

In all other countries, sold and distributed
by Springer,
P.O. Box 322, 3300 AH Dordrecht, The Netherlands.

Printed on acid-free paper

springeronline.com

Printed in the Netherlands.

TABLE OF CONTENTS

PART 2. VISUAL AND HETEROGENEOUS REASONING

3. Visual reasoning and representation

Boris Kovalerchuk

4. Representing visual decision making: a computational architecture for heterogeneous reasoning

Dave Barker-Plummer and John Etchemendy

5. Algebraic visual symbolism for problem solving: iconic equations from Diophantus to the present

Boris Kovalerchuk and James Schwing

PART 5. VISUAL AND SPATIAL PROBLEM SOLVING IN GEOSPATIAL DOMAINS

20. Algorithm development technology for conflation and area-based conflation algorithm

Michael Kovalerchuk and Boris Kovalerchuk

21. Virtual experts for imagery registration and conflation

Boris Kovalerchuk, Artemus Harper, Michael Kovalerchuk, and Jon Brown

PREFACE

Visual problem solving has been successful for millennia. The Pythagorean Theorem was proved by visual means more than 2000 years ago. The entire study of geometry existed as a visual problem-solving field more than one and a half millennia before René Descartes invented symbolic coordinates. Albert Einstein wrote in 1953 that the development of Western Science is based on two great achievements: the invention of the formal logic system (in Euclidean geometry) and reasoning based on systematic experimentation during the Renaissance. In the context of this book, it is important to notice that the formal logical system in Euclidean geometry was visual.

Consider two other important historical examples of visual problem solving and decision making. Maritime navigation by using the stars presents an example of sophisticated visual problem solving and decision-making. Then in the 19th century, John Snow stopped a cholera epidemic in London by proposing that a specific water pump be shut down. He discovered that pump by visually correlating data on the city map. Of course, there continue to be many current examples of advanced visual problem solving and decision-making.

This book presents the current trends in visual problem solving and decision-making making a clear distinction between the **visualization of an already identified solution** and **visually finding a solution**. Thus, the book focuses on two goals:

(G1) displaying a result or solution visually, and

(G2) deriving a result or solution by visual means.

The first goal has two aspects: G1(a) **displaying results to a novice** and G1(b) **convincing a decision maker**. Recently mass media (US News and World Report, Dec. 2003, p.30) reported that intelligence analysts knew the danger of coming September 11 but convincing decision makers was one of their major challenges: "There were people who got it at the analyst level, at the supervisory level, but all of us were outnumbered". A novice simply does not know the subject but has no prejudice, priorities, special interests or

other preconceived notions that may prevent the digesting of information. Decision makers may have all of these characteristics.

The second goal, G2 is even more difficult to achieve especially for non-structured problems. Obviously, there are many intermediate goals that fall between the two extremes and the journey from G1 to G2 is not a short, non-stop flight. This is the reason that we consider this bsook to be the first step in a future series "Visual decision making and problem solving."

A typical example of G1 is the animation of a known algorithm for a novice. Here the intention is to show visually the algorithm's steps. A different situation arises relative to G2 when we use animation to discover properties of an algorithm visually such as the number of loops and the amount of space required for the task. If the animation tool permits viewing loops and space used, then it can serve as a visual problem-solving tool. Thus, both tasks might use the same technique. This observation shows that the essence of the transition from G1 to G2 is not the visual technique itself (e.g. animation). The essence of achieving G2 is in **matching a decision-making/problem-solving task with a visual technique**. The same type of matching may be required to convince a decision maker. Note that it is likely that a simple animation technique appropriate for the novice would not be sufficient for an advanced visual decision-making on algorithm efficiency by an experienced analyst. On the other hand, there are also situations, which show that after a solution is discovered through the use of sophisticated analytical and visual means a very simple visualization can be sufficient and desirable for convincing a decision maker.

We use the term brute force visual problem solving for the approach in which every available visual and/or analytical technique is tried for the task. **A task-driven approach** is a better alternative and is one theme that is discussed throughout the book. It can be implemented at a variety of levels from the global decision level to subpixel sensor level. Such an approach can involve the automatic generation/selection of a visual tool based on a user's query to the automatic generation of composite icons matched to user's queries. Task driven approaches can also involve of the use of hierarchical, visual, decision-making models and the recording the analyst's visual decision-making procedures.

The book emphasizes the difference between visual decision making and visual data mining. **Visual data mining** discovers useful regularities visually or visualizes patterns discovered by common data mining tools such as neural networks and decision trees. We will cover this subject along with a look at spatial data mining, which combines analytical and visual mining of spatial data. **Visual decision making** relies heavily on visual data mining, but useful regularities are only a part of the entire decision-making process. In finance, visual data mining helps to discover market trends. Visual invest-

ment decision-making is used to produce buy/sell signals and to help select a portfolio. In medicine, visual data mining helps with patient diagnosis. Decision on a course of treatment is also heavily based on the diagnosis, but it is only a part of a treatment decision. For instance, a treatment decision should avoid a patient's allergic reactions and other negative side effects. Diagnostic data mining does not cover this issue. In defense applications, visual data mining can provide valuable intelligence clues for planning an operation. A visual decision about a military strike requires that much more information be considered, including the availability of military forces and political realities. Visual data mining is also useful in breaking drug trafficking rings, but the actual decision on these operations needs to involve more aspects of the problem.

The foundations and applications of visual problem solving and decision making are many and varied. To organize these topics, the book has been divided into five parts: (1) visual problem solving and decision making, (2) visual and heterogeneous reasoning, (3) visual correlation, (4) visual and spatial data mining, and (5) visual and spatial problem solving in geospatial domains.

As noted, Part 1 addresses visual decision making and is divided into two chapters. Chapter 1 provides a broad overview of current trends in visual decision making and problem solving. This overview includes: further differentiating the visualization of a solution and the generation of a solution visually. The chapter describes general, hierarchical, visual decision-making models using structural information and ontologies.

Chapter 2 provides an extensive discussion of the efficiency of information visualization techniques. It suggests an informal model (called information visualization value stack model) that predicts a problem area called "sweet spot", where information visualization will most likely achieve utilization. This model is based on a set of qualitative problem parameters identified in the chapter.

Visual and heterogeneous reasoning is the focus of Part 2 and consists of Chapters 3-7. Reasoning plays a critical role in decision making and problem solving. Chapter 3 provides a comparative analysis of visual and verbal (sentential) reasoning approaches and their combination called heterogeneous reasoning. It is augmented with a description of application domains of visual reasoning. One of the conclusions of this chapter is that the fundamental iconic reasoning approach introduced by Charles Peirce is the most comprehensive heterogeneous reasoning approach.

Chapter 4 describes a computational architecture for applications that support heterogeneous reasoning. Heterogeneous reasoning is, in its most general form, reasoning that employs representations drawn from multiple representational forms. Of particular importance, and the principal focus of

this architecture, is heterogeneous reasoning that employs one or more forms of graphical representation, perhaps in combination with sentences (of English or another language, whether natural or scientific). The architecture is based on the model of natural deduction in formal logic. This chapter describes and motivates modifications to the standard logical model necessary to capture a wide range of heterogeneous reasoning tasks.

Chapter 5 provides a discussion of mathematical visual symbolism for problem solving based on an algebraic approach. It is formulated as lessons that can be learned from history. Visual formalism is contrasted with text through the history of algebra beginning with Diophantus' contribution to algebraic symbolism nearly 2000 years ago. Along the same lines, it is shown that the history of art provides valuable lessons. The evident historical success provides a positive indication that similar success can be repeated for modern decision-making and analysis tasks. Thus, this chapter presents the lessons from history tuned to new formalizations in the form of iconic equations and iconic linear programming.

Chapter 6 describes an iconic reasoning architecture for analysis and decision-making along with a storytelling iconic reasoning approach. The approach provides visuals for task identification, evidence, reasoning rules, links of evidence with pre-hypotheses, and evaluation of hypotheses. The iconic storytelling approach is consistent hierarchical reasoning that includes a variety of rules such as visual search-reasoning rules that are tools for finding confirming links. The chapter also provides a review of related work on iconic systems. The review discusses concepts and terminology, controversy in iconic language design, links between iconic reasoning and iconic languages and requirements for an efficient iconic system.

Chapter 7 considers directions for visual reasoning and discovery. Currently, computer visualization is moving from a pure illustration domain to visual reasoning, discovery, and decisions making. This trend is associated with new terms such as visual data mining, visual decision making, and heterogeneous, iconic and diagrammatic reasoning. Beyond a new terminology, the trend itself is not new as the early history of mathematics clearly shows. This chapter demonstrates that we can learn valuable lessons from the history of mathematics for visual reasoning and discovery.

Visual correlation is the thrust of Part 3, which consists of Chapters 8-10. Chapter 8 introduces the concept of visual correlation and describes the essence of a generalized correlation to be used for multilevel and conflicting data. Several categories of visual correlation are presented accompanied by both numeric and non-numeric examples with three levels (high, medium and low) of coordination. The chapter presents examples of multi-type visual correlations. The chapter also provides a classification of visual correlation methods with corresponding metaphors and criteria for visual correlation

efficiency. The chapter finishes with a more formal treatment of visual correlation, providing formal definitions, analysis, and theory.

Chapter 9 presents the state-of-the-art in iconic descriptive approaches to annotating, searching, and correlating that are based on the concepts of compound and composite icons, the iconic annotation process, and iconic queries. Specific iconic languages used for applications such as video annotation, military use and text annotation are discussed. Graphical coding principals are derived through the consideration of questions such as: How much information can a small icon convey? How many attributes can be displayed on a small icon either explicitly or implicitly? The chapter also summirizes impact of human perception on icon design.

Chapter 10 addresses the problem of visually correlating objects and events. The new Bruegel visual correlation system based on an iconographic language that permits compact information representation is described. The description includes the Bruegel concept, functionality, the ability to compress information via iconic semantic zooming, and dynamic iconic sentences. The formal Bruegel iconic language for automatic icon generation is outlined. The chapter is devoted to case studies that describe how Bruegel iconic architecture can be used.

Part 4 addresses visual and spatial data mining and consists of Chapters 11-16. Chapter 11 introduces two dynamic visualization techniques using multi-dimensional scaling to analyze transient data streams such as newswires and remote sensing imagery. The chapter presents an adaptive visualization technique based on data stratification to ingest stream information adaptively when influx rate exceeds processing rate. It also describes an incremental visualization technique based on data fusion to project new information directly onto a visualization subspace spanned by the singular vectors of the previously processed neighboring data. The ultimate goal is to leverage the value of legacy and new information and minimize re-processing of the entire dataset in full resolution

In Chapter 12, the main objective of the described spatial data mining platform called SPIN! is to provide an open, highly extensible, n-tier system architecture based on the Java 2 Platform, Enterprise Edition. The data mining functionality is distributed among (i) Java client application for visualization and workspace management, (ii) application server with Enterprise Java Bean container for running data mining algorithms and workspace management, and (iii) spatial database for storing data and spatial query execution. In the SPIN! system, visual problem solving involves displaying data mining results, using visual data analysis tools, and finally producing a solution based on linked interactive displays with different visualizations of various types of knowledge and data.

Chapter 13 begins by looking at the Predictive Model Markup Language (PMML), an XML-based industrial standard for the platform- and system-independent representation of data mining models. VizWiz, a tool for the visualization and evaluation of data mining models that are specified in PMML, is presented. This tool allows for the highly interactive visual exploration of a variety of data mining result types such as decision trees, classification and association rules or subgroups.

Chapter 14 describes new neural-network techniques developed for the visual mining of clinical electroencephalograms (EEGs). These techniques exploit the fruitful ideas of the Group Method of Data Handling (GMDH). The chapter briefly describes the standard neural-network techniques that are able to learn well-suited classification modes from data presented by relevant features. It then introduces and applies an evolving cascade neural network technique that adds new input nodes as well as new neurons to the network while the training error decreases. The chapter also presents the GMDH-type polynomial networks trained from data. New neural-network techniques developed to derive multi-class concepts from data are described and applied.

Chapter 15 discusses how to represent scientific visualization and data mining tasks in a simpler form so that visual solutions become possible. Visualization is used in data mining for the visual presentation of already discovered patterns and for discovering new patterns visually. Success in both tasks depends on the ability of presenting abstract patterns as simple visual patterns. A new approach called inverse visualization (IV) is suggested for addressing the problem of visualizing complex patterns. The approach is based on specially designed data preprocessing, which is based on a transformation theorem proved in this chapter. A mathematical formalism is derived from the Representative Measurement Theory. The possibility of solving inverse visualization tasks is illustrated on functional non-linear additive dependencies.

Chapter 16 describes a new technique for extracting patterns and relations visually from multidimensional binary data. The proposed method relies on monotone structural relations between Boolean vectors in the n-dimensional binary cube and visualizes them in 2-D as chains of Boolean vectors. Actual Boolean vectors are laid out on this chain structure. Currently the system supports two visual forms: the multiple disk form and the "Yin/Yang" form.

Part 5 concludes the book with geospatial data analysis, decision making and problem solving and consists of chapter 17-21. This focus is not accidental – geospatial problems are naturally visual and spatial. Chapter 17 features a general framework for combining geospatial datasets. The framework is task-driven and includes the development of task-specific measures,

the use of a task-driven conflation agent, and the identification of task-related default parameters. The chapter also describes measures of decision correctness and the visualization of decisions and conflict resolution by using analytical and visual conflation agents. Finally, the chapter elaborates mathematical (geometric and topological) techniques for decision making and problem solving for combining geospatial data.

Chapter 18 addresses imagery conflation and registration problems by providing an Analytical and Visual Decision Framework (AVDF). This framework recognizes that pure analytical methods are not sufficient for solving spatial analysis problems such as integrating images. Without AVDF, the mapping between two input data sources is more opportunistic then definitive. A partial differential equation approach is used to illustrate the modeling of disparities between data sources for a given mapping function. A specific case study of AVDF for pixel-level conflation is presented based on Shannon's concept of mutual entropy. The chapter also demonstrates a method of computation reduction for defining overlapping image areas.

Chapter 19 looks at spatial decision making and analysis, which heavily depend on the quality of image registration and conflation. An approach based on algebraic invariants for the conflation/registration of images that does not depend on identifying common points is developed. This new approach grew from a careful review of other conflation processes based on computational topology and geometry. This chapter describes the theory of algebraic invariants and describes a conflation/registration method and measures of correctness for feature matching and conflation.

Chapter 20 presents technology for conflation algorithm development with a wide applicability domain. The sequence of steps starts from vague but relevant expert concepts and ends with an implemented conflation algorithm. The generic steps are illustrated with examples of specific steps from the development history of an area-based "shape size ratio" conflation algorithm. The fundamental "shape size ratio" measure underlying the algorithm has rather strong invariance properties, including invariance to disproportional scaling.

Chapter 21 presents an Artificial Intelligence technique for generating (on-the-fly) rules of visual decision making for use by experienced imagery analysts. The chapter addresses the construction of a methodology and tools that can assist in building a knowledge base of imagery analysis. Further, the chapter provides a framework for an imagery virtual expert system that supports imagery registration and conflation tasks. The approach involves three strategies: recording expertise on-the-fly, extracting information from the expert in an optimized way using the theory of monotone Boolean functions, and using iconized ontologies to build a conflation method.

This book is the first guide to focus on visual decision making and problem solving in general and for geospatial applications specifically. It combines theory and real-world practice. The book includes uniformly edited contributions from a multidisciplinary team of experts. Note that the book is not a collection of independent contributions, but rather a book of interconnected chapters. The book is unique in its integration of modern symbolic and visual approaches to decision making and problem solving. As such, it ties together the monograph and textbook literature in this new emerging area. Each chapter ends with a summary and exercises.

The intended audience of this book is professionals and students in computer science, applied mathematics, imaging science and Geospatial Information Systems (GIS). Thus, the book can be used a text for advanced courses on the subjects such as modeling, computer graphics, visualization, image processing, data mining, GIS, and algorithm analysis.

We would like to begin our acknowledgements by thanking all the contributing authors for their efforts. A significant part of work presented in this book has been sponsored by the US Intelligence Community, the Department of Defense, and the Department of Energy. Authors of individual chapters have made such acknowledgements in their respective chapters. Several other chapters have been supported by European funding agencies that are also acknowledged in the individual chapters. All support is gratefully acknowledged. Special thanks go to ARDA/NGA GI2Vis Program and NGA Academic Research Program managers, panel members, and participants for their interest, stimulating discussions and a variety of support. Several students contributed to this book as co-authors and others assisted us in other forms. Richard Boyce, Mark Curtiss, Steven Heinz, Ping Jang, Bea Koempel-Thomas, Ashur Odah, Paul Martinez, Logan Riggs, Jamie Powers, and Chris Watson provided such assistance.

Please find book-related information at www.cwu.edu/~borisk/bookVis.

Boris Kovalerchuk, James Schwing

PART 1

VISUAL DECISION MAKING

Chapter 1

DECISION PROCESS AND ITS VISUAL ASPECTS

Boris Kovalerchuk
Central Washington University

Abstract: This chapter provides a conceptual link between the decision making process, visualization, visual discovery, and visual reasoning. A structural model of the decision making process is offered along with the relevant visual aspects. Examples of USS Cole incident in 2000 and the Cholera epidemic in London in 1854 illustrate the conceptual approach. A task-driven visualization is described as a part of the decision making process and illustrated with browsing and search tasks.

Key words: Decision making, visualization, visual reasoning, visual discovery, task-driven approach.

A picture is worth a thousand words.
Proverb
A picture tells a thousand lies.
Hany, 2003
Sometimes, a picture is better than a thousand words.
NGA, Pathfinder, 2004

1. CURRENT TRENDS

Visual problem solving has been known for millennia as both great success and failure in science, mathematics, and technology. The quotes above sum up these facts in a few words. Below we present current trends in this area that indicate that the field is moving (1) from mostly geometric visuals to more abstract algebraic, symbolic visuals; (2) from the visualization of solutions to finding solutions visually, (3) from visual data mining to finding solutions visually; (4) from drawing tools to visual discovery and conceptual analysis, and (5) from abstract decision models to visual decision models.

1.1 From geometric visuals for algebraic symbolic visuals

The Pythagorean Theorem was proven by visual means more than 2000 years ago. The entire study of geometry existed as a visual problem-solving field more than one and a half millennia before René Descartes invented symbolic coordinates.

In 1953, Albert Einstein wrote that the development of Western Science is based on two great achievements: the invention of the *formal logic system* (in Euclidean geometry) and reasoning based on systematic *experimentation* during the Renaissance. For us it is important to notice that the formal logical system in Euclidean geometry was visual.

Historically in mathematics visuals are associated with geometry that can be traced to the concept of the number in Greek mathematics. This contrasts with "non-visual," abstract mathematics that began with Descartes and analytic geometry [Schaaf, 1930]. In other words, this is the fundamental conceptual difference between concrete visual forms (in geometry) and abstract forms (in algebra) of mathematics. Typically, an abstract algebraic form is not considered to be a visual form although abstract symbols, icons, are graphical representations.

However, the point is that the algebraic abstract form is also visual and in some sense, the *algebraic visual form* is more general than the *geometric visual form.* To be abstract does not necessarily mean to be non-visual. The concept can be visual, abstract and very productive simultaneously. Important differences between abstract and concrete hide significant similarities between geometric and algebraic approaches – both of them are visual forms, but concrete and abstract respectively.

The geometric form often is an individual invention for a specific task and that is not applicable to other tasks. This was one of the main reasons that mathematics moved from the geometric proofs of Greek mathematics to the more abstract Cartesian mathematics that permitted to work on geometric problems in an algebraic form.

Chapter 5 and 7 of this book show the productivity of the algebraic visual approach using historical examples. This productivity derives from the fact that solving algebraic equation in symbolic form is much more efficient than using words or geometry.

1.2 From visualization of solution to finding solution visually

Visualization of a solution is quite different from finding a solution visually. Thus, there is a significant conceptual and practical difference between

(1) visualization **of an already identified solution** and
(2) **visually finding the solution**.

The second task is much more ambitious and difficult especially for non-structured problems. A typical example of the first task is the **animation** of a known algorithm for a novice, to visually demonstrate the steps of an algorithm.

A different situation arises with the second task when we use animation to visually discover properties of an algorithm such as the number of loops and the amount of space required for the task.

If an animation tool permits viewing loops and space used, then it can serve as a visual problem-solving tool. Thus, it is possible that the same technique could be used for both tasks. This observation shows that the essence of the transition from task 1 to task 2 is not the visual technique itself (e.g. animation).

The essence of task 2 is in *matching a decision-making, problem-solving task with a visual technique.* Note that it is likely that a simple animation technique appropriate for the novice would not be sufficient for an advanced visual decision-making on algorithm efficiency by an experienced analyst.

1.3 From visual data mining to visual decision making

Visual data mining discovers *useful regularities visually* or visualizes patterns discovered by common data mining tools such as neural networks and decision trees. Visual decision making produces *decisions* that rely heavily on visual data mining, but useful regularities are only a part of the whole decision-making process.

In finance, visual data mining helps to discover market trends. Visual investment decision making is used to produce buy/sell signals and to help select a portfolio.

In medicine, visual data mining helps with patient diagnosis. Decisions about a course of treatment are heavily based on the diagnosis, but this is only a part of a treatment decision. For instance, a treatment decision should avoid a patient's allergic reactions and other negative side effects. Diagnostic data mining does not cover this issue.

In defense applications, visual data mining can provide valuable intelligence clues for planning an operation. A visual decision about a military strike requires that much more information be considered, including the availability of military forces and political realities. Visual data mining is also useful in breaking drug trafficking rings, but actual decision on these types of operations should involve more aspects of the problem.

1.4 From drawing tools to visual discovery and conceptual analysis

Let us begin our considerations here with an analysis that uses architectural Computer Aided Design (CAD) tools. Typical CAD tools help an architect *implement* an architectural solution similar to how a pen helps us recording our everyday solutions. This is obviously useful but it is not *guiding* an architectural solution. Finding an architectural solution is an extremely difficult task, because in essence architecture is art. This means that we need a *visual discovery* tool that is more complex than *visualization* of already discovered solution. One might argue that the architect can start sketching without a clear solution and can discover it in the process of using a CAD tool. Thus, the CAD tool would be a discovery tool too. We would disagree. Leo Tolstoy wrote and rewrote some of his famous works many times with a simple pen. Should we call his pen a discovery tool?

It is extremely difficult to distinguish between a genuine visual discovery tool and a less sophisticated tool. There are two extremes:

1. visual tools that provide an *algorithmic solution* and
2. visual tools that only help in *recording a solution.*

Most available tools lie somewhere between these two extremes. They support recording and visualization of the solution and partially support discovery of the solution.

At first glance, it maybe surprising but one of the best-known visual techniques that provides an algorithmic solution is basic elementary school arithmetic known for centuries. For instance, adding 35 and 17 we (a) write them one under another, (b) add 5 and 7, (c) write 2 which is the lesser digit of the sum 5+7 below the result line, (d) write 1 that is the greater digit of the sum 5+7 and known as the carry in a separate location, (e) add next digits 3 and 1, (f) add carry to them, and (g) write the result 5 below the result line next to the 2. This explanation text is less clear than a graphical form where 17 is really written under 35.

What is important in this example? The example shows an algorithmic process, *not an art*, where the result varies from person to person. Every person with an elementary school background should produce the same number 52 when the sequence described above is followed.

In architectural design, different architects can add two buildings to each other very differently. The CAD tool will support any of these solutions without guiding the solution. Thus, this process is not algorithmic but rather artistic. Somebody can argue that adding a building is much more complex task than adding numbers. It is true, but thousands years ago adding numbers was not an easy task either nor is it an easy task for elementary school students today. Later in this book, you will find the Chapter 4 written by D.

Barker-Plummer and J. Etchemendy which discusses in depth the visual reasoning problems in architectural design.

Visual problem-solving and decision-making tasks and approaches have a natural scale. This is the **assistance level** that can be provided by a decision support system (DSS). Such scale is a semantic scale that represents a specific type of semantic differentials [Osgood, 1952]. Our semantic scale has two extremes:

- recording and visualization tools, and
- mathematical, algorithmic tools and proofs.

This scale is depicted in Figure 1. Typical current CAD systems are obviously not fully algorithmic.

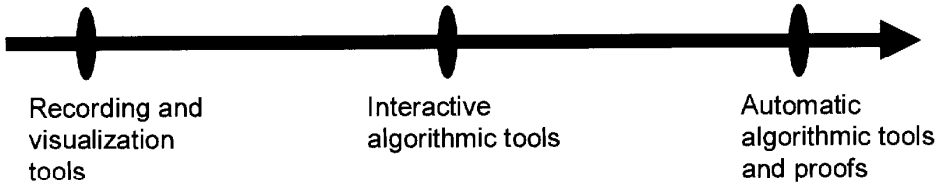


Figure 1. Assistance level scale

Recording and visualization tools are exemplified by visualization of the proof of Pythagorean Theorem. Note that for 2000 years there has been no automatic algorithmic tool for *discovering the theorem* $a^2+b^2=c^2$ better than a heuristic search in the set of all possible formulas with the form $a^n+b^m=c^k$. There are only *verification tools* that can prove already discovered theorem statement, $a^2+b^2=c^2$.

Another field that is moving from drawing tools to a deeper conceptual analysis is **designing software specification** especially in enterprise information systems.

Visual tools include traditional whiteboards and drawing tools with boxes and various connectors such as Visio and Unified Modeling Language (UML) diagrams. These tools are not limited by diagram drawing. UML permits one to capture the whole object-oriented logic of a design for further detailed programming.

However, visual support for upper-level conceptual data modeling with domain experts is arguably less than ideal [Halpin, 2000]. Other options that facilitate conceptual level design with visuals now include the Object Role Modeling (ORM) language combined ontologies presented in DAML-OIL language. This would permit the derivation of UML class diagrams integrated with ontologies.

1.5 From abstract decision model to visual decision models

The traditional mathematical decision-making and problem-solving approach assumes that a problem should be structured and presented as a **formal mathematical model**. This often requires a **double conversion**: from a *visual form* into a *formal model* and then from the formal model back to a visual form (see Figure 2).

For many real-world problems such as CAD and imagery analysis, building a formal model can be too complex to be realistic. On the other hand, the efforts to build a formal mathematical model can be unnecessary when solving those problems visually.

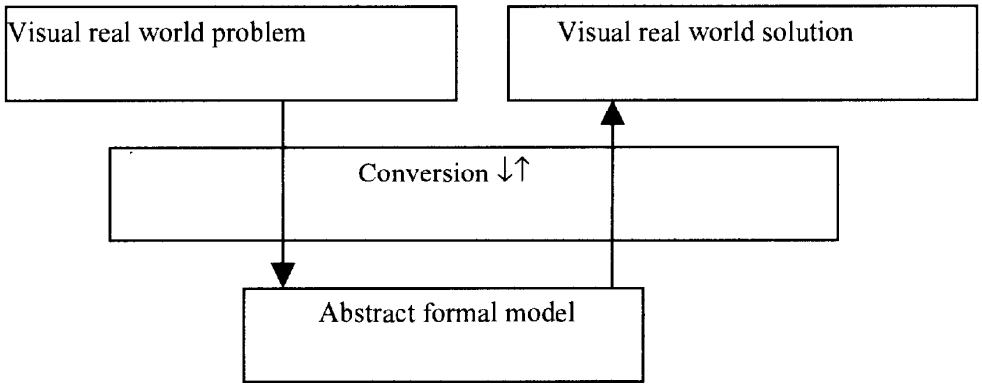


Figure 2. Double conversion process

There is a potential that a *visual model* can substitute/augment an abstract model to some extent. The above-mentioned example of a visual proof of the Pythagorean Theorem 2000 years ago shows that this has been feasible for a long time. Now is the time for expanding this field of study for problems that are posed visually from the beginning. Problems involving satellite imagery and map fusion, geospatial data conflation are among them. There are many other such problems in a variety of fields. The challenge is to find a way for *minimizing* double conversion effort.

Conversion is so common now that we often do not notice it. Modern digital computer architecture relies on the internal binary representation of numbers. This is not a convenient, compact visual form that a normal human being can operate - the decimal number 1024 is equivalent to binary number 10000000000. Thus, decimals are converted to binary and the results are converted to decimals again. Recall that some early computer architectures in 1940s and 1950s were decimal based (ENIAC) while others were based on Roman numerals (THROBAC I) without conversion to binary. The last example is especially notable, because Claude Shannon known as a father of

the information theory designed it. His THrifty ROman-numeral BAckward-looking Computer was able to add, subtract, multiply and even divide numbers up to 85 working only with Roman numerals [Calderbank & Sloane, 2001]. It is interesting to understand why the double conversion happened in the computer architecture field. The answer is that binary digital logic devices were simpler to build and conversion to and from binary was not too difficult. In other problems, conversion maybe and often is a significant or even formidable challenge. Often the original representation of a task is too complex to be solved, and a relatively simple visualization is impossible if we try to build the visual for the task "as is." **Simplification** of a task's representation may be the first step for visual data mining and decision-making as discussed in Chapters 15 and 16.

2. CATEGORIES OF VISUALS

2.1 Illustration, reasoning and discovery

Visuals can be classified in three categories: (1) illustration, (2) reasoning, and (3) discovery. **Illustration** means showing the essence of objects, events, solutions, decisions, or statements. **Reasoning** means showing why these are relevant objects, events, solutions, decisions, or statements. And **discovery** means showing how to find relevant objects, events, solutions, decisions or statements. These categories form another semantic scale that we call the **creativity scale** for visuals. This scale is illustrated in Figure 3 using the term statement as an umbrella term for any category to be presented visually. Illustration and discovery are the two extremes in this scale with many intermediate mixed cases. Reasoning occupies the middle of this scale.

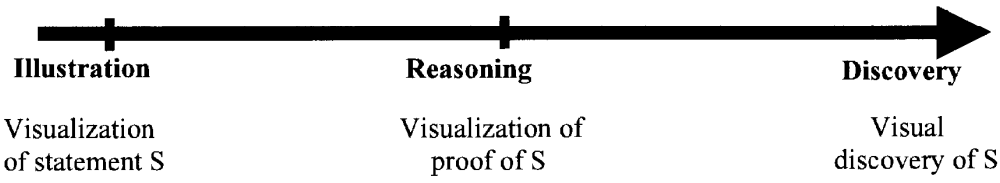


Figure 3. Scale of creativity levels of visual problem solving methods

A pure illustration of this is chess notation that permits us to replay a wining game, but does not give us a clue on how to select a winning strategy. The discovery category is clearly the most difficult; thus, it is not accidental that there is no visualization in this category for the Pythagorean Theorem although the Theorem has existed for 2000 years and was proved using geometric diagrams many times. Indeed, more than 370 different geo-

metric proofs have been published [Loomis, 1968]. All of them represent a *reasoning* category on creativity scale. One of visual proofs, shown in figure 4, is in essence the same as that used by Euclid. Now it is an animated Java applet [Morey, 1995]. Figure 5 is quite different from figure 4 fits more readily into the *illustration* category. Figure 5 both visualizes and visually proves the statement of the Theorem for a triangle with sides of length 3, 4, and 5, i.e. $3^2+4^2=5^2$. It does not prove the Theorem's general statement that $a^2+b^2=c^2$ for any right triangle.

Both figures 4 and 5 lack the ability to show us how the theorem could be *discovered.* A proof deals with a hypothesis that should be proved (verified) or refuted. To do this we first need to generate that hypothesis. That is to have a visual process helping generate an initial set of reasonable hypotheses that should include a true theorem statement. Next, we need to test hypotheses visually. Without a discovery process the situation would be similar to finding an exit in the maze using random trials. Thus, we distinguish three categories of visuals applied to theorems:

1) **Illustration**: visualization of the theorem statement (Figure 5),
2) **Reasoning (verification)**: visualization of the proof process for the theorem's statement (see Figure 4, triangles are moved without changing their areas), and
3) **Discovery**: visualization of the discovery process that identifies theorem's statement as a hypothesis.

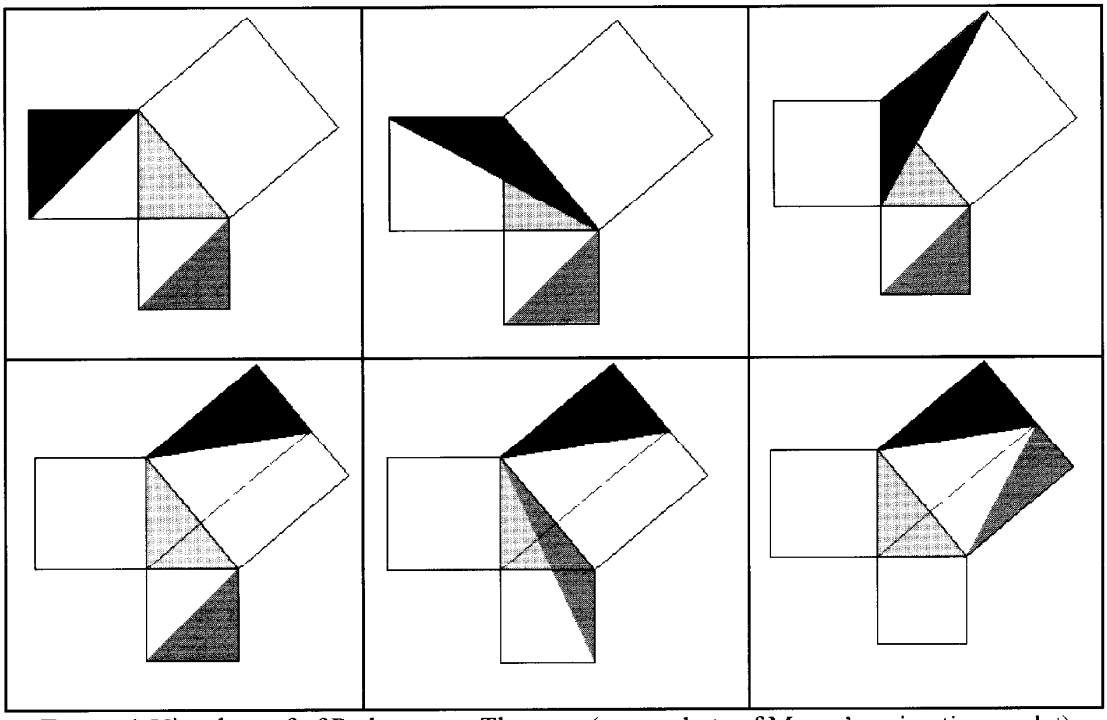


Figure 4. Visual proof of Pythagorean Theorem (screenshots of Morey's animation applet)

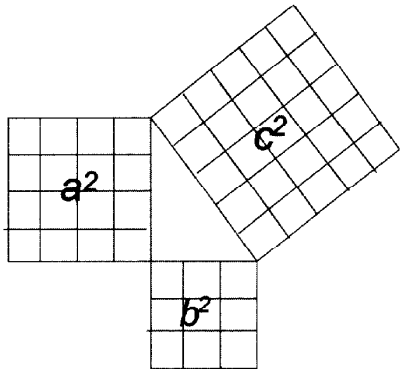


Figure 5. Result visualization for Pythagorean Theorem

For the Pythagorean Theorem when we are not proving the theorem, but using its proved result ($a^2+b^2=c^2$) in a particular situation (e.g., a=3, b=4), the reasoning step will be computing the specific numeric result, $c = 5 = \sqrt{3^2 + 4^2}$, by *applying the theorem*. Thus, for mathematical tasks based on the use of the theorems we can describe categories of visuals in the following way:.

- *Illustration*: visualization of the solution of an individual task based on the use of the theorem, (e.g., Figure 4 illustrates both the general statement of Pythagorean Theorem and an individual task with specific sides 3, 4 and 5).
- *Reasoning:* verification (proof) of the theorem, and computation of the result by applying the theorem, e.g., computing side *c* of the right triangle given sides *a* and *b*.
- *Discovery*: visualization of the discovery process that identifies theorem's statement as a hypothesis.

In visual **decision making** the listed categories have their counterparts:

- *Illustration*: visualization of the decision (solution) statement
- *Reasoning*: explanation of why this is a correct decision, verification of the hypothesis, and visualization of the reasoning process that leads to the decision statement using a verified hypothesis.
- *Discovery*: visualization of the process of hypothesis discovery.

2.2 Informal, heuristic, and rigorous visual decision making

The scale described in Figure 3 represents a *creativity level* for visuals ranging from illustration through reasoning and onto discovery. It does not however cover another important aspect of visual decision making and problem solving – the **algorithmic-level scale**. This scale can be characterized

over the following range: informal ("artistic") approach, heuristic algorithm, rigorous (full-solution) algorithm. This scale is shown in Figure 6.

An informal algorithm can be an instruction: "Follow good previous practice" accompanied by few examples of "good practice." The success of a heuristic algorithm depends strongly on the case and the skills of the subject matter expert (SME). A full-solution algorithm provides a rigorous and unambiguous way of producing a solution.

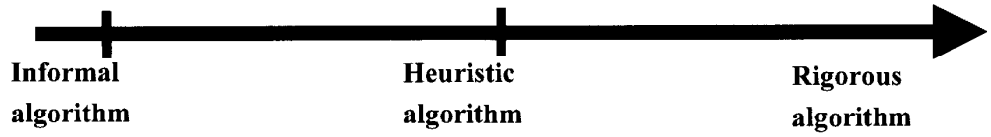
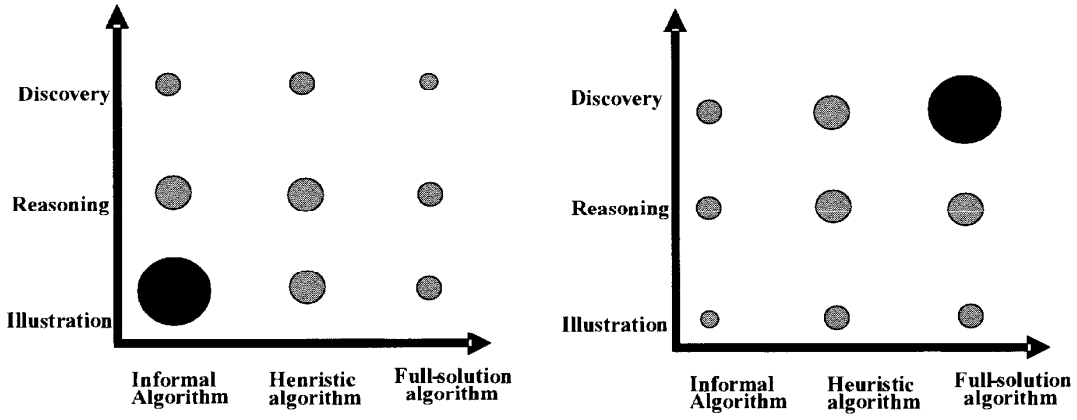


Figure 6. Scale of algorithmic level of a visual problem solving method

Figure 7 combines the scales for algorithmic and creativity levels. Here the sizes of circles indicate the relative number of methods currently available (part a) and the relative number of methods desired (part b). It is obvious from this figure that new methods dealing with full-solution algorithms for discovery tasks are in short supply.



(a) Available methods (b) Desired new methods

Figure 7. Combined scales for algorithmic and creativity levels

Below we provide some examples of tasks that occupy the extreme cells in Figure 7. The first one is an illustration (I) combined with informal algorithm (IA) for accomplishing a task, <I, IA>. Architectural drawing using CAD tools falls into this category. Another extreme is a discovery task (D) combined with a full-solution algorithm (FA) for discovery, <D, FA>. Discovery of the number π by interpolating a circle with polygons belongs to this category. Increasing the number of sides in the polygon increases accuracy of the solution. The two other extremes are represented by <D, IA> and

<I, FA>. Here <D, IA> means the discovery of the solution with informal algorithm in hand. A variety of guidelines in architectural and engineering design reside in this category. Such tools do not go beyond providing insight for finding a solution. The case described by <I, FA> is an illustration of an algorithm that provides a full-solution. For instance, it can show the regular polygon with, say, 100 sides and the approximation to the number π that it provides. Figure 5 represents another example in this category.

An example of the combination of reasoning (R) with a full-solution algorithm <R, FA> is the visual verification of the Pythagorean Theorem (see Figure 4).

The center point on both scales in Figure 7 is a combination of reasoning with a heuristic algorithm (HA) or <R, HA>. The visual, interactive scheduling of jobs using heuristic, greedy strategies such as "largest jobs first" is a representative of this category.

The creativity scale can be further elaborated by distinguishing between the Discovery of an Individual solution (DI) and the Discovery of a Process (DP) that can lead to several individual solutions.

Reasoning also has two subcategories. The first one is finding a solution by Applying a verified solution Process (AP), e.g., finding the hypotenuse for the triangle with sides 3 and 4 by applying the general statement of the Pythagorean Theorem. The second more challenging task is Verification of the solution Process (VP), e.g., proving the Pythagorean Theorem.

In Chapters 5 and 7, we provide more examples of visualization as illustration, visual reasoning, and visual discovery from the history of mathematics. The examples include the process of discovery (number and visual counting. Such analysis establishes a background for developing visual decision-making processes for modern tasks.

3. A MODELING APPROACH

Every day mass the media shows impressive visualizations of events in the World. One of them was published by "Time" magazine about the attack on USS Cole in Aden in October 2000 [Ratnesar, 2000]. This visualization provided a rich multilevel visualization. The visual is a sequence of increasingly focused pictorials that starts with a view of the World and ends up depicting an individual injured sailor. The visual presents six levels of detailed visualization in the process: (1) World, (2) region, (3) port, (4) ship, (5) damage area, and (6) sailor. While the visualization shows many details about USS Cole's equipment including armament, it does not help much in decision making – namely, how to prevent such deadly attacks. There are two reasons for that: (1) decision making is not a mass media goal and (2)

"rich" information is actually scarce for decision making. Figure 8 shows an iconic summary of this visualization that we constructed. This summary quickly shows that visualization from "Time" while being creative and impressive does not contain much information useful for decision making. It is clear that this is an example of the illustrative level of visualization only, but when the task is that of guiding reasoning and decision making, higher levels of visualization are needed.

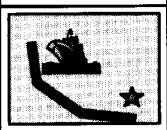
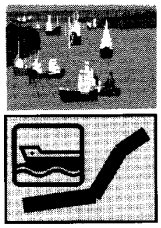
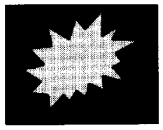
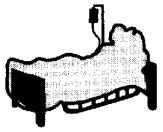
World map with a marked region of the incident	Region map with route and city of incident	Ship moves in the port with the route of the boat-bomb	Ship with identified armament and damage area	Ship's damage area	Injured sailor in the hospital bed
					
Level 1	*Level 2*	*Level 3*	*Level 4*	*Level 5*	*Level 6*

Figure 8. Structure of visualization of the attack on USS Cole

Consider next, another example as described by E. Tufte [Tufte, 1997] on the cholera epidemic in London in 1854 using an original work by Dr. J. Snow [Snow, 1855]. This example includes several visualizations. Some of them show growing death toll day by day in September 1854 ("within two hundred and fifty yards of the spot where Cambridge Street joins Broad Street, there were upwards of five hundred fatal attacks of cholera in ten days" [Tufte, 1997; Snow, 1855]). However, these visualizations did not help make decisions regarding how to stop the epidemic.

There was another visualization that matched/correlated the death tolls with locations of water pumps/wells. Specifically, Dr. Snow marked deaths from cholera on a map, along with locations of the area's 11 community water pump/wells. This visualization proved extremely useful for decision-making (DM). It prompted the authorities to shut down a specific pump located in the area with a high death toll.

An analysis of these examples shows that the first case (Aden) *lacks a discovered relation* between the attack and attributes useful for decision making to prevent such attacks in the future.

The second example obviously has such a relation, which was discovered by Dr. Snow on September 7, 1854, and which allowed the Board of Guardians of St. James's Parish to make a decision to prevent a further spread of the cholera by shutting down that well on September 8, 1854. The epidemic ended within two weeks.

These two examples help us to make a point about the concept of *decision making visualization* (DMV); namely, that is visualization useful for decision making based on:

- *a discovered relation/pattern* (DRP) *and*
- *a decision making model* (DMM).

The first example (Aden) is creatively impressive, but does not include the components DRP and DMM. The second example (London) includes both of them:

1. **Discovered relation** – people who used water from well *d (death)* on Broad St. died more often from cholera than people who used any other well:

$$\forall i\ (i \neq d)\ D(d) > D(i),$$

where $D(i)$ is the number of dead after drinking water from well i.
"There were only ten deaths in houses situated decidedly nearer to another street Broad St. pump." [Tufte, 1997; Snow, 1855].

2. **Decision making model** – shut down a well d if the death toll of people who used this well is higher than that for people who used other wells:

$$\forall i\ (i \neq d)\ D(d) > D(i) \Rightarrow Shutdown(d).$$

The DMM is very simple and people often do not even notice that the model is there. However, this simple model is a result of very non-trivial discovery by Dr. Snow of the relation between use of well water and death toll [Tufte, 1997, Snow, 1855].

Next we note that two categories of DDM models are necessary:

(a) A model for the decision-maker (e.g., city managers or the board of guardians of the parish) and

(b) A model for the analyst (e.g., Dr. Snow) who discovers relations for a decision-maker.

The model (a) for the *decision maker* can and should be *simple*, similar to the decision making model in (2.) above. The model (b) for the *analyst* must be *complex* enough to cover a wide range of possible decision alternatives. In the London example, the decision making model (2.) produced a single decision alternative – to shut down the pump/well d. A model of type (b) for the analyst might include many other **alternatives** to be explored:

1. Restrict the access of new people to the city,
2. Restrict the contact between people in the city limit,
3. Restrict the consumption of certain foods,
4. Use certain medications,
5. Restrict the contact of the population with certain animals,
6. Restrict the consumption of some drinks.

Actually, the research of Dr. Snow resulted in the last alternative (specifically to restrict/prohibit consumption of water from the well *d*). We have no historic evidence that Dr. Snow really considered all the alternatives (1) - (6). It is most likely that he came to the well water alternative without a formalized decision making model such as the model (b). Our goal is to show that if his decision-making and visualization process had been driven by a DMM with alternatives (1)-(6) then the water alternative (6) would have surfaced naturally and would have been investigated. This alternative can guide an investigation (including **exploratory visualizations**) instead of relying on insight of such extraordinary people as Dr. Snow. We illustrate the concept of model-based approach in Figure 9.

This conceptual model has two components. The first component involves an analyst, who builds a DMM model and discovers some relationships. The second component involves a decision maker who works with relationships discovered by the analyst. This work is based on discovered and visualized relations and a DMM for actual decision.

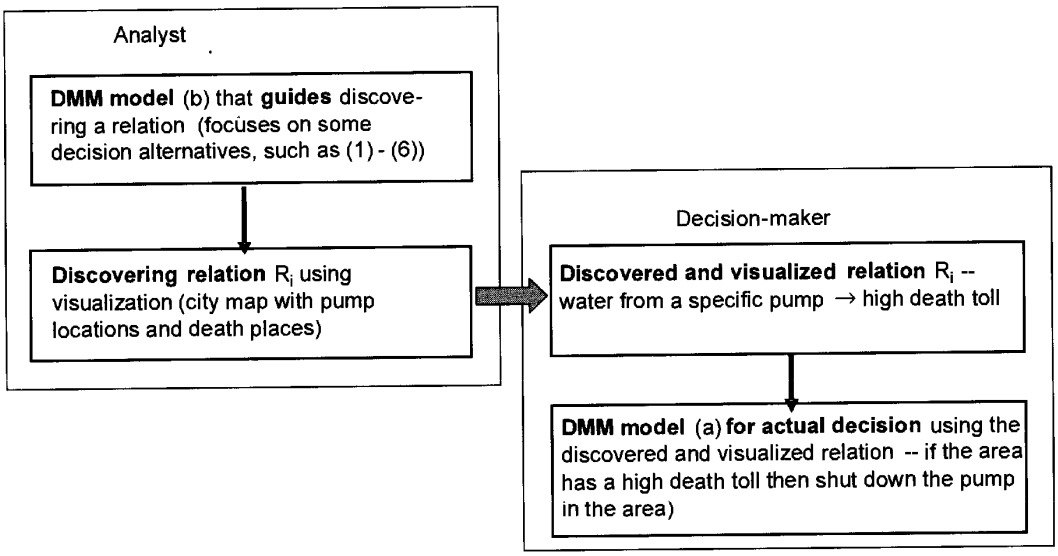


Figure 9. Conceptual Decision Making model structure and visualization

4. DMM MODEL AND DISCOVERY OF RELATIONS

In this section, we clarify connections between the decision making model of type (b) with the alternatives (1)-(6) and discovering supporting R_i relations. The decision making model (b) is not formulated in terms of any specific relation. It should be elaborated to include more objects and then investigate relations between them. For instance, alternative (6) can be developed to include such objects as water pumps, water distribution from the

pumps, methods of water treatment, and the type of population. These objects may be suspected of being related to the high death toll. We call this **structured information** for the DM. Thus, the DM model will grow like a tree (see Figure 10). The rectangles show relations to be investigated.

After providing such structural information, an analyst can investigate relations between death toll and each of the components: pumps, distribution routes, methods of water treatment and type of population. Currently this process is done by spatial data mining techniques (see Chapter 12 on SPIN system in this book). Visualization is a natural element in this analytical process.

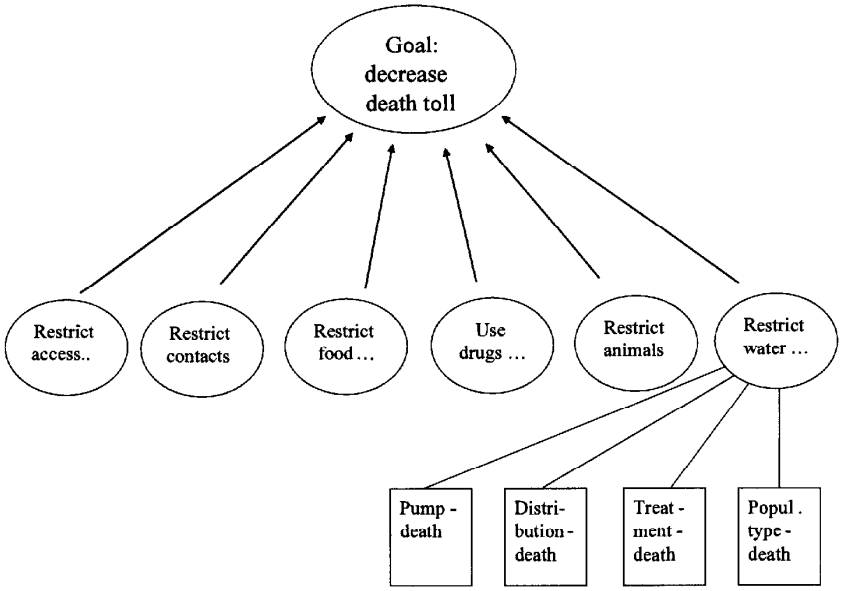
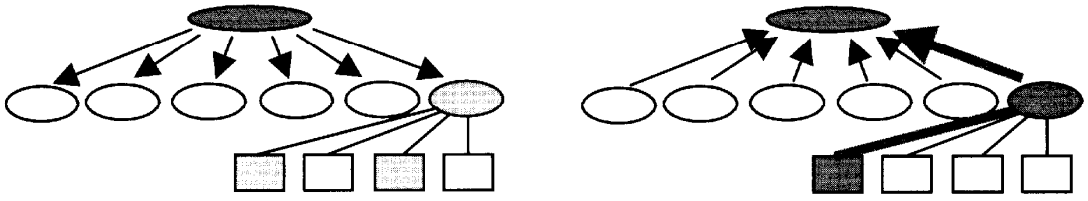


Figure 10. DM model with potential alternatives and relations



(a) (b)

Figure 11. Visual correlation in the process of discovery of relations

After such investigation an analyst can conclude that water treatment (boiling, pump operations, and so on) are the most highly correlated objects to the death toll. It is visualized by marking these components (see filled ellipses and rectangles in Figure 11a). The next stage of investigation could discover a very important relation between pump and the death toll (as marked in Figure 12b). Conceptually such discovery means matching the

knowledge with *likely states* based on the context of the problem [Marsh, 2000]. Sometimes this stage is called *understanding*.

Now the analyst can report the discovery to decision-makers (city managers). If they want to be sure that the analyst did not overlook some important alternatives then graphs (a) and (b) from Figure 11 provide them this information. If the decision-makers just want to consider a *course of action* based on the discovered relation then only the simple path marked on Figure 11 (b) is needed. The details of this path are presented on Figure 12. It shows the discovered pattern, its visual correlation with the decision (shut down the pump) and the visual correlation of the decision with the ultimate goal – decrease death toll.

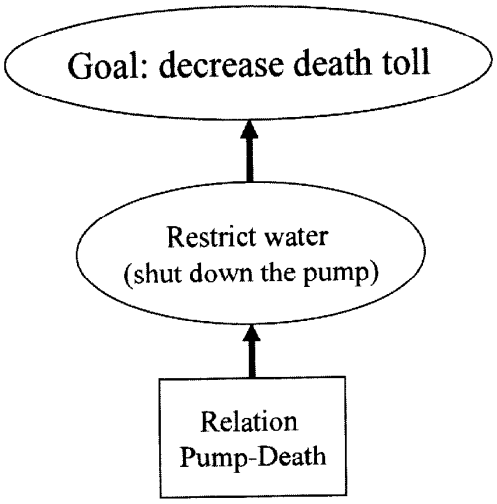


Figure 12. Final decision making model with discovered relation: visual correlation approach

Conceptually this stage means interpreting the currently *understood situation* in terms of the *desired end states* and *choosing the response* that best meets the objective. Marsh [2000] suggested the term **appreciation** for this stage.

Next we consider how visualization can help to discover the relation between pumps and the death toll. This, of course, is the classical work of Dr. Snow [Tufte, 1997; Snow, 1855]. Figure 13 presents the idea of Snow's visualization that is common now in geospatial visualization and spatial data mining. The idea is to bring together a city map with pumps (circles) and death locations (squares).

Figure 13 shows a higher death toll around one of the pumps. This visualization was critical for discovering the relation between the death toll and pumps, but once it has been discovered many **other visualizations** can serve as well as this one for convincing decision makers that the pump has to be shut down to save lives.

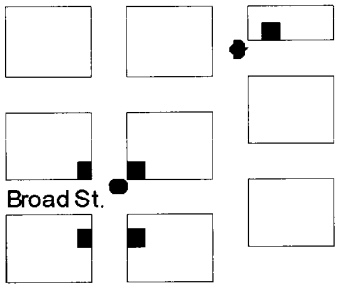


Figure 13. Mapping pumps and death toll

Figure 14 shows a simple alternate visualization. To get the death toll in Figure 13 we need to use a specific area around each pump. Figure 14 shows this plot for distance 250 yards from pumps, other similar graphs can be drawn for distances of 500, 1000, and 1500 yards. Obviously, Figure 14 is simpler than the information presented in Figure 13 especially if all 11 pumps studied by Dr. Snow along with all the city areas associated with these pumps were presented on one map. The map would contain a lot of information irrelevant to decision making on the cholera epidemic.

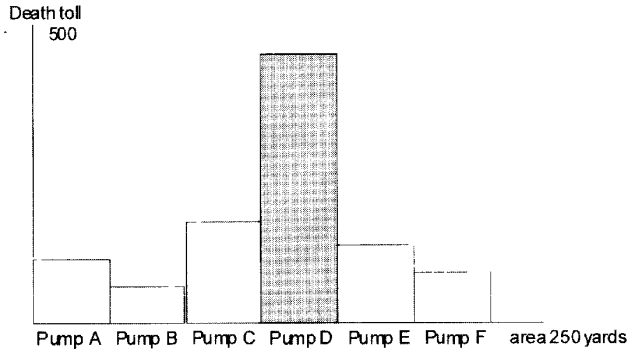


Figure14. Visual correlation

Next, note that the visualization in Figure 14 is not new to decision makers; they are familiar with this type of plot. This is a standard plot of the relation used for checking correlation. Thus, decision makers can concentrate on making decisions instead of studying a new method for presenting data.

We did not find evidence that, with respect to final decision making, the simple visual correlation (Figure 14) has any disadvantages in comparison with maps such as Figure 13 when a relation is already discovered. It seems that a simple visual correlation can serve very well. Moreover, it is possible that *new developments* of visual correlation methods for the final decision-making *are not needed.*

Consistent use of known visual correlation methods and their combinations has an obvious advantage – decision makers know and trust them. In the next section, we review more specifically visualizations suitable for this

stage. We note, however, that the situation can be much different when visual correlation is needed as a tool for *discovery of unknown relations*. Discovery of relations and their visual aspects is a major subject in such areas as computational intelligence, data mining, machine learning, and knowledge discovery (see for example [Kovalerchuk & Vityaev, 2000]).

5. CONCEPTUAL DEFINITIONS

Visualization and visual correlation can support decision-making tasks more efficiently if their role in this process would be clarified on the conceptual level. To do this we need a conceptual model of decision making process itself. Marsh [Marsh, 2000] provided a new conceptual model for the decision-making process (see Figure15) and contrasted it with the traditional model.

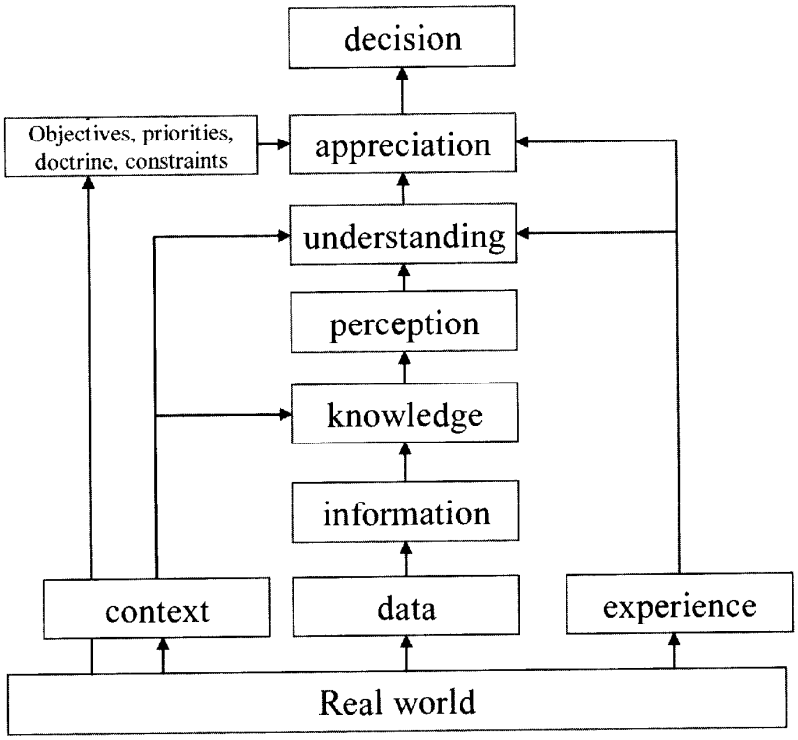


Figure 15. A new view of decision making (based on Marsh, 2000)

Each model element can be implemented with some level of visual support. Ideally, visualization of an element is derived from its role in the model. For instance, visualization of data, information, and knowledge is selected using the goal of the decision-making process as discussed above for the epidemic example.

The model suggested by [Marsh, 2000] operates with concepts that include data, information, knowledge, understanding, perception, context, experience, decision, and goal. Other concepts used in the model are appreciation, priorities, a doctrine, and constraints. Constraints may include tactics, techniques, and procedures (TTPs).

There are several and somewhat contradictory interpretations of these concepts in literature. Watson [Watson, 2002] defines **data** as properties of things and **knowledge** as a property of agents predisposing them to act in particular circumstances. Boisot [Boisot, 1998] defines **information** as a subset of the data residing in things that activates an agent – it is filtered from the data by the agent's perceptual or conceptual apparatus. Marsh [Marsh, 2000] defines knowledge, understanding, and appreciation as follows:

- **Knowledge** is the matching of available information to *known entities* and behaviors in the real world.
- **Understanding** is the matching of the knowledge with one or more *likely states* based on the context of the problem.
- **Appreciation** is interpreting the currently understood situation in terms of the *desired end states* and *choosing the response* that best meets the objective.

Thus, appreciation is defined as a higher form of reasoning that incorporates both knowledge and understanding. Figures 9 and 10 adapted from [Marsh, 2000] show the *central role of correlation* in this view of the decision-making process. It is consistent with our view on decision-making process as described above (see Figures 9-14).

According to Figure 16 **correlation** is:

a) a procedure matching *information based on observations* with multiple *alternatives* (hypotheses) regarding the current situation, or
b) a procedure matching *perception of the situation* (based on knowledge and assumptions) with multiple alternatives (hypotheses) regarding the current situation.

This concept of correlation is somewhat more general than the traditional correlation concept. The traditional concept often assumes that we correlate entities of the same modality, e.g., we may correlate stock market data for different days. In the correlation concept given above, entities are correlated with entities of potentially different nature: information is correlated with hypotheses about information; perception of situation is correlated with hypotheses about the situation. In essence, entities are correlated with their possible explanations, which may have a very different structure and nature.

There is also a significant difference between (a) and (b) in the level of human involvement. For example, in the case of Challenger catastrophe as noted in [Tufte, 1997], correlation between low temperature and a high fail-

ure rate was established. That is, correlation (a) was in place, but correlation (b) the perception of the situation was not highly correlated with the high failure rate. Thus, knowledge existed, but *understanding* of the situation did not.

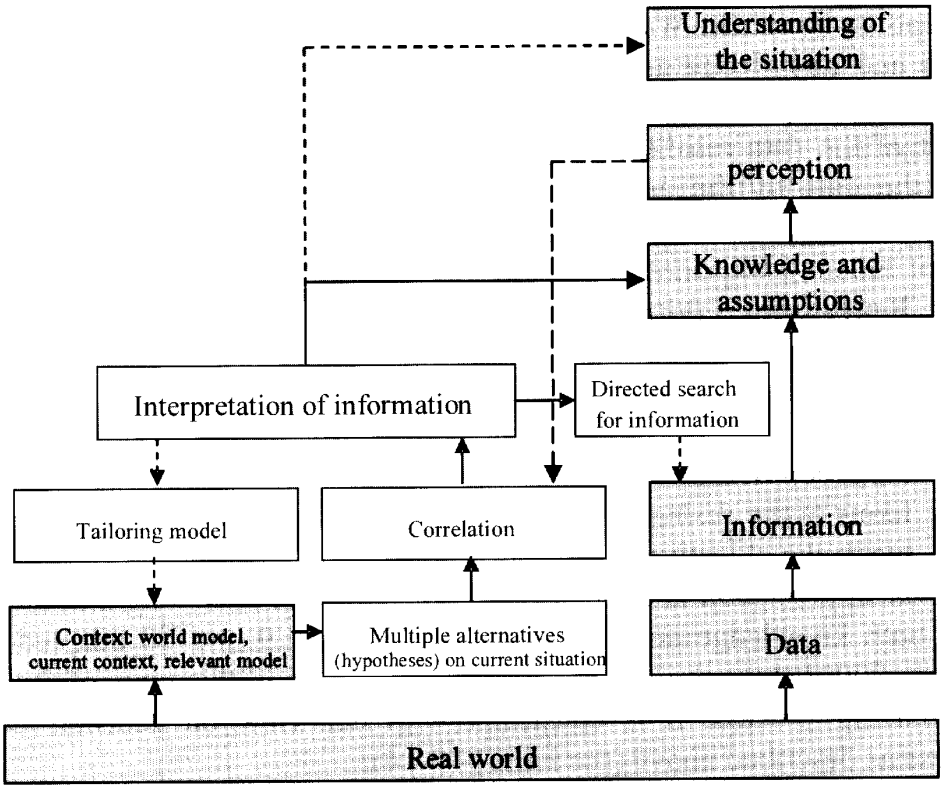


Figure 16. Building knowledge from information and understanding from knowledge (based on [Marsh, 2000])

This definition fits well with the classical mathematical concept of correlation, but at first glance, it does not include correlation between observations without any specific hypothesis (alternatives) explicitly formulated. A close look at the concept of correlation in fact assumes that there are some alternative hypotheses behind the scene. If somebody told us that A and B are correlated we would ask in what sense, i.e., we want to know what it means that A and B are correlated. The answer could be that pair (A, B) is correlated with the hypothesis of a linear relation between A and B, $b_i = ka_i$, where B = $\{b_i\}$ and A = $\{a_i\}$.

There could be many other possible alternative hypotheses such as $b=ka^2$. Thus, the commonly used expression that A correlates with B actually is a simplification of more exact statement that **pair (A, B) is correlated with hypothesis H,** where H states the *type of relationship* between the data A and B.

In the cholera epidemic example above, correlation between A_D = "location of pump D" and B_D = "death rate at the location of pump D" discovered by Dr. Snow means that pair (A, B) correlates with the hypothesis H, where H is the *relationship* between A and B given by: "death rate at the location of pump D is high". More formally this relation can be written as H = High(A, B)

Alterative hypotheses might correlate high death rate with other pumps: H_E = "death rate at the location of pump E is high", or H_Q = "death rate at the location of pump Q is high" using available data, i.e., High(A_E, B_E) and High(A_Q, B_Q). This example also illustrates that when we correlate the available information we may find that we know something, but we may not understand it [Marsh, 2000].

In the example above, we know that location of pump D is *correlated* to a high death rate, but we *do not understand* why this is the case. Historically, such an understanding came much later when cholera bacteria were discovered. Yet for decision making to stop the cholera epidemic, we do not need that deep an understanding. We just need to **understand** how the correlation can be *exploited* to stop the epidemic. Thus, we need to **interpret** the discovered correlation in a practical way. Having in mind that understanding was defined as matching the knowledge with one or more *likely states* based on the *context* of the problem [Marsh, 2000], we need to match the discovered relationship for pump D with the *current status* of the pump D. We may find out that it is turned on. This will be our *level of understanding of the situation.*

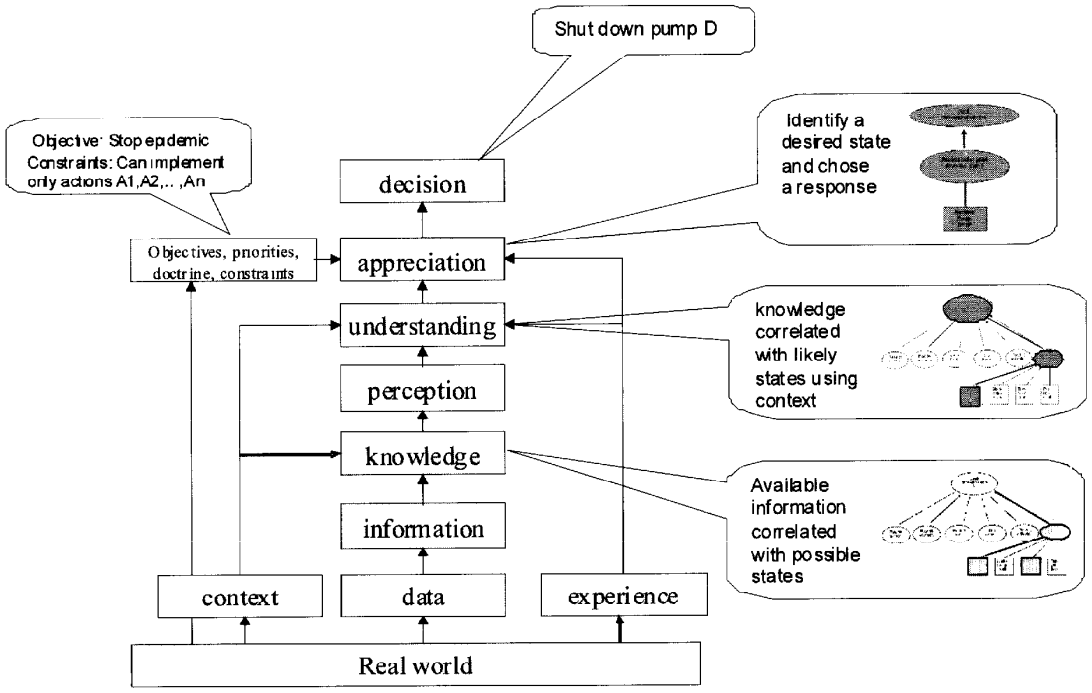


Figure 17. Visualization of epidemic decision making model

Applying Marsh's approach, the next step is to appreciate the situation, i.e., to interpret the currently understood situation in terms of the desired states and choose the response that best meets the objective.

Recognizing that shutting down pump D is a *desired state* would be the **appreciation** of the situation. Then the *actual decision making* could be to order pump D to *shut down*. This is illustrated in Figure 17, which combines Figures 11, 12 and 15.

6. VISUALIZATION FOR BROWSING, SEARCHING, AND DECISION MAKING

Visualization research and practice has shown that efficiency of visualization depends on both **type of data** and **type of task** visualized. However, this common wisdom often is ignored and visualization selection is made using only data type. Below we analyze specific requirements for three most common tasks that need visualization support: browsing, searching and decision making. At first, we clarify both common and significantly different features of these tasks. Common features can receive the same visualization support and features that are significantly different may need different visualization support. The main goal of visualization that supports **browsing** is to *identify a type of relevant information more easily*. That is, in browsing we wish to identify the search criteria. Thus, we distinguish browsing from **searching** in which the search criteria is already identified. In browsing the main goal is to get a clue, to *formulate criteria* about how actual relevant information may look.

In searching we look for *actual relevant information,* knowing the *search criteria* from the browsing stage. For instance, by browsing variety of houses a user can identify that having a lake nearby is more attractive than a nearby stadium. Searching focuses on finding, in a specific area, actual houses for sale with lakes nearby. Browsing could be done in the Boston area to identify the criteria "lake nearby". An actual search might then be done in the Seattle area where a person wants to buy a house. Finally, in this example the goal of *visualization to support decision making* is to make generating a decision easier.

From the visualization viewpoint, browsing and decision-making have similar requirements: support for observing several entities simultaneously to determine a preference. At first glance, it means that both tasks need the same visualization support. However, in the example above, even when the entities involved in the comparison are the same (houses and neighborhoods), the outputs are different.

In browsing, it is the *features* of houses and neighborhoods that are important, while in decision making it is a house as *a whole*. Thus, the process of browsing needs the support of making all the features of two or more houses visible simultaneously. In the extreme case, only features would be visualized for such browsing without showing the house as a whole.

For the decision-making task, searching provides houses with the needed features. This requires seeing all the features together. It is already known that searching provides these features. We actually need to see differences between features that do not explicitly belong to our list of search criteria. Alternatively, we may be interested in looking at other features (not used as search criteria) which will make the difference in choosing between the found houses.

If we view searching as a procedure with well-defined search criteria, what we are looking for as a visualization tool for searching can be quite different from visualization tools for browsing and decision making. It should support a query design (visual query) which speeds up search if the search is not fully automatic. Visualization of the search space can make searching interactive and faster.

7. TASK-DRIVEN APPROACH TO VISUALIZATION

Years of psychological studies had shown that graphics can expedite task-specific information processing [Ullman, 1964; Larkin & Simon, 1987; Casner, 1991]. For *computational tasks*, it was noticed that **quick perceptual inferences** such as determination of *distance, size, spatial coincidence*, and *color comparisons* judgments are much easier and faster than *logical inferences* such as mental arithmetic or numerical comparisons.

For the **search tasks,** it was also noticed that grouping related information in a single spatial locality, and encoding it by coloring, shading, and spatial arrangement efficiently supports preattentive and parallel visual search.

The BOZ system was one of the first visualization systems based on explicit task analyses [Casner, 1991]. It substitutes simple perceptual inferences in place of more demanding logical inferences, and streamlines information search. This system analyzes the **logical description of a task** and designs a provably equivalent **perceptual task.** Next it produces a graphic to support **perceptual inference** and minimization of visual search along with a perceptual procedure describing how to use the graphic to complete the task. BOZ generates *different presentations of the same information* customized to the requirements of different tasks. Experiments have shown that this

significantly shortened users' performance time in five different *airline reservation tasks*.

Casner [Casner, 1991] provides an extensive argumentation for **task-driven visualization**:

> Generalizations made about the observed usefulness of a graphic for one task are highly inappropriate since using the same graphic for different tasks often causes the usefulness of the graphic to disappear. Graphic design principles that do not take into account the nature of the task to be supported (e.g., "line graphs are best for continuous data") are too underspecified to be useful in general....

Casner also refers to similar previous judgments of [Jarvenpaa & Dickson, 1988] and concludes that effective graphic design should begin with a task analysis and be focused on finding the parts that might be performed more efficiently using a graphic.

To succeed, the task-driven approach needs a mechanism for matching the task (that is not originally in a graphic form) with a graphic. If the semantics of a task are expressed in propositional formalism then we need a graphic represented as *sentences* in a formal graphical language that has the same *matched, precise syntax and semantics* as the propositional formalism. Mackinlay [Mackinlay, 1986] designed such a system, APT, for the specific task of providing 2D presentations of relational information. Intensive studies of this subject are presented in [Mackinlay & Genesereth, 1985; Mackinlay, 1986; Mackinlay, 1988; Card & Mackinlay, 1997].

Another idea is to select a presentation format (from a set of predefined formats) that *best matches* the characteristics of the task. This approach was popular in 1980s in systems such as the following: AIPS [Zdybel, Greenfield, Yonke, & Gibbons, 1981], BHARAT [Gnanamgari, 1981], VIEW [Friedell, 1982], and APEX [Feiner, 1985].

The task-driven approach continues to be a focus of visualization research for the last ten years [Feijs & de Jong, 2000; Kerpedjiev & Roth, 2000; Kerpedjiev, Carenini, Roth, & Moore, 1997; Shaffer, Reed, Whitmore & Shaffer, 1999; Zhou, 1999; Beshers & Feiner, 1993]. In general, the **task-driven approach** (also called the *task-analytic approach,* the *mission-specific approach, and task-specific appraoch*) to the design of graphics in which graphic presentations are viewed as perceptually manipulated *data structures* helps to improve task performance.

The discussion above allows the formulation of general steps of the task-driven approach: (1) analyzing a *user task,* (2) generating *equivalent perceptual tasks* that can be performed more efficiently, and (3) designing accompanying graphics to support efficiency of the perceptual task. Accomplishing

steps (1)-(3) is a significant research challenge even for relatively simple tasks.

8. CONCLUSION

In this chapter, we have discussed problems of decision support by using discovered and visualized patterns. Visualization helps domain experts discover hidden patterns and correlations by directly augmenting computational intelligence methods. For domain experts (analysts and decision makers) discovery and visualization of hidden patterns is important but only as *a part of their decision making process.* We provided a conceptual view of the decision-making process, a structural model and relevant visual aspects of this process. A structural model of the decision-making process has been linked with the visual discovery of its components and visual reasoning with those components. This consideration has been illustrated with examples from USS Cole incident in 2000 and cholera epidemic in London in 1854.

A task-specific visualization approach that is a part of the conceptual view has been illustrated by providing conceptual differences between visual means that are needed to support three different tasks such as decision making, browsing, and searching.

9. ACKNOWLEDGEMENTS

This research has been sponsored by the US Intelligence Community via Advanced Research and Development Activity (ARDA) and National Geospatial-Intelligence Agency (NGA). This support is gratefully acknowledged.

10. EXERCISES AND PROBLEMS

1. Find a visualization of the NASA Columbia disaster in 2003 in the mass media. Build an iconic structure (summary) of that visualization similar to presented in Figure 8 and identify the level of the visualization (illustration, reasoning, decision making) for the iconic summary and the original mass media image. If illustration, reasoning, and decision making levels do not fit, formulate your own level category.

2. This assignment differs from #1 only in one aspect: the visualization should be taken not from Mass Media outlets but from NASA sources including an official report on Columbia disaster.

3. Compare the levels of the visualizations in assignments 1 and 2 from a decision making perspective. Discuss differences and similarities. Build your own visualization for Columbia disaster that intends to meet decision making intent. Provide justification for your design based on analysis of deficiencies of visualizations used in assignments #1 and #2.
   Tip: Start from adapting Figures 10 - 12 to this assignment.

4. Build a conceptual description of your design for assignment #3.
   Tip: Start from adapting Figure 9.

5. Find or design visualization that fits a searching task. Justify your solution.

6. Find or design visualization that fits a browsing task. Justify your solution.

7. Find or design visualization that fits a decision-making task. Justify your solution.

8. Describe general steps of a task-driven visualization approach. Give one example of such visualization from literature and design one example on your own. Identify the general task-driven steps in both examples.

Advanced

9. Elaborate conceptually the general steps of a task-driven visualization approach: (1) analysis of a user task, (2) generation of equivalent perceptual tasks that can be performed more efficiently, and (3) design of accompanying graphics to support the efficiency of the perceptual task. Tip: Your elaboration should decompose these three steps to smaller substeps. Be specific and provide examples for the substeps.

11. REFERENCES

Beshers, C., Feiner, S. AutoVisual: Rule-based design of interactive multivariate visualizations. IEEE Computer Graphics and Applications, 13(4), 1993, 41-49. http://www.cs.columbia.edu/graphics/projects/AutoVisual/AutoVisual.html

Boisot, M. Knowledge assets, securing competitive advantage in the information economy, Oxford: Oxford University Press, 1998.

Calderbank, R., Sloane, N. Claude Shannon 1916-2001, Nature, 410 (6830) April 12, 2001, 768. Accessed Jan. 2004 http://www.research.att.com/~njas/doc/ces5.html

Casner, S. A Task-Analytic Approach to the Automated Design of Graphic Presentations, ACM Transactions on Graphics, 10 (2) April 1991, 111-151, (Online Version: http://portal.acm.org/citation.cfm?doid=108360.108361)

Card, S. K., Mackinlay, J. The Structure of the Information Visualization Design Space. IEEE Symposium on Information Visualization, Phoenix, AZ, 1997, 92-99. http://www2.parc.com/istl/projects/uir/pubs/pdf/UIR-R-1996-02-Card-InfoVis97-DesignSpace.pdf.

Feijs, L. and de Jong, R. 3D visualization of software architectures, Communications of ACM, 41 (12) 2000, 73-78.

Feiner,S. APEX: An experiment in the automatic creation of pictorial explanation, IEEE Comput. Graph. Appl.,Nov., 1985, 29-37.

Friedell, M. Context-sensitive, graphic presentation of information. Computer Graphic. 16 (3) 1982, 181-168.

Gnanamgari, S. Information presentation through default displays. Ph.D. dissertation, Univ. of Pennsylvania, May 1981.

Halpin, T., UML Data Models From An ORM Perspective, http://www.orm.net/uml_orm.html, 2000.

Hany, F., A Picture Tells a Thousand Lies, New Scientist (09/06/03) 179 (2411) 2003, 38.

Jarvenpaa, S. L., Dickson, G. W. Graphics and managerial decision making Research Based Guidelines. Commun. ACM 31 (6) 1988, 764-774.

Kovalerchuk, B., Vityaev, E., Data Mining in Finance: Advances in Hybrid and Relational Methods, Boston: Kluwer Acad. Publ., 2000.

Kerpedjiev, S., Roth, S., Mapping Communicative Goals into Conceptual Tasks to Generate Graphics in Discourse, Proceedings of Intelligent User Interfaces (IUI '00), New Orleans, LA, January, 2000, 60-67 (Online Version: http://www-2.cs.cmu.edu/Groups/sage/Papers/IUI00/gmicro9.pdf)

Kerpedjiev, S., Carenini, G., Roth, S. F., and Moore, J. D. Integrating Planning and Task-based Design for Multimedia Presentation, International Conference on Intelligent User Interfaces (IUI '97), Orlando, FL, January 1997, ACM, 145-152 (On line Version: http://www-2.cs.cmu.edu/Groups/sage/Papers/IUI-97/IUI-97.html)

Larkin, J., Simon, H. Why a diagram is (sometimes) worth 10,000 words. Cognitive Sci. 11, 1987, 65-99.

Loomis, E.S. The Pythagorean Proposition: Its Demonstrations Analyzed and Classified and Bibliography of Sources for Data of the Four Kinds of ``Proofs", 2nd ed. Reston, VA: National Council of Teachers of Mathematics, 1968. 284 p

Mackinlay, J. Automating the design of graphical presentations of relational information. ACM Trans. Graph. 5,2, Apr. 1986, 110-141 (Online Versión: http://www2.parc.com/istl/projects/uir/pubs/pdf/UIR-R-1986-02-Mackinlay-TOG-Automating.pdf)

Mackinlay, J. D. Applying a Theory of Graphical Presentation to the Graphic Design of User Interface, Proceedings of the ACM SIGGRAPH Symposium on User Interface Software (UIST '88), 1988, 179-189. http://www2.parc.com/istl/projects/uir/pubs/pdf/UIR-R-1988-05-Mackinlay-UIST88-Applying.pdf

Mackinlay, J. D. and Genesereth, M. R. Expressiveness and Language Choice, Data and Knowledge Engineering 1(1, June), 1985, 17-29. http://www2.parc.com/istl/projects/uir/pubs/pdf/UIR-R-1985-01-Mackinlay-DKE-Expressiveness.pdf.

Marsh, H.S. Decision Making and Information Technology, Office of Naval

Research, 2000. Retrieved January 2004 from: http://www.onr.navy.mil/sci_tech/information/docs/dec_mak_it.pdf

NGA, Pathfinder, May-June, 2004, http://www.nga.mil/NGASiteContent/StaticFiles/OCR/mayjune2004.pdf

Osgood, C E,. The nature and measurement of meaning. Psychology Bulletin, 49, 1952, 197-237.

Ratnesar, R. Sneak Attack, Time, Oct. 23, 2000 vol. 156 no. 17. http://www.time.com/time/magazine/article/0,9171,1101001023-57755-2,00.html

Snow, J. On the Mode of Communication of Cholera, London, 1855.

Shaffer E., Reed D., Whitmore S., Shaffer B., Virtue: Performance visualization of parallel and distributed applications, Computer, v. 12, 1999, 44-51.

Tufte, E., Visual Explanations: images and quantities, evidence and narrative, Connecticut: Graphics Press, 1997.

Ullman, S. *Visual routines*. Cognition 18, 1964, 97-159.

Watson, I., Applying Knowledge Management: techniques for building organizational memories, Advances in Case-Based reasoning, 6th European Conf. ECCBR 2002, LNAI 2416, Springer, 2002, 6-12.

Zdybel, F., Greenfield, N, R., Yonke, M.D., and Gibbons, J. An information presentation system. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981, 978-984.

Zhou M., Automated Generation of Visual Discourse, Ph.D. thesis, Columbia University, 1999. Retrieved January 2004 from: ftp://ftp.cs.columbia.edu/pub/zhou/Thesis.tar.gz

Chapter 2

INFORMATION VISUALIZATION VALUE STACK MODEL

Stephen G. Eick
Univeresity of Illinois and SSS Research, Inc., USA

Abstract: Visual decision-making involves a problem, users, and a software distribution model. We describe the information visualization value stack and identify a framework that defines problems where it creates significant value. We also describe successful models that support visualization deployment and characterize different types of visualization users.

Key words: Value model, applications, sweet spot.

1. THE INFORMATION VISUALIZATION VALUE STACK PROBLEM

Over the last decade information visualization has emerged as an exciting research area that is addressing a significant problem: how to make sense of the ever increasing amounts of information that has become widely available. With the growth of networking and decreasing cost of storage it has become technically feasible and cost effective to store and access vast sets of information. The academic, business, and government challenge is how to make sense of this information and translate the insights into value-producing activities.

As a new emerging field there will certainly be opportunities for information visualization technology. There have already been some early successes and also some failures. Unfortunately, not all information visualizations create enough value so that users will switch over from conventional users interfaces to adopt new visual interfaces. This paper presents a simple frame-

work that predicts problem areas where information visualization will achieve utilization -- that is being useful enough so that users will adapt new visual interfaces.

Information visualizations are exciting and the demos inevitably generate interest among potential users. Unfortunately, however, visualization, as exciting as it is, only involves the user interface or presentation layer in a technology stack. Useful information applications solve problems that involve collecting data, manipulating it, organizing it, performing calculations, and finally presenting the results to users. The value of the application is captured by the complete system. It is often the case that each system component individually is not particularly useful. For example, tires are not useful without a car, but better tires improve a car's performance. The presentation layer, like beauty, is only "skin deep" and the usefulness of the application comes from the whole solution and not just the "lipstick."

Thus, by itself, information visualization is naturally a feature of system and rarely is a complete application by itself. This, unfortunately, makes utilization difficult. With a few exceptions, the technology must be part of an application to capture sustainable value. Information visualization "makes it better" but does not make it. The information visualization value stack challenge is to find applications where information visualization creates enough value, either by itself or as part of an application, to support utilization.

2. WHERE DOES INFORMATION CREATE HIGH VALUE?

At its most basic level, information visualization is a technique for helping analysts understand information. This section describes four classes of information problems, illustrated by examples, where visualizations create value.

2.1 User interface is the application

In certain cases, the user interface is essentially a complete application. The canonical example of this is **computer games** which are innovative and sophisticated user interfaces that involve, relatively speaking, little computation and no data integration. Successful games must have a great user interface that challenges and engages prospective players within the first few seconds.

Visual Presentation and Branding involves creating custom, 3D displays of information for presentations that are visually exciting. It frequently incorporates aspects of branding and has a high glitz and wow factor. Typical presentation and branding techniques include animations and colorful 3D displays.

Figure1 shows two examples of information visualizations for presentation and branding. The visualization on the left shows activity on the NASDAQ stock exchange and the visualization on the right shows website activity.

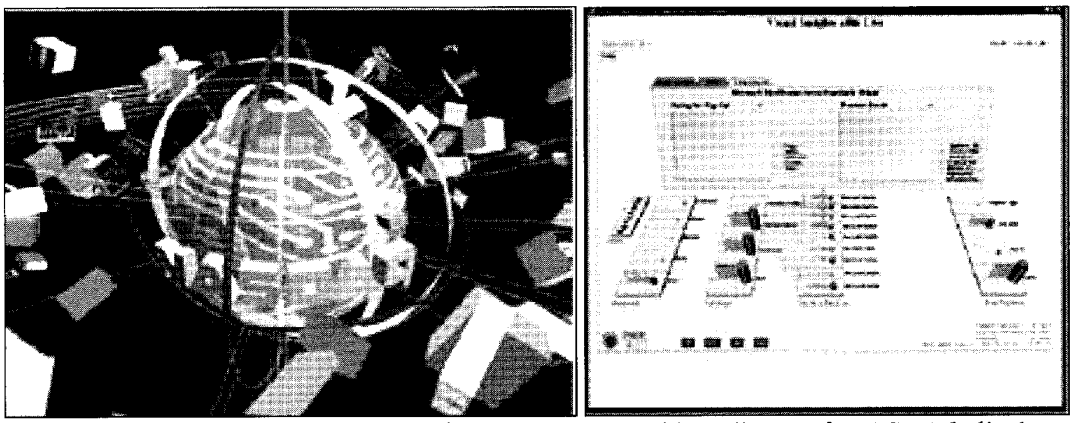


Figure 2. Information visualizations for presentation and branding. Left NASDAQ display and Right: Visual Insights' eBizLive product for showing website activity. See also color plates.

Executive Dashboards provide decision-makers with instant access to key metrics that are relevant for particular tasks. Much of the intellectual content in dashboards is in the choices of metrics, organization of information on the screen, and access to supporting, more detailed information. Information visualization techniques improve this presentation. Executive dashboards may include the ability to export result-sets to other tools for deeper analysis.

State-of-the-art implementations of active executive dashboards are web-based, interactive, and dynamic, involve no client-side software to install, and often include **action alerts** that fire when pre-defined events occur. End user customizations include **sorting, subsetting, rearranging layouts** on the screen, and the ability to **include** or **exclude** various metrics. It is common for visual reports to be distributed via email, published on a corporate intranet, or distributed through the internet.

Real-time Visual Reports are related to executive dashboards but provide an active presentation of an information set consumable at a glance. Although the distinction is subtle, visual reports usually involve fat client-side software and thus can provide richer presentations of the information. Visual Reports exploit the idea that a picture is worth a thousand words and,

in particular, for many tasks a picture is more useful than a large table of numbers.

Visual reporting systems are:

1. Easy to use for both sophisticated and non-sophisticated user communities,

2. Suitable for broad deployments, and

3. Provide capabilities for flexible customization.

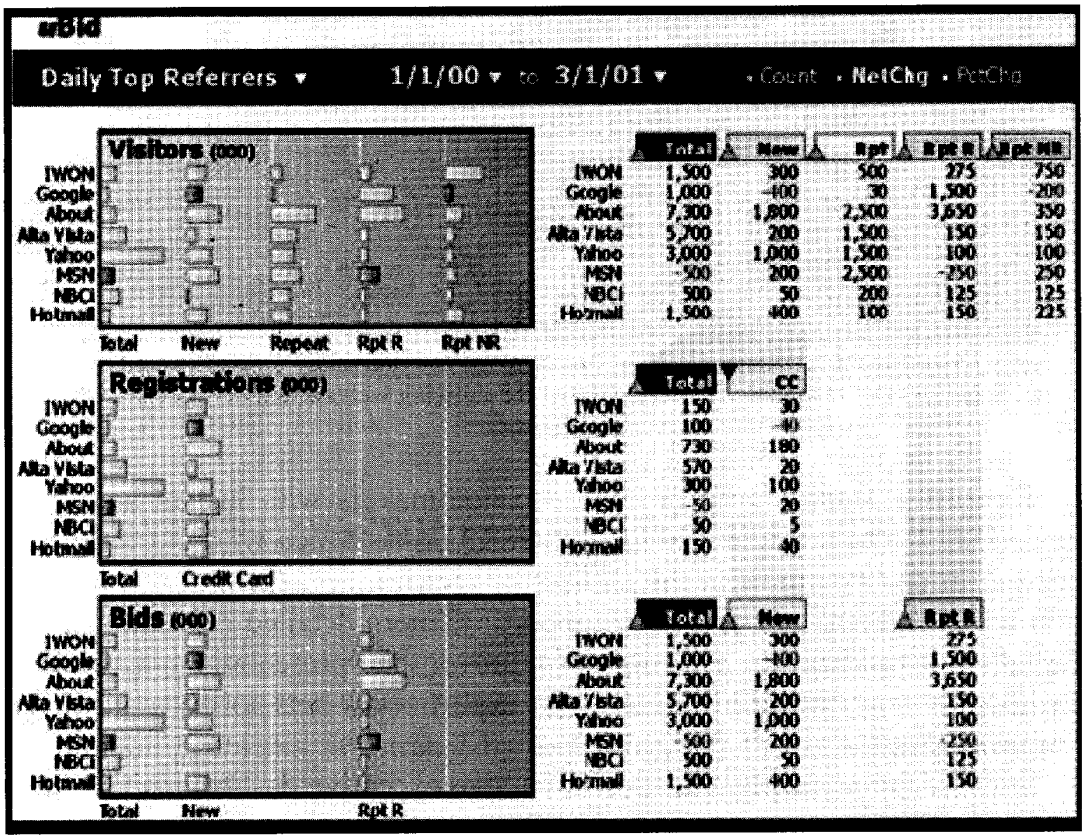


Figure 2. Executive Dashboard courtesy of Bill Wright. See also color plates.

Visual reports, as with all reports, are a tool for **assumptive-based** analysis. Reports answer "point questions": How much of a particular item is in stock? Where is it? How long will it take to get more? Reports are ideal for operational tasks, but do not provide full analytics, or enable an analyst to automatically discover new information that a user has not thought to ask about.

This is a well-known characteristic of all report-based analytical solutions. The reports pre-assume relationships that are reported upon. The diffi-

culty with this approach is that most environments are too complex for a predefined report or query to be exactly right. The important issues will undoubtedly be slightly, but significantly different. This is particularly true for complex, turbulent, environments where the future is uncertain. There are two common solutions to this problem. The first is to create literally hundreds of reports that are distributed out to an organization, either using a push distribution mechanism such as email or a pull mechanism involving a web-based interface. The second involves adding a rich customization capability to the reporting interface that increase UI complexity. Unfortunately, neither works particularly well. Although a report containing novel information might exist, finding it is like finding a needle in a haystack. Adding UI features makes the reporting system difficult to use for non-specialists.

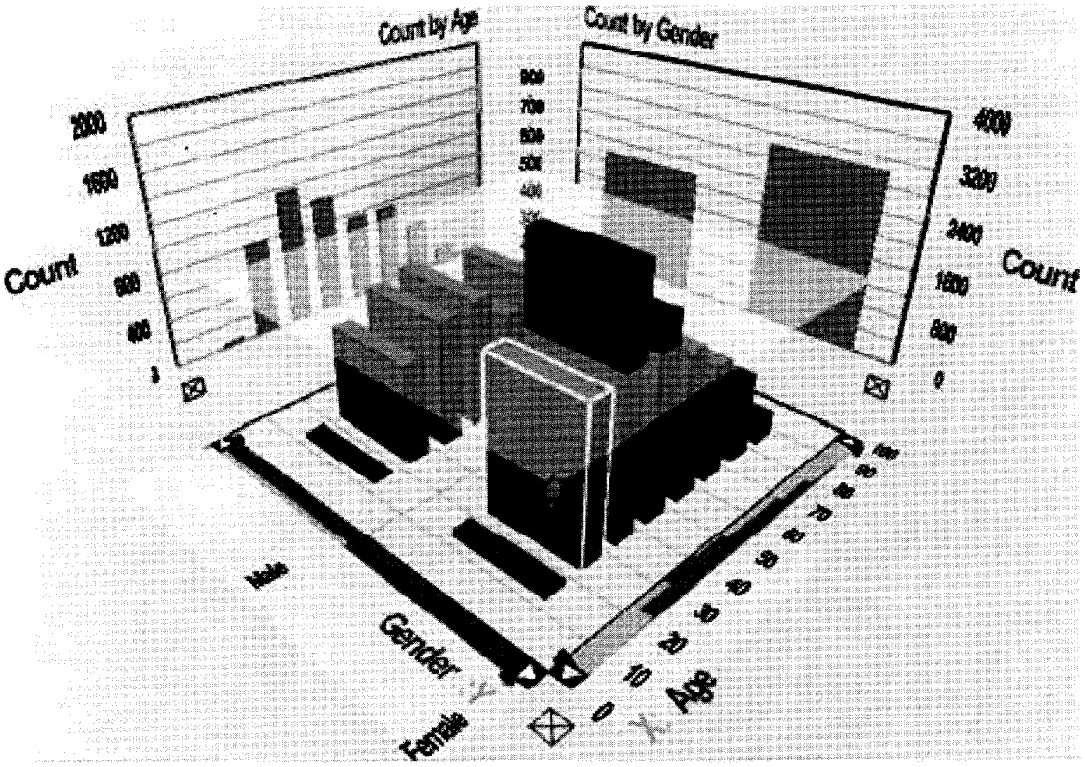


Figure 3. Real-time 3D Visual Report courtesy of Visual Insights.
See also color plates.

2.2 Information discovery applications for deep analysis

Visual discovery-based analysis addresses the shortcomings assumptive-based analytics by providing a rich environment to support novel discovery.

Systems supporting visual discovery are used by analysts and frequently combine data mining, aspects of statistics, and also predictive analytics. Visual discovery is domain specific and iterative. Information visualization improves visual discovery by enabling discoveries to often "jump" out and may lead to "why" questions. For example, in a supply chain management analysis, visual discovery might identify an unusual inventory condition that would lead to a subsequent investigation into why it occurred and how to fix it.

ADVIZOR [Codd, 1977] is an example of system for visual discovery. It consisted of a workspace with standard data acquisition capabilities, and a set of visual metaphors, e.g. views, each of which showed data in a particular way. Some of the views were conventional (e.g., barchart, linechart, piechart) and some were novel (Data Constellations, Multiscape, Data Sheet). For visual analysis, the views could be combined into fixed arrangements called perspectives.

Within any perspective the views could be linked in four ways: by *color, focus, selection and exclusion.* Components linked by color used common color scales and those linked by *focus, selection and exclusion* were tied by data table row state using a *case-based* model [Eick, 2000a].

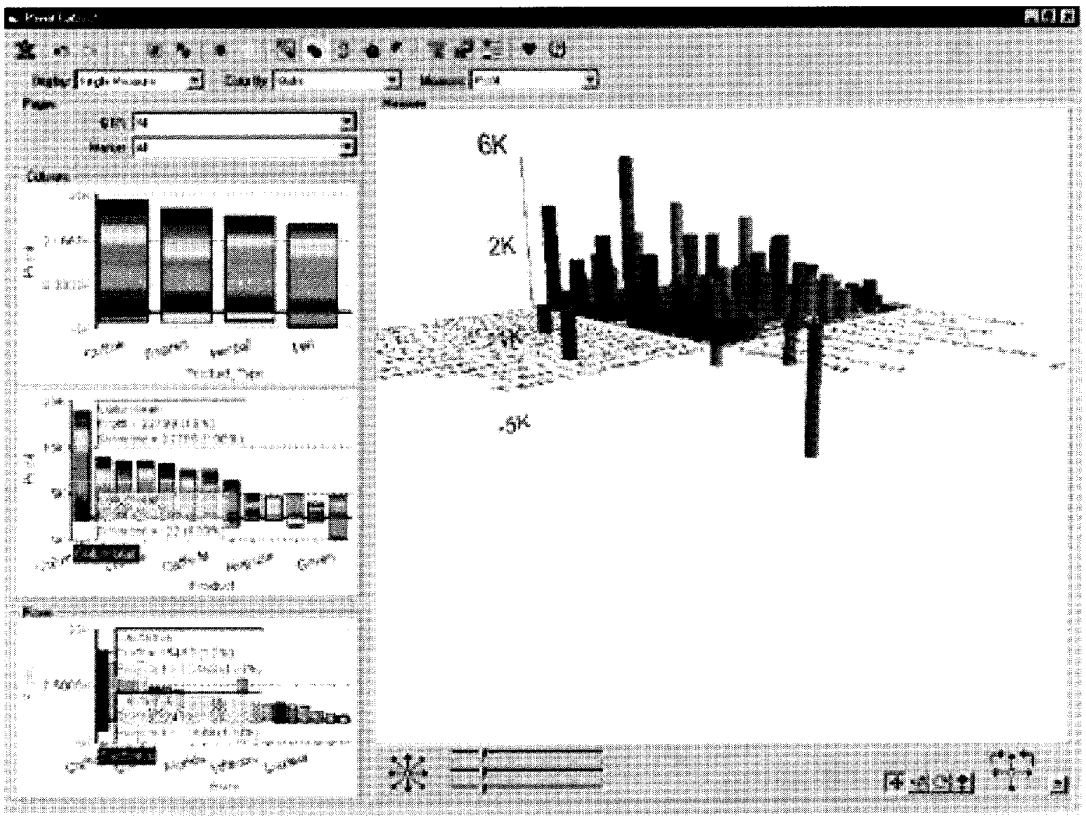


Figure 4. Advizor 2000 Visual discovery and analysis tool.
See also color plates.

ADVIZOR contained three interesting ideas:

1. **Perspectives** extend general linked view analysis systems by reducing complexity for non-expert users. Perspectives are "authored" by "power users" who are ADVIZOR experts. Analysts, who are domain experts, but not power users, use the perspectives as a starting point for analysis and as a guiding framework. The output from their analysis, visual reports, may be published and distributed for use by casual users, executives, and decision-makers. The ADVIZOR user model is similar to that employed by spreadsheets where there are spreadsheet authors, users, and consumers.

2. **Visual Design Patterns** are recurring patterns within perspectives that are broadly useful and apply to many similar problems. Following the object-oriented programming community [Eick, 2000b], recognizing, cataloging, and reusing design patterns have the potential for significantly improving information visualizations.

   Examples of design patterns are Shneiderman's *Information-seeking Mantra*: *Overview Zoom, Filter, Details on Demand* [Card, 1999]. The overview shows the entire dataset, e.g. all movies in the dataset, and supports the ability to zoom in on interesting movies and query the display with the mouse to extract additional details. This design pattern incorporates interactive filters, frequently bar and pie charts that enable you to filter out uninteresting folders so that you display only the data that is interesting. Filtering might be by category, numeric range, or even selected value.

   Another design pattern, called *Linked Bar Charts,* is particularly strong for data tables containing categorical data. Categorical data, sometimes called contingency tables, involves counts the number of data items organized in various bins or subcategories. This design pattern employs one bar plot for each categorical column with the height of the bar tied to the number of rows having that particular value. In statistical terms each of the bar charts shows a marginal distribution. As the user selects an individual bar, the display recalculates to show one-way interactions. Using exclusion and selection shows two-way interactions.

3. **Visual Scalability**, the third interesting idea, involved the realization that each visual metaphor has an inherent scalability limitation [Eick, 2003] - a large enough dataset will overwhelm any visual technique. Scalability is determined by controllable factors such as visualization technique, monitor resolution, computing environment,

data structures, algorithms, and uncontrollable factors such as human perception.

The important and exciting observation is that there is a set of tools and techniques that can be used to "ratchet up" the scalability of any visual component. These include interactivity, panning and zooming techniques, identification and selection, focus and context, multi-resolution visual metaphors, automatic aggregation, and improved labeling algorithms. When combined, the tools can give rather stunning increases in visual scalability, e.g. two to three orders of magnitude. See Figure 5.

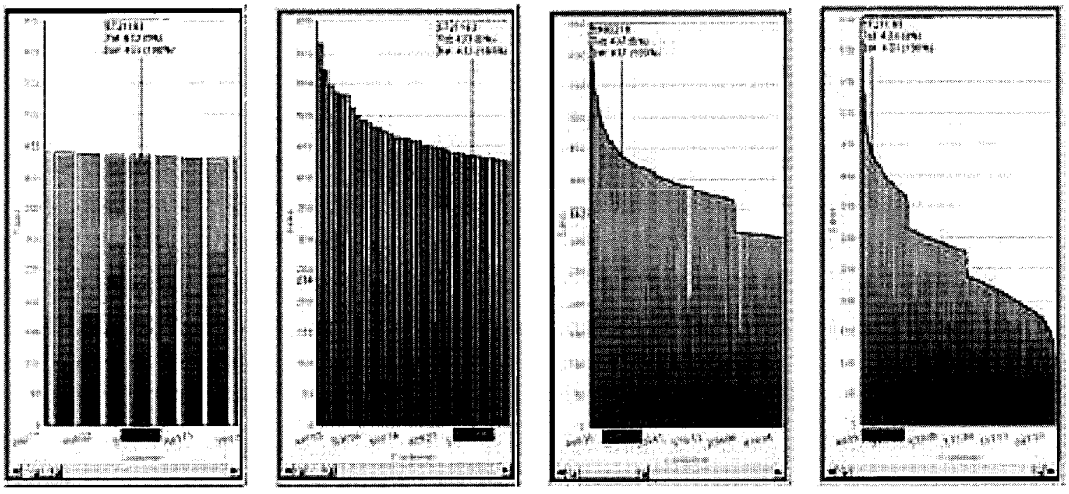


Figure 5. Bar chart scalability is increased by using levels of rendering detail and a red overplotting indicator at the top of the view. Scalability in this case facilitates locating and then focusing attention on particular bars.
See also color plates.

2.3 Information visualizations for searching and exploration

Information visualizations focused on visual searching involves undirected knowledge discovery against massive quantities of uncategorized, heterogeneous data with varying complexity. This scenario is typical of web searching where users recognize information when they find it. Searches are iterative, intuitive, and involve successive refinements.

The key measures for the performance visual searching systems revolve around the amount of information per unit of search effort expended. The search effort may be measured in user time, number searches, personal energy, etc. The results, or information found, may be measured in articles,

references, relevance, novelty, ease of understanding, etc. Different systems exploit various design points trading off these factors.

2.4 Task-specific visualizations

Task-specific visualizations help users solve critical, high-value tasks. Examples include visualizations to:

1. Execute on-line equity trades (Figure 6),
2. Manage complex communications networks, and
3. Operate nuclear power plants.

These visualizations are tuned to particular problems often delivered as part of a complex system. They are highly valuable, frequently involve fusing of a large number of information streams, and serve both as an output presentation for information display and also control panel and input interface for user operations.

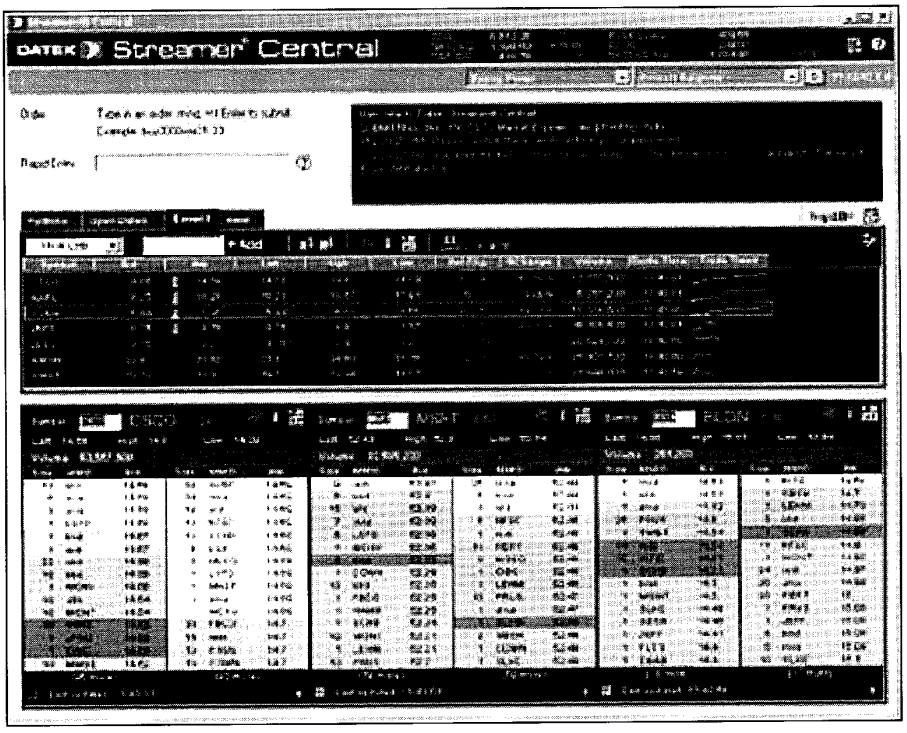


Figure 6. Information visualization for on-line stock trading

3. INFORMATION VISUALIZATION SWEET SPOT MODEL

Information Visualization problems can be defined by three dimensions:[1]

1. *Dataset size* is a measure of the total amount of data to be analyzed. Although some might disagree, information visualization techniques are not needed for small datasets containing tens to hundreds or perhaps even a few thousand observations. In these cases reports, spreadsheet graphics, and standard techniques work fine. More powerful techniques are unneeded.

   Conversely, information visualization techniques do not scale to analyze massive datasets containing gigabytes of information. The basic problem is that information visualization is technique that makes human analysts more efficient and human scalability is quite limited. The exact scalability limits of information visualization are subject to debate and are an active research area [Crow, Lantrip, Pennock, Pottier, Schur, Thomas, et al., 1995; Eick, 2003]. Most researchers would agree, however, that massive datasets containing hundreds of thousands to millions of observations are too big and need to be subdivided, aggregated, or in some way reduced before the information can be presented visually. Information visualization, it would seem, cannot be applied to analyze massive image databases containing millions of images, but might be applied to meta data associated with the images.

2. *Dataset complexity* can be measured by the number of dimensions, structure, or richness of the data. Information visualizations are not needed for (even large) simple datasets with low dimensional complexity. Statistical reduction tools such as regression work fine and are sufficient in this situation.

   Conversely, datasets of massive complexity containing thousands of dimensions are too complex for humans and thus for information visualizations. Some have argued that information visualizations can cope with as many as fifty dimensions, although a more practical upper limit is say half to a dozen dimensions.

3. *Dataset change rate* is a measure of how frequently the underlying problem changes. Static problems, even for very complex problems, can eventually be solved by developing an algorithmic solution. The algorithmic solution has a huge advantage over an information visu-

[1]The original version of this idea is due to Doug Cogswell.

alization-based solution since the algorithm can be applied repeatedly without the need for expensive human analysts.

Conversely, analysis problems involving change or other dynamic characteristics are extremely difficult to automate because the problem keeps moving. In these cases human insight is essential. Humans, however, cannot cope with problems that change too quickly. We are incapable of instantaneous responses. Human analytical problem solving occurs on a time scale of minutes to months. We must automate problems needing faster response and partition problem those involving longer time scales.

Table 1. Attributes with low and high values

Attribute	*Low Value*	*High Value*
Dataset size	10^2-10^3	105 to106
Dataset complexity	2 or 3 dimensions	50 dimensions
Dataset change rate	minutes	Months

As shown in Table 1 the sweet spot for information visualization involves analysis problems of moderate data sizes, rich, but not overwhelming, dimensional structure, that change, are not easily automated, or for some reason need human involvement. Examples of prototypical applications include the following.

Network Management for complex networks where the system is dynamic, constantly changing with new protocols, new devices, and new applications. The systems are instrumented and collect alarms with complex dimensional structure. It is frequently the case that the number of events (alarms) exceeds the capacity of network visualizations and must be algorithmically reduced.

Customer Behavior involving human buying patterns and transaction analysis is an ideal candidate for information visualizations. Human behavior is complex, unpredictable, and dynamic. Furthermore, although aggregate numbers of transactions are large, for any individual or set of individuals the numbers of transactions are not overwhelming and easily suitable for analysis.

Intelligence Analysis is an ideal candidate for information visualization. It is difficult to automate, involves complex dimensional data, is dynamic, and necessarily involves human analysts.

4. SUCCESSFUL DEPLOYMENT MODELS FOR INFORMATION VISUALIZATIONS

Successfully deploying information visualizations involves solving a technical problem and creating a business model that supports widespread utilization. Broadly speaking, there are three classes of business models for software companies.

Custom software is written to solve a specific problem, usually for a single customer. The problem being addressed must be significant, valuable, important, and yet specialized enough so that general solutions do not exist. The projects often involve next generation technology and new approaches to problems.

Typical price points for custom software projects usually start at $250K. Custom software is sold directly by the vendor with six months to two-year sales cycle. The sales team is highly specialized and the sales process frequently involves company executives.

Organizations involved with customer software include universities, government labs, large commercial organizations, and boutique specialty shops. Although it might seem surprising to some, research universities and government labs are custom software developers where the funding agencies effectively hire university principal investigators using BAAs and solicitations to solve important custom problems. In this setting the principal investigators function as both sales professionals and also lead fulfillment efforts with "graduate student" development teams.

In the large organizations that sponsor customer software development there are commonly multiple roles. It is often the case particularly with government-sponsored projects that the funding organization is not the organization that will eventually use the software and the users of the software may not receive the value from its use. These separate organizational roles complicate the software sales process. For example, the National Science Foundation funds research to build software for scientists to use. Scientists use the software to solve important national problems. Thus citizens are the ultimate beneficiary. In the commercial environment the CFO (Chief Financial Officer) funds a project that is implement by the CIO (Chief Information Officer) for a business unit. Thus three organizations are involved.

Enterprise software, sold by commercial companies, is essentially a flexible template that is "implemented" on site, either by the vendor or a "business partner." In the implementation phase the template is customized for a particular customer by connecting up data sources, define the specific reports a company needs, populating tables, e.g. inserting employee names into a payroll file, etc. For an enterprise application, data integration is essential.

Since enterprise software is reusable, it can be sold more economically than custom software. Generally, price points for enterprise software range from $25 to $250K. The sales model for enterprise software may be direct at the higher price points, e.g. SAP, or through local business partners who are "certified" by the vendor.

Shrink-wrap software is highly functional software that solves a specific problem very well. The software usually is customer installed and provides for little or no customization. Customer support, if provided, is usually self-serve via a web site or perhaps with limited help desk support.

Shrink-wrap software is almost always sold by distributors[2] or OEMed to the hardware vendors and sold as part of a bundle. As a mass-market item, the price point for shrink-wrap software is less than $25K and more frequently less than $1K.

Relating the business models back to the visualization deployments, most of the demand for information visualization has been met with custom research software built by universities, government labs, and large communications companies. The customers are the military, intelligence community, biomedical researchers, and other highly specialized users. Demand for information visualization within the research community is healthy.

Within the enterprise category we might expect information visualization-enabled applications to emerge. In this category, the value is provided by the whole application and an information visualization presentation layer could be described as a software feature or add-on product. Within this category there have been some early successes. Perhaps the most notable success has been Cognos Visualizer. Cognos sold 300K[3] units of Cognos Visualizer, an add-on for Cognos PowerPlay, at $695 per unit and some of the other "Business Intelligence" software vendors have had similar experiences.

The "Gorilla" analytic application within shrink-wrap category for visualization is Microsoft Excel. It is generally considered to be good enough for 90% of problems and essentially everybody has it.

5. USERS OF VISUALIZATION SOFTWARE

There are three broad classes of potential information visualization users: *Scientists*, *Analysts* including both intelligence and commercial analysts, and *Business User.*

[2]Microsoft, the largest producer of shrink-wrap software, sells essentially all of its software through distributors.

[3]For comparison, a software application that sold 2,000 to 5,000 units would generally be considered successful.

Scientists have deep needs for information visualization, are extremely technical, and work on the most significant problems. They want powerful tools for cutting-edge analyses.

Analysts, particularly in commercial companies, also have a strong need for information visualization, but tend to have specialized needs. They are not as sophisticated as scientists are and will not tolerate raw software packages.

Business users need simple information visualizations and are easily frustrated by complex software. Business users are numerous, have budget, but need solutions to problems and are not inherently interested in the complexly that excites scientists and analysts.

These three classes of users have different needs and varying tolerances for complex software. A research challenge is to create software that is sophisticated enough to solve scientific problems and yet easy to use for business users.

6. CONCLUSION

Information visualization involves the presentation layer with is naturally a feature of many products. By itself, it usually has insufficient value to support widespread usage and deployment.

It is generally a feature of an application and a critical component of a solution. This paper describes several of these cases, illustrates them with examples, and defines a simple model for information visualization's sweet spot.

7. ACKNOWLEDGEMENTS

Many of the visualizations were created by the Visual Insights staff.

8. EXERCISES AND PROBLEMS

1. Identify other factors that might be included in the information visualization value stack model.

Advanced

2. Extend information visualization value stack model to ASP-based applications.

3. Formalize information visualization value stack model.

9. REFERENCES

Card, S.K., Mackinlay, J.D. and Shneiderman, B, Readings in Information Visualization: Using Vision to Think. San Francisco, California: Morgan Kaufman, 1999.

Codd, E.F. Extending the Database Relational Model to Capture More Meaning, Association for Computer Machinery, 1977.

Crow, V., Lantrip D., Pennock, K., Pottier, M, Schur, A., Thomas, J., et al. Visualizing the Non-Visual: Spatial Analysis and Interaction with Information from Text Documents. In Information Visualization '96 Proceedings; 1995 October 30. Atlanta, Georgia: IEEE Computer Science Press 1995.

Eick, S. G. ,Scalable network visualization, Visualization Handbook 2003.

Eick, S.G. Visual discovery and analysis, IEEE Transactions on Computer Graphics and Visualization January-March 2000; 6:44–59.

Eick, S. G., Visualizing Multi-Dimensional Data, IEEE Computer Graphics and Applications 2000; 34: 44-59.

Eick, S. G., and Fyock, D.E., Visualizing Corporate Data. AT&T Technical Journal 1996; 75:74-86.

Eick, S. G., Karr, A. F., Visual scalability, Journal of Computational Graphics and Statistics March 2002; 11:22–43.

Eick, S. G., Wills, G.J., High Interaction Graphics. European Journal of Operational Research 1995; 81:445–459.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J., Design Patterns. Addison-Wesley, 1995.

Hanrahan P., Stolte, C., and Tang, D. Polaris: A System for Query, Analysis, and Visualization for Multidimensional Relational Databases. IEEE Transactions on Visualization and Computer Graphics 2002; 8:52-63.

Havre, S. Hetzler, E., Nowell, L., and Whitney, P. Theme River: Visualizing Thematic and Computer Graphics 2002; 8:9-20.

Hill, W.C., Hollan, J.D., McCandless, T., and Wroblewski, D., Edit Wear and Read Wear: Their Theory and Generalizations, CHI '91 Conference Proceedings 1991

Keim, D.A., Information Visualization and Visual Data Mining. IEEE Transactions on Visualization and Computer Graphics 2002; 8:1-8.

PART 2

VISUAL AND HETEROGENEOUS REASONING

Chapter 3

VISUAL REASONING AND REPRESENTATION

Boris Kovalerchuk
Central Washington University, USA

Abstract: Reasoning plays a critical role in decision making and problem solving. This chapter provides a comparative analysis of visual and verbal (sentential) reasoning approaches and their combination called heterogeneous reasoning. It is augmented with a description of application domains of visual reasoning. Specifics of iconic, diagrammatic, heterogeneous, graph-based, and geometric reasoning approaches are described. Next, explanatory (abductive) and deductive reasoning are identified and their relations with visual reasoning are explored. The rest of the chapter presents a summary of human and model-based reasoning with images and text. Issues considered include: cognitive operations, difference between human visual and spatial reasoning, and image representation. One of the main our statements in this chapter is that the fundamental iconic reasoning approach proclaimed by Charles Peirce is the most comprehensive heterogeneous reasoning approach.

Key words: Visual reasoning, spatial reasoning, heterogeneous reasoning, iconic reasoning, explanatory reasoning, geometric reasoning.

The words or the language, as they are written or spoken,
do not seem to play any role in my mechanism of thought.
Albert Einstein

1. VISUAL VS. VERBAL REASONING

Scientists such as Bohr, Boltzmann, Einstein, Faraday, Feynman, Heisenberg, Helmholtz, Herschel, Kekule, Maxwell, Poincare, Tesla, Watson, and Watt have declared the fundamental role that images played in their most creative thinking [Thagard & Cameron, 1997; Hadamard, 1954; Shepard & Cooper, 1982].

Problem solving and decision making is based on reasoning, where the result of such reasoning is a solution or decision. Herbert Simon [Simon, 1995] pointed out that Aristotelian logic and Euclidean geometry were major and abiding contributions of the Greeks to reasoning in language (natural or formal) and drawing inferences from diagrams and other *pictorial sources* to solve problems of logic and geometry. Note that despite the frequent references to Greek mathematics as an origin of visual reasoning, the Chinese and Indians knew a visual proof of the Pythagorean Theorem in 600 B.C. before it was known to the Greeks. This visual proof is shown in Figure 1 [Kulpa, 1994].

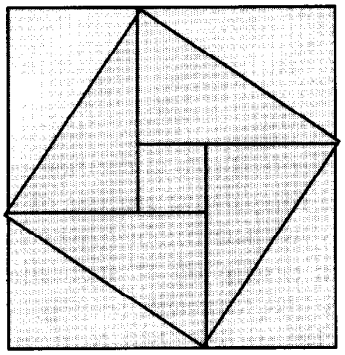


Figure 1. Pythagorean Theorem known to the Chinese and Indians [Kulpa, 1994]

It is widely acknowledged that visual diagrammatic, **iconic representations** of reasoning are better understood than verbal explanations because diagrams and icons are symbols that better resemble what they represent than text [Thagard & Cameron, 1997].

Simon [Simon, 1995] discusses the *heuristic nature of human reasoning* in problem solving as an argument for using non-verbal visual reasoning using diagrams as a tool to foster reasoning and to find answers. According to Simon, traditional reasoning in non-visual formal logic is much more helpful in **testing the correctness** of the reasoning than in **identifying the statement** to be inferred (finding a solution).

Another direction to consider for a visual approach is to foster **problem finding** in the space of alternative problems vs. problem solving that we discussed above. Einstein and Infeld [Einstein & Infeld, 1938] stated:

> The formulation of a problem is often more essential than its solution, which may be merely a matter of mathematical or experimental skill. To raise new questions, new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advance in science.

It is well known this process is extremely informal not only in scientific discovery but in design (architectural and others). Several attempts have

been made to provide a comprehensive structural picture of this informal visual process. Hepting [Hepting, 1999] formulated this picture as a collection of 16 design principles. We structured them into two processing categories:

I. **Support for a model of problem solving by**: (1) supporting restructuring and reorganizing of alternatives, (2) encouraging the user to discover things personally, (3) supporting the systematic exploration of a conceptual space, (4) allowing the user to concentrate on the task, (5) providing an external representation of the possible choices, and (6) allowing each user to apply personal judgment.

II. **Technical tools to support problem solving by:** (1) supporting the user in working directly with images, (2) supporting the user in combining images, and their elements, (3) allowing the user to employ heuristic search techniques, (4) recording all aspects of the visual representations of the design, (5) supporting collaboration, (6) assisting the user in choosing images, (7) allowing the user to interact with the images, (8) supporting multiple visual representations, and (9) supporting the use of current solutions in future enquiries.

Once again, this list shows that the process is very informal (and probably sometimes illogical). It is also illustrates the huge role of visual reasoning in all stages of the design process.

Another argument for visual reasoning is that **more than one medium** provides information for reasoning that includes text and pictures, but formal logic is limited by sentences [Shin & Lemon, 2003]. This argument is used for supporting both pure visual reasoning and for **heterogeneous reasoning** that combines text, pictures, and potentially any other medium [Barwise & Etchemendy, 1995].

The next argument for visual and heterogeneous reasoning is related to the speed and complexity of reasoning. Reasoning with diagrams and without **re-expressing the diagrams** in the form of a sentence can be simpler and faster. It avoids an unnecessary and non-trivial information conversion process, by working directly with **heterogeneous rules of inference**, e.g., First Order Logic and Euler/Venn reasoning [Swoboda & Allwein, 2002; Swoboda & Barwise, 2002].

2. ICONIC REASONING

One of the founders of the modern formal logic Charles Peirce (1839-1914) argued for use of visual inference structures and processes a long time

ago. Recently it has become increasingly clear that one of the fundamental difficulties of automatic computer reasoning is that it is extremely hard to incorporate the human **observational and iconic** part of the reasoning process into the computer programs. Indeed, the extreme opinion is that it is simply impossible [Tiercelin, 1995].

Peirce stated that in order for *symbols* to convey any *information, indices and icons* must accompany them [Peirce, 1976; Hartshorne, Weiss & Burks, 1958; Robin, 1967; Tiercelin, 1995]. It is probably not accidental that in addition to being a logician and a mathematician, Peirce was also a land surveyor at the American Geodesic Coast Survey. In this capacity, he would have first hand experience with real world visual and spatial geographic reasoning. We should note here that several chapters in this book deal with visual and spatial reasoning and problem solving related to combining geographic maps, aerial and satellite photos. Charles Peirce distinguished the **iconic, indexical and symbolic functions** of signs. Table 1, based on [Tiercelin, 1995], summarizes Peirce's view of icons and diagrams.

Table 1. Peirce's concepts of icons and diagrams

Icons	**Icon:** an object that may be purely fiction, but must be *logically possible* **Main icon function:** *exemplification or exhibition* of an object (its characteristics) **Secondary icon function:** *resemblance* to the object By direct observation of an icon, other truths concerning its object can be discovered. The ideal iconic experimentation warrants an accord between the model and the original. *Icons are formal* and not merely *empirical* images of the things. Icons represent the *formal* side of things.
Diagrams	species of icons

According to Peirce, in general, mathematical reasoning and deduction involve appeal to the observation of **"iconic" representations** and it *cannot be reduced to purely symbolic* (i.e. rule-governed) transformations [Peirce, 1933; Ransdell, 1998]. In modern studies, the term "diagrammatic" principally replaced the Peirce's term "iconic". Peirce stated that *all thinking is in signs,* and that signs can be *icons, indices, or symbols* [Thagard & Cameron, 1997; Goudge, 1950; Hartshorne & Weiss, 1958]. We believe that Peirce's original term "iconic" has an important flavor that has faded in the modern, more "scientific" term "diagrammatic". We discuss iconic representations further in Chapters 9 and 10.

Diagrammatic reasoning is an active area of research with a variety of subjects being considered [Shin & Lemon, 2003; Hegarty, Meyer & Narayanan, 2002; Chandrasekaran, Josephson, Banerjee, Kurup & Winkler, 2002; Chandrasekaran, 2002; Chandrasekaran, 1997; Anderson, Meyer & Ovier, 2002; Anderson, 1999; Magnani, Nersessian & Pizzi, 2002; Glasgow,

Chandrasekaran & Narayanan, 1995; Chandrasekaran & Narayanan, 1990; Barwise & Etchemendy, 1999; Barwise & Etchemendy, 1998; Barwise & Etchemendy, 1995; Swoboda & Allwein, 2002; D'Hanis, 2002; Shimojima, 2002; Magnani, 1999].

Applications of diagrammatic reasoning range from teaching introductory logic classes [Barwise & Etchemendy, 1994] to motions of vehicles and individuals engaged in a military exercise [Chandrasekaran, et al., 2002].

There is no full consensus on the definition of a diagram. Shin and Lemon [Shin & Lemon, 2003] consider any picture to be a diagram and distinguishing them as either an *internal mental representation* or an *external representation.* According to cognitive science, reasoning as performed by humans involves both types of diagrams.

It has also been shown now that diagrammatic systems can have the same *logical status* as traditional linear proof calculi [Shin & Lemon, 2003; Barwise & Etchemendy, 1995]. This is an important result, because traditionally diagrams are considered a **heuristic tool** for discovering, explaining, and exploring a proof, but *not as a part of a proof* [Greaves, 2002; Shin & Lemon, 2003].

Diagrammatic reasoning is part of a wider field of study known as **heterogeneous reasoning** [Barwise & Etchemendy, 1995] (discussed in section 4 and Chapter 4) that itself is a part of even more general study known as **visual explanatory (abductive) reasoning.**

The history of mathematics has seen several successful diagrammatic systems: Euler's circles, Venn diagrams, Lewis Carroll's squares, and Charles Peirce's existential graphs [Euler, 1768; Venn, 1881; Carroll, 1896; Peirce, 1933; Shin & Lemon, 2003]. Some of these systems have the considerable expressive power of a first order logic language that stimulates their use. In the next two sections, we discuss the Euler and Venn systems.

Here it is appropriate to contrast the dimensionality of sentential and diagrammatic examples. We note that sentential languages based on acoustic signals are sequential in nature, whereas diagrams, being inherently two-dimensional, are able to display some relationships without the intervention of a complex syntax [Shin & Lemon, 2003; Stenning & Lemon 2001]. This is in general true for symbolic logic languages, but for some natural languages especially hieroglyphic languages, a sequential one-dimensional nature does not always hold. Individual hieroglyphs and their composition can be two-dimensional. Chapter 7 (Sections 2, 3) provides such examples.

Table 2, based on [Shin, Lemon, 2003; Stenning, Lemon 2001; Barwise, Shimojiima, 1995; Thagard, Cameron, 1997], summarizes strengths and weaknesses of diagrams.

Table 2. Strength and weakness of diagrammatic reasoning

Strength	Weakness
More flexibility in comparison with sequential one-dimensional languages (natural languages, symbolic logic) because of ability to use **intuitive** two-dimensional spatial relations to represent relations between objects. Example: Euler Diagram	*Less flexibility* in comparison with 3-D languages. Exploits only 2-dimensional spatial relations. Cannot exploit non-planar 3-D and higher dimensional relations. Example: Non-planar graphs
More compact representation in comparison with sequential languages because it can exploit a less formal grammar. Sequential languages need to encode two-dimensional spatial relations in a one-dimensional language that required a more complex and elaborated syntax.	*Often less compact* representation in comparison with sequential languages in representations of *abstract objects and relationships* Example: Euler Circles (see section 2.2 in this chapter).
Faster reasoning ("free ride" [Barwise, Shimojiima, 1995]) about two-dimensional spatial relations in comparison with sequential languages. Humans can read off 2-D spatial relations directly from 2-D sentences (diagrams) without reasoning. Example: A map vs. a verbal description of a landscape.	*More effort* is needed to extract relations between objects from 2-D sentence (diagram) than from an ordinary one-dimensional sentence. Many 2-D sentences (diagrams) are incomplete and *complexity* of inference with the diagrams is NP-hard.
Faster discovering causal relations in comparison with sequential verbal rule-based representation. The later may require a longer search.	*Can discover irrelevant (non-causal) relations* if these relations are visualized and read.
Often better understood than verbal, sentential explanations because icons better resemble what they represent.	Can be *less understood* if irrelevant relations are visualized and read.

3. DIAGRAMMATIC REASONING

3.1 Euler Diagrams

One of the most productive mathematicians Leonhard Euler (1707 - 1783) invented what we call now **Euler diagrams** [Euler, 1768]. Below we present Euler's own examples (see Figure 2) adopted from [Shin & Lemon, 2003].

Example 1. All *A* are *B*. All *C* are *A*. Therefore, all *C* are *B*.

This example can be rewritten as follows:

IF All *A* are *B* *and* All *C* are *A* THEN all *C* are *B*.

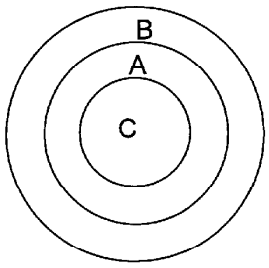


Figure 2. Euler Diagram

More exactly this is equivalent to two formal sentences. The first sentence is in the set-theoretical form and the second one is in the predicate form:

$$\forall x\ [(A\ \exists x \Rightarrow B\ \exists x)\ \&\ (C\ \exists x \Rightarrow A\ \exists x)\ ] \Rightarrow (C\ \exists x \Rightarrow B\ \exists x) \quad (1)$$

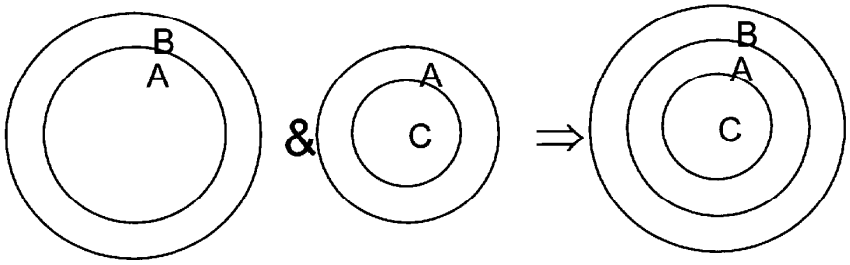
$$\forall x\ [(A\ (x) \Rightarrow B\ (x))\ \&\ (C(x) \Rightarrow A(x)\ ] \Rightarrow (C(x) \Rightarrow B(x)\ ) \quad (2)$$

Statements (1) and (2) can be formally proved in classical first order logic. The proof does not require a mathematician to be involved since it can be carried out by an automatic theorem prover, a computer program.

The Euler diagram above does not provide a formal proof itself, but it makes it obvious for a human that the statement is true. Human visual inspection of this diagram goes through the following steps:

1) match the statement "All *A* are *B*" with two circles *A* and *B* and confirm that *A* is nested in *B* (this can also be done by a computer program that has a diagram as input in vector file format),
2) match the statement "All *C* are *B*" with two circles *C* and *B* and confirm that *C* is nested in *B*,
3) test visually that circle *C* is nested in circle *A*.

These steps are not shown in the diagram and should be learned. Thus, diagram itself does not provide a proof, but make it easier to conceptualize. We "animate" the proof process to make the recording of these steps explicit and clear (see Figure 3.).



IF (All *A* are in *B*) & (All *C* are in *A*) THEN (All *C* are in *B*)

Figure 3. Reasoning with Euler diagrams

The power of this representation lies in the following facts [Shin & Lemon, 2003]:

- Object membership is easily conceptualized by the object lying inside a set.
- Set relationships are represented by the same relationships among the circles.
- Every object x in the domain is assigned a unique location in the region R.
- Conventions above are sufficient to establish the meanings of these circle diagrams.

Now let us consider another original Euler example that involves an **existential statement:** "Some A is B" presented in [Shin & Lemon, 2003].

Example 2. No A is B. Some C is A. Therefore, some C is not B.

Euler's solution is shown in Figure 4.

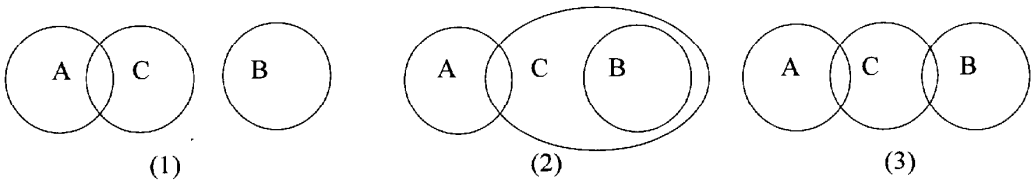


Figure 4. Euler solution

The idea of this solution comes from the fact that C may have several alternative relationships with A and B. Figure 4 shows three such cases:

1. C overlaps with A and without overlapping with B,
2. C contains B and overlaps with A,
3. C overlaps with both A and B.

There are many other possible relationships of C with A and B. To identify them we can construct a 6-dimensional Boolean vector $(x_1, x_2,\ldots,x_6)$, where

x_1=1 if C overlaps with A, but neither contains A nor is contained in A

x_2=1 if C overlaps with B, but neither contains B nor is contained in B

x_3=1 if C contains A,

x_4=1 if C contains B,

x_5=1 if C is contained in A,

x_6=1 if C is contained in B.

Each x_i is equal to 0 if the respective condition is not true. Potentially we have 2^6=64 combinations, not just the three cases Euler listed. Some of them are not possible or do not satisfy premises of example 2, but obviously more than three cases are actually satisfy the premises.

Before analyzing some of these other cases, it is instructive to consider Euler's probable view of the "some" quantifier. It appears that Euler interpreted "some" not as a modern existential quantifier $\exists$ that assumes that if something is true for all it is also true for some: $\forall x\, P(x) \Rightarrow \exists x\, P(x)$. It seems

that Euler's some is "*only for some*", that is if $P(x)$ is true only for some x then another y exists that $P(y)$ is false: Some x $P(x) \Rightarrow \exists y \neg P(x)$.

Now let us turn back to those cases not drawn by Euler. For instance, the case <000011> where both A and B contain C is not possible, as that would contradict the premise, "No A is B". Another case not drawn in Euler's proof is obtained when C contains both A and B, <001100>. Under our usual understanding of "some" this should be drawn since if A and B do not overlap this case satisfies both premises "No A is B" and "Some C is A" because all A is C. Yet under Euler's probable interpretation of some, this case would not be drawn, as there is no x in A that is not also in C. Now we want to check if Euler's solution is complete assuming the interpretation of "some" given above. Shin and Lemon [Shin & Lemon, 2003] do not accept Euler's solution.

> However it is far from being visually clear how the first two cases lead a user to reading off this proposition, since a user might read off **"No *C* is *B*"** from case 1 and "**All *B* is *C*"** from case 2. The third diagram could be read off as "**Some *B* is *A*,**" "Some A is not B," and "Some B is not A" as well as "Some A is B.

We disagree with this argument. Shin and Lemon accepted the first Euler example shown in Figure 2 as delivering clear statements about relations between sets. We can add more nested circles to Figure 2 and it will not be clear what nested relation the user may want to read off. Recall, we added a guide to the user, a reasoning diagram displayed with Euler diagrams (Figure 3). For example 2, perhaps the guide should at least be a verbal explanation that Euler wants to show only those of 64 potential situations ("possible worlds" in more modern terminology) where the statement is true.

From our viewpoint, Euler provided unique and *constructive clarity* in example 2. He actually generated all possible cases given his use of the "some" quantifier. This constructive algorithmic approach is very sound from modern computer science viewpoint.

The need to develop another representation is coming from the fact that when n, the number of predicates A, B, C grows the number of diagrams for a single statement may grow exponentially with n and the compactness of visual representation will be lost. This actually happened with Venn diagrams.

3.2 Venn Diagrams

Venn diagrams were invented in 19th century [Venn, 1880, 1881] and are widely used. Below we discuss examples provided in [Shin & Lemon, 2003]

Example 3. Display "Nothing is *A*." Figure 5 displays this statement as a Venn diagram.

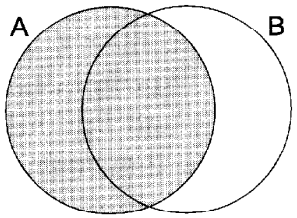
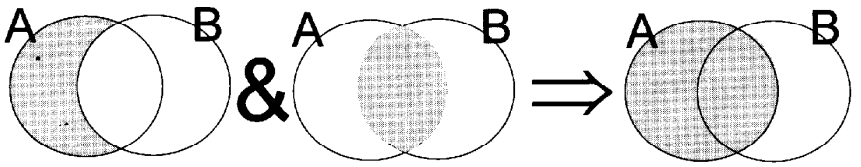


Figure 5. Venn diagram for empty set A.

To represent "Nothing is *A*" the diagram conveys two statements, "All *A* are *B*" and "No *A* is *B*". These statements are not contradictory only if *A* is nothing, empty. The first statement is represented by the part of the diagram shown in Figure 6 on the left. The second statement is shown in Figure 6 on the middle and their combination is shown on right.



All *A* are *B* & No *A* is *B* ⇒ Nothing is *A*

Figure 6. Reasoning with Venn diagrams

At first glance, there is no intuitive meaning in representing "All *A* are *B*" and "No *A* is *B*" with diagrams shown in figure 6. At least these diagrams do not match to meaning of Euler diagram. Venn added shading to the legend to represent the empty part of the diagram. In the left diagram, the shaded part of *A* indicates that this part is empty. Using Euler approach we can read that this part of *A* does not belong to *B*. Combine this with the fact that it is empty and we conclude that "All *A* are *B*". Similarly, in the middle diagram, we note that the shaded area consists of the common elements of *A* and *B*. Because the area is shaded, it is empty, that is, there is no *A* that is also *B*, "No *A* is *B*". Overlaying one diagram over another diagram, we obtain the resulting diagram shown in Figure 6. It is important to notice that knowing the Venn legend, we can read the right-hand diagram directly, – every part of A is empty, noticing that all parts of *A* are shaded.

Figure 6 has two important properties it shows the inference of the result and an intuitive graphical way to prove the statement:

All *A* are *B* & No *A* is *B* ⇒ Nothing is *A*

Venn diagrams use a "primary diagram" legend that shows a general possible disposition of two sets overlapping. This is a departure from the original Euler idea that permitted combining several Euler diagrams into one. In Euler form, this example is presented in Figure 7.

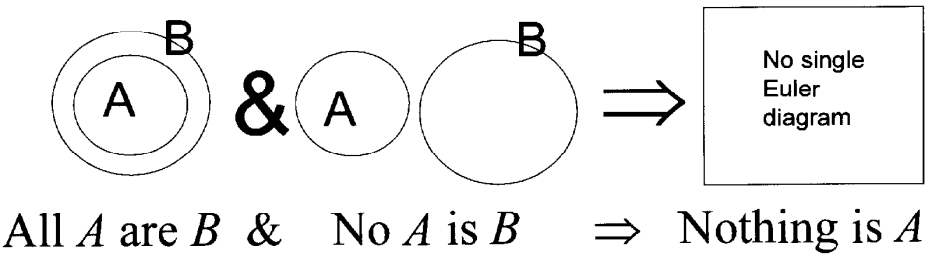


Figure 7. Euler diagram for "Nothing is *A*"

3.3 Peirce and Shin diagrams

Example 4. Figure 8 represents the statement, "All *A* are *B* or some *A* is *B*", which neither Euler's nor Venn's system can represent in a single diagram.

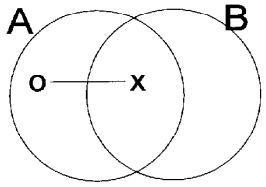


Figure 8. A Peirce diagram "All *A* are *B* or some *A* is *B*"

Figure 8 uses Peirce's legend [Peirce, 1933], where

- 'o' represents emptiness,
- 'x' represents "some" (existential quantifier ∃), and
- a linear symbol '-' connecting the 'o' and 'x' symbols represents disjunctive information.

In this visual language, Figure 8 indicates that part of *A* that does not belong to *B* is empty as it uses the empty symbol 'o', i.e., "All *A* are *B*."

Example 5. Figure 9 represents the proposition "*Either* all *A* are *B* and some *A* is *B*, *or* no *A* is *B* and some *B* is not *A*."

Most of the people including Peirce himself agree that this diagram is too complex in comparison with very intuitive Euler diagram. An alternative visualization of example 5 is shown in Figure 10

[Shin, 2003; Shin & Lemon, 2003]. This visualization uses the following legend:

- Venn's shadings are used to designate emptiness,
- Peirce's 'x' is used for existential properties, and
- Peirce's connecting line between x's is used for disjunctive information.

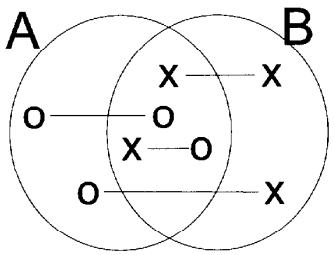


Figure 9. Peirce diagram

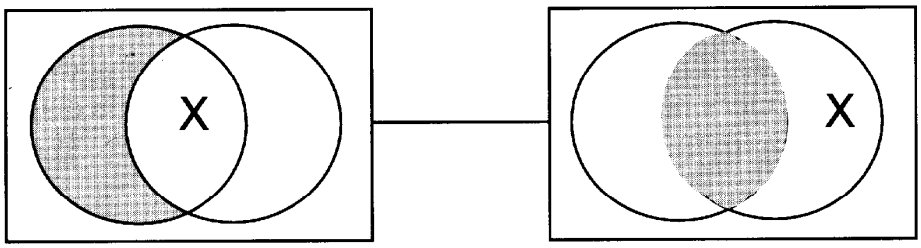


Figure 10. Shin diagram [Shin & Lemon, 2003]

Shin and Lemon [Shin & Lemon, 2003] state that this Shin diagram demonstrates increased expressive power without suffering the loss of visual clarity that happened in Peirce's diagram. While this is true, it seems that the Shin diagram is also limited in scalability. Let us assume that we have more that two predicates *A* and *B*, say, four or five predicates or sets with similar relations between them.

The number of Shin diagrams can grow exponentially. Consider will happen with for five sets *A, B, C, D, and E*. We may need ten pairs of diagrams of the type shown in Figure 10. Use of additional graphical elements such as color and texture can make scalability of the problem less severe.

Regardless of this limitation, Shin diagrams have several important properties that make them equivalent to rigorous systems expressed in formal logic. This formal system is sound and complete in the same sense that some symbolic logic is complete [Shin & Lemon, 2003].

Table 3 provides a summary of diagrammatic systems for representing relations between sets.

Table 3. Characteristics of diagrammatic systems

System	*Characteristics*
Euler diagrams [1768]	Clear intuitive homomorphic relation between circles and sets. Forces the display of more relations than needed or known
Venn diagrams [1880]	"Primary diagram" concept and shading for encoding emptiness to represent partial knowledge about relations between sets
Peirce diagrams [1933]	Symbols for existential and disjunctive information with loss of visual clarity because of the introduction of more arbitrary conventions
Shin diagrams [2003]	Two or more Peirce diagrams for restoring Euler and Venn visual clarity
Hammer and Shin [1998]	Restored Euler's homomorphic relation between circles and sets, by adopting Venn's primary diagrams, but without existential statements

3.4 Graph-based diagrammatic reasoning

Explanatory reasoning can be naturally described in terms of two graphs G_1 and G_2, and a rule R: $G_1 \Rightarrow G_2$ that represents an explanation. Here graph G_2 visually conveys something that should be explained and graph G_1 conveys something that explains G_2, for short G_1 explains G_2. Graph G_1 can be viewed as a hypothesis if it is not known that G_1 is actually true [Thagard & Cameron, 1997].

The *explanation rule* R can be accompanied with a transformation procedure that shows how to get G_2 from G_1. The visual proof of Pythagorean Theorem is an obvious example of such explanatory visual reasoning where G_2 is a diagram of the Theorem statement ($a^2+b^2=c^2$) and G_1 is another diagram with a known relations between its components. The proof of the Pythagorean Theorem is a visual transformation of G_1 to G_2.

Figure 4 in Chapter 1 shows this transformation in six steps, where there are four graphs G_{11}, G_{12}, G_{13}, and G_{14} between G_1 and G_2. In more formal terms, the rules of transformation R_{11}:$G_1 \Rightarrow G_{11}$, R_{12}:$G_{11} \Rightarrow G_{12}$ R_{13}:$G_{12} \Rightarrow G_{13}$, and R_{14}:$G_{13} \Rightarrow G_2$ form the graph reasoning grammar [Thagard & Cameron, 1997]. The major difference between graph transformations and standard verbal transformation rules is that graph transformations have a natural visual representation.

Full, general explanatory reasoning involves visual transformations along with verbal reasoning as proof the Pythagorean Theorem shows. A future multimodal theory of explanatory reasoning may include smell, touch, and emotion as [Thagard & Cameron, 1997] suggest in noting that physicians sometimes diagnose patients using odor. A **multimodal reasoning theory** is also called **heterogeneous reasoning** and is discussed in the next section.

4. HETEROGENEOUS REASONING

Jon Barwise late director of the Visual Inference Laboratory at Indiana University continued visual approach taken by Charles Peirce for inference structures and processes. Barwise with his colleagues developed a visual notation and the Hyperproof reasoning system that uses diagrams to make collections of logical expressions more intuitive [Barwise & Etchemendy, 1994; Allwein & Barwise, 1996]. A further development of this heterogeneous reasoning approach is presented in Chapter 4. The description of the Hyperproof system below is adapted from [Barwise & Etchemendy, 1994]. It is a **heterogeneous reasoning system**, that uses two types of an initial (given) information: (i) a *diagram* depicting a block world (called the situation), and (ii) *sentences* in first-order logic. Typically, sentences describe the *goal* and a visual mean (diagram) provides a situation description. The system supports 27 different types of goals including various "diagrammatic" goals and sentence proving goals. Some of the goals are:

- to proof a particular sentence using the given information,
- to show that the sentence can not be proven from the given information, and
- to determine characteristics of the blocks in the diagram such as the size or shape of a particular block using the diagram and given sentences.

For instance, the *goal* sentence can be to prove that block *c* is the same shape as *d*, *SameShape(c, d),* or to determine the size, shape or location of the highlighted block. The Hyperproof system uses a diagram legend described in Table 4. See also Figure 11.

Table 4. Diagram legend

Icon (syntactic element)	*Meaning (semantics)*	*Kripke 3-valued logic (true, false, unknown) predicate sentence or term*
Barrel	A block of unknown size	Size(block)=U, where U is a term "unknown"
Orange triangle	Indicator of tetrahedron	Tetrahedron(block)=T (true)
Question mark	Indicator of a block of unknown size *and* shape	Size(block)=U & Shape(block)=U
Block on checkers board cell (i,j)	Indicator of location of the block on the checkers board	Board(block,i,j)=T
Block outside checkers board	Block with unknown location on the checkers board	∀ i,j Board(block,i,j)=F
Roman letters	Names of blocks	a,b,c,d,…,z

During the reasoning process, the system may ask the user to determine the size of some block, say block *e* (see Figure 11). Barwise and Etche-

mendy noticed that constructing *proofs of antecedently specified sentences* such as A⇒B is rarely performed in everyday life. More commonly, it must determine whether the *goal can be satisfied* with the information at hand. They point out that (1) Sherlock Holmes *is not told* that the butler is the murder, and asked to prove this assertion, and (2) Holmes' goal is to find out who did it and possibly recognize that more evidence must be gathered.

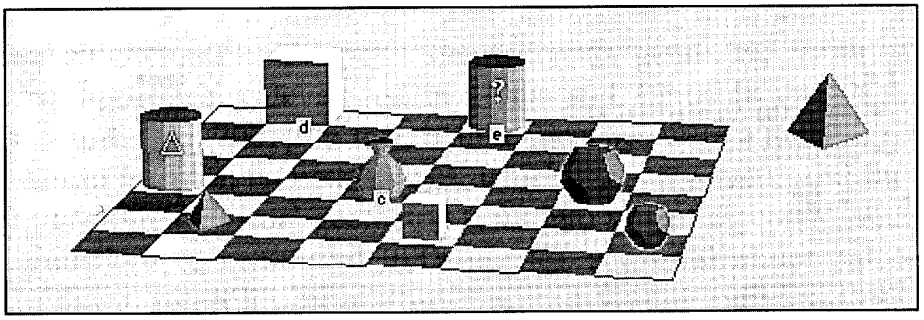


Figure 11. Hyperproof (with permission from Barwise & Etchemendy, http://www-csli.stanford.edu/hp/Hproof2.html). See also color plates.

Table 5 and Figure 12 provide an example of a problem on which Hyperproof work is based [http://www-csli.stanford.edu/hp/Hproof2.html].

The visual solution provided by Hyperproof follows this intuitive logic, where humans seamlessly combine visual information from the diagram with sentences. Barwise and Etchemendy have shown that the solution of this problem completely converts to non-visual, first-order logic sentences requiring *several hundred steps*. Here, based on sentences and the diagram, a user immediately recognizes that all blocks on the left of the dodecahedra in column 6 are irrelevant to the task, which cuts the search significantly.

Table 5. Hyperproof example

Goal Questions	Find out whether the block named "*a*" can be identified from the given visual and textual information and find this block on the diagram if it can be identified.
Given information	(1) Block *b* is a dodecahedron, Dodec(*b*) (2) Block *b* is to the left of block *a* (from our perspective), LeftOf(*b,a*) (3) Block *a* is large. Large (*a*) (4) The tetrahedron that is farther right from b is large, Tetra(*x*) & RightOf(b,x) & FartherOf(b,x) (5) A diagram is shown in Figure 12 (only one block on the right from dodecahedra and this block is a tetrahedron)
Intuitive reasoning	According to (1) block *b* is a dodecahedron. According to the diagram, block *b* must be one of the two dodecahedra in the sixth column. According to (2), block *a* is on the right from block *b*. According to the diagram, there is only one block right to block *b* and this block is father right than one of the dodecahedra. According to (4) this block is large. Thus, block *a* should be a large tetrahedron.

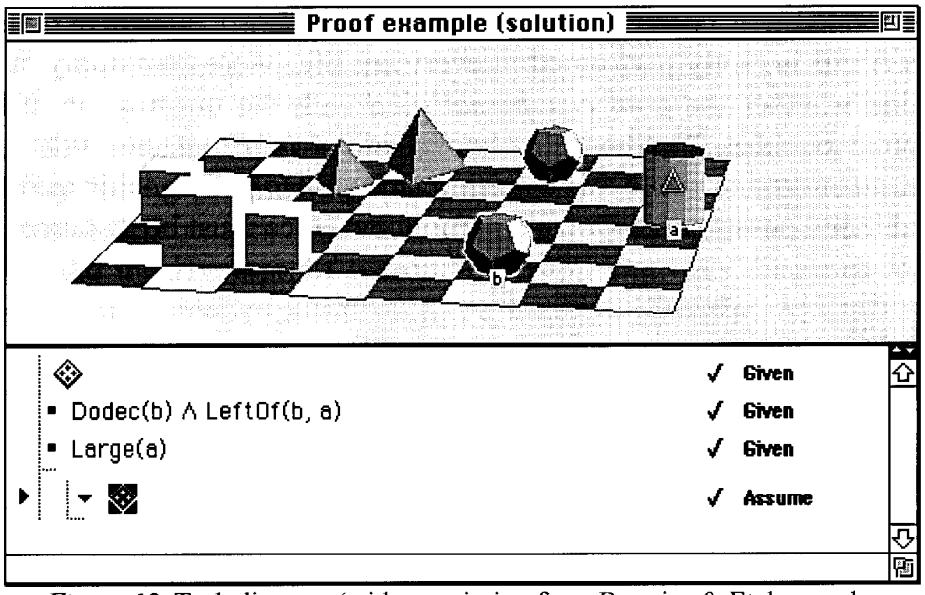


Figure 12. Task diagram (with permission from Barwise & Etchemendy, http://www-csli.stanford.edu/hp/Hproof3a.html). See also color plates.

If a diagram is not given, then several diagrams are generated and the same task is solved with each of them. A diagram is tested if it is compatible with sentences and a block *a* can be identified using given sentences and the assumed diagram.

Barwise and Etchemendy [Barwise & Etchemendy, 1995] claim that it is not necessary to create an Interlingua to be able to reason with heterogeneous information. We feel that some clarification is needed here. The diagram in the Hyperpoof is defined formally using the same predicates that used in sentences. There is a one-to-one mapping of the visual legend of the diagram with predicates to terms in sentences and a formal description of the diagram.

Thus, in some sense, the Interlingua is not needed here because of the way both representations were designed. If the diagram were described in, say, the traditional terms of computer graphics, e.g., as OpenGL code with concepts such as lines, rectangles, and textures or as a single raster bitmap image, an Interlingua would surely be needed. This problem is well known in scene analysis, robotics and geospatial imagery analysis where information is not only **heterogeneous** but also obtained from **disparate sources** that have not been coordinated in advance.

5. GEOMETRIC REASONING

Geometric reasoning has a long and inspiring history traced back to the Greeks. One might hope from the term that modern geometric reasoning

may continue the intuitively clear visual geometric line of reasoning inherited from Greeks. In fact, as our short review will show, it is not yet the case.

In 1950s, one of the first and seminal efforts of Artificial Intelligence (AI) research was to simulate human geometric reasoning in a computer program [Gelenter, 1959]. "This research activity soon stagnated because the classical AI approaches of **rule based inference** and heuristic search failed to produce impressive geometric, reasoning ability" [Kapur & Mundy, 1989].

The next attempts of computer simulation for geometric reasoning were **algebraic approaches** developed in 1980s and 1990s [Kapur & Mundy, 1989; Chou & Gao, 2001]. In fact, both approaches were a modern return to the Cartesian tradition in mathematics, which was very successful for centuries by transforming visual geometric tasks into sets of algebraic, vector and matrix equations or non-visual If-Then rules. As we shall see below, both rule-based and algebraic approaches departed from intuitively clear visual geometric proofs.

Below we describe the frameworks of both approaches using work adapted from [Chou & Gao, 2001]. In *algebraic approach* geometric statements are converted to a set (conjunction) of equations

$$h_1(y_1, y_2, \ldots, y_m)=0$$
$$h_2(y_1, y_2, \ldots, y_m)=0$$
$$\ldots$$
$$h_r(y_1, y_2, \ldots, y_m)=0$$
$$c(y_1, y_2, \ldots, y_m)=0.$$

It is usually assumed that coefficients in equations are rational numbers. Thus, the algebraic form of the geometry equation would be

$$\forall y\ [h_1(y)=0\ \&\ldots\&\ h_r(y)=0) \Rightarrow c(y)=0].$$

Chou, Gao, and Zhang [Chou, Gao & Zhang, 1996] also demonstrated that a revived AI *rule-based approach* was able to provide valuable results -- short proofs of tasks where an algebraic solution of polynomial equations was long. A geometric rule or axiom used in their Geometry EXpert (GEX) system has the following form:

$$\forall x\ [P_1(x)\ \&\ldots\&\ P_k(x)) \Rightarrow Q(y)],$$

where x is a point occurring in the geometry predicates P_i, Q. The following is one of the rules used in GEX (a diagram is presented in Figure 13):

$AB \parallel CD$ if and only if Angle(AB, PQ) = Angle(CD,PQ)

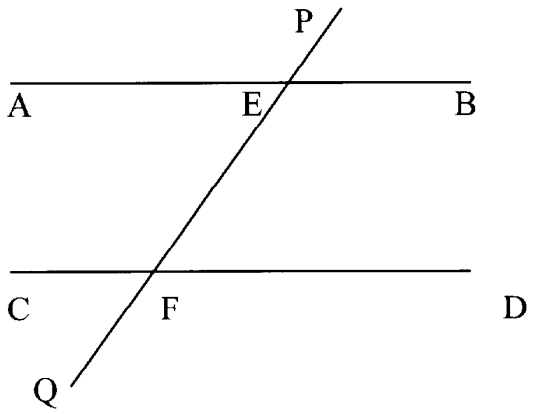


Figure 13. Diagram for non-visual rule R1

The GEX system implements several methods. According to [Chou, Gao & Zhang, 1996] one of the methods (Wu's method, based on polynomial equations with the characteristic set) has been used to prove more than 600 geometric theorems. Another method (based on high-level geometric lemmas about geometric invariants) produced short, elegant, and human-readable proofs for more than 500 geometry theorems. Other methods use the Groebner basis method for polynomial equations, the calculation of vectors, complex numbers, and full-angles. The full-angle method is a rule-based method that produced very short proofs in cases where all other methods fail because very large polynomials arise during in the proving process.

6. EXPLANATORY VS. DEDUCTIVE REASONING

The visual reasoning models we considered above focused on deductive reasoning. A typical **deductive reasoning model** distinguishes data from hypotheses, explains data by mapping hypotheses to data, and does not explain hypotheses. This is a common situation in many machine learning and data mining situations. Thagard and Cameron [Thagard & Cameron, 1997] claim that such models are **not adequate for pictorial discovery**. They argue that

1. explanatory (abductive) reasoning models should be used instead and models should include **explanatory hypotheses** that are explicitly formed and evaluated, and
2. Explanatory reasoning should not be equated with a relatively simple formal logical deductive reasoning models, such as models presented in [Bylander, Allemang, Tanner & Josephson, 1991; Konolige, 1991].

These arguments are supported by the example of visual explanatory reasoning from archeology that we describe in section 7 below.

The term "**abductive**" for explanatory reasoning was coined by Charles Peirce a hundred years ago along with the term "iconic" reasoning [Hartshorne & Weiss, 1958; Thagard, 1988].

Explanatory reasoning is informally described as defining a set of characteristics concerning the explanatory hypothesis. Indeed, this serves as the major characteristic of explanatory reasoning. More comprehensively, explanatory reasoning should deal with:

- a hierarchy of the hypotheses where some hypotheses explain others (layered hypotheses);
- a set of hypotheses is not given in advance and need to be discovered or constructed (creative hypotheses);
- hypotheses may contradict to each other and it is necessary to establish a domain theory (revolutionary hypotheses);
- Hypotheses may not explain the all data (incomplete hypotheses);
- Some hypotheses are not verbal, but visual iconic hypotheses;
- Constraints for building hypotheses are known only partially and methods for testing constraints satisfaction may not be obvious.

For more detail, we refer to [Thagard & Cameron, 1997]. The motivation for introducing many of characteristics listed above is obvious. There is also an argument that deductive reasoning is not necessarily explanatory.

> For example, we can deduce the height of a flagpole from information about its shadow along with trigonometry and laws of optics. However, it seems odd to say that the length of a flagpole's shadow explains the flagpole's height... Some additional notion of causal relevance is crucial to many kinds of explanation, and there is little hope of capturing this notion using logic alone [Thagard & Cameron, 1997].

The hope is that **visually layered reasoning** will be especially useful in capturing, justifying (and possibly refuting) **causal relations**.

7. APPLICATION DOMAINS

Applications of diagrammatic, explanatory reasoning to architectural design are explored in [Barwise & Etchemendy, 1998; Barker-Plummer & Etchemendy, 2003] and in Chapter 4. The role of visual problem solving in graphic design in discussed in [Lieberman, 1995]. Design reasoning involves multiple representations of information, complex rationale, and goal structures. Design reasoning naturally fits the diagrammatic approach. A new application domain for diagrammatic reasoning has emerged recently in the area of automated theorem proving [Jamnik, 2001; Shin & Lemon, 2003].

Another application of the diagrammatic approach is the analysis of locations and motions of vehicles and individuals engaged in a military exercise [Chandrasekaran et al., 2002]. The goal of this work is use the diagrammatic reasoning to infer maneuvers, plans and intentions of the two sides with intention of improving the efficiency of decision making. Use of diagrammatic reasoning for this task intends to

- decrease *overload* for a decision maker,
- automatically generate *hypotheses* about emerging *threats* and *deviations* of a side's behaviors from the expected behaviors,
- *summarize* a vast amount of detail in diagrams,
- provide *insights* about diagrammatic constructs that are the most important, and
- provide insights about diagrammatic constructs that are supportive of specific types of *inferences*.

A variety of questions is of interest is this application, such as:

Is there a movement of an opposing force toward the left flank of the current force F1?

The answer should be inferred from the map with iconic representation of the sides, left and right flanks of both friends and enemies.

Our next example in the use of diagrammatic reasoning is presented in [Pisan, 2003] for reasoning about supply and demand of cassette tapes The input information is presented as a plot of two linear functions *Supply(quantity) = price* and *Demand(quantity) = price*, which is sketched in Figure 14. The visual reasoning system called SKETCHY is able to interpret the plot and answer for the questions such as:

At what point is Supply equal to Demand?

What is the price for the Supply line when the quantity is 350?

If a user changes the plot, the system adjusts its answers.

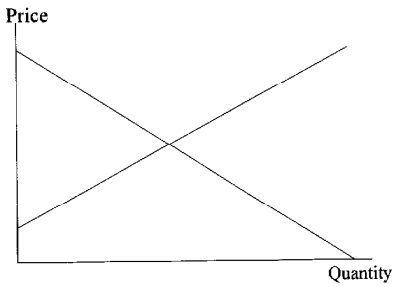


Figure 14. Plot used for automatic visual reasoning

Another example of visual explanatory reasoning has been provided in [Thagard & Cameron, 1997] from the area of archeology. The visual fact to be explained consists of two unusual notches in the skullcap of an australopithecine. The placement, depth, and direction of notches can be expressed

as formal sentences with spatial predicates, but they are more naturally given visually.

Two hypotheses have been generated in this example:

Hypothesis 1: The notches had been inflicted by two separate blows from a weapon wielded by another *hominid.*

Hypothesis 2: The notches had been created by a *leopard,* which had taken the australopithecine's head in its mouth.

These hypotheses were accompanied with several explanations, another hypotheses, scenarios, and facts. These are presented in Table 6.

Table 6. Hypotheses, explanations, facts, and scenario

	Hypothesis 1
Explanation 1	Notches had been made at *divergent* angles from the centerline (that is clearly visible on the picture without any formalization).
Domain theory, scenario 1	Human evolution had been driven by murder and cannibalism ("killer ape" hypothesis).
	Hypothesis 2
Explanation 2.1	The lower canine teeth of leopards diverge and are about the right distance apart.
Explanation 2.2	A fossil leopard jaw from the same site fits the notches fairly well.
Supporting fact 2.1:	The entrance to the cave was a vertical shaft when the australopithecine remains were deposited and those remains are mostly comprised of skull fragments.
Supporting fact 2.2	Leopards visit similar shaft caves in the area today and use the trees, which grow around the entrances as places to consume prey out of the reach of hyenas.
Supporting fact 2.3	Leopards tend to destroy the skeletal material of their primate prey with the exception of the skulls.
Explanation scenario	The skulls have fallen from the trees and into the cave shafts

Formal, non-visual reasoning would require conversion of visual input such as human visual memory of site observation, picture, and video into non-visual sentences. This is complex work and one of the major drawbacks and reasons for failure in development of knowledge bases in many domains. In addition, this approach may require conversion back from non-visual sentences to task's original visual form since that is natural for domain experts. Finally, we may have a complex and unnecessary double conversion: visual → non-visual → visual, trying to follow a non-visual approach (see Chapter 1 (Section 1.5) for more detail on double conversations).

8. HUMAN AND MODEL-BASED VISUAL REASONING AND REPRESENTATIONS

8.1 Spatial reasoning vs. visual reasoning

There is a widespread belief that human visual reasoning with material that is easy to visualize *speeds up problem solving* more than with material that is hard to visualize. Knauff and Johnson-Laird [Knauff & Johnson-Laird, 2000] stated that the literature does not provide consistent evidence for such claim. To explore this issue four types of verbal relations were identified in [Knauff & Johnson-Laird, 2000; Knauff, Fangmeier, Ruff & Johnson-Laird, 2003]:

(1) **spatio-visual relations** that are easy to envisage both spatially and visually (e.g., above-below)
(2) **"control" relations** that are hard to envisage either spatially or visually (e.g., *better-worse*)
(3) **spatial relations** that are hard to envisage visually but easy to envisage spatially, and
(4) **visual relations** that are hard to envisage spatially but easy to envisage visually (e.g., *cleaner-dirtier*).

The following deductive reasoning task from [Knauff & Johnson-Laird, 2000] illustrates reasoning with visual but not spatial relations *cleaner-dirtier*:

> The dog is cleaner than the cat.
> The ape is dirtier than the cat.
> Does it follow: The dog is cleaner than the ape?

In experiments, these authors measured time for solving similar tasks with different types of relations. The speed of reasoning (solving these problems) was in accordance with the order of the relations listed above, where (1) was the fastest. Reasoning with relations of type (1) took 2200 ms, with relations of type (2) took 2384 ms and with relations of type (4) took 2654 ms. Note, the speed difference between (1) and (2) was not statistically significant. Also the fact that reasoning with (4) was slower than with (2) might not be expected in advance. Knauff and Johnson-Laird [Knauff & Johnson-Laird, 2000] provided the following explanation of this result.

> A relation that has a natural spatial model should speed up the process of reasoning. In contrast, a visual relation, such as *dirtier*, may elicit irrelevant visual detail. One imagines, say, a cat caked with mud, but such a representation is irrelevant to the transitive inference. It takes *additional*

time to replace this vivid image with one in which dirtiness is represented in degrees. In other words, the visual relations, which are hard to envisage spatially, lead to a mental picture.

Another experiment conducted in [Knauff, Fangmeier, Ruff & Johnson-Laird, 2003] supported the explanation about the addition time:

All relations elicit mental models that underlie reasoning, but visual relations in addition elicit visual images.

This experiment used functional magnetic resonance imaging to identify types of brain activities during work with relations (1)-(4), that were presented acoustically via headphones (without any visual input).

These provided experiments are important for understanding limits of efficient visual reasoning and problem solving.

8.2 Cognitive operations

Human image-processing abilities are critical in everyday problem solving and have been very efficient in important scientific discoveries [Shepard & Cooper, 1982]. Biologically inspired models for reasoning with images can be potentially very efficient. In this section, we review the current cognitive theory on this subject as a potential base for models of visual and spatial reasoning and decision making.

Table 7 below outlines a cognitive theory of human image processing based on [Kosslyn, 1999], it includes four operations: image generation, inspection, maintenance, and transformation.

Table 7. Human image processing [Kosslyn, 1999]

Image generation	One **generates an image** of an object by looking up a "visual code" in associative memory, which in turn activates a visual memory in the ventral sys-tem. A spatial pattern of activation is imposed on topographically organized regions in the occipital lobe. If a detailed image is required, one now accesses associative memory and looks up the most distinctive part or property as well as its location. One then shifts attention to the appropriate location.
Inspection of imaged patterns	One **inspects the imaged pattern** by shifting the attention window over it to encode previously unconsidered properties (e.g., the shape of an animal's ears). Imaged patterns are recognized by matching the input to stored visual memories. Spatial relations are encoded using the dorsal system. Once one has formed an image, one can "see" parts that are embedded in the shape, and were not noticed explicitly when the object was first encoded.
Image maintenance	One can maintain it <image> by re-activating the visual memory representations in the ventral system; these representations eventually degrade due to adaptation, hence an image cannot be retained indefinitely.
Image transformation	One can transform an imaged pattern by shifting the pattern in spatially organized regions of the occipital lobe.

Thus, according to this theory human image processing starts from input visual data "as is" without identifying specific properties and parts. Then the process of detailed image generation starts, where first most distinctive parts and properties are identified before identifying other properties. This identification is done by matching the input to stored visual memories.

8.3 Visual representation and reasoning models

Visual reasoning and problem solving depend heavily on the visual representation used. Below we describe some visual representation models found in computer science and cognitive science summarized in Table 8.

Table 8. Visual representation models

Array-based visual semantic representation	**Location** of the item **in the array** is matched approximately to its location in 2-D image or 3-D scene. An item can be represented as a subarray if it has its own internal subitems. Location and sizes of the items in the array permit to represent relations such as *left-of* and *north-of*. Limitation of the model: arrays do not represent directly a relation when a single top triangle is over both of the lower triangles.
CaMeRa model	Pictorial information consists of a bitmap and associated **node-link** structures that provides semantic metadata.
Semantic network with scene graphs	A node within **semantic network** can represent a complete concept. A node in a semantic network can be linked to a scene graph. A node within a **scene graph** could be a part of a hierarchy representing a single visual object.
Knowledge-based model	The stored scene knowledge includes: • the **observed features and relationships** such as part-whole relationships, constraints among the subparts (algebraic constrains, "restriction graphs"), and relationships over time; • **expected objects** (presented as models, parameterized object models, "object graphs", "observation graphs", slots, filler frames); • **prediction tools** ("prediction graphs") to predict expected objects using observed features and relationships; • **"interpretation graphs"** to eliminate inconsistencies in match of observed features and relationships to the models using more reasoning; • hypotheses (primarily top-down) to drive prediction and interpretation.
Probabilistic model	Knowledge-based model refined with **probabilistic information**
2D iconic model	2D iconic representations built from **different views** of the **3D model** to simplify matching. 2D strings for iconic indexing built as pairs of one-dimensional strings that represent the symbolic projections of the objects on the x and y axis.
Deformable object model	Deformable objects built using **statistical** rather than geometric relationships. Such representations can be **learnt from examples**.

This table is based on [Croft & Thagard, 2002; Baxton & Neumann, 1996; Glasgow & Papadias, 1992; Tabachnik-Schijf, Leonardo & Simon, 1997]. These models include array-based models, node-link based models, semantic networks with scene graphs, knowledge-based models, probabilistic models, 2D iconic models and deformable object model.

Iconic image representations are considered **biologically plausible** [Buxton, Neumann, 1996]. The term icon is used in a variety of meanings. One of them was discussed in section 2 above that followed Peirce's approach. Nakayama [1990] and Rao and Ballard [1995] use the term iconic to describe small visual templates, which constitute visual memory.

Specifically in [Rao, Ballard, 1995] a set of icons is just a vector of *numeric parameters* associated with a pixel or patch, extracted from the image using some local filters of different size of localities, and used to identify rotation. Parameters are called icons because they have visual equivalent and they are small like icons because they cover small patches.

These icons can be called low-level icons. They show just line direction or pixel distribution. High-level icons represent real world concepts such as a house or bridge. Both icon types can make image representation shorter if the image can be described only by patches with complex patterns. In addition, iconic representation is biologically plausible, that is mental images possibly are generalized real images in the form that resembles icons [Buxton, Neumann, 1995; Rao, Ballard, 1995].

Rao and Ballard based their iconic model on biological evidence [Field, 1994; Kanerva, 1988] about the primate visual system. Specifically, this system takes advantage of the redundancy in the visual environment by producing a sparsely distributed coding that aims to minimize the number of simultaneously active cells.

According to [Kanerva, 1988], the memory operates on features and creates internal objects by chunking together things that are similar in terms of those features and relatively invariant to the changes in the environment.

Rao and Ballard [Rao & Ballard, 1995] hypothesize that visual memories could consist of iconic representations stored in a *distributed* manner which can be activated by an incoming visual signal or other iconic representation. In the context of this hypothesis, visual perception is an activation of memory.

Thus, relatively invariant *iconic feature vectors* may be viewed as an effective medium for vision-related memory storage. The Bruegel visual correlation system presented in Chapter 10 is icon based and derives benefits from such properties of human perception.

9. CONCLUSION

Reasoning plays a critical role in decision making and problem solving. This chapter provided a comparative analysis of visual and verbal (sentential) reasoning approaches and their combination called heterogeneous reasoning. It is augmented with the description of application domains in visual reasoning. Specifics of iconic, diagrammatic, heterogeneous, graph-based, and geometric reasoning approached have been described. Next, explanatory (abductive) and deductive reasoning are identified and their relationships to visual reasoning are explored. The chapter also presents a summary of human and model-based reasoning with images and text. Issues considered include cognitive operations, differences between human visual and spatial reasoning, and image representation.

The experience of generations of scientists such as Bohr, Einstein, Faraday, and Watt have shown that visual representations and reasoning can greatly improve the ability for finding and testing explanatory hypotheses. The hope is that systematic visual and heterogeneous reasoning will serve the same role although this remains to be seen. We believe that the fundamental iconic reasoning approach proclaimed by Charles Peirce is the most comprehensive heterogeneous reasoning approach and as such it need to be further developed.

We share the vision expressed by M. Greaves [2002] as to why structured graphics as part of heterogeneous reasoning has been largely excluded from contemporary formal theories of axiomatic systems. He concluded that there are no other reasons than historical and philosophical heritage stretching from the Greeks to the early twentieth-century work of David Hilbert.

10. EXERCISES AND PROBLEMS

Simple

1. Construct an Euler diagram and a reasoning diagram similar to that shown in Figures 2 and 3 for the statement "No A is B. All C are B. Therefore, no C is A." Comment on the visual clarity of your result.

2. Adapted from [Lemon & Pratt, 1997]. Using Euler Circles, try to represent the following premises:

$$A \cap B \cap C \neq \varnothing$$
$$B \cap C \cap D \neq \varnothing$$
$$C \cap D \cap A \neq \varnothing$$

Show that this is impossible using diagram shown in Figure 15, where $A \cap B \cap C \cap D \neq \varnothing$.

Try to modify diagram in Figure 15 to satisfy $A \cap B \cap C \cap D = \varnothing$. Does it continue to be a set of Euler Circles?

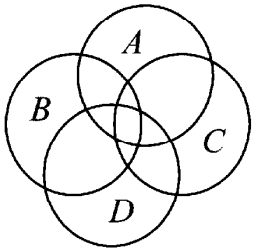


Figure 15: An Euler's Circles representation exhibiting Helly's Theorem

Advanced

3. Adapted from [H. Simon, 1995]. Assume that somebody wrote: "I notice a balance beam, with a weight hanging from a two-foot arm. The other arm is one foot long." Then somebody asked the question: "How much force must I apply to the short to balance the weight?"

   What kind of reasoning would be appropriate for this situation? Is it verbal reasoning? If so, what are its axioms and rules of inference, and where do they come from? Are the axioms logical, or do they embody laws of physics? What kind of heterogeneous reasoning might you suggest?

11. REFERENCES

Allwein, G., Barwise, J., Eds. Logical Reasoning with Diagrams. Oxford Univ. Press, Oxford, 1996.

Anderson, M., Meyer, B., Ovier, P, *Diagrammatic Representation and Reasoning*, Springer, 2002.

Anderson, M., Diagrammatic Reasoning Bibliography, 1999 http://zeus.cs.hartford.edu/~anderson/biblio.html

Barker-Plummer, D., Etchemendy, J., Applications of heterogeneous reasoning in design Machine Graphics & Vision International Journal, Vol.12, Issue 1, January 2003, 39-54.

[0]Barwise, J., Etchemendy, J., A Computational Architecture for Heterogeneous Reasoning, in Theoretical Aspects of Rationality and Knowledge, I. Gilboa, ed., Morgan Kaufmann, 1998, 1-27. http://www-csli.stanford.edu/hp/TARKpaper.pdf

Barwise, J., Etchemendy, J., Language, Proof, and Logic, Stanford: CSLI Press, 1999.

Barwise, J. Etchemendy, J., Computers, Visualization, and the Nature of Reasoning, with, in The Digital Phoenix: How Computers are Changing Philosophy. T. W. Bynum and James

H. Moor, eds. London: Blackwell, 1998, pp. 93-116. http://www-csli.stanford.edu/hp/CVandNR.pdf

Barwise, J., Etchemendy, J., Heterogeneous Logic, In Diagrammatic Reasoning: Cognitive and Computational Perspectives, J. Glasgow, N. H. Narayanan, and B. Chandrasekaran, eds., Cambridge, Mass: The MIT Press and AAAI Press, 1995, pp. 211-234.

Barwise, J., Etchemendy, J., Hyperproof, Stanford University, Lecture Notes; New York: Cambridge University Press, 1994. http://www-csli.stanford.edu/hp/#Hyperproof

Barwise, J., Shimojima, A., 1995, Surrogate Reasoning, Cognitive Studies: Bulletin of Japanese Cognitive Science Society, 1995, 4(2): pp. 7-27.

Buxton, H., Neumann, B., Visual Interpretation and Understanding, ECVNet, 1996, http://www-prima.inrialpes.fr/ECVNet/Policy/Interpretation/node7.html Accessed Dec 1, 2003.

Bylander, T., Allemang, D., Tanner, M., and Josephson. J., The computational complexity of abduction. Artificial Intelligence, 49, 1991, pp. 25-60.

Carroll, L., Symbolic Logic. New York: Dover, 1896.

Chandrasekaran, B., Diagrammatic Representation and Reasoning: Some Distinctions, an invited paper presented at the AAAI Fall 97 Symposium Series, Diagrammatic Reasoning, Boston, MA. 1997, http://www.cis.ohio-state.edu/~chandra/dr-II.pdf

Chandrasekaran, B., What Does It Mean for a Computer to Do Diagrammatic Reasoning? A Functional Characterization of Diagrammatic Reasoning and Its Implications, In Diagrammatic Representation and Inference, Second International Conference, Diagrams 2002, Eds: M. Hegarty, B. Meyer, N. Hari Narayanan, 2002, pp.1-3. Springer, LNCS Volume 2317. http://www.cis.ohio-state.edu/~chandra/What-Does-It-Mean-for-a-Computer-to-do-diagrammatic-reasoning.pdf

Chandrasekaran, B., Josephson, J., Banerjee, B.,Kurup U., and Winkler, R., Diagrammatic reasoning in support of situation understanding and planning, Proceedings of Army Science Conference, Dec 2002, Orlando, FL. http://www.cis.ohio-state.edu/~chandra/DR-paper-ASC.pdf

Chandrasekaran B., and Narayanan, N., Towards a theory of commonsense visual reasoning. in Foundations of Software technology and Theoretical Computer Science, Nori, K.V. and Madhavan, C.E.V. (eds.), Springer, 1990.

Chou, S., Gao, X., and Zhang, J., An Introduction to Geometry Expert, Proc. CADE-13, LNAI 1104, Eds. M. A. McRobbie and J. K. Slaney, 1996, Springer, p. 235-239.

Chou, S., Gao, X., Automated reasoning in geometry, in Handbook of automated reasoning, A. Robinson, A. Voronkov, Eds., Elsevier, 2001, pp.3-47.

D'Hanis, I., A Logical Approach to the Analysis of Metaphors, In L. Magnani, N. Nersessian, C. Pizzi (Eds.), Logical and Computational Aspects of Model-Based Reasoning, Kluwer, 2002.

Glasgow, J., Narayanan, N., and Chandrasekaran, B., (Eds.), Diagrammatic Reasoning, Cognitive and Computational Perspectives, MIT Press, Boston, 1995.

Glasgow, J., Papadias, D., Computational imagery. *Cognitive Science*, 17930:355-394, 1992

Croft, D., Thagard P. Dynamic imagery: A computational model of motion and visual analogy. In L. Magnani and N. Nersessian (Eds.), Model-based reasoning: Science, technology, values. New York: Kluwer/Plenum, 2002, pp. 59-274. http://cogsci.uwaterloo.ca/Articles/Pages/diva.pdf

Einstein, A., Infeld, L., The Evolution of Physics: From Early Concepts to Relativity and Quanta, Simon and Schuster, New York, 1938, 1966.

Euler, L., Lettres à une Princesse d'Allemagne. St. Petersburg; l'Academie Imperiale des Sciences, 1768

Field, D., What is the goal of sensory coding? Neural Computation, 6, 1994, pp. 559-601.

Gelernter, H., Realization of a geometry-theorem proving machine. Proc. International Conf. On Information Processing (ICIP). UNESCO House, Paris, pp. 273-282. 1959, (Reprinted in: Feigenbaum E.A., Feldman J. (Eds.): Computers and Thought. McGraw-Hill, New York, 1963, pp. 134-152).

Goudge. T., The thought of C. S. Peirce. University of Toronto Press, Toronto, On., Canada, 1950.

Greaves, M., *The Philosophical Status of Diagrams*. Stanford: CSLI Publications, 2002

Hadamard, J., The Psychology of Invention in the Mathematical Field, Dover, 1954.

Hammer, E., Shin, S., Euler's Visual Logic, History and Philosophy of Logic, 19, 1998, pp. 1-29.

Hartshorne, C., Weiss, P., Burks, Eds. Collected papers of Charles Sanders Peirce. Harvard University Press, Cambridge, Mass., U.S.A., 1958.

Hegarty, M., Meyer, B., Narayanan, N., Diagrammatic Representation and Inference: Second International Conference, Diagrams 2002, Proceedings, LNCS Volume 2317, Springer, 2002.

Hepting, D., A new paradigm for exploration in computer-aided visualization, Dissertation, Simon Fraser University, 1999, http://fas.sfu.ca/pub/cs/theses/1999/DarylHeptingPhD.pdf

Jamnik, M., Mathematical Reasoning with Diagrams. Stanford: CSLI Publications, 2001.

Kanerva, P., Sparse Distributed Memory, Cambridge, MA, Bradford Books, 1988.

Kulpa. Z., Diagrammatic representation and reasoning. Machine Graphics & Vision 3(1/2), 1994, pp. 77-103. http://www.ippt.gov.pl/~zkulpa/diagrams/Diagres.pap.ps.z

Kapur D., Mundy J., (eds), Geometric Reasoning, MIT Press, 1989

Knauff, M., Johnson-Laird, P., Visual and spatial representations in spatial reasoning. In Proceedings of the Twenty-second Annual Conference of the Cognitive Science Society, Mahwah, NJ: Lawrence Erlbaum Associates, 2000, pp. 759-765. http://www.cognition.iig.uni-freiburg.de/research/publications/Knauff-CogSci2000.pdf

Knauff ; M., Fangmeier, T., Ruff , C., Johnson-Laird, P., Reasoning, Models, and Images: Behavioral Measures and Cortical Activity, Journal of Cognitive Neuroscience, Vol.15 N. 4, 2003, pp. 559 – 573.

Konolige, K., Abduction versus closure in causal theories. Artificial Intelligence, 52, 1991, pp. 255-72.

Kosslyn, S., Visual Mental Images as Re-Presentations of the World: A Cognitive Neuroscience Approach, In Visual and Spatial Reasoning in Design, Eds. Gero, J., Tversky, B., University of Sydney, Sydney, Australia, 1999, http://www.arch.usyd.edu.au/kcdc/books/VR99/Kosslyn.html

Lemon, O., 2002, Comparing the Efficacy of Visual Languages, in Barker-Plummer et al. (eds), 2002, pp. 47-69.

Lieberman, H., The Visual Language of Experts in Graphic Design, IEEE Symposium on Visual Languages, Darmstadt, Germany, September 1995. http://web.media.mit.edu/~lieber/Lieberary/Graphic-Design/Graphic-Design.html

Magnani, L., Abduction and Geometrical Analysis. Notes on Charles S. Peirce and Edgar Allan Poe, In Magnani, L., Nersessian, N., and Thagard, P. eds., Model-based reasoning in scientific discovery, Kluwer, 1999.

Nakayama, K. The iconic bottleneck and the tenuous link between early visual processing and perception. In C. Blakemore, Editor, Vision: coding and efficiency, pp. 411-422, NY, Cambridge University Press, 1990

Pisan, Y., Visual Reasoning with Graphs, http://citeseer.nj.nec.com/167073.html, accessed Nov 15, 2003

Rao, R., Ballard, D., An active vision architecture based on iconic representations. *Artificial Intelligence*, 78:461--506, 1995

Peirce, C., Collected Papers. Cambridge, MA: Harvard University Press, 1933
Peirce, C., New Elements of Mathematics. C. Eisele, ed., Mouton, The Hague, 1976.
Pineda, L., Diagrammatic Inference and Graphical Proof, In L. Magnani, N. Nersessian, C. Pizzi (Eds.), Logical and Computational Aspects of Model-Based Reasoning, Kluwer, 2002.
Shepard, R., Cooper, L., Mental Images and their Transformations, Cambridge, MA, MIT Press, 1982.
Ransdell, J., The Visual Inference Laboratory, 1998 http://members.door.net/arisbe/menu/links/vilab.htm
Robin, R., The Annotated Catalogue of the Papers of CS Peirce, Amherst, Mass., 1967 (unpublished manuscripts collected in Harvard University Library).
Shin, S., Lemon, O., Diagrams, Stanford Encyclopedia of Philosophy, 2003, http://plato.stanford.edu/entries/diagrams/
Shin, S., The Iconic Logic of Peirce's Graphs. Cambridge: MIT Press, 2003.
Simon H., Foreword. In: J. Glasgow, N. Narayanan and B. Chandrasekaran (Eds.), Diagrammatic Reasoning, Cognitive and Computational Perspectives, MIT Press, Boston, 1995.
Swoboda, N., Allwein, G., Modeling Heterogeneous Systems, In Diagrammatic Representation and Inference: Second International Conference, Diagrams 2002, Proceedings, Eds: M. Hegarty, B. Meyer, N. Hari Narayanan (Eds.), LNCS Volume 2317 / 2002, Springer, 2002, pp. 131-145.
Swoboda, N., Barwise, J. Implementing Euler Venn Reasoning Systems, In Diagrammatic Representation and Reasoning, Anderson, M., Meyer, B., Ovier, P., (Eds.), Springer, 2002,
Swoboda, N., Allwein, G., A Case Study of the Design and Implementation of Heterogeneous Reasoning Systems, In L. Magnani, N. Nersessian, C. Pizzi (Eds.), Logical and Computational Aspects of Model-Based Reasoning, Kluwer, 2002.
Shimojima, A., A Logical Analysis of Graphical Consistency Proofs, In L. Magnani, N. Nersessian, C. Pizzi (Eds.), Logical and Computational Aspects of Model-Based Reasoning, Kluwer, 2002.
Stenning, K., Lemon, O., 2001, Aligning Logical and Psychological Perspectives on Diagrammatic Reasoning, Artificial Intelligence Review, 15(1-2): 29-62. (Reprinted in Thinking with Diagrams, Kluwer, 2001.)
Stenning, K., 2002, Seeing Reason: image and language in learning to think, Oxford: Oxford University Press.
Tabachneck-Schuf, H. J. M., Leonardo, A. M., & Simon, H. A. (1997). CaMeRa: A computational model of multiple representations. Cognitive Science, 21,
Thagard, P., Computational philosophy of science. MIT Press, Cambridge, Mass., 1988.
Tiercelin, C., The Relevance of Peirce's Semiotic for Contemporary Issues, in Cognitive Science, Acta Philosophica Fennica, vol.58, 1995, pp. 37-74. http://jeannicod.ccsd.cnrs.fr/documents/disk0/00/00/01/99/ijn_00000199_00/ijn_00000199_00.htm
Venn, J., On the diagrammatic and mechanical representation of propositions and reasoning, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 9, 1880,. Pp. 1-18.
Venn, J., Symbolic Logic, London: Macmillan, 1881, 2nd ed., 1894.

Chapter 4

REPRESENTING VISUAL DECISION MAKING

A Computational Architecture for Heterogeneous Reasoning

Dave Barker-Plummer and John Etchemendy
CSLI, Stanford University, USA

Abstract: In this chapter, we describe a computational architecture for applications that support heterogeneous reasoning. Heterogeneous reasoning is, in its most general form, reasoning that employs representations drawn from multiple representational forms. Of particular importance, and the principal focus of this architecture, is heterogeneous reasoning that employs one or more forms of graphical representation, perhaps in combination with sentences (of English or another language, whether natural or scientific). Graphical representations include diagrams, pictures, layouts, blueprints, flowcharts, graphs, maps, tables, spreadsheets, animations, video, and 3D models. By "an application that supports heterogeneous reasoning" we mean an application that allows users to construct, record, edit, and replay a process of reasoning using multiple representations so that the structure of the reasoning is maintained and the informational dependencies and justifications of the individual steps of the reasoning can be recorded. Our architecture is based on the model of natural deduction in formal logic. In this chapter we describe and motivate the modifications to the standard logical model necessary to capture a wide range of heterogeneous reasoning tasks. The resulting generalization forms our computational architecture for heterogeneous reasoning (CAHR).

Key words: heterogeneous reasoning, diagrammatic representation, sentential representation, decision support, rationale capture, constrained graphical editing.

1. INTRODUCTION

Until recently, theoretical accounts of reasoning have been limited to homogeneous linguistic reasoning; that is, reasoning in which all information is represented in the form of sentences of some language, either natural or formal. This presents a major obstacle to the application of insights about the structure of reasoning to real-world problem solving, for example the

reasoning involved in the construction of designs, which typically involves graphical representations.

This chapter presents a theoretically informed account of real-world problem solving based on an understanding of the nature and structure of reasoning, or more precisely, rational justification; historically the province of logic. We believe that the overall structure of rational justification described by the theory of natural deduction [Fitch, 1952; Gentzen, 1935; Prawitz, 1965; Prawitz, 1971] is a reasonable model of the gross structure of everyday reasoning. We understand natural deduction in its broadest sense, in which proofs are seen as recursively structured rationales that display the overall structure of reasoning and permit the nesting of proofs within proofs.

We will begin by describing the traditional logical model of deduction and its representation by formal proofs. This model assumes that information is presented as formal sentences in a single language, and that inference proceeds by applying rules based on the syntax of these sentences.

We then illustrate the generalizations necessary to handle heterogeneous deduction. The introduction of diagrammatic representations requires extensions to the traditional model. Most obvious among these are the fact that diagrammatic inference typically proceeds by making incremental modifications to a single representation, while sentential inference proceeds by the introduction of new sentences into a proof. In addition, the reliance on formal syntax to drive inference must be generalized.

Finally, we describe further generalizations to this model, which permit the representation of reasoning not subject to the constraints of formal deduction. This results in an architecture which can in principle be applied to numerous domains in which heterogeneous reasoning is carried out, and in which formal criteria of validity are not available. Our account allows us to describe a large class of real-life reasoning, such as that involved in the design of complex artifacts. In particular, the generalization encompasses three important features that are not considered by traditional models of reasoning: heterogeneity of representation, of rationale and of goals.

- **Heterogeneity of representation.** Most reasoning and design problems require the marshaling, manipulation and communication of information represented in a wide variety of formats ([Glasgow, Narayanan & Chandrasekaran, 1995] presents a collection of influential articles in this area.) For example in addition to circuit diagrams, an electrical engineering team may use state machine diagrams, timing diagrams and logic ladder diagrams, together with algebraic or natural language specifications of the desired input/output behavior, in order to produce a design that meets the client's needs [Harel, 1988; Johnson, Barwise & Allwein, 1993].

- **Heterogeneity of rationale.** The formal model of proof is far too constrained to countenance the kinds of reasoning involved in design and practical problem solving. In particular, the model is too strict to countenance forms of justification based on matters of cost, efficiency, safety, style, aesthetic judgment, probabilistic considerations, and the like, justifications that arise at almost every turn in real-world problem solving, see for example [Mitchell, 1990]. A model of heterogeneous rationales is particularly important for collaborative reasoning, where a consensus must be achieved in spite of competing justifications (say aesthetic versus structural) of divergent decisions.
- **Heterogeneity of goals.** Historically, logic has focused almost exclusively on a single type of reasoning in which the goal is to show that a conclusion is a necessary consequence of some given information. In real-world problem solving, by contrast, the goals of a particular reasoning task can be quite varied, see for example [Barwise & Etchemendy, 1994]. Design problems typically admit of a wide variety of solutions that meet the primary specifications, and selection among competing solutions is made on the basis of subsidiary, comparative criteria. The primary goal is thus to find any solution that satisfies certain requirements, not a unique solution entailed by those requirements, and secondary goals guide the choice among candidate solutions to the primary goal. For example, in the design of a circuit, the primary goal will be a correct circuit, but other criteria distinguish between competing correct circuits, for example, the expense of fabricating the circuit in silicon. Research in logic has historically not addressed reasoning with such complex goal structures.

Our computational architecture for heterogeneous reasoning (CAHR) allows each of these kinds of heterogeneity to be represented within a structured document encoding a piece of reasoning. This chapter is an extended version of [Barker-Plummer & Etchemendy, 2003].

2. SENTENTIAL NATURAL DEDUCTION

We take the traditional model of formal logic known as natural deduction [Fitch, 1952; Gentzen, 1935; Prawitz, 1965; Prawitz, 1971] as our starting point. An example of a proof constructed using this model is presented in Figure 1. The particular presentation of natural deduction that we use is due to the logician Fitch [Fitch, 1952].

In order to maintain an appropriate flow of information, a reasoning application maintains a data structure representing the proof. A proof is more than an unstructured collection of information-bearing representations but rather records a structured reasoning process that represents a solution to a reasoning task. It is important to note that the structure of a proof does not represent the temporal structure of the reasoning, but rather the underlying logical structure of the reasoning.

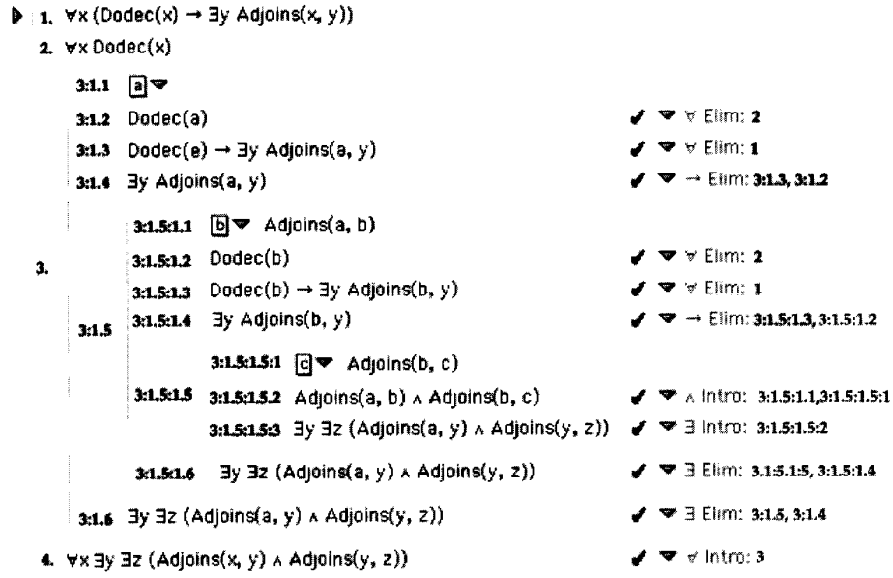


Figure 1. A Sentential natural deduction proof

A *(sentential) proof* is a recursive structure consisting of an ordered sequence of nodes. Each node contains either:

1. a sentence (which expresses information asserted at that node), illustrated in Figure 1 at steps 3:1.2 and 3:1.3, for example.
2. a nonempty set of proofs (called "subproofs" or "cases" of the parent proof). This situation is illustrated in step 3 of Figure 1. This step contains one subproof, consisting of steps 3:1.1--3:1.6.

The initial node of a proof (or subproof) is called the *assumption step*. A small horizontal tick in the left margin serves to separate the assumption step from the remainder of the proof. In our presentation, the main (outer) proof can have several assumption steps, though this is a syntactic convenience.

With the exception of assumption steps, each step in a sentential proof must be justified by the use of in inference rule and citation of support. Together the rule and support provide an unambiguous justification for the deduction of the new sentence from the support steps.

Nodes in a proof may optionally be numbered for ease of reference. One possible numbering scheme that captures the recursive structure of a proof is illustrated in Figure 1.

The steps in each subproof are numbered sequentially, tagged with an identifier for the subproof. A subproof's identifier is that of the step in which it is contained, appended with the sequence number of this subproof's case within that step. The tag 3:1.6, near the bottom of Figure 1 should be read (backward) as naming the sixth step in the first subproof of the third step (of the main proof).

2.1 Inference

The sentences occupying the atomic steps of the example proof are written in a formal language known as first-order logic (FOL). This language has an unambiguous syntax and semantics. The sentence ∀x Dodec(x) in step two for example, means that every object (in the domain) is a dodecahedron.[1]

The use of a justification is illustrated at step 3:1.2. The inference rule used here is known as **∀-Elim** (universal elimination). In this step the support is the sentence appearing in step 2 of the proof. The inference here is from the universal claim that every object is a dodecahedron, to the particular claim that a is a dodecahedron, where a is some object in the domain of the problem.

The fact that the claim at step 3:1.2 may be deduced from the claim at step 2 using the **∀-Elim** inference rule can be checked syntactically. To do this we note that the support is a universally quantified sentence, and that the justified sentences is the matrix (body) of the sentence, with the bound variable x replaced with a name a. This is a correct application of the rule according to the definition of the system.

The number and nature of the available inference rules will depend on the particular logical system. In the case of logical deduction, the key requirement is that each inference rule must be stated in completely unambiguous terms, and it should be clear when the conditions on the correct application of the inference rule have been met.

2.2 Support and the structure of a proof

In order to ensure logical validity, we may only use information from earlier in the proof as support for a later step. The notion of "earlier in the proof" is made precise by the following observations.

[1] Under our conventional interpretation of the predicate Dodec.

A node is *accessible* from a second node, and hence available as a potential support for it, if either a) both nodes are in the same (sub)proof and the first strictly precedes the second, or b) the second node is in a subproof that is contained by a node from which the first node is accessible (either directly or recursively in virtue of this clause). When node A is accessible from node B, we write $A < B$. Note in particular that a node in a subproof is not accessible from any node in its parent proof (or in any of its ancestors), nor from nodes in sibling subproofs, that is, subproofs with the same parent proof.

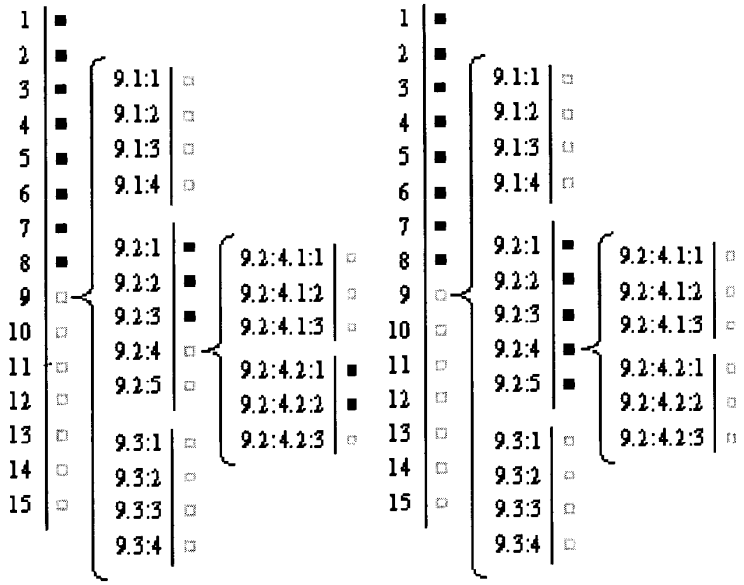


Figure 2. The accessibility relation

On the left of Figure 2, we indicate in black all the nodes accessible from node 9.2:4.2:2, which is shown in gray. On the right of the same figure, we indicate in black all nodes accessible from node 9.2:5. Here, notice that node 9.2:4, which contains a range of cases (subproofs), is accessible from node 9.2:5, but that the nodes within those cases are not accessible from this node.

The accessibility relation determines the inheritance of information in a proof. That is, if $A < B$, then the information expressed by representations contained at node A is also available at node B.

It is important to note that although the nodes of a proof form a partial (not linear) order, when we restrict attention to the nodes accessible from a specified node, these nodes are linearly ordered. This may be emphasized by renumbering the nodes in Figure 2 in the manner shown in Figure 3. Thus, each node in a proof has a unique, linear history which we call the

node's provenance. The information available at a node N of a proof is the sum of the information expressed at nodes in N's provenance.

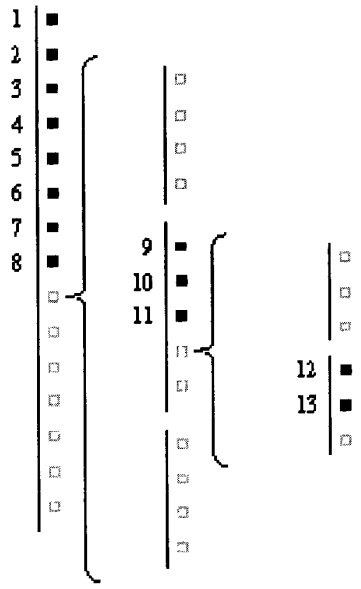


Figure 3. The provenance of a node

The articulation of a proof into nodes, proofs, and subproofs allows us to isolate two senses in which information is present at a given stage of reasoning. On the one hand, each node in a proof typically adds new information potentially relevant to the solution of the reasoning task in question. This additional information may be presented by a new sentence, or by a set of cases attached to the node. These cases are interpreted disjunctively, that is, as representing a range of possible alternatives. We call the information contained in a single step the *incremental information* associated with the node. The justification mechanism allows the user to explain and justify the incremental information introduced at a node. But this incremental information is not the only information available at that stage in the reasoning. The accessibility relation defines the paths of legitimate information inheritance through the proof. At any point in the reasoning, the total information state is defined by the totality of incremental information accumulated along the nodes in the current node's provenance. We call this the *cumulative information* associated with the node.

Because the cumulative information available at a node is represented by the provenance of that node, any step contained in the provenance is available for citation as support for an inference at that node.

3. GENERALIZING TO HETEROGENEOUS DEDUCTION

With the standard model of natural deduction in hand, we turn our attention to the task of formalizing the more general domain of heterogeneous deduction. Figure 4 illustrates a heterogeneous deduction.

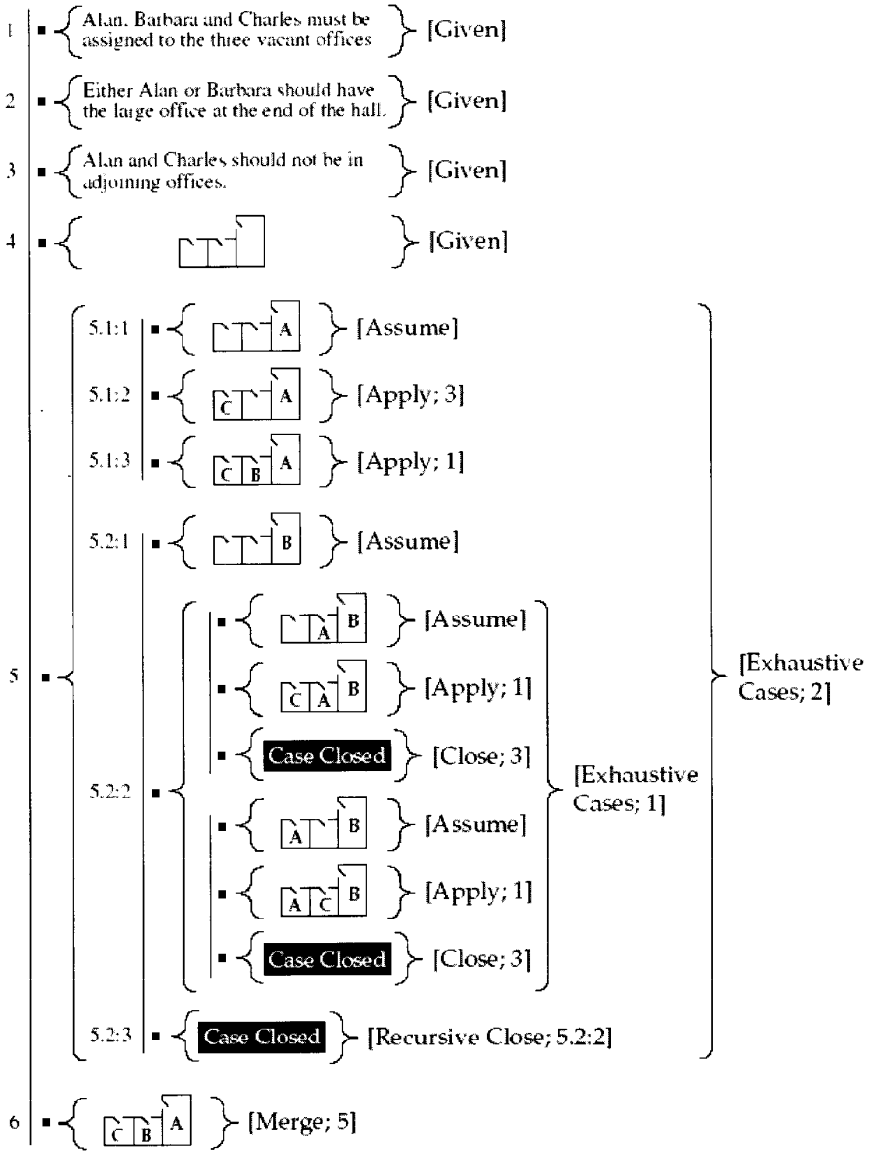


Figure 4. An office assignment problem

The reasoning recorded concerns the assignment of offices, represented diagrammatically in the proof, based on a variety of constraints that are expressed sententially. The reasoning shows that there is only one possible as-

signment that satisfies the stated constraints.[2] In this section, our intent is to consider the generalizations to the model of sentential deduction that are necessary to model the heterogeneous case.

The first thing to notice about Figure 4 is that the structure of the reasoning is very similar to that of a typical natural deduction proof. The reasoning proceeds by splitting into cases represented by subproofs each headed with a new assumption, and then elucidating the consequences of those assumptions within the subproof. At the end of a subproof, information is exported to the main proof according to some inference rule.

The obvious difference between the two cases is that the individual steps of the proof contains diagrams, rather than sentences. In fact, in the example of Figure 4 all of the steps in the proof except the initial assumptions contain diagrams, though in general some steps might contain sentences.

3.1 Graphical representations

The presence of graphical representations in a proof requires special techniques for properly handling information inheritance for graphically displayed information, as well as appropriate editing restrictions that respect the inheritance structure of the proof.

In order to describe these techniques and restrictions, we introduce some terminology. Every graphical representation is introduced into a proof at a specific node, which we call the *node of origin* of the representation. This can but need not be the first node of the proof. A graphical representation G introduced at node A can be modified at any subsequent node in the proof from which A is accessible, subject to certain editing constraints described below. In general, the modification of a graphical representation G at node B does not affect the display of G at earlier nodes. Thus, a single graphical representation will present different displays at different nodes of a proof.

The specific display of a graphical representation at a particular node is called a *graphic*, and the graphic is an instance of the graphical representation. If graphic G_A at node A and graphic G_B at node B are instances of the same graphical representation G, and $A < B$, then G_A is said to be an *ancestor* of G_B, and G_B is said to be a *descendant* of G_A.

Being an instance of the same graphical representation is an equivalence relation among graphics, i.e., it is reflexive, symmetric and transitive. In

[2] The reasoning of Figure 4 is very similar to that implemented in our Hyperproof program [Barwise & Etchemendy, 1994] with the key difference being that the kind of diagram used in this proof (Hyperproof uses diagrams of the placement of objects on a checkerboard.)

particular, notice that two graphics, neither of which is at a node accessible to the other, can still be instances of the same graphical representation. This will happen if (and only if) both graphics result from modifying a common graphic appearing at a node accessible to both. If G_A and G_B are instances of the same graphical representation G but neither A nor B is accessible from the other, then G_A and G_B are said to be *cousins*.

The left of Figure 5 illustrates a simple proof structure containing a single graphical representation. The node of origin of this graphical representation is node 1. All of the graphics in the proof are descendants of G_1, except for G_1 itself. Graphic $G_{2.1:2}$ is also a descendant of graphic $G_{2.1:1}$, and graphic $G_{2.2:2}$ is a descendant of graphic $G_{2.2:1}$. Graphics $G_{2.1:1}$ and $G_{2.1:2}$ are cousins of graphics $G_{2.2:1}$ and $G_{2.2:2}$. Graphic G_3 is a descendant of G_1 and a cousin of the remaining graphics in the proof.

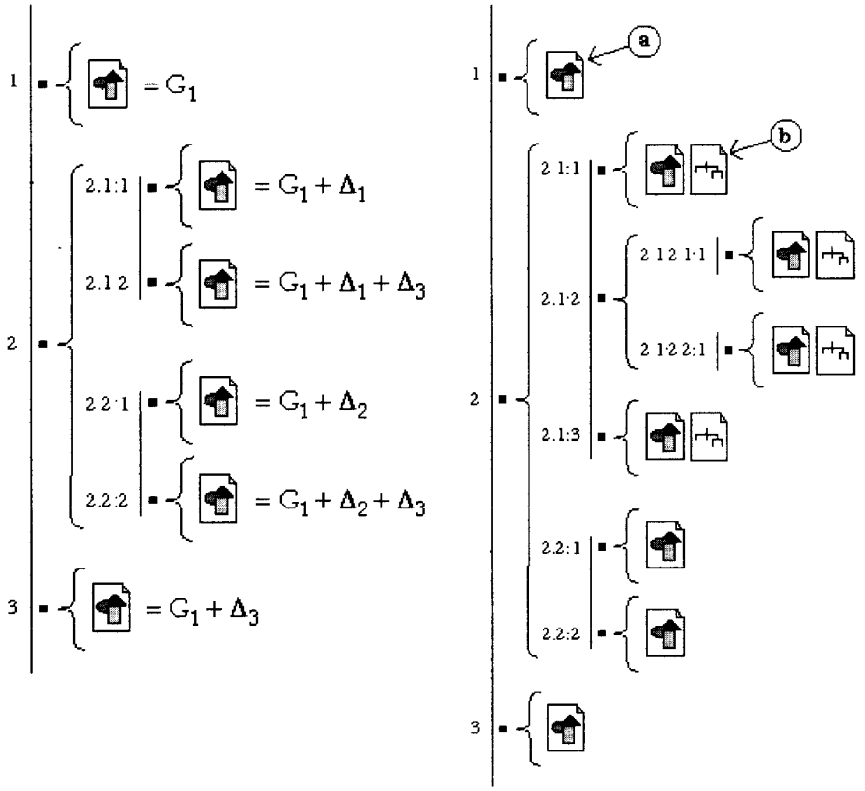


Figure 5. Availability of information in a proof

If a graphical representation G is introduced at node A, then we say that G is *available* at A and at any node from which A is accessible. If A is not accessible from node B, then we say that G is *not available* at B. A representation G can be displayed and edited at a node if and only if it is available at that node. On the right of Figure 5, we illustrate the difference in availability of two graphical representations appearing in a proof. The first

graphical representation, marked *a*, is introduced at node 1, and is consequently available at every node in the proof (since node 1 is accessible from every node in the proof). The second representation, marked *b*, is introduced at node 2.1:1. It is available throughout subproof 2.1 (including the subproofs of this subproof), but not at any other node in the proof. Thus, no instance of this representation appears at nodes 1, 2.2:1, 2.2:2, or 3, and it cannot be modified at these nodes.

3.2 Display and editing of graphical representations

A basic insight underlying our architecture is that sentential and graphical reasoning display similar structural features, features that can be captured by means of the proof structure and accessibility relation defined earlier. Where these characteristic types of reasoning differ is in how incremental and cumulative information must be handled. With sentential reasoning, it is generally possible to express by means of a single sentence the precise incremental information added at any point in the reasoning. This incremental information can thus be isolated from the cumulative (sentential) information inherited at that node, since sentences expressing the inherited information remain at earlier nodes. They are still accessible from the current node, but are isolated from the incremental information by virtue of their location in the proof.

In contrast, when reasoning employs a graphical representation, incremental information in typically expressed by means of modifications of the representation. But these modifications are specified or defined in relation to other information-bearing features of the representation. For example, if we add a mark indicating a location to an existing map of a city, both the graphical modification and the incremental information it conveys presuppose the presence of the existing features of the map. For this reason, it is not possible to graphically isolate the incremental information from the cumulative information inherited from earlier stages in the reasoning. The incremental information, which includes, for example, the distance from the new point to any other location in the city, is distributed throughout the graphic. The CAHR architecture handles these characteristics of graphical representations by means of special techniques governing the display and modification of graphical representations in a proof.

An implementation of CAHR keeps track of where (at which node) modifications to a graphical representation are made in a proof. Generally speaking, the location of a modification has an impact both on where the modification is displayed and on what subsequent modifications can be

made to the representation at later nodes in the proof. The basic intuition is that since a modification at a particular node introduces incremental information, that information is inherited at later nodes. And since inheritance of graphically expressed information is explicit, subsequent modifications to the representation should preserve that information. Thus, the information content of a graphical representation should increase monotonically as we follow any accessibility path through a proof.

This section describes two types of graphical editing that may be permitted in a CAHR-based application enforcing this flow of information: *incremental* and *presentational* editing.

3.3 Incremental editing

Incremental editing is used for modifications that conform to the inheritance structure of the proof, and hence is the appropriate edit type to use when making informationally significant changes to a graphical representation. When an incremental edit is made to a graphical representation G at node N of a proof, the modification is displayed in the graphic G_N and in all descendants of G_N, until the modification is superseded by changes made at subsequent nodes. If a subsequent edit superseding the change made at N is made at node $L > N$, the modification made at N is displayed in the graphic G_N and in any graphic G_M where $L > M > N$. Incremental edits made to G_N have no effect on the ancestors or cousins of G_N.

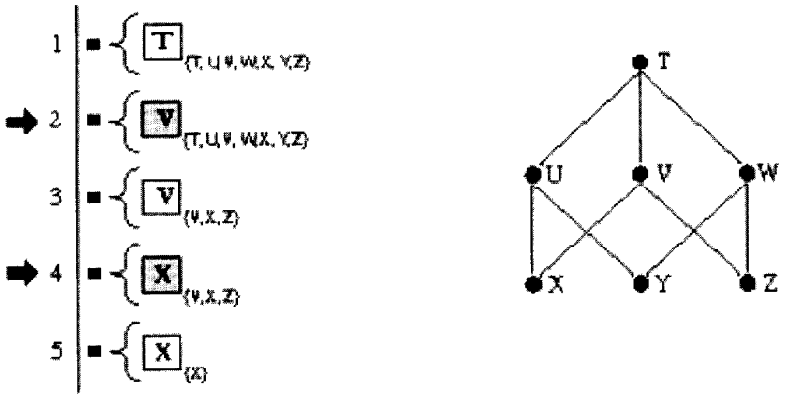
The office assignment example of Figure 4 illustrates the notion of incremental editing. As we move through the proof, the (tentative) assignment of an individual to an office is represented by the placement of the appropriate letter in an appropriate location on the diagram. For example, at step 5.2:1, Barbara is assumed to be assigned the large office. In each of the subproofs contained in step 5.2:2 this information is inherited and represented for the duration of the subproof. Only when the scope of this subproof is closed at step 6 is the information not displayed at that node (because 5.2:2 is not accessible from step 6.)

Note that editing a graphic at a node does not impose additional editing constraints on the graphic at that same node, but rather on descendant graphics. Indeed, anything it is possible to do at a particular graphic, it should be possible to "undo" or take back at that same graphic. It is only when the user moves to subsequent nodes that she commits to the modifications made at the earlier node. Or, to put it another way, at any given node the user is freely allowed to modify the incremental information at that node, but must respect any constraints imposed by the graphically displayed information inherited at that node.

The example has an important feature, namely that each edit is permanent for the duration of its scope. Once an individual has been assigned to an office, this assignment is not subject to further modification. We call such edits *permanent incremental edits*. Permanent incremental editing can be effectively used to enforce the monotonicity requirement discussed above. If a piece of information has been established at a particular node in a proof, then that information is available at any node from which the node in question is accessible. If that information is recorded as a modification to a graphic, then the specific modification must be inherited by all descendant graphics. Permanent incremental editing guarantees that this will be the case, since the modification cannot be subsequently altered at a later (descendant) node. Permanent edits can, however, be retracted or modified at a later time, but the modification must be made at or before the node in which the original edit was made. This imposes a practical restriction that aids users in the successful completion of the reasoning task: A conclusion or decision that was previously made cannot be retracted without navigating to the node associated with that modification and reviewing the original justification. This restriction becomes particularly important if the proof is being constructed by multiple users or if the construction takes place over an extended period of time.

Permanent incremental editing is a species of a more general notion that we call *constrained incremental editing*. A modification to a graphic G_N is a constrained incremental edit if, in addition to modifying G_N, it narrows the range of possible modifications that can be made to descendant graphics. Permanent editing is the most highly constrained form of incremental editing.

Figure 6 illustrates constrained incremental editing. We assume a graphical representation of a single attribute that can be given one of seven possible values.



Constrained incremental edit Information semilattice

Figure 6. Constrained incremental editing

We have assumed that the values have fixed conventional interpretations, and that the relative significance of the values is expressed by the information semilattice shown on the right of the figure. The value *T* is a "null" value, consistent with any possible assignment to the attribute, while the values *X*, *Y*, and *Z* are maximally informative and mutually incompatible. The values *U*, *V*, and *W* are intermediate values: *U* is consistent with either *X* or *Y*; *V* is consistent with *X* or *Z*; and *W* is consistent with *Y* or *Z*. The proof shown on the left depicts incremental edits made at nodes 2 and 4.

The value assigned to the attribute at each node is shown in the square, and the square is annotated with the set of permissible values that can be assigned at that node. At node 2, the value of the attribute is changed from *T* to *V*. This increments the information contained at the node, and hence subsequent changes to the graphical representation are constrained to those that express further refinements of that information. At node 4, the value is changed from *V* to *X*, again incrementing the information provided by the graphic. Since this value is maximally informative, no further modifications are allowed at node 5.

3.4 Post-editing

Since the temporal order in which reasoning occurs rarely conforms to the logical structure of the reasoning, it is necessary for any CAHR based application to permit the editing of the proof structure, as well as any representation, after its initial construction. However, both forms of "post-editing" must respect the display and editing constraints imposed by incremental edits made to the graphical representations contained in the proof.

Since the display of graphics in a proof is determined by the original graphic plus a sequence of incremental "deltas," the deletion of a node in an existing proof can change the graphics displayed at subsequent nodes. Figure 7 illustrates the impact of deleting a node containing modifications to a graphical representation. In this figure, we examine the effect of deleting node 2 from the proof of Figure 6. After the deletion, where the value *V* was assigned, descendant graphics no longer display the effects of this modification. Thus at node 3, the attribute retains the value *T*, until the modification at node 4.

Post-editing a graphic located at a node can alter subsequent graphics in two distinct ways. On the one hand, changing the existing value of an attribute will have the expected effect on how the attribute is displayed in descendant graphics: the new value is displayed until superseded by a different value assigned in a descendant graphic. An example of this sort of post-editing is depicted in the center of Figure 8, where the proof on the left is

modified by changing the value of the attribute at node 2 from *V* to *U*. The new value is then inherited by descendant graphics until it is subsequently changed to *X*.

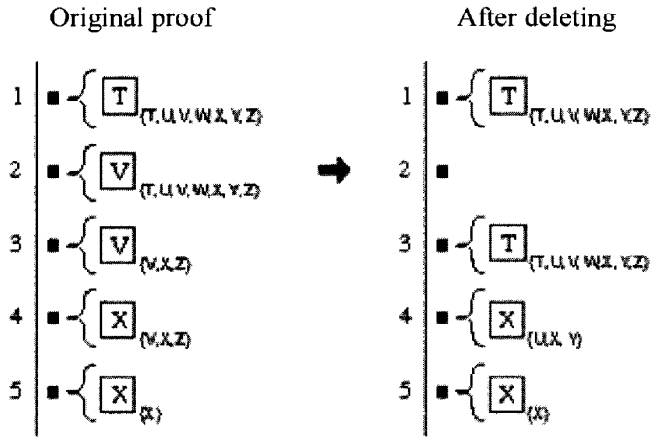


Figure 7. Deleting a node

On the other hand, the modification of an existing value can alter the editing constraints imposed on descendant graphics, and so render an edit impermissible that was formerly permitted. An example is depicted on the right of Figure 8, where the value at node 2 of the proof on the left is changed from *V* to *W*. A consequence of this change is that the range of permissible values at node 4 no longer includes the value *X*. Thus the change at node 2 precludes the assignment of *X* at node 4, and the latter assignment is wiped out. In situations of this sort, when a modification precludes an edit already made at a descendant node, a CAHR-based application will typically warn the user of the effects of the modification.

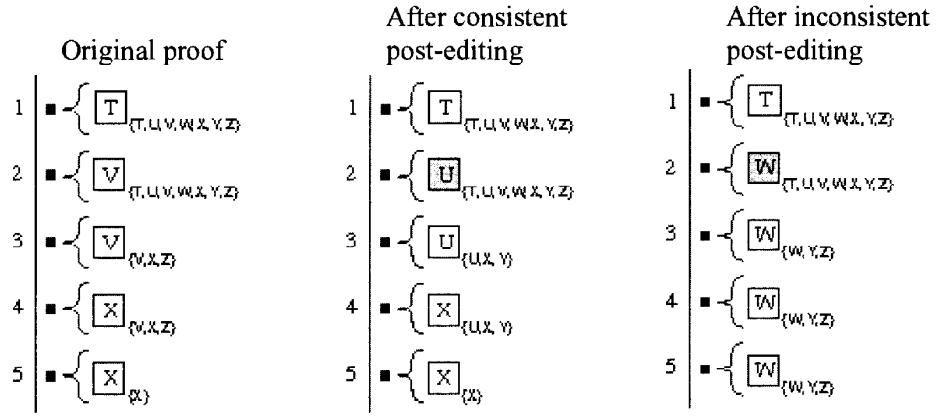


Figure 8. Post-editing

3.5 Heterogeneous inference rules

It is widely assumed that with the introduction of graphical representations into a proof comes a necessary informality in the reasoning that can be carried out. Our Hyperproof application, [Barwise & Etchemendy, 1994], demonstrates that this is not in general the case. Provided that the non-sentential representation has a sufficiently well-defined semantics and that there is some overlap between the expressive power of the graphical and sentential representations participating in a heterogeneous system, it is possible to implement a completely formal heterogeneous deduction system.

Consider the office assignment example of Figure 4. In this figure the sentential information has been expressed in the English language, for ease of presentation. We trust that the semantics of these sentences is sufficiently clear that the possibility of formalizing these sentences and the inferences contained within is apparent.

An example of a heterogeneous deduction rule is the Apply rule used to justify several steps in the proof illustrated in Figure 4. In these examples, information expressed in sentential form is "applied" to the graphical representation; that is, information is used to justify specific modifications of that representation. For example, at step 5.1:2 we apply the sentence at step 3 to the diagram at step 5.1:1. There is a unique way to update the diagram at step 5.1:1 consistent with the information at step 3, and so provided the semantics of the two representations were explicitly elucidated this inference could be formalized and automatically validated.

Another heterogeneous rule is used at the last step of subproof 5.2:2.1 of Figure 4. Here we "close" a subproof on the basis that the information available is contradictory. In sentential reasoning, this is standardly achieved by deriving a sentence and its negation, but here we are asserting that the diagram in step 5.2:1.1:2 contradicts the sentence at step 3 of the proof, since Charles and Alan are depicted as occupying adjoining offices.

A final example of heterogeneous deduction is given by the splitting into cases and subsequent merging of those cases. In sentential reasoning, a disjunctive sentence is usually used to demonstrate that a collection of subproofs exhausts the range of possible situations. For example, a sentence might assert that a particular object is either a tetrahedron or a dodecahedron. To utilize this fact we would construct two subproofs, one (sententially) assuming that the object is a dodecahedron, and the other assuming that it is a tetrahedron. If it is possible to reach a conclusion common to both of these cases, then that conclusion can be promoted out of the subproofs into the containing proof.

In standard first-order logic, the splitting and promotion is achieved by the use of a single inference rule called disjunction elimination. However,

we can isolate two facets to the rule. One is the production of an range of cases, which must exhaust all of the alternatives represented by the disjunction, and then second is the promotion rule which permits the extraction of information common to the subproofs into the containing proof. We can generalize these observations to the heterogeneous case.

Notice that in Figure 4 the individual subproofs of step 5 are headed not by sentences but by diagrams, each of which represents a different assignment to the large office. If we are to extract information from these cases we need to demonstrate that these cases are exhaustive, as indeed they are given the sentence in step 2. Since diagrams are typically bad at expressing disjunctive information, it is not unusual for a sentence to be the source of the justification that a set of cases is exhaustive.

At step 6 of the example of Figure 4 we have used an inference rule called Merge to promote information common to the cases into the main proof. In the sentential case the analog is to establish a sentence common to all subproofs, and to promote this sentence to the containing proof (on the basis that since the cases exhaust all possibilities, and each case entails this information, then it must be valid to assert the information outside of any specific case.) In the diagrammatic case, we could insist that each case contains the same diagram, and that this diagram is promoted, but to be more general we can allow the promotion of the diagram that contains the information common to diagrams in all subproofs.

4. GENERALIZING TO HETEROGENEOUS REASONING

Figures 9 and 10 represent examples of the kind of reasoning that we hope to capture using applications based on the full CAHR.

Figure 9 shows a simple proof prepared by an architect designing an addition to an existing house (shown at step 1). The client wants to add a guest bedroom and bath; the architect's proposed solution is shown at step 3. This solution is the result of a structured reasoning process that is recorded in the proof. The proof is a record of the rationale for the decisions incorporated into the final design, and can be used to present the reasoning to colleagues and clients.

Figure 10 illustrates an application of CAHR to the very different domain of financial planning. In this example, the user is deciding between a mortgage at 7.5% interest, with two points, and one at 8% interest, with no points. The user envisages two salient scenarios: one in which she is transferred out of town in five years; the second where she keeps the house for

ten years. The reasoning shows that under either scenario, the optimal choice is the former mortgage.

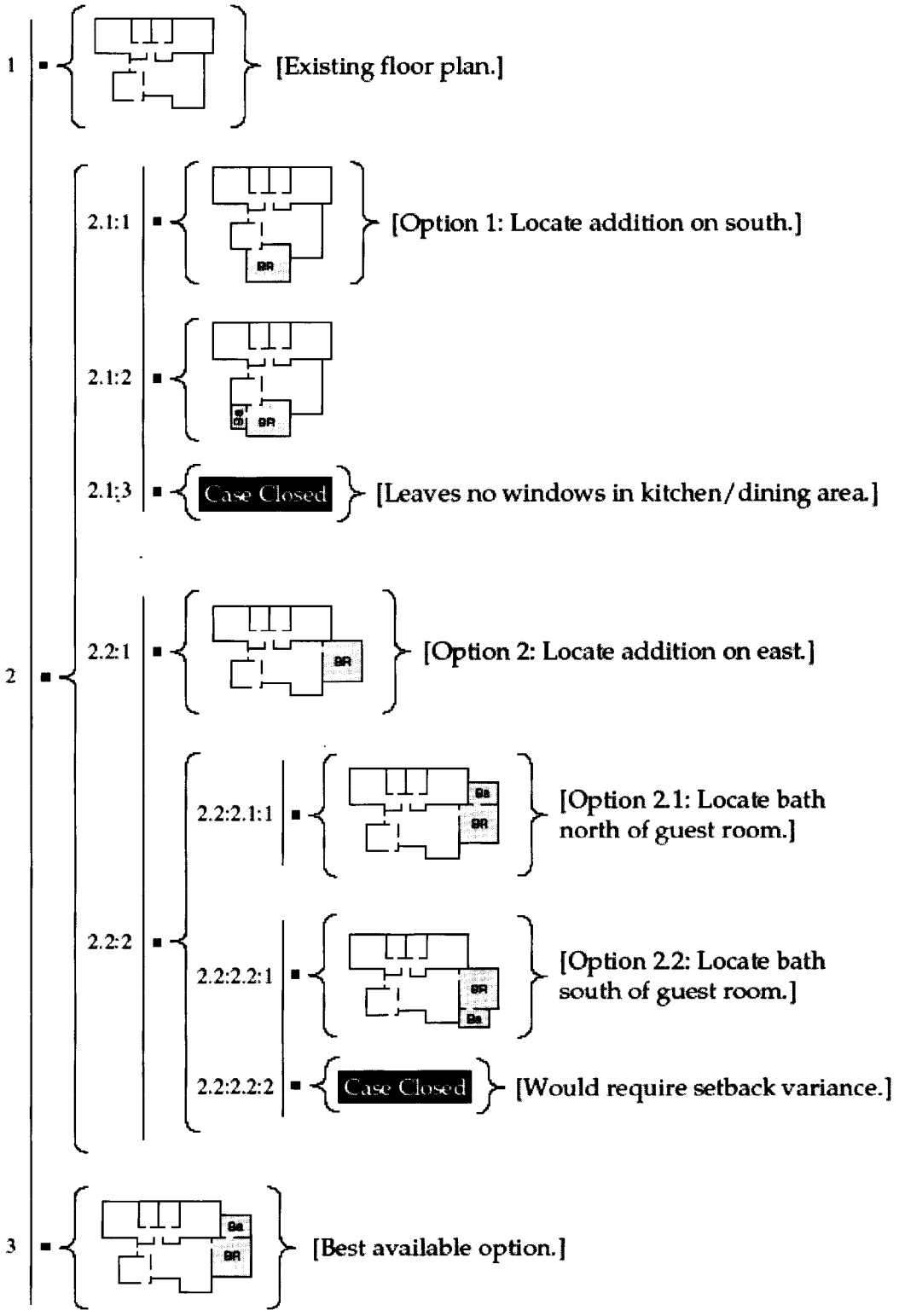


Figure 9. House design example

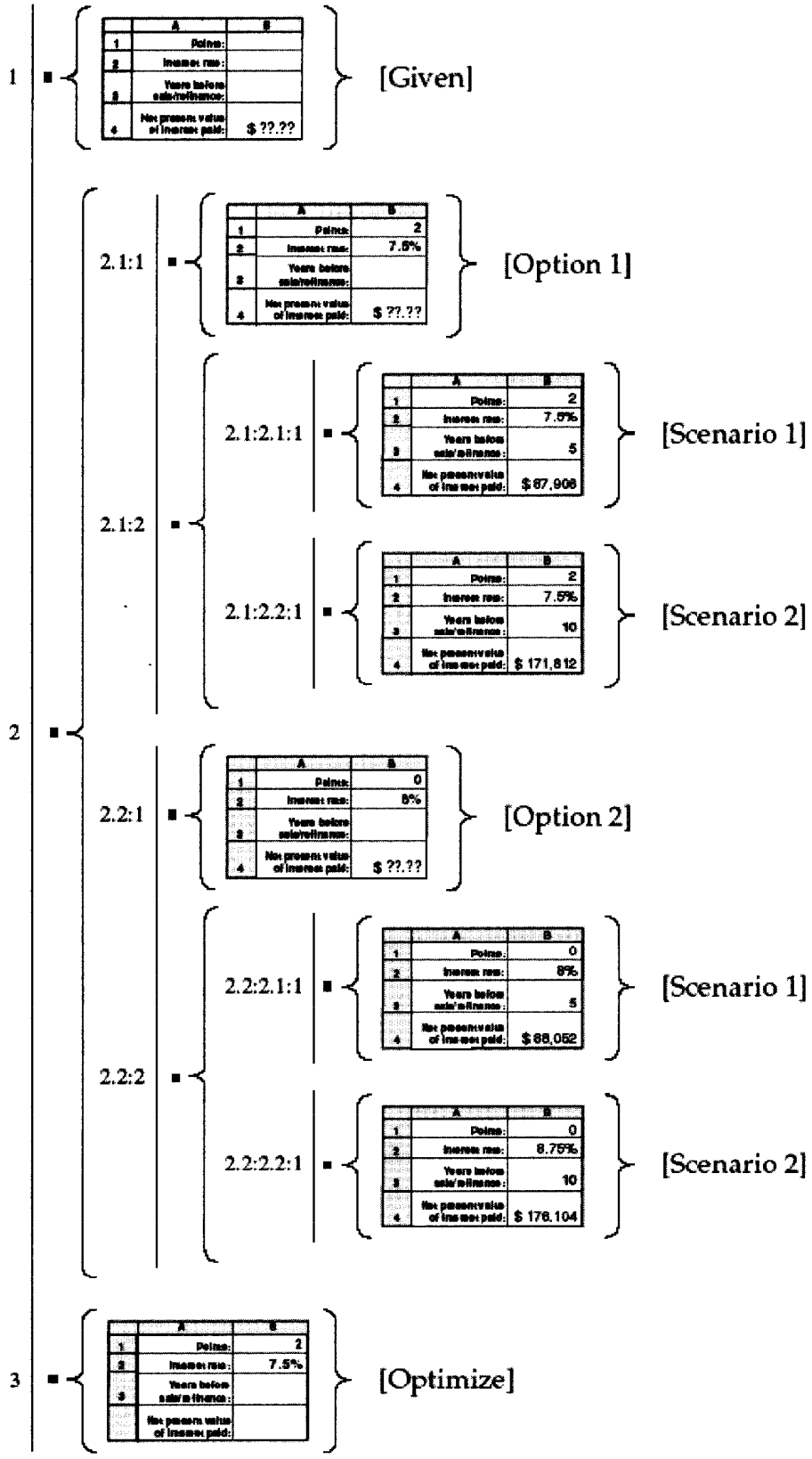


Figure 10. Financial planning example

We note that in neither of these examples is the user of the application performing deductive reasoning, but rather engaging in a kind of informal reasoning which is appropriate for these domain. In neither case, for example, does the user consider a provably exhaustive set of alternatives, and then reason to a conclusion necessitated by consideration of these alternatives. In these applications we are not using a CAHR-based application to provide any kind of logical certainty, but rather to simply structure and record the reasoning that is being carried out in the search for a an acceptable solution for our task. We believe that such reasoning is the rule rather than the exception in real-world reasoning situations.

Our CAHR allows the construction of applications that support this kind of reasoning. Systems of graphical representation vary widely in the extent to which they have fixed or intended interpretations. For example, circuit diagrams, architectural drawings, and perc charts have highly conventional interpretations, while the representations created in free-form drawing programs have no fixed interpretation, yet can be employed to represent a wide range of contents. Because of this, implementations of CAHR will differ in the extent to which they can antecedently recognize which modifications of a graphical representation are intended by the user to be informationally significant. Hence, implementations will differ in the extent to which they rely on the user to guarantee monotonicity versus imposing editing constraints whose goal is to enforce it.

In the example of Figure 9 changes to the plan are made by a series of incremental edits that are consistent with the structure of the proof. Within each subproof the information is accumulated on the diagrammatic representation. However, when using a diagrammatic representation which lacks a good specification of information bearing modifications to the diagram, or to provide the most flexible tool possible for a particular domain, we introduce the notion of an *unconstrained incremental edit.*

Unconstrained incremental edits are similar to the constrained incremental edits introduced earlier, but such an edit made to graphic G_N places no additional constraints on the modifications that can be made at descendant graphics. Thus, if we think of an arbitrary attribute of the graphic G_N as having both a displayed value and a range of permissible values (those that can be assigned to that attribute at G_N), then an unconstrained incremental edit to G_N modifies the displayed values of one or more attributes in G_N (and its descendants), but does not change the range of permissible values associated with any attributes of descendant graphics. In particular, we note that the edit could be undone without returning to the node at which it was made, since the information represented before the edit (whatever that was) is still among the range of permissible values.

In Figure 11 we contrast the effects of an unconstrained incremental edit and a permanent incremental edit made at the same node. In both proofs an incremental edit is made at node 3, where the value of the attribute is changed from *T* to *X*. Where the permanent edit differs from the unconstrained edit is in its effect on the range of permissible values of the attribute at subsequent nodes. In the proof on the right, the possible values reduce to *X* in the nodes following the one at which the change is made. Thus in the proof on the left, the user would be permitted to further edit the value of the attribute at nodes 4 and 5; in the proof on the right, the value is fixed at these nodes.

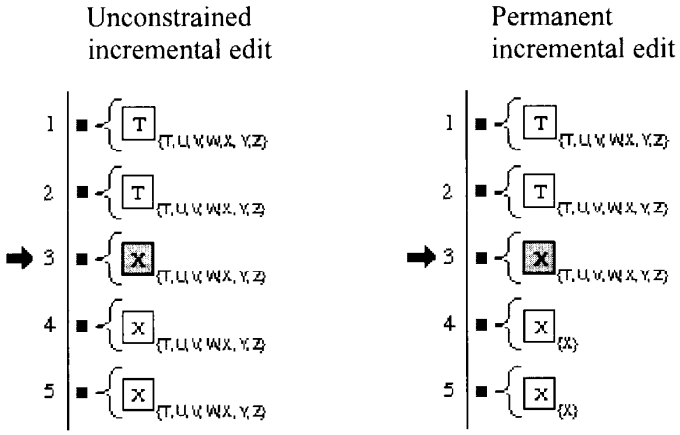


Figure 11. Constrained and unconstrained editing

Some implementations will allow unconstrained incremental editing, either because the graphical representations they employ do not have sufficiently well-defined interpretations capable of supporting an antecedently specified system of editing constraints, or because the nature of the reasoning tasks targeted by the application require the use of unconstrained editing. This is frequently the case when the reasoning supported has a temporal or planning dimension. In such applications, a piece of reasoning may begin with a graphical representation of the current state of the subject matter in question (e.g., the current design of a product), and proceed to reason about possible modifications of that state (design). In such cases, any feature of the current state that is potentially subject to modification in the solution will be represented by an unconstrained incremental edit in the initial graphic, while changes to the design can appropriately be made with constrained (or permanent) incremental edits. Applications that allow both permanent and unconstrained incremental editing may provide devices for distinguishing features of a graphic that are permanent from those that are not, and may also provide functions that enable users to "lock" a previously unconstrained incremental edit at a node, that is, make the feature behave like a permanent incremental edit made at that node.

4.1 Presentational editing

A significant component of the value of a CAHR-based application for reasoning is that the record produced serves as a kind of documentation for other stakeholders in the reasoning process. For example in the case of architectural design there may be many designers on a project, each specializing in a different aspect of the design, there will be a client with interest in the justifications of decisions made, and there will be construction engineers with a interest in understanding the design for the purposes of realizing an artifact. The use of a common document recording decisions made using representations in common use within each community serves to facilitate communication between the different stakeholders.

To allow the most flexible tools possible, an implementation of CAHR may allow the user to make certain kinds of modifications whose purpose is not to increment the information contained by the graphic in which the modification is made, but rather to comment on or make explicit some aspect of the reasoning that otherwise might be missed or undervalued. We describe two types of modification of this sort: spot edits and backdrop edits.

A *spot edit* made to a graphical representation G at node N is displayed only in the graphic G_N appearing at N. A spot edit does not affect either the display or editing constraints in effect in any other instance of G. The primary purpose of spot edits is presentational rather than informational. A spot edit is useful for highlighting or annotating particular characteristics of the graphic at a specific node. For example, a spot edit might be introduced to bring the user's attention to subtle incremental edits made at that node. (The gray highlighting in Figures 6, 8 and 11 is an example of spot editing.)

A *backdrop edit* made to G at node N is displayed at every graphic in the proof that is an instance of G, except a) in graphics that contain, or are descendants of graphics that contain, incremental edits which preclude the display of the backdrop edit, or b) in graphics that contain spot edits which preclude the display of the backdrop edit.

The value of a backdrop edit can be modified at any node in which it is displayed. Thus for purposes of editing and display, a backdrop edit is treated like an unconstrained incremental edit made at the node of origin of the graphical representation.

The purpose of backdrop edits is to introduce or change pervasive features of the graphical representation throughout the proof. They are appropriate for graphical features that are either meant to carry no significant information (e.g., the background color of a diagram or the font in a spreadsheet cell) or meant to represent background information presupposed throughout the reasoning (e.g., the property lines in an architectural drawing).

A graphical editor provided by a CAHR-based application may or may not support these or other forms of presentational editing, and those that do support presentational editing may restrict its use in various ways. For example, an application may restrict spot and backdrop edits to special "layers" of the graphic, allowing only incremental editing on the graphic's primary layer.

4.2 Justifications in heterogeneous reasoning

A CAHR-based application will, in general, provide a collection of *rules* for justifying reasoning steps. The notion here is a generalization of that of inference rules in deduction. Applications may generalize these constraints for reasons of convenience or intractability. It may not be deemed necessary that every move is justified in any way, for example, or justifications may be expressed in natural language which may leave them unverifiable by any syntactic means. It may also provide a facility for users to introduce their own rules for justifying steps.

An implementation of CAHR can provide various levels of support for rules. An application in a domain where it is possible and desirable for all reasoning steps to be formally checked, can provide *verification support* for a given rule or rules within the system. *Partial verification support* is available for systems which are not able to verify some but not all uses of the rule, for reasons of tractability perhaps. *Citation support* for a given rule allows the user to mark a node as justified by the rule and requires the user to indicate the supports for the justification, but cannot verify the legitimacy of the rule's application. The example of Figure 4 is intended to represent an application in which at least citation support is available as indicated by the presence of the names of rules and the cited support to the right of each step. Finally, an application may omit the justification mechanism entirely, preserving only the proof structure of the user's reasoning and the ability to make unstructured annotations to the node.

This variety is intended to make the architecture applicable to a wide range of domains. If applied to the realm of architectural design, for example, some design decisions may be made on aesthetic grounds, and any requirement that this be formally justified would impede the recording of such decisions (pending a formal theory of aesthetics.) Other decisions in the same domain may be made on the basis of the legal building code. While it is presumably possible to determine whether a design meets the building code, it may be impossible in practice to implement such a check. Citation support at least permits the user to cite the parts of the legal code that, say, put a proposed design outside of the code as part of the justification for not further considering the design.

For example, the justification at step 2.2:2.2:2 in Figure 9 presumably appeals to the building code in force as part of the justification, which is not, but perhaps could be, represented as part of the proof. Similarly, the justification at step 3 presumably appeals to some criteria enabling the judgment that this is the best solution, for example the specification of the project, the budget for the project, or the architects aesthetic sense. In the first two cases, it is again possible that this information might be represented within the proof structure to make this appeal to them more explicit.

The rules provided by an application can be either homogeneous or heterogeneous. Homogeneous rules specify legitimate reasoning steps that employ a single type of representation. Examples include rules that allow the inference of a sentence from other sentences, or the inference of a Venn diagram from other Venn diagrams. Heterogeneous rules specify legitimate reasoning steps that employ or can be applied to more than one type of representation, for example a rule that legitimates the inference of a sentence from a Venn diagram (for a discussion of inference with Venn diagrams see [Hammer, 1995; Shin, 1995]). Here we single out five important types of rules:

1. **Assumption rules:** Each proof (including subproofs) contains a (possibly empty) initial sequence of nodes that provide information assumed for the remainder of the proof or subproof. These nodes are justified using an assumption rule and require no supports. An application may provide a variety of assumption rules to be used in different contexts, corresponding to different types of assumptions.

   The proof illustrated in Figure 10 uses three types of assumption rules, which we have called Given, Option, and Scenario. The first marks the information assumed in the problem; the second indicates possible choices open to the user; and the third allows the user to entertain possibilities outside her control. Steps justified by means of these different assumption rules may figure differently in other reasoning steps and procedures. The Given rule is also used explicitly in Figure 10 and implicitly at the first step of Figure 9.
2. **Transfer rules:** Transfer rules allow the transfer of information from one representation to another. Examples include rules that allow the user to express in sentential form information that is present in a graphical representation at an earlier accessible node, or rules that allow the modification of a graphical representation based upon information expressed in sentences at accessible nodes. The Apply rule is an example of a diagram to sentence transfer rule.

   We note that in the general case, there may be many representations in play during a piece of reasoning, and that transfer rules may involve transferring information between representations in a many-to-many fashion.

3. **Case rules:** As observed above, case rules may be used to justify a node that contains a set of subproofs. The most important class of case rules is what we call exhaustive cases rules. An exhaustive cases rule allows the user to break into a collection of alternatives ("cases") that are jointly exhaustive (that is, one of which must hold). The cases are specified by the initial (assumption) nodes of the subproofs contained by the node being justified. Exhaustive cases rules permit cases that are specified by various types of representation (sentential or graphical), and which are supported by nodes containing various types of representation.

   Not all case rules will require an exhaustive set of cases. In the financial planning example of Figure 10 the user is deciding between two different mortgage options. In practice there may be very many available deals, and in principle they might be considered as a range of exhaustive cases, but more likely only representative or extremal examples need be considered.

   The complete range of cases may not be available even in principle. The architectural example of Figure 9 considers two locations for the extension, to the south or to the east of the main house, but (assuming continuous space) there is an infinity of possible options, with varying dimensions for the new addition. Again, only extremal or representative cases are likely to be considered in practice.
4. **Promotion rules:** Promotion rules allow users to extract information from a set of exhaustive cases, that is, to promote information contained in one or more of the embedded subproofs to a subsequent node in the embedding (parent) proof. A promotion rule is used to justify a node in the parent proof and cites as support an accessible node containing a set of subproofs. Logically, promotion rules allow users to extract information from a set of exhaustive cases, that is, to promote information contained in one or more of the embedded subproofs to a subsequent node in the embedding (parent) proof. A promotion rule is used to justify a node in the parent proof and cites as support an accessible node containing a set of subproofs. One important class of promotion rules are what we call merge rules: The application of a merge rule is legitimate when the information extracted is present in each of the cases not containing a "close" declaration at any of its nodes (see below).

   The last steps in Figure 9 illustrates the use of a Merge rule. In this case, one of the subproofs has been closed, indicating that the case under consideration is inconsistent. The merge rule then promotes the information from the sole remaining case as representing "the way things must be". If there are multiple open cases, the strongest representable information common to all open cases is promoted into the containing proof.

   The final step in Figure 10 shows a different kind of promotion rule, one in which information is promoted as a consequence of the application

of a metric to the various options, in this example the option with the minimum interest payment in corresponding scenarios.

We note that different promotion rules might be applicable depending on the nature of the cases that they are acting upon. For example, we discussed above the assumption rules Option, which reflects a choice on the part of the user, and Scenario, reflecting a forseeable state of the world that might arise outside of the user's control. When promoting an outcome from a range of cases considering different Options, we should use an optimizing promotion rule which preserves the best case. In contrast, it is not obvious whether to use an optimizing, averaging, or pessimizing promotion rule when promoting outcomes from a range of Scenarios. The most conservative approach would be to use a pessimizing rule in a manner analogous to the familiar minimax procedure.

In the case of logical deduction, promotion rules are pessimizing in the sense that only information known in all cases can be promoted from a range of Assumption cases (and then only if the cases are known to be exclusive.)

Reasoning by cases is fundamentally hypothetical and disjunctive. It is hypothetical in that the reasoning within a particular case is based on assumptions that need not hold. It is disjunctive in that we are in general left with multiple open alternatives when we conclude our consideration of the cases. For this reason, promotion rules are essential if our goal is to find (and justify) a unique or optimal solution to a reasoning task.

5. **Declaration rules:** Declaration rules are used to justify assertions made about the state of the proof at the node in question. Examples include declarations that a case (subproof) is closed or that a case is consistent with the information assumed in the proof. Declaration rules specify the conditions under which the declarations can be made.

   Several examples of the Close rule are illustrated in Figures 4 and 9. These proofs also highlight the variety of reasons for which cases may be closed in the course of solving a reasoning task. In Figure 9, cases are ruled out for aesthetic and legal reasons; in Figure 4, cases are closed because they are inconsistent with the constraints given in the problem.

   Other declarations are possible. It might be necessary to demonstrate that the information expressed in a set of representations is consistent, when the representations are interpreted conjunctively. For example, imagine being presented with a collection of constraints, and a candidate solution. The goal of the reasoning is to show that the candidate solution is in fact a solution to the problem posed by the constraints. We need to conclude the proof with a declaration that the constraints are all satisfied in the solution design.

A CAHR-based application may implement rules explicitly or implicitly. By this we mean that the user may or may not be required to understand and

choose rules from an explicitly presented range of options. For example, an application may treat the first node in every proof or subproof as an assumption; it may treat every node containing a set of subproofs as an exhaustive range of cases; and it may contain routines that automatically apply an appropriate promotion rule in the node immediately following a range of cases.

5. APPLICATIONS OF THE ARCHITECTURE

As we have hinted throughout this chapter, we see applications based on the CAHR architecture and being useful in a variety of domains, in particular in the realm of design. There are a number of reasons for this. The first is that the architecture explicitly supports the use of graphical representations which are pervasive in design domains. Because designed artifacts occupy space in the world, graphical representations of space such as architectural plans, wiring diagrams, circuit diagrams and the like are natural for such tasks. Other similarly spatial tasks, such as determining possible foldings of complex molecules would also naturally fall into this category (indeed this too can be seen as a design task, to design a folding that matches the constraints obtained from experimental data.)

The design of complex artifacts typically involves teams of designers each with a different specialization. For example, the design of a building might involve designers concerned with the structure of the building, electricians concerned with the placement of generators, and the wiring of the building, plumbers concerned with routing pipes, landscape architects concerned with exterior access to the building and so on. Each of these specialists may use a different representation of the artifact appropriate for their own interests. However, these specialists do not work in isolation, they are collaborating to build a single artifact, and consequently, the design reasoning will naturally involve all of these representations interpreted conjunctively. The need for an architecture able to support heterogeneous reasoning naturally falls out of this kind of collaborative task.

We think that one of the most exciting aspects of this architecture is the promise of producing documents that record the rationale for the decisions made in the process of producing designs of complex artifacts, or more generally of complicated decision making processes. Documents constructed using this architecture will allow designers to replay their reasoning for colleagues, clients and for themselves at a later date. Rather than presenting a client with a proposed design for the extension to their house, an architect would be able to replay the reasoning resulting in that design. If the client or a colleague were to disagree with the justification associated with a step, the reasoning at that step might be reconsidered. For example, the architect may

judge a candidate design unacceptable on various grounds, which are in fact acceptable to the client.

More importantly, the ability to capture the rationale for the design of a complex artifact will ease the task of maintaining that artifact over time. This is of particular importance when the longevity of the artifact approaches that of the team that designed it. If the rationale for the particular decisions are lost, then future maintenance becomes problematic since the features of the design that must be maintained are not necessarily obvious.

We recognize that the task of recording the structure of reasoning and providing justifications for individual inference steps can interfere with the performance of the reasoning task. The degree to which such information is demanded by an application based on our architecture will depend on the perceived importance of collecting this information for a given reasoning task. In any case, user interfaces must be designed to minimize the intrusiveness of the process of recording this information about the reasoning task.

6. CONCLUSIONS AND FURTHER WORK

We have described a computational architecture for heterogeneous reasoning. The architecture is a general framework for building applications to support users in heterogeneous reasoning tasks -- tasks involving multiple representations for information. Of particular interest are graphical and sentential representations which have markedly different characteristics. Heterogeneous reasoning tasks are very common; indeed, we believe that heterogeneous reasoning tasks are at least as common as homogeneous, sentential tasks.

Our architecture has been developed by consideration of the generalizations necessary to the notion of formal reasoning represented by natural deduction proofs. In our architecture a proof, represents a (possibly incomplete) piece of reasoning. Proofs are recursively structured, whose basic elements are nodes each of which may contain instances of information-bearing representations. Central to the architecture is an analysis of how information flows through proofs, and how this constrains the actions available to the reasoner at different nodes in the proof.

The architecture does not mandate the specification of justifications or the manner in which they are to be collected if their use is supported. We believe that the degree to which specifying and collecting justifications for reasoning is critical varies from domain-to-domain, and that the corresponding degree of intrusion into the reasoner's process that is warranted by the desire to collect justifications varies proportionally. As a consequence it is

up to the individual implementer to design appropriate user interfaces for eliciting relevant information during the reasoning process, in a manner that is sensitive to the domain at hand..

We believe that our architecture may be useful in a wide range of application domains, including but not limited to design of complex artifacts. Proofs may be viewed as means for capturing rationales involved in the solution of complex reasoning problems, since they record not only the eventual outcome of the reasoning, but also the history of the decisions made in producing that outcome.

We are currently implementing our architecture, as a Java application framework called *Openproof* when implemented the framework will accept implementations of editors and inference engines for particular representations as plug-ins. By providing combinations of such plug-ins, we hope to facilitate the development of arbitrary heterogeneous reasoning environments.

7. EXERCISES AND PROBLEMS

1. Using the CAHR representation, prepare a heterogeneous proof sketch demonstrating the solution to the following problem. Your sketch should resemble that of Figure 4, with plausible justification rules of your own devising. For each such rule describe clearly the criteria for it successful application, and classify the rule according to our five types of rule described earlier.

   Harry and his wife Harriet gave a dinner party. They invited Harry's brother, Barry, and Barry's wife Barbara. They also invited Harry's sister, Samantha, and her husband Samuel. Finally, they invited Nathan and Nathan's wife Natalie. While they were seated around the table, one person shot another.

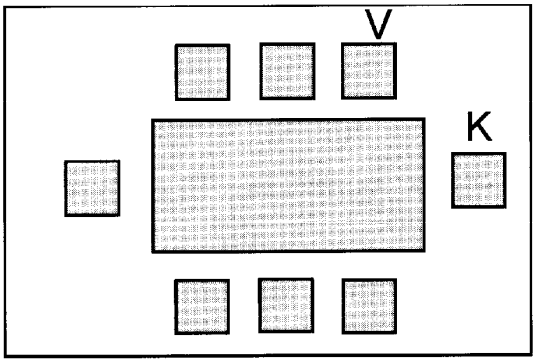


Figure 12.

The chairs were arranged as shown in the diagram. The killer sat in the chair marked K. The victim sat in the chair marked V. Every man sat opposite his wife. The host was the only man who sat between two women. The host did not sit next to his sister. The hostess did not sit next to the host's brother. The victim was the killer's former spouse.

2. Speculate on the costs and benefits of adopting a CAHR-based enviroment for capturing the structure of arguments and rationales for applications such as: architectural design of an individual residence, planning a mission to Mars, designing a CPU chip for the next generation of computer, designing a nuclear power station, developing a survey instrument, which will be reused for over forty years, for collecting complex economic data. What features would a task have to make the adoption of CAHR most necessary and/or desirable?

3. Formal deduction, in mathematics for example, provides logical certainty of results that are deduced. By allowing arbitrary justifications that are not formally verifiable by software, the CAHR-architecture has sacrificed this certainty. What (if anything) has been gained in its place? What would be lost if one were to insist on the formal verifiability of all justifications? Would mathematical practice be representable in such a system, for example could Wiles' proof of Fermat's theorem be so represented (in principle and in practice)?

4. When working with a patient a doctor typically describes only the diagnosis reached and actions taken, and not the alternatives that were considered and rejected. To what extent do you as patient care about these alternatives? What stakeholders might benefit from a complete record of the doctor's reasoning and which would not? What social forces would act to encourage and resist the widespread adoption of CAHR-based rationale capture if it were available in the medical domain?

8. REFERENCES

Barker-Plummer D., Etchemendy J. Applications of Heterogeneous Reasoning in Design. Machine Graphics and Vision 2003; 12(1):39--54

Barwise J., Etchemendy J., Hyperproof. Stanford and Cambridge: Cambridge University Press 1994. Program by Gerard Allwein, Mark Greaves and Mike Lenz.

Fitch F.B. Symbolic Logic: An Introduction, Luvain, 1952.

Gentzen G. Investigations into Logical Deduction, In The Collected Papers of Gerhard Gentzen, M.E.Szabo, ed. Amsterdam: North-Holland, 1935/1969

Glasgow J. Narayanan H. and Chandrasekaran B. Diagrammatic Reasoning: Cognitive and Computational Perspectives. AAAI Press and MIT Press, 1995.

Hammer E. Logic and Visual Information. Number 3 in Studies in Logic, Language and Information. CSLI Publications 1995.

Harel D. On Visual Formalisms. Communications of the ACM, 1988. 31(5):514—530

Johnson S.D., Barwise J. and Allwein G. Toward the Rigorous Use of Diagrams, in Reasoning About Hardware. In Working Papers on Diagrams and Logic, G Allwein and J. Barwise eds Indiana University Logic Group, Preprint Series, 1993.

Mitchell W. The Logic of Architecture. Cambridge: MIT Press. 1990.

Prawitz D. Natural Deduction: A Proof-Theoretical Study. Stockholm: Almqvist and Wiksell, 1965.

Prawitz D. Towards a Foundation of a General Proof Theory. In Proceedings of the 1971 International Congress of Logic, Methodology and Philosophy of Science. Bucharest. 1973.

Shin S-J. The Logical Status Of Diagrams. Cambridge University Press. 1995.

Chapter 5

ALGEBRAIC VISUAL SYMBOLSM FOR PROBEM SOLVING

Iconinc equations from Diophantus to the present

Boris Kovalerchuk and James Schwing
Central Washington University, USA

Abstract: This chapter provides a discussion of mathematical visual symbolism for problem solving based on the algebraic approach. It is formulated as lessons that can be learned from history. The visual formalism is contrasted with text through the history of algebra beginning with Diophantus' contribution to algebraic symbolism nearly 2000 years ago. Along the same lines, it is shown that the history of art provides valuable lessons. The evident historical success provides a positive indication that similar success can be repeated for modern decision-making and analysis tasks. Thus, this chapter presents the lessons from history tuned to new formalizations in the form of iconic equations and iconic linear programming.

Keywords: Mathematical visual symbolism, problem solving, algebraic symbolism, iconic equations, iconic linear programming, iconic algebraic expressions.

1. VISUAL SYMBOLISM VS. TEXT

1.1. Diophantus and beginning of algebraic symbolism

Mathematicians know two famous visual forms of problem solving:

- *algebraic symbolic reasoning* and
- *geometric diagrammatic reasoning*

although typically algebraic formalism is not viewed as a visual reasoning approach. In this chapter we show that in fact *algebraic reasoning is visual reasoning* and that it has been very efficient throughout history. Indeed, in many cases it is a better choice than textual or geometric reasoning. Inven-

tion of algebraic symbolism was critically important for progress in generating the algebra necessary for solving linear, quadratic and other equations. The reason is obvious -- it is almost impossible to solve an equation expressed as text.

A simple modern algebraic expression $(12 + 6n)(n^2-3)$ is much longer in its textual (verbal, rhetorical) form: "a sixfold number increased by twelve, which is multiplied by the difference by which the square of the number exceeds three". The expression can be easily transformed to $6n^3-12\ n^2 + 18n - 36$ using symbols, but it would be extremely difficult to accomplish this using text only.

The history of the invention of algebraic mathematical symbolism is quite dramatic and spins out over some 2000 years. It is traced to Babylonian mathematics and the "father of algebra," Greek mathematician Diophantus (about 200-284 A.D.).

Europe was not aware of his "Arithmetica" until 1570, when the book written in Greek and preserved by the Arabs was translated into Latin. But even this translation, delayed for more than 1000 years, was not published. The first translation with known impact on European mathematics is due to Bashet, who translated it in 1621. Ferma read it, commented on it, and was influenced by Arithmetica [Swift, 1956; O'Connor & Robertson, 1999].

It seems that the history of geometric proof was no less dramatic. "There are some arguments that it is unlikely that the Greeks could have invented their notion of proof so rapidly and in isolation. Instead, it is suggested that the notion of geometric proof was a *secret* that was jealously guarded from all but the "inner sanctum" of the Egyptian priesthood" [Altshiller-Court, 1964].

1.2. Mathematics: from verbal algebra to symbolic visual algebra

Developing mathematical symbology progressed in several ways and steps over the centuries:

1. progressing from text to symbols *directly* and
2. progressing from text to symbols by first going to *abbreviations* and then to symbols that may not have direct link to a textual representation and
3. progressing to more and more sophisticated *syntax* defining how symbols could be combined.

Below we illustrate this progress with examples from Diophantus' Arithmetica [Geller, 1998] starting from examples on *direct transition from text to symbols*. Diophantus used:

- as a symbol for an *unknown* the Greek small letter sigma when it is written at the end of a word, ς, the symbol differs from the standard small sigma σ, this sigma ς, is called the final small sigma and
- symbols for numerical coefficients correspond to alphabetic Greek numerals (α, β, γ, δ, ε, ... → 1, 2, 3, 4, 5, ...).

Diophantus also used abbreviations progressing from text to symbols. He introduced:

- the symbol for *constant term* 1 as a capital M with a small circle above (this symbol is the abbreviated Greek word *monades* for "units");
- the symbol for an "*unknown squared*" (our modern x^2) as the first two letters of the Greek word *dunamis* for "power";
- the symbol for an "*unknown cubed*" (our modern x^3) as the first two letters of the Greek word *kubos* for "cube";
- the symbol for "*minus*" originated from the first two letters of the Greek word *leipis* for "lacking."

Diophantus' **symbolic syntax** has several components:

- the summation symbol is omitted and expressed by sequential writing the terms to be added,
- all negative terms follow the minus symbol, and
- the power of the unknown precedes the numerical coefficients.

There are some indirect indications that Diophantus may have been influenced by the Hindus syncopation that was quite similar to that of Diophantus [Geller, 1998].

1.3. Lessons from the history of algebra and calculus

Below we present an extract from the English translation of the real historical mathematical text, the first book in algebra (~ A.D. 830), "*Al-jabr wa'l-muqabala*" by M. Al-Khwarizmi [English translation: Al-Khwarizmi, 1974; Parshall, 1988]. The term "*algebra*" came from this book as well as the term "*algorithm,*" which are derived from the name of the author.

"...a square and 10 roots are equal to 39 units.

The question therefore in this type of equation is about as follows: what is the square which combined with ten of its roots will give a sum total of 39?

The manner of solving this type of equation is to take one-half of the roots just mentioned. Now the roots in the problem before us are 10. Therefore take 5, which multiplied by itself gives 25, an amount which you add to 39 giving 64. Having taken then the square root of this which is 8, subtract from it half the roots, 5 leaving 3.

The number three therefore represents one root of this square, which itself, of course is 9. Nine therefore gives the square."

The modern symbolic (iconic) representation and solution is much shorter and clearer [Parshall, 1988]:

Solve $x^2 + 10x = 39$ for x^2.

Solution: $x^2 + 2{\cdot}5x + 25 = 39 + 25$; $x \cdot x + 2{\cdot}5x + 25 = 39 + 25 = 64 = 8{\cdot}8$;
$(x+5)(x+5) = 8{\cdot}8$; $(x+5)^2 = 8{\cdot}8$; $x + 5 = 8$; $x = 3$, $x^2 = 9$.

This evident historical success intrigued us to repeat it for modern decision-making and analysis tasks. Section 2 demonstrates such a development for iconic equations. Tables 1 - 3 provide another appealing example of the advantages of symbolic (iconic) representation.

Table 1. Description of linear equations using text and iconic symbology

Rhetorical, textual representation of a problem	*Compressed text content in iconic form*
Three multiplied by an unknown value plus five multiplied by another unknown value minus seven is equal to twelve	$3x + 5y - 7 = 12$
Six multiplied by an unknown value plus four multiplied by another unknown value plus two is equal to eight	$6x + 4y + 2 = 8$
Find both unknown values if they exist.	$x, y?$

Table 2. Solution of linear equations in symbolic iconic form

1.	$2(3x+5y-7) = 2*12$	6.	$y = 32/6$
2.	$6x+10y-14 = 24$	7.	$3x = 12+7-5y$
3.	$10y-4y-14-2 = 16$	8.	$x = (12+7-5y)/3$
4.	$6y-16 = 16$	9.	$x = (12+7-5*32/6)/3$
5.	$6y = 32$; $6x+4y+2 = 8$; $-(6x+4y+2) = -8$		

Table 3. Textual solution: first two steps

1. Two multiplied by open parenthesis three multiplied by an unknown value plus five multiplied by another unknown value minus seven close parentheses is equal to twelve. Six multiplied by the same unknown value as in the first sentence plus four multiplied by another unknown value (that is the same as the second unknown value in the first sentence) plus two is equal to eight.
2. Two multiplied by open parenthesis three multiplied by an unknown value plus five multiplied by another unknown value minus seven close parentheses is equal to twelve. Minus multiplied by open parentheses six multiplied by the same unknown value as in the first sentence plus four multiplied by another unknown value (that is the same as the second unknown value in the first sentence) plus two close parenthesis is equal to minus eight.

As noted above, it is almost impossible to solve systems of equations using words alone. Table 1 gives a linear system of equations. Table 2 gives a symbolic, iconic solution. Table 3 shows only the first two steps of the solution presented in the textual form.

The example demonstrates that iconic reasoning for finding a solution is much shorter than reasoning that uses only a textual representation of the task. Comparison of Tables 2 and 3 highlights the obvious advantages of a solution method using symbols/icons. Here it is important to note that the symbolic/iconic representation has a much more ambitious goal than merely visualizing the task and the solution. It really provides an efficient visual solution method.

Today the advantages of symbolic, iconic representation and reasoning in mathematics are obvious, but such was not the case at the time that the iconic representations were invented. And it is here that we can learn an important lesson from history. The first symbolic, iconic form was rejected. As noted at the end of section 1.2, in A.D. 250, Diophantus invented symbolic iconic representations [Geller, 1998; Miller, 2001], specifically:

$$\varsigma, \Delta^{T}, K^{T}, \Delta^{T}\Delta, \Delta K^{T}, K^{T}K$$

for the algebraic unknowns that have become the modern symbols

$$x, x^2, x^3, x^4, x^5, x^6.$$

For Diophantus the algebraic expression $x^3 + 2x - 3$ would be

$$K^{T}\alpha\varsigma\,\beta\,\Uparrow M\,\gamma$$

Table 4 derives the equivalence. Not surprisingly, the textual form at the bottom is ambiguous and much longer than either the modern or Diophantus' notations.

It is not clear from the text whether we have: $x^3+x\cdot 2-3$ or $x^3+x\cdot(2-3)$. To avoid this ambiguity the text should be even longer, perhaps: "Cube of unknown number plus the same unknown number multiplied by two and minus three from the whole expression presiding to three". In both modern and Diophantus' notations there are 8 characters vs. 130 characters for the text based representation. That is, Diophantus' notation is quite competitive even now.

The advantages of symbolism become even more evident when we try to combine two expressions. Let us add $x^3+x\cdot 2-3$ and $2x^3+3x-2$. Diophantus'

notation requires 16 symbols to write down these two expressions before combining them:

$$K^{T}\alpha\varsigma\beta \pitchfork M\gamma$$

$$K^{T}\beta\varsigma\gamma \pitchfork M\delta$$

and the result can be produced by adding columns

$$K^{T}\gamma\varsigma\varepsilon \pitchfork M\varepsilon$$

This is equivalent to $3x^3+5x-5$ in modern notation. In both cases we have still used 8 symbols.

Table 4. Example of Diophantus' symbology

Diophantus' symbology [Geller, 1998; Miller, 2001]	$K^{T}\alpha\varsigma\beta \pitchfork M\gamma$							
Modified Diophantus' notation with space for +	$K^{T}\alpha\ \varsigma\beta \pitchfork M\gamma$							
Diophantus notation sliced with extra space for +	K^{T}	α		ς	β	$\pitchfork$	M	γ
Modern algebraic notation (sliced)	x^3	1	+	x	2	-	1	3
Modern algebra (with multiplication sign)	$x^3 \cdot 1$		+	$x \cdot 2$		-	$1 \cdot 3$	
Modern algebra (without multiplication sign)	x^3		+	$2x$		-	3	
Modern algebra (compact)	x^3+2x-3							
Textual form	Cube of unknown number plus the same unknown number multiplied by two minus three							

In text we would need about 260 symbols before combining expressions. Note that text has no a simple positional way of combining components such as we used in the symbolic form.

The next example looks almost like a modern intelligence puzzle. A collection of epigrams known as the *Greek Anthology* contained among its mathematical problems a puzzle about Diophantus himself [Cohen & Drabkin, 1958; Parshall, 2002]:

God granted him to be a boy for the sixth part of his life, and adding a twelfth part to this, He clothed his cheeks with down; He lit him the light of wedlock after a seventh part, and five years after his marriage He granted him a son. Alas! late-born wretched child; after attaining the measure of half his father's life, chill Fate took

him. After consoling his grief by this science of numbers for four years he ended his life.

Questions posed based on this text are: When was he married? When did he get his first son? When did he die?

The answer provided is: Diophantus married at age thirty-three, had a son when he was thirty-seven, and died when he was eighty-four.

How can one answer these questions without iconic symbols? It is a real challenge using reasoning in a natural language as we have seen in examples above, but the solution using iconic notation is simple:

$$x/6 + x/12 + x/7 + 5 + x/2 + 4 = x;\ 75x/84 - x = -9;\ -9x/84 = -9;$$
$$x/84 = 1;\ x = 84.$$

Table 5, based on [Schroeder, 1997; Miller, 2001], presents the trend found in later mathematics from using text and moving on to iconic symbology in attempt to rediscover Diophantus' dormant invention. This trend demonstrates that mathematics again went through the same three steps Diophantus had earlier:

(1) text ⇒ (2) abbreviations ⇒ (3) icons.

In this way, mathematics was able to reach the highest level of rigorous reasoning and problem solving that probably would not have been reached without such iconographic knowledge representation or minimally would have taken much more effort.

Table 5. Trend of mathematics from text to iconic symbology

Time	*Presentation*	*Example*	
Before 15th century	Text	Al-Khwarizmi:	"Square and 10 roots are equal to 39 units" (modern x2+10x=39)
		Fibonacci:	"Divide .a. by .b. to obtain .e." (modern a/b=e)
Renaissance	Abbreviation	Cardan:	"1.quad.aeq.10.pos.p.144" (modern $x^2 = 10x + 144$)
From 17th century	Icons, symbols	Modern	$x^2 = 10x + 144$

Table 6 details the history of mathematical symbology [Miller, 2001] showing what happened after Diophantus' invention was forgotten. The first 1200 years were dark years; in 14th century Europe, everything except numerals were still written out in words.

In 1463, Benedetto introduced a symbol for the **unknown** – the Greek letter ρ – rediscovering what Diophantus did more than 1000 years before. In 1494, Pacioli used m~ as an icon for minus for the first time.

The history of calculus symbology (see Table 7) provides further insight into the role of iconic symbology in problem solving. Newton's symbology for derivatives is simpler than Leibniz's symbology and is well suited for functions of one variable $f(x)$ such as velocity and acceleration.

On the other hand, while it is more complex, Leibniz's symbology is better suited to functions of two and more variables, $f(x_1, x_2, ..., x_n)$. "British mathematicians who patriotically used Newton's notation put themselves at a *disadvantage* compared with the continental mathematicians who followed Leibniz." [O'Connor & Robertson, 1997]. Note also that Leibniz's notation abstracts and visualizes a more complex concept.

Table 6. History of mathematical symbology (based on [Miller, 2001])

Year	*Icon/symbol*	*Inventor*
1489	+, –	Widmann
1525	√.	Rudolff
1557	=	Recorde
1570	a, b as known positive numbers	Cardan
1580.	literal coefficients	Viète
~ 1600	division ÷	Stevin
1631	multiplication ×, and ±	Oughtred
	less, < ; greater, >	Harrio
1637	x,y,z as unknowns,	Descartes
	Equation $ax + by = c$, for positive a, b, c	
1657	a, b, c as positive and negative numbers	Hudde
1698	Multiplication (·)	Leibniz
1734	≥	Bouguer

Table 7. Alternative symbology for derivatives

Newton's symbology	*Leibniz's symbology*
$\dot{x}, \ddot{x}$	dx/dt

The important question arising from history is: "How was it possible that, in spite of obvious advantages of algebraic symbolism, it was not used for 1250 years after Diophantus invented it?"

> The *dominance of Euclidian geometrical algebra* is typically considered as the major reason that Diophantus' invention was not used or further developed for a millennium [Parshall, 1988].

Was this accidental? Close consideration shows that it was not accidental in a way similar to the fact that Ptolemy's geocentric model was not accidental. Geometric algebra is more directly visual; that is, it is similar to di-

rect modeling of **physical entities** that we perceive directly, such as lines, angles and the rotating of the Sun around the Earth. In contrast in Diophantus' algebra, we are forced to operate with **abstract entities** that we cannot observe directly. Diophantus' algebra operates with **visual but abstract concepts** of unknown values, constants and arithmetic operations. Geometric algebra operates visually with **concrete objects** of the real world. More exactly, it operates with objects that have a direct match in the real world. Thus, geometric algebra was much easier to understand. It relies much less on abstract thinking.

Now we can return to the present and ask: "Why is there still no sophisticated visual reasoning system?"

We feel that the reason is basically the same as it was in the time of Euclid, Pythagoras, Diophantus and Ptolemy – sophisticated visual reasoning requires the *visualization of abstract concepts* not only objects that have direct match in the real world. Visual reasoning with abstract concepts is the real challenge that still needs to be addressed.

1.4. Lessons from history of art

In the previous section, we demonstrated advantages of symbolic, iconic representation for solving mathematical tasks. The history of art provides us even more fascinating examples. For centuries artists were able to express large textual pieces of the Bible in a single painting. If we consider small paintings and sketches that were made for viewing from the same distance as text we can notice that they occupy much less space than the corresponding Bible's text.

Thus, artists **"compress" the space** occupied by bible's contents. Artists also "compressed" other texts such as myths and proverbs. The important advantage of "reading paintings" is that we can see the "whole story" at once using our parallel visual processing abilities. Thus pictures also **compress the time** required for "reading" a picture in comparison with a sequential reading of the text.

Note that the text itself can be "compressed" into other text forms known as proverbs and sayings. Table 1 in Chapter 8 illustrates the case and permits the comparison of textual and visual "compressions." It is based on Pieter Bruegel's painting "Blue Cloak" that "compresses" 78 Flemish proverbs. Fragments of Bruegel's painting serve as role models for **icons for complex concepts.**

Text, which has been compressed into a text, based metaphor or proverb still needs to be read sequentially, but an image based metaphor or icon can be "read" as a whole. Above we illustrated a merging (summation) operation for two algebraic expressions in symbolic form, showing that the merged

expression occupies the same space as each of the original components. Table 1 in Chapter 8 illustrates how this is done in art. The text based metaphor is doubled practically, but the merged picture metaphor occupies almost the same space as each of its components, showing the same "compression" efficiency as in the mathematical examples above.

One can ask: "How is this example from art related to use of visual symbolism in solving decision-making problems that we have seen in mathematical examples?" At first glance, there is no such relation, but the efficiency of symbolic mathematics in large part came from its compressed information representation. Thus, any efficient compressed visual presentation can be potentially insightful for the decision-making and problem-solving tasks.

Not every relation in the world can be easily presented in text. Often spatial adjacency is a natural way to foster a vision of non-textual relations between entities. Merging two proverbs visually reveals a deeper meaning of each proverb and their interrelation than a text only representation does.

2. SOLVING ICONIC EQUATIONS AND LINEAR PROGRAMMING TASKS

2.1. Iconic algebraic expressions

Assume that we need to compute a simple sum, $S = 2 + 5 - 1 + 4 = 10$. Next assume that we have two associated data types (lions and birds):

S = two lions + five birds – one lion + four birds.

At first, there is some uncertainty about what this sum means. If we ignore the data types, we will get 10 abstract entities or living creatures.

But if we pay attention to specific data types, we will have to have two numbers in our solution: one (lion) and 9 (birds) that cannot be called a sum in a standard arithmetic. To be consistent with the standard arithmetic two separate equations: 2-1 = 1 (for lions) and 4+5 = 9 (for birds) can be generated. Data with a hundred different data types will be expressed in a hundred equations in the standard arithmetic. Visual, iconic arithmetic allows us to do this in one equation.

Such **visual iconic arithmetic** is shown in Figure 1. It can be viewed as a special type of **data fusion**, not just counting but a process of **combining data**.

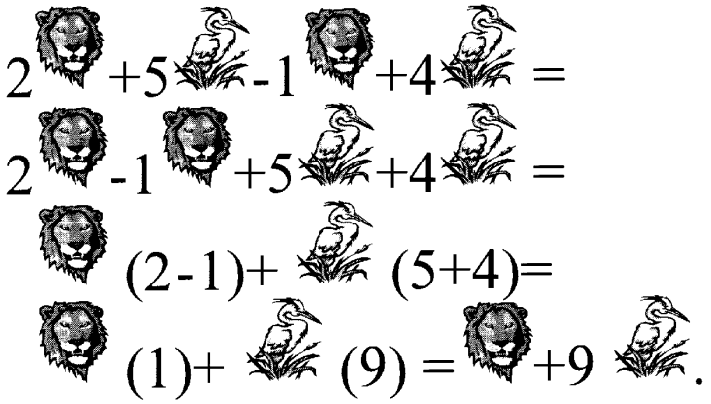


Figure 1. Multi-sort visual/iconic arithmetic

Similarly, human population dynamics, separated by gender, could be computed using male and female icons. One could further compute the total population by ignoring gender icons.

The same **fusion task** without visual iconic representation would require introducing explicit textual data types of the entities, for example: creature, animal, human, male and female. In physics, we set up physical measurement scales such as kg and sec. Physics rules then prohibit us from computing 2 km + 3 sec + 5 kg - 2 sec+ 16 kg as this summation involves different physical modalities and measurements. To be correct we have to compute separately sums for km, sec, and kg: (1) 2 km, (2) 3 sec - 2 sec, and (3) 5 kg + 16 kg. On the other hand, computing 2 km+ 3300 m + 500 m – 2 km + 160 m can be done correctly by converting all components to the same units, say km, 2 km+ 3.3 km + 0.5 km – 2 km + 0.160 km = 3.96 km.

Iconic language syntax permits us having a single equation for lions and birds by adding **shading.** Shading could then be interpreted as ignoring animal types. Obviously, it makes sense only if a meaningful **supercategory** such as animal or creature exists. The **ontology** of the domain provides us with a natural way of testing for the existence of a supercategory before trying to compute a result. Technically this ontological approach is implemented like the idea of semantic zooming that is described in detail in Chapter 10 (Section 2.2).

Semantic zooming in Figure 1 can convert both lions and birds to a general "creature" category as shown in Figure 2.

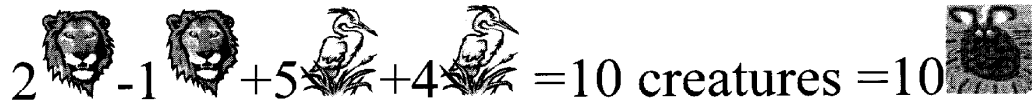


Figure 2. Iconic arithmetic with shaded data types

Having a special icon for this supercategory we can transform computation to a more standard form as shown in Figure 3.

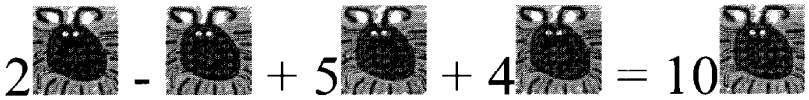


Figure. 3. Iconic arithmetic for a supercategry – creature

Figures 4 and 5 illustrate the same equations with different creatures and corresponding icons. Both examples are based on Egyptian hieroglyphs.

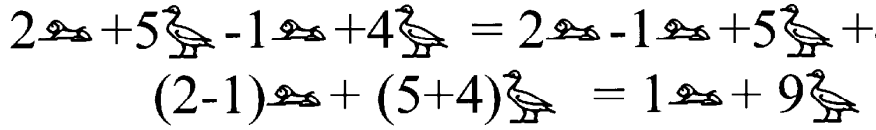


Figure 4. Equations with hieroglyphic icons

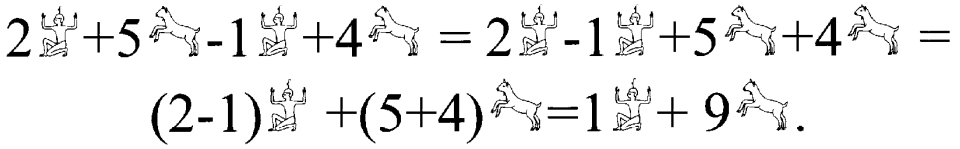
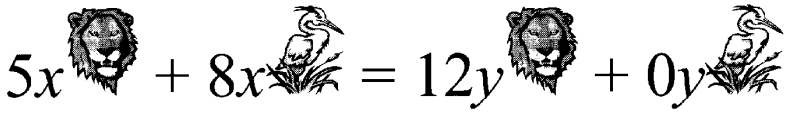


Figure 5. Equations with hieroglyphic icons

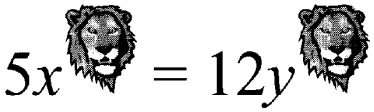
2.2. Visual multi-sort iconic equations and their solutions

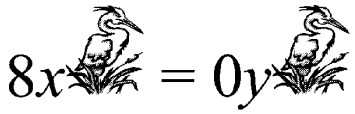
Consider an iconic multi-sort equation with unknown x:

What does it mean? We can interpret this equation as:



and construct two equations from it - one for lions and other for birds:





where icons could indicate a data type similar to our use of kg, sec, m, and $. For instance lion icon can be lion's weight, price or size of habitat.

In general, icons can be interpreted not only as **data type modality** indicators but also as indicators of **measurement units** similar to our use of sec, min, hour, day, week, month, and year.

In addition, we can use icons as indicators of **additional variables** *with a specific modality and measurement units*. For instance, the lion icon can be interpreted as amount of food consumed by a lion per day in kg. Similarly, the bird icon can indicate the same of bird. Obviously food for lions and birds is quite different even if its amount is expressed in common units such as kilograms. Ignoring these differences for many situations and tasks can be incorrect. At this stage we deliberately do not specify any of these interpretations for lion and bird icons in equations above. We try to use a purely syntactic manipulation with icons as long as possible and go to a specific interpretation only when further syntactic manipulation cannot continue without an interpretation.

The first equation with lions has many solutions. Namely, every pair (x, y) for which $x = (12/5)\,y$ is a solution. The second equation also has many solutions which take the form $(0, y)$. Here x and y could be the number of lions and birds, respectively.

This method is based on reduction of the multi-sort equation into two single-sort equations. An alternative approach would be to simplify the multi-sort equation before interpreting it by combining all lions and birds together:

$$(12y - 5x)\,\text{lion} = 8x\,\text{bird}$$

$$\text{bird} = (12y-5)/8x\,\text{lion}$$

If now one choose a particular case $x = y = 1$, we can write down this case as

$$\text{bird} = 7/8\,\text{lion}$$

How the last equation can be interpreted? It might represent a relation between prices. Consider for example, the case of an exotic bird and a wild lion. The interpretation could be that the price of the bird is 7/8 of price of the lion. Another interpretation could involve habitats. The size of bird's habitat could be 7/8 of the size of lion's habitat. In this case, lion and bird icons are interpreted as *additional variables*: variable a is price of a lion and

variable *b* is price of a bird. Thus the last equation can be rewritten in a standard way:

$$a=(7/8)b.$$

The iconic presentation is convenient because we *do not need to carry interpretations into the iconic equations*. We can operate with iconic equations syntactically just like classical algebraic equations. The interpretation can then be external as the semantic meaning of syntactical operations in iconic algebra. This follows the standard algebraic practice where abstract equations such as $3y = 15$ and $y = 15/3 = 5$ are used instead of interpretation specific equations $3y$ kg = 15 kg or $3y$ miles = 15 miles.

Currently, with an increasing amount of **heterogeneous, multimodal data** coming from a huge variety of different sources **multi-sort iconic arithmetic** can be helpful in **data fusion and integration**. Typically, at the beginning, the task is vague and we often do not know exactly what we want to do. For instance, we might want to operate separately with specific entities (e.g., birds and lions, or the female population) or with more general categores of entities (animals or humans). We might be uncertain about the *level of ganulatiry* that we need to carry. This is often because our goal is not yet clearly defined and our ability to reach such a goal is also uncertain.

Assume that we want to plant some crops in an area with a known total available size. We are uncertain about the planting plan. It could be a very detaied plan involving specific individual varieties of wheat, grass and cotton or much more generalized plan for the three categries: wheat, grass or cotton. In the latter case, the model would be much simpler and could use average costs of planting and other prices. The information available may not be sufficient for a more specific planning model but this may not be clear in advance. For instance, information may exist for individual crop varieties that turns out to be unreliable.

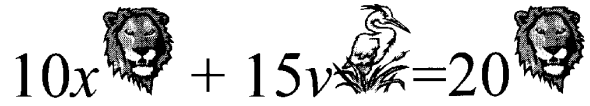
Syntactic manipulation with icons permits us: (1) postponing the exact task specification, (2) providing great flexibility for task specification and (3) avoid being stuck with some task specification prematurely.

The example below illustrates the use of iconic equations and syntactic manipulation without explicit interpretation of lion and bird icons and variables *x* and *v* in advance.

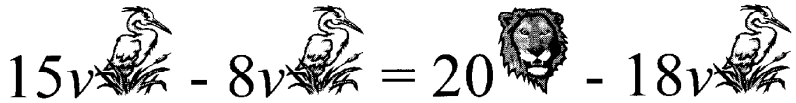
(1)

(2)

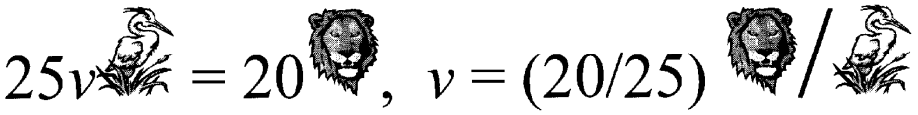
We can solve these two equations using an **iconic version of the classical Gaussian elimination** method. This iconic generalization can be done in a matrix form. Multiplying Equation (1) by 5 and Equation (2) by 2 we will get:



The subtracting the second equation above from the first one eliminates x.



After simplification this is equivalent to



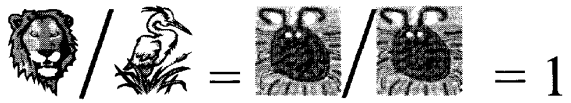
Now we need to interpret components of equations (1), (2) and the equation derived from them including the "ratio" of icons in the last equation. In both equations let x be the number of lions and v be the number of birds. Lion and bird icons are interpreted as additional variables a and b, that stand for prices of a lion and a bird.

Thus,  is a ratio R between prices for a lion and a bird. If R = 25, then the number of birds v is (20/25)25 = 20. In this interpretation, equation (1) means that the total price of $2x$ lions plus $3v$ birds is the same as the price of 4 lions. Equation (2) has a similar interpretation. In classical terms equations (1) and (2) are:

$$2xa + 3vy = 4a$$

$$5xa + 4vb = 9vb.$$

This means that exactly the same Gaussian elimination can be done in classical terms. The idea of iconic equations is that a human when formulating the problem to be solved can use icons to assist in problem formulation and initial reasoning. Then iconic equations can be converted to a traditional systems of equations and solved analytically. If we generalize birds and lions as creatures from the beginning then $R = 1$ and $a = b$.



This will produce another solution of the equations (1) and (2) that can be produced analytically in a regular algebraic way.

2.3. Systems of iconic equations and iconic linear programming

Further development of this approach permits us to write and *solve systems of iconic linear equations* and perform **linear programming** optimization tasks. Such optimization tasks can represent highly formalized decision-making tasks that are typically outside the capabilities of iconic visualization and the visual decision-making process.

Figure 6 illustrates this iconic problem solving approach with a goal of maximizing a linear objective function that includes lions and birds and satisfying two constrains given as inequalities, one for lions and another for birds.

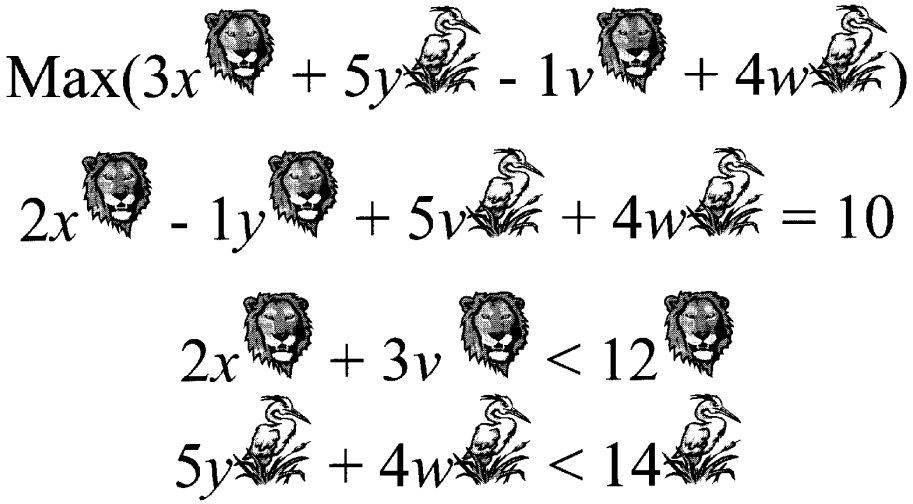


Figure 6. Multi-sort iconic linear programming task

The exercise section below contains several tasks that are open problems in iconic equations and iconic optimization, such as non-linear, discrete and stochastic iconic optimization problems.

3. CONCLUSION

This chapter has discussed the algebraic visual symbolism for problem solving and lessons learned from the history of algebraic equations from Diophantus to the present. It was shown that often the textual form of an equation is ambiguous and much more verbose than either modern or Diophantus' notations. These lessons from history led us to new concepts for iconic equations and iconic linear programming tasks.

We discussed several questions. How it was possible that, in spite of obvious advantages of algebraic symbolism, algebraic notation was not used for 1250 years after Diophantus invented it? Was this accidental? The conclusion was that this was not accidental. It was a result of the dominance of Euclidian geometric algebra and the fact that Diophantus' algebra operates with abstract entities that are not directly observable. Diophantus' algebra operates with visual but abstract concepts of unknown values, constants and arithmetic operations. Geometric algebra operates visually with concrete objects that have a direct match in the real world. Thus, geometric algebra was much easier to understand. It relies much less on abstract thinking. Geometry is closer to direct modeling of physical entities.

It was shown that the history of art provides valuable lessons in the same vane as the history of algebraic equations. Artists "compress" the space occupied by texts such as Bible, myths and proverbs. The important advantage of "reading paintings" vs. reading text is that we can see the "whole story" at once using our parallel visual processing. Thus, pictures compress the time required to "read" a concept in comparison with sequentially reading text. Multi-sort iconic equation representation is convenient because we do not need to carry interpretations into the iconic equations. We can operate with iconic equations syntactically similar to what is done in classical algebraic equations. Interpretation can be external as the semantic meaning of syntactical operations in iconic algebra. We also noted that with an increasing amount of heterogeneous, multimodal data coming from a huge variety of different sources multi-sort iconic arithmetic can be helpful in data fusion and integration. For decision-making tasks, iconic equations are important because they permit one to work with the high level of uncertainty that is natural for the initial stages of decision making and problem solving.

4. EXERCISES AND PROBLEMS

1. Try to solve a system of two equations $x + y = 5$ and $3x - y = 2$ in text. Record and compare your time for this solution with the algebraic sym-

bolic solution. Compare the amount space used to produce both solutions.

2. Design an iconic version of the Gaussian elimination method for solving systems of linear equations with three variables.

Advanced

3. Design an iconic version of the Gaussian elimination method for solving systems of linear equations with n variables in a matrix form.

4. Design an iconic version of the simplex method for solving the linear programming task with n variables.

5. Design an iconic version of a non-linear optimization problem with n variables.

6. Design an iconic version of a stochastic optimization problem with n variables.

7. Design an iconic version of a discrete optimization problem with n variables.

5. REFERENCES

Al-Khwarizmi, M. Al-jabr wa'l-muqabala, [English translation: Al-Khwarizmi, 1974.

Altshiller-Court, N. The Dawn of Demonstrative Geometry. Mathematics Teacher, 57, 1964, 163-166.

Cohen, M., Drabkin, I. Source Book in Greek Science. Cambridge, MA: Harvard University Press, 1958.

Geller, L. Start of Symbolism, 1998, http://www.und.nodak.edu/instruct/ lgeller/

Miller, J. Earliest Uses of Symbols for Variables, http://members.aol.com/jeff570/variables.html, 2001

O'Connor, J. Robertson, F. An overview of the history of mathematics 1997, http://www-gap.dcs.st-and.ac.uk/~history/HistTopics/History_overview.html#17.

O'Connor, J. Robertson, F. Diophantus of Alexandria, 1999, http://www-groups.dcs.stand.ac.uk/~history/Mathematicians/Diophantus.html

Parshall, K The art of algebra from Al-Khwarizmi to Viète: a study in the natural selection of ideas, History of Science, Vol. 26, No. 72, 1988, pp.129-164.

Parshall, Biography of Diophantus, 2002. http://www.lib.virginia.edu/science/parshall/diophant.html

Schroeder M. Number Theory in Science and Communication: with Applications in Cryptography, Physics, Digital Information, Computing, and Self-Similarity (3rd Ed)), Springer-Verlag, 1997.

Swift, J. Diophantus of Alexandria. American Mathematical Monthly, *63* (1956), 163--70.

Chapter 6

ICONIC REASONING ARCHITECTURE FOR ANALYSIS AND DECISION MAKING

Boris Kovalerchuk
Central Washington University, USA

Abstract: This chapter describes an iconic reasoning architecture for analysis and decision-making along with a storytelling iconic reasoning approach. The approach includes providing visuals for task identification, evidences, reasoning rules, links of evidences with pre-hypotheses, evaluation of hypotheses. The iconic storytelling approach is consistent hierarchical reasoning that includes a variety of rules such as visual search-reasoning rules that are tools for finding confirming links. The chapter also provides a review of related work on iconic systems. The review discusses concepts and terminology, controversy in iconic language design, links between iconic reasoning and iconic languages and requirements for an efficient iconic system.

Key words: iconic reasoning architecture, analysis and decision-making, storytelling iconic approach, iconic language.

1. INTRODUCTION

The goal of an **iconic evidentiary reasoning (IER)** is to convey complex multi-step analytical reasoning and decision making in a more efficient and condensed way than traditional text reasoning. IER is defined as a *visual support mechanism* for the following general problem-solving steps:

- defining the *problem*,
- generating initial alternatives for hypotheses that are called *pre-hypotheses*,
- linking pre-hypotheses with *evidences*, and
- generating *hypotheses* by evaluating pre-hypotheses against evidences.

Typically these steps are repeated several times in the process of refining hypotheses and evidences in scientific research as well as in intelligence analysis, engineering and architectural design, market analysis, health care, and many other areas.

To explain the IER approach we use a modified task of ongoing monitoring of the political situation in some fictitious country. The final reasoning result condenses all of the most important visuals from analytical steps in a single compact picture. This picture combines several types of icons and arrows that indicate a conclusion, its status and a chain of evidences that support the conclusion. For presentation purposes the final part of the reasoning chain can be enlarged and the icons replaced by actual maps, imagery and photographs of people involved.

Major components of the IER architecture are:

- Collecting and annotating analytical reports as *inputs* using a markup language, e.g., XML, DAML;
- Providing iconic representation for *hypotheses*, *evidences, scenarios*, *implications*, *assessments*, and *interpretations* involved in the analytical process;
- Providing iconic representation for *confirmations*, and *beliefs* categorized by levels;
- Providing iconic representation for evidentiary *reasoning* mechanisms (propositional, first-order logic, modal logic, probabilistic and fuzzy logics);
- Providing scenario-based visualization and visual discovery of changing *patterns and relationships;*
- Providing a condensed version of iconic representation of evidentiary *reasoning* mechanisms for *presentation and*

These components are depicted in Figure 1. The use of iconic visuals permits a user to reach a high condensing ratio level. Experiments reported in Chapter 10 show that iconic sentences can occupy space that is 10 times smaller than space occupied by text, that is a compression factor of 10 is possible. Also people can work with *multidimensional icons* two times faster than with text [Spence, 2001]. A similar time and space compression is expected to communicate analytical results (including underlying reasoning) to decision-makers and fellow analysts using IER. Moreover at some moment with such advantages and the ongoing proliferation of visualization technology, iconic reasoning can become a **major way of reasoning and communication** in general. The visual correlation approach described in chapters 8-10 can be naturally combined with visual reasoning to improve problem solving.

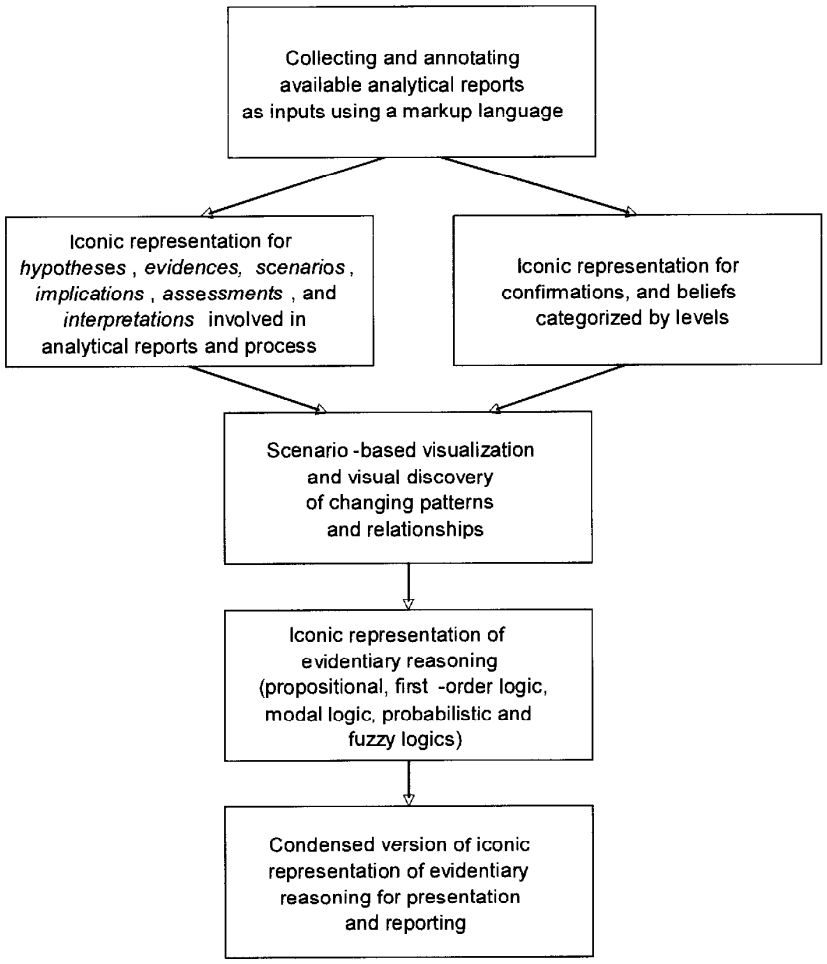


Figure 1. Iconic evidential reasoning architecture

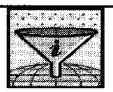
2. STORYTELLING ICONIC REASONING ARCHITECTURE

2.1 Task identification: output characteristics and pre-hypotheses

In this section we discuss how the process of defining the problem and generating pre-hypotheses can be done visually. The *problem* is defined as *ongoing monitoring of a political situation* in some fictitious country [AQUANT, 2002]. This may include identification of the country, and selection of processes to be monitored such demographic, economic, democratic processes, research & development and military activity. Assume that the user selected a task of monitoring democratic processes and identified overall output characteristics to be evaluated as one of the judgments: positive

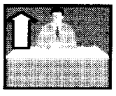
change, no significant change, negative change, mixed change at this moment. In IER task identification is also done using an iconic user interface where the user picks up the task from the iconic menu of tasks and characteristics as shown in Table 1.

Table 1. Task identification

Alternative tasks	*Select icon*	*Selected icon*
Monitoring democratic processes		☑
Monitoring economic processes		☐
Monitoring military activity		☐
Monitoring research & development processes		☐
Monitoring demographic processes		☐

Next the user selects output characteristics to be monitored from the menu provided in Table 2.

Table 2. Defining the output characteristics

Characteristics	*Select icon*	*Selected icon*
Description of a new process		☐
Change direction (positive, no change, negative, mixed)		☑
Rate of change (low, medium, high)		☐
Description of an emerging leader		☐

After that the user picks up a pre-hypothesis from the menu for the selected task. The menu contains all logically possible alternatives for changes in country Y in the selected scale shown in Table 3.

Table 3. Pre-hypotheses and their legends. See also color plates.

	Pre-hypotheses	*Legend 1*	*Legend 2*	*Selected icon*
H_1	Change is positive in country Y at this time (green shapes)			
H_2	No significant change in country Y at this time (gray shapes)			
H_3	change is negative in country Y at this time (orange shapes)			
H_4	change is mixed in country Y at this time (orange/green shapes)			

Icons in Table 3 show alternative legends for pre-hypotheses. For instance, mixed change alternative (H_4) is presented visually in three ways: as a rectangular with orange and green components (green indicates a positive change, and orange indicates a negative change. A two-color flag and two flags express the same idea a little bit differently. Analysts have an option to select an icon legend that best fits her/his preferences and perceptual abilities.

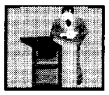
2.2 Visual evidences

In this chapter, we assume that visual evidences are already collected. They also can be already encoded in a predicate form or in XML form that can make automated iconization of them easier. Table 4 presents examples of evidences.

Evidence E_{11} states that a new person that supports democracy is in power. This evidence is depicted by the icon of an official with a positive, green background. A speaker with a neutral, gray background is an icon for evidence E_{12} that there are no new indications of suppressing free speech. Altering the color in the icon for the first evidence produces an icon for evidence E_{21}, that there are no indications that new people with alternative views are in power.

Further alternation of the color in the first icon along with two dots produces an icon for evidence E_{31} where orange background is interpreted as negative with icon meaning that several new persons that oppose democracy are in power.

Table 4. Evidence and iconic representations See also color plates.

	Evidence	*Icon description*	*Icon*
E_{11}	A new person that supports democracy is in power	An official with a *positive, green* background	
E_{12}	No new indications of suppressing free speech	A speaker with *a neutral, gray* background	
E_{21}	No indication that new people with alternative views are in power	An official with *a neutral, gray* background	
E_{31}	Several new persons that oppose democracy are in power	An official with *negative, orange* background for opposition to democracy and two dots for "several" officials	
E_{32}	New indications of suppression of free speech	*Orange* background encodes negative fact- suppression of free speech	
E_{41}	A new persons that oppose democracy is in power	An official with *negative, orange* background for opposition to democracy	
E_{42}	New indications of free speech	*Green* background encodes *positive* fact- free speech indications	

The color language and icon content used for icons in Table 4 is described in Table 5. This language is easy to learn. It had only two iconic elements (iconels) on content (a speaker and an official), three colors and presence and absence of dots for quantity providing total 3*2*2=12 icons.

Table 5. Evidence encoding legend

Icon element (iconel)	*Semantic indicator*
Green background	positive change, positive statement
Gray background	neutral change, positive statement
Orange background	negative change, positive statement
Person sitting at the desk	Official in power
Person giving a talk	Speech
Two dots	Several people

2.3 Visual reasoning rules

Evidences provided in section 2.2 can be combined in if-then reasoning rules such as shown in Table 6 in both a natural language and in a formal logic. The first rule "If a new person that supports democracy is in power (E_{11}) and no new indications of suppressing free speech (E_{12}) then positive

change in country Y (H_1) is possible (with some confidence)" has its formal equivalent, E_{11}& $E_{12} \Rightarrow_{possible} H_1$.

Table 6. Reasoning rules, templates

	Natural language rules for the hypotheses H_i	*Rules*
R_1	If a new person that supports democracy is in power (E_{11}) and no new indications of suppressing free speech (E_{12}) then positive change in country Y (H_1) is *possible* (with some confidence).	
	IF Evidences E_{11} and E_{12} take place then H_1 (positive change) is *possible*	E_{11}& $E_{12} \Rightarrow_{possible} H_1$
R_2	IF Evidences E_{12} and E_{21} take place then H_2 (no significant change) is *highly probable*	E_{12} & $E_{21} \Rightarrow_{highly\ probable} H_2$
R_3	IF Evidences E_{31} and E_{32} take place then H_3 (negative change) is *true*	E_{31} & $E_{32} \Rightarrow_{true} H_3$
R_4	IF Evidences E_{11} and E_{32} take place then H_4 (mixed change) is *true*	E_{11} & $E_{32} \Rightarrow_{possible} H_4$

These rules can be visualized in different visual forms. Figure 2 shows visualization of rule R_1 in two graphical forms. The first line shows the rule in an **abstract block diagram paradigm**. The significant difference from the classical formal logic here is in the use of the colors to indicate positive (green) meaning of the terms E_{11} and H_{12} and neutral (gray) meaning of the term E_{12}. The second line presents the **iconic storytelling paradigm** where terms E_{11} and E_{12} are presented as icons with the same green and gray backgrounds. In both forms an arrow indicates inference, where P stands for possible (modal logic operator). The iconic form reveals more information than abstract one and it is more appealing perceptually.

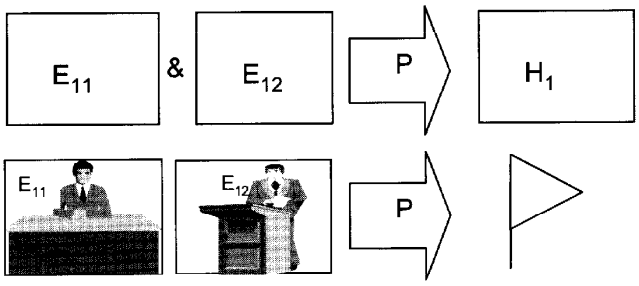


Figure 2. Traditional and iconic visualizations of rule R_1. See also color plates.

Figure 3 shows inferences for other rules R_2, R_3, and R_4 from Table 6. The middle line reveals firm reasons (black arrow) for an orange flag -- "several new persons that oppose democracy are in power" and there are "new indications of suppressing free speech". Thus, a black arrow indicates sure conclusion (true statement). Thus, orange line of reasoning in the middle row immediately and preemptively indicates a negative line of events. Similarly a

gray line indicates a neutral line of reasoning and conclusion. A mixed color line indicates a mixed conclusion. Similarly, to Figure 2 this figure illustrates that the iconic form conveys more information, is more appealing and permits to convey a reasoning statement easier.

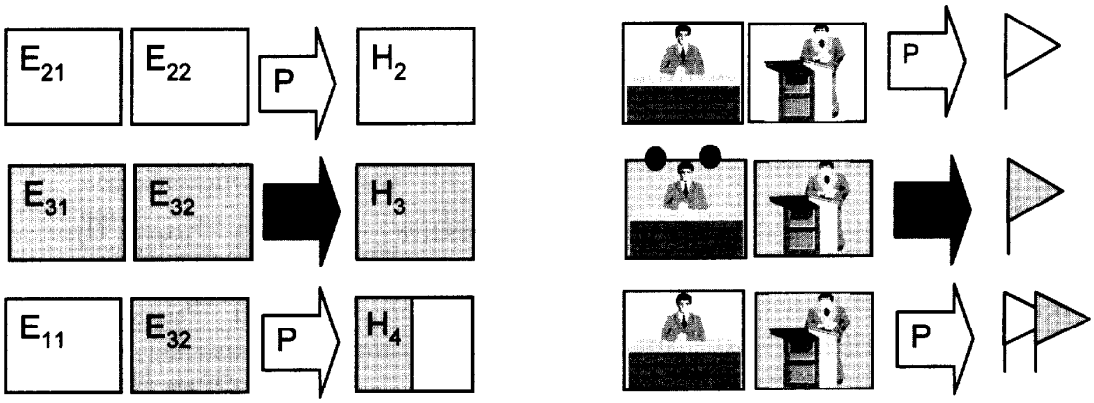
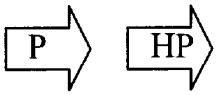


Figure 3. Traditional and iconic visualizations of rules. See also color plates.

Table 7 presents some arrow icons used in IER. Arrow icons have a hierarchy, that is if we want only to encode that fact that the result is possible we can use the first icon in Table 7, but if we want to encode the possibility more specifically we can use text markers such as HP and LP for highly possible and low level of possibility. Another option is use of partially filled arrows to identify the level of conclusion certainty as shown in Table 7.

Table 7. IER selected arrow icons

Icon	*Interpretation*
	P - Possible conclusion HP - Highly possible conclusion
	Sure conclusion (classical logic true)
	Conclusion with 50:50 chances
	A search which may yield nothing, a correct result or an incorrect result
	Conclusion based on the use of a Device
	Conclusion based on a Human judgment

2.4 Linking evidence and pre-hypotheses using reasoning rules

Next we link available evidences and a pre-hypothesis using reasoning rules as matching templates. This is a first step for evaluating pre-hypotheses and finding plausible hypotheses. If a pre-hypothesis H_1 is selected and evidences listed in table 4 are available, then we look through rules listed in Table 6 to see if some of them match hypothesis H_1 with at least some of available evidences. Rule R_1 is such a rule, since it matches H_1 with E_{11} and E_{12}:

$$E_{11} \,\&\, E_{12} \Rightarrow_{\text{possible}} H_1.$$

To find matching rule R_1 we can compare texts in Tables 3 and 4 or icons in Figure 2, Tables 3 and 4. Comparing icons has some advantages because they contain more information than symbols E_{11} and E_{12} and represent their meaning more directly.

2.5 Pre-hypotheses evaluation against evidence using visual rules

Any logically possible alternative is a pre-hypothesis but only some of them are meaningful hypotheses (or hypothesis for short). We assume that *meaningful hypotheses* are those pre-hypotheses that (i) have some confirmation by evidences and opinions or (ii) it is expected that such confirmation by evidences or opinions can be found. Thus, we differentiate pre-hypotheses and hypotheses conceptually.

In the example above, rule R_1 matched pre-hypothesis H_1 and available evidences. This match happened to be complete, that is every term in the if-part of the rule R_1 has been found in the database (Table 4). This creates the base for evaluation of the pre-hypothesis H_1 as "possible" according to the description of rule R_1 presented above. It is easy to see that here we followed a rule-based approach typical in knowledge-based reasoning systems. The significant difference is that we can accomplish this reasoning by iconic means by human and automatic iconic search.

3. HIERARCHICAL ICONIC REASONING

In the previous consideration we assumed that evidences listed in Table 4 are already given, but in fact they often need to be established. Let us consider evidence E_{11} = "a new person that supports democracy is in power" as a **candidate evidence**. Actually, this statement is a new hypothesis H_{11} of

lower *level 2*. To confirm or refute it we introduce and visually represent using icons two new evidences E_{111} and E_{112} shown in Table 8. These evidences are of the next level 2. Reasoning rules for evidences of this level are shown in Table 9. Rule R_{13} is a conjunction of two rules R_{11} and R_{12}.

Table 8. Hypotheses and their legends. See also color plates.

	Evidences (level 2 hypotheses)	*Icon*
E_{111}	a new pro-democracy person X stands next to the president in a recent photograph (partial green background encodes positive change)	
E_{112}	A new pro-democracy person X is appointed to the cabinet as the national security advisor (*green* background and *arrow up* encode positive change and high rise)	

Table 9. Reasoning rules, templates

	Natural language rules for the hypotheses H_i	*Rule*
R_{11}	A new pro-democracy person X stands next to the president in a recent official photograph If E_{111} then H_{11}	$E_{111} \rightarrow H_{11}$
R_{12}	A new pro-democracy person X is appointed to the cabinet as the national security advisor If E_{112} then H_{11}	$E_{112} \rightarrow H_{11}$
R_{13}	A new pro-democracy person X stands next to the president in a recent official photograph and a new pro-democracy person X is appointed to the cabinet as the national security advisor If E_{111} & E_{112} then H_{11}	E_{111} & E_{112} $\rightarrow H_{11}$

A new level 2 reasoning rule R_{13} (If E_{111} & E_{112} then H_{11}) is shown visually in Figure 4. Now keeping in mind that $H_{11}=E_{11}$ we can visually combine reasoning steps from Figures 2 and 4 to produce a **reasoning chain** (see Figure 5).

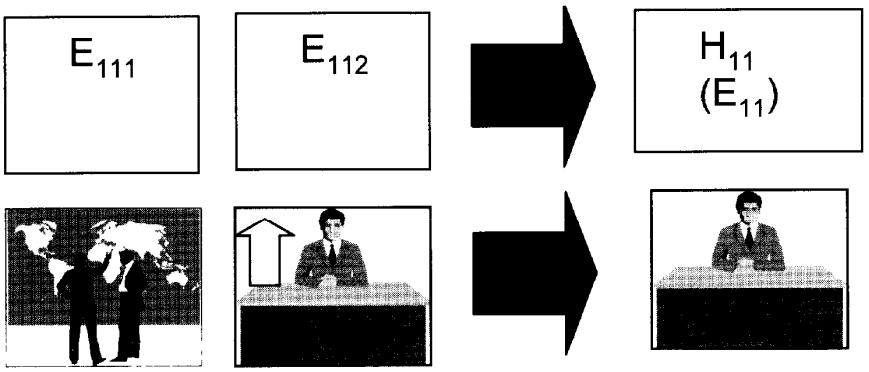


Figure 4. Comparison of two visual reasoning alternatives. See also color plates.

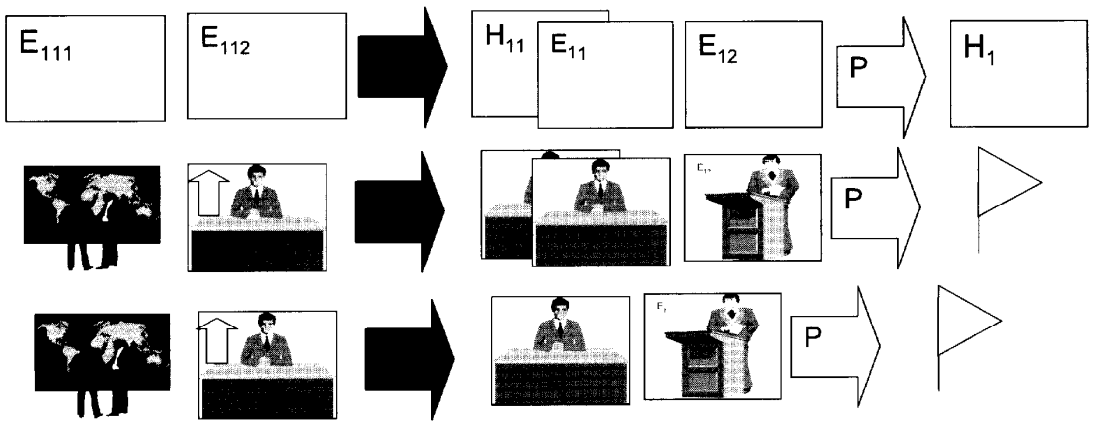


Figure 5. Reasoning chains. See also color plates.

In this figure the first line shows the match found between H_{11} and E_{11} in rules visualized by partial overlapping the blocks for H_{11} and E_{11}. Similarly, the second line matches icons for H_{11} and E_{11} by overlapping them. The third line shows a completed match with a full overlapping of matched H_{11} and E_{11} icons. A user can do this visually by dragging one icon over another one and animate the process and the result.

Dragging is an additional intuitive element of visual reasoning. It is also possible in abstract reasoning (first line in Figure 5), but it requires remembering that H_{11} and E_{11} are the same. In contrast icons reveal similarity of these concepts *instantly*. Figure 5 makes the reasoning chain evident and easy to communicate. The first step of reasoning is firm (black arrow), but the second one is only possible (arrow with letter P). Having a longer reasoning chain or a tree an analyst and a decision maker can quickly see the most questionable reasoning steps that may need more attention.

4. CONSISTENT COMBINED ICONIC REASONING

In this section we elaborate the process of combining iconic rules in more detail. As we can see, iconic rules combine evidences and hypotheses uniformly. We start from visualizing reasoning that combines two visual rules to produce a new rule as shown in line 1 in Figure 6.

The first rule is "If a new person that supports democracy is in power (E_{11}) then positive change in country Y (H_1) is possible". The second rule is "If there are new indications of free of speech (E_{42}) then positive change in country Y (H_1) is possible". The produced rule on the right is "If a new person that supports democracy is in power (E_{11}) and there are new indications of free of speech (E_{42}) then positive change in country Y (H_1) is possible".

This is a rule R_1 from Table 6. Such conjunction is generic for combing any rules and we will call it a **conjunction metarule**.

We can also generate another compact visual reasoning rule shown in line 2 in Figure 6. This visual rule shows that if a mixture of "gray" and "green" properties implies a positive change then a consistent analyst should accept that two "greens" also should imply a positive change. This rule is based on principle of **monotonicity**. The storytelling visual rule is much *shorter and intuitively clearer than text* of this rule:

IF (If a new person that supports democracy is in power (E_{11}) and no new indications of suppressing free speech (E_{42}) then positive change in country Y is possible)

Then (If a new person that supports democracy is in power (E_{11}) and there are new indications of free speech (E_{12}) then positive change in country Y is possible)

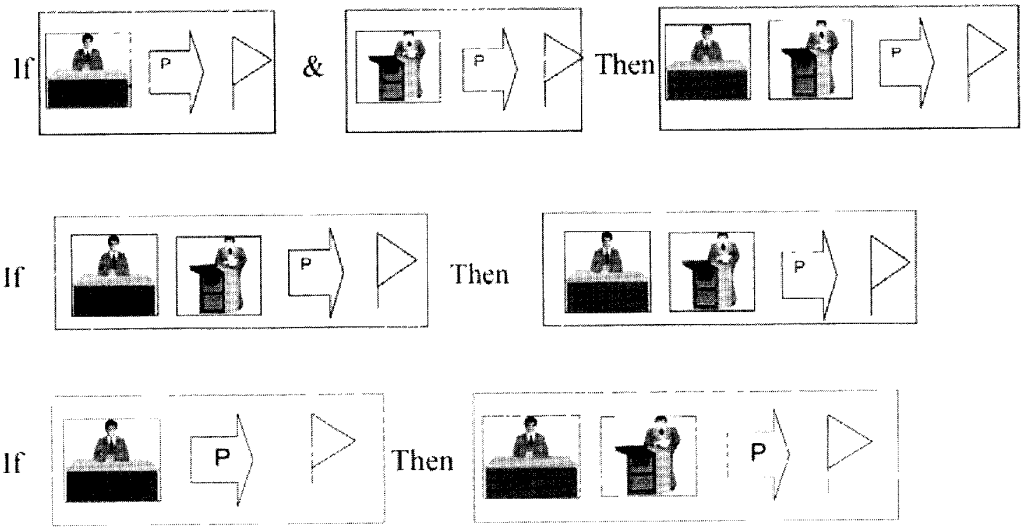


Figure 6. Visual reasoning rules. See also color plates.

The importance of this monotonicity rule is in the fact that we do not need to write this rule in advance. We can generate a specific form of this rule automatically using the principle of monotonicity. This metarule (rule applied to rules) will be called **monotonicity metarule**.

In the same way another short visual rule is generated in line 3 of Figure 6. It can represent analyst's opinion: "If a positive change is possible because of a pro democracy person is in power then positive change is possible even if there is no progress in free speech". A visual presentation of this statement reveals its structure clearly and is shorter. This rule also has its metarule counterpart – **neutral metarule** –adding a neutral statement (with & operator) does not change reasoning result.

4.1 Visual search-reasoning rules

The process of searching for other candidate evidences can bring us to the lower level hypotheses similarly to the discussion above. In this process, candidate evidences are considered as pre-hypotheses and new evidence candidates for them are generated and visualized on the next *third level.* We are making search an explicit part of the reasoning process by introducing **search rules** such as rule R_{111} shown in line 2, Figure 7:

> If the name of X is known then **search** in the list of foreign chiefs for this name and if found retrieve the post occupied.

Thus, this approach visualizes **integration of declarative and operational knowledge,** where search rules represent an operational knowledge. Let us assume that search produced the following result -- Mr. X is a national security advisor in country Y. The line of reasoning that produced this result can be expressed visually by rule R_{111} shown in the second line in Figure 7.

The textured arrow indicates that the search *result can be incorrect or not guaranteed.* For instance, the name may not be in the search list or the list contains the name, but it is another person with the same name.

Let us assume that (1) we have a *candidate evidence* E_{111} ="A new pro-democracy person X stands next to the president in the recent official picture"; (2) the analyst has found a photograph with a new person that stands next to the president during a recent visit to a foreign country, and (3) the analyst does not know who is Mr. X.

If the name is not known we need a reasoning rule of the next *forth level.* For instance we may have rule R_{1111}:

> If name is not known then *run face recognition software* (FRS) of the selected face against all annotated images available from country Y.

Figure 7 provides visuals for this reasoning by depicting rules R_{1111} and R_{111} used sequentially.

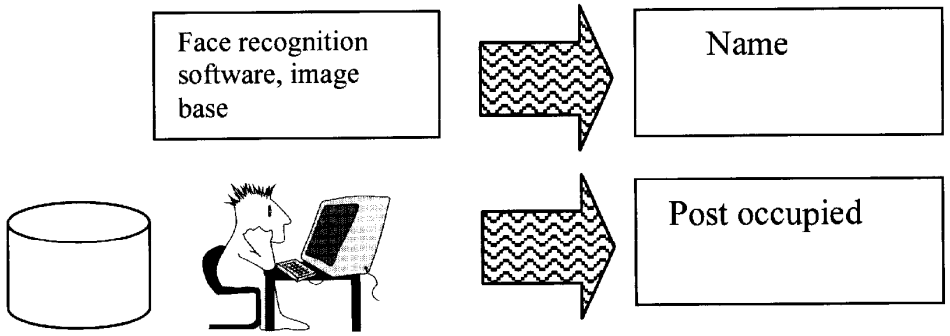


Figure 7. Search rules

4.2 Search for confirming links

Now we need to check that Mr. X is pro democracy as stated in evidence E_{111}, his name should be in Dem.Name file from an independent source. This request is presented as a visual rule in the line 1 in Figure 8. We may also have a negative rule R_{1112} depicted in line in Figure 8:

> If searching for political opinion of an official in non-democratic country Y then do not rely on local official media, search independent data.

The crossed search arrow in line 2 indicates the negative rule, not to search using local media.

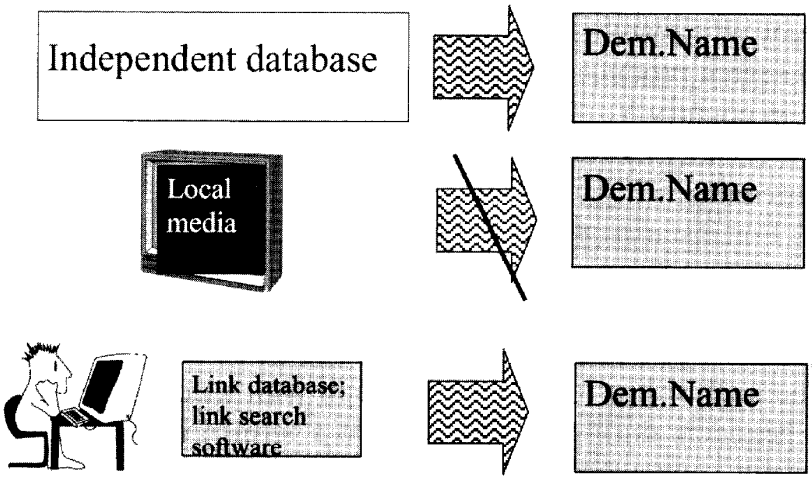


Figure 8. Search rules with negation

Now if the independent database is not available we apply another level 4 rule R_{1113}:

> If searching for political opinion of an official in non-democratic country Y then **search for confirming links** of Mr. X.

This rule is depicted in line 3 in Figure 8 as a green "link" block that requires running interactive link analysis software. The search is indicated by the search arrow. Link analysis software found a telephone call from country Y to Mr. X's home in 1998. A caller (Mr. W) confirms that Mr. X is pro democracy ("yes" callout in visual representation). This rule is shown in Figure 9, where letter "H" in the arrow indicates a confirmation from a human.

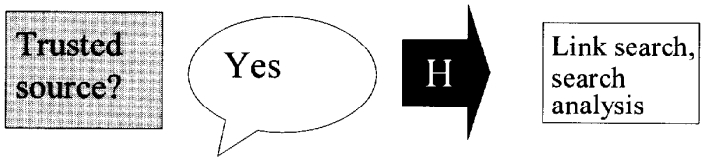


Figure 9. Visual source quality rule

Mr. W is a trusted source (green block "trust"). To make this conclusion we use a lower level rule such as rule R_{111211}:

If Mr. W was tested using a polygraph successfully before then X can be trusted.

Figure 10 shows this as a visual reasoning rule, where "D" in the arrow means "*confirmed*" by a device. Thus, Figures 8-10 visualize logic of search.

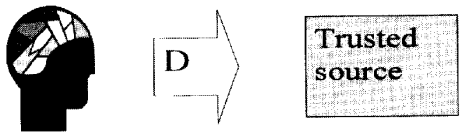


Figure 10. Device-based source quality rule

4.3 Integrating visual reasoning components

Now successful reasoning steps can be combined into a single visual evidentiary reasoning scheme (Figure 11). This single picture shows all eight reasoning steps, levels of fidelity of conclusions, and points of linking of individual reasoning steps. For instance, it shows that only three steps out of eight steps are firm in final conclusion about positive change in country Y.

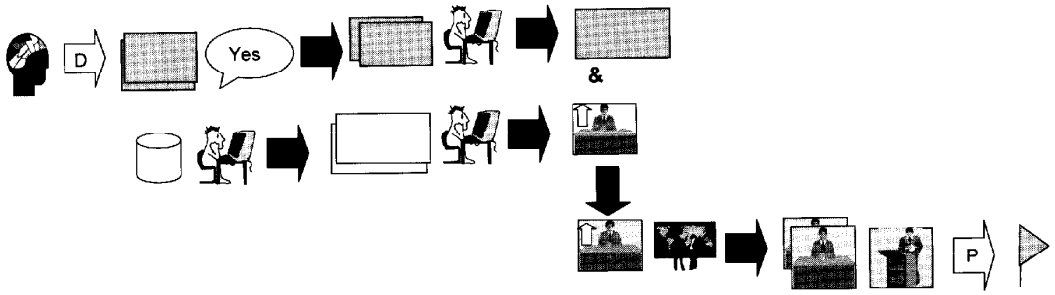


Figure 11. Integrated visual evidentiary reasoning scheme. See also color plates.

Three conclusions have no guarantee in conclusion (they came from search). One conclusion came from a device and one came from a human.

This picture can also be augmented with more elaborated levels of confidence of conclusions, their contradiction and quality of the source.

The example shows tight integration of three stages: Storytelling Iconographic Visualization, Collecting Information, and Evaluating Hypotheses. An analyst is able to see from visualizations similar to shown in Figure 11 a current status of the analysis, which can show that only few hypotheses have been tested and tested against only a small number of evidences included in reasoning rules. An appropriate part of this visualization can be converted into a visual report to decision makers.

More complex hypotheses require more complex and dynamic icon development. Ideally iconic representation should be automated. This subject is discussed in Chapter 10.

4.4 Visual reasoning for handling signal uncertainty

Similarly visual evidentiary reasoning can be applied to improve handling signal uncertainty in identifying location of a signal source using visualization combined with probabilistic and fuzzy logics. A sketch on Figure 12 illustrates a type of visual reasoning and presentation that is applicable here.

A traditional approach uses ellipses to convey uncertainty of location based on radar information [Mikulin, Elsaesser, 1995] with simple ellipses. A more elaborated visual representation provides additional information: probabilistic distribution of individual fixes (see Figure 13).

A visual reasoning technique conveys additional information: probabilistic distribution of individual fixes and their mixtures. Areas with higher values of a distribution function have more points rendered.

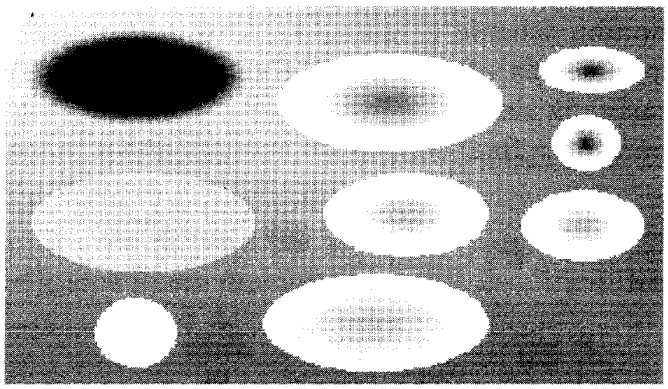


Figure 12. Signal ellipses. See also color plates.

Figure 13 shows other alternatives to visualize uncertainty of location using iconic approach. It permits to convey visually, in a condensed way and quickly the differences between alternative combinations of fixes, where lines 2 and 3 visualize the traditional ellipsoid approach.

This visualization also permits naturally convey a mixture of distributions as shown on the line 4 in Figure 13. Ellipses from Figure 12 also can be used in such visual reasoning.

Similarly this approach can accommodate in visual reasoning new developments in reasoning about radar emitters such as "what-and-where" fusion for recognition and tracking of multiple radar emitters based on a neural network learning technique [Granger, Rubin & Grossberg, 2001].

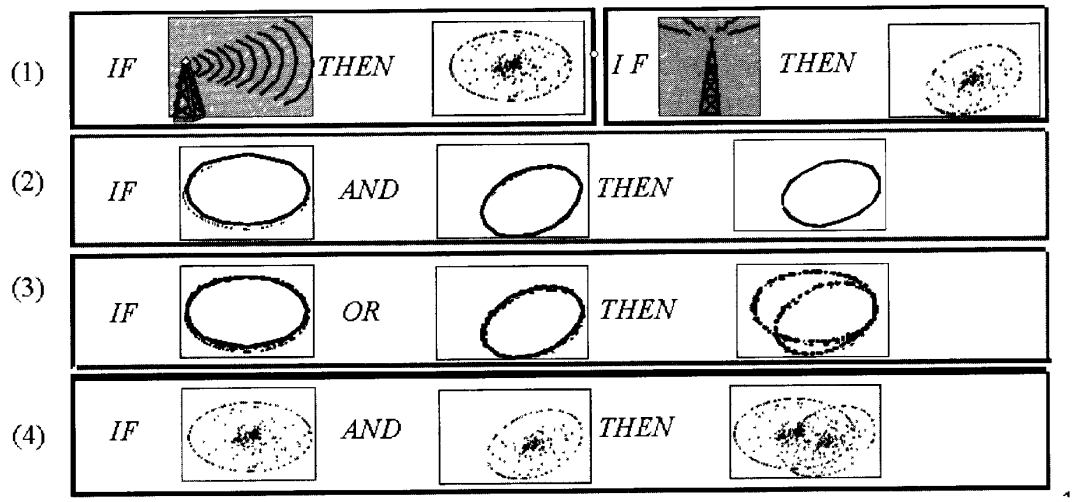


Figure 13. Visual rules with signal ellipses. See also color plates.

5. RELATED WORK

5.1 Concepts and terminology

Data representations are characterized by several dimensions [Sloman, 1995]. Two dimensions are important in the context of visual reasoning: formal-natural and verbal-visual [Nadin, Küpper, 2003]. Frixione et al. [1997] discuss the role of **image-like representations** in the computational modeling of mental processes in Artificial Intelligence and Cognitive Sciences. Authors trace earlier research starting from the theorem proving machine [Gelerntner, 1959], the use of **diagrammatic representations** in problem solving proposed by Funt [1980], accounts of mental imagery in terms of pictorial representations [Kosslyn, 1980], and "opposition" between image-like representations and more linguistically oriented approaches [Block, 1981]. Later revival studies moved to **blending** features of images, diagrams, and propositional systems, e.g., [Chandrasekaran, Simon, 1992; Gardin, Meltzer, 1989; Kulpa, 1994]. **Iconic reasoning** is a term actively used now, e.g., [Frixione at al., 1997]. This subject is also presented in Chapter 3. Often tasks are very different and the term is interpreted differently. **Iconic communication** is another umbrella term [King, 1999; Yazdani, Barker, 2000] with focus on developing iconic languages for human communication. From our viewpoint the major difference between iconic communication and iconic reasoning is not in the subject (icons) but in the application area: *problem solving* (e.g., robot navigation) for iconic reason-

ing and *human conversation* for iconic communication. Note the term "iconic communication" itself is more general than its scope in current research. Thus, we suggest using this term as an umbrella term for both areas. A variety of icons are developed for both purposes [Dreyfuss, 1972, 1984].

The term an **iconic language** is another term with different meanings. Valiant at al. [1995, 1997] define an iconic language as a language with *absence of a specified syntax*. In contrast, **iconic programming languages** are defined with a specified syntax.

Pierce categorized the patterns of meaning in visual signs as iconic, symbolic and indexical as shown in Table 10 based on [Moriarty, 1995; Nadin, Küpper, 2003].

Table 10. Peirce's sign categories

Sign	*Description*	*Example*
iconic sign	looks like, resembles what it represents	picture of a dog; garbage can icon
indexical sign	a clue that links or connects things in nature; the marks left by an object	smoke as a sign of fire; icicles as a sign of cold; fingerprint, wind arrow as marks left by an object
symbol	meaning is determined by convention	the US flag; the Statue of Liberty; Roman and Hindi-Arabic numbers

More terms based on [King, 1999; Valiant, 1997] are described in Table 11.

Table 11. Sign terminology

Sign	*Description*	*Example*
natural sign	a universally intelligible iconic or indexical sign	a picture of a dog; icicles as a sign of cold
abstract sign	a symbol, conventional sign that have to be learned	Hindi-Arabic numbers
ideogram	an icon that encodes an idea/concept	Bliss alphabet [Bliss, 1965]
figurative icon/picture	a metaphorical icon /sign	the Statue of Liberty
semem	a traditional text message made up of linguistic entities in a natural language	any text in English

This is not a complete set of the iconic terms used. For instance, *semiom* is defined as a message composed with icons which do not necessarily match up to linguistic entities. Icons that express predicates are called *predicative icons*. *Single word icons* can be expressed with a single word in a natural language and *multiple word icons* correspond to more than one word in a natural language

5.2 Controversy in iconic language design

Blissymbolics [Bliss, 1965] is an example of a sign system that was driven by comprehensibility of signs; semantics of signs intends to capture a deep meaning of a concept depicted in the icon. A roman number system is another attempt to create a comprehensible sign system. Compare Roman numbers three (III) and eight (VIII) with the same Hindu-Arabic numbers 3 and 8. The first two are more "natural" (III directly shows three units "I") and last two (3 and 8) are abstract, conventional signs without a direct link between their appearance and contents. History made its choice for Hindu-Arabic numbers, in spite of the fact that they are not "natural", but they have advantages indicated in [King, 1999]:

- encode a complex concept in a simple sign (8 is simpler than VIII),
- support arithmetic simpler than Roman numbers,
- can be easily learned and distributed (compare 1999 and its Roman equivalent),
- supported by billions of users around the Globe.

Two related extreme claims are listed in [King, 1999]: (1) all sign systems and methods of representation are *inherently arbitrary* and (2) pursuing intrinsic comprehensibility of a sign is a chimera.

Cruickshank and Barfiel [2000] develop an approach that augments textual communication with **user-created icons**. The authors argue that this approach can overcome difficulties of alternatives such as Bliss symbology intended to *replace* textual communication by using a *fixed iconic vocabulary.*

> "Bliss faltered and ultimately failed because people are generally not prepared to learn and communicate with a rigid new language, syntax and grammar; the required investment in time and understanding outweighs the potential benefits" [Cruickshank, Barfiel, 2000].

It is suggested that a modern symbol system must be able to grow. This is especially useful for attempts to create a *universal* iconic language that can augment text in many possible situations of human communications (e.g., e-mail). However, this ambitious goal is not necessarily the goal of every problem solving iconic communication. For instance, a music symbolic language, digital logic, military and traffic symbologies are sophisticated iconic languages but are not intended to be universal and hardly can be produced quickly in the course of communication itself. Such languages should be consistent and have relatively lower ambiguity in contrast with a language for an unrestricted domain. These examples also illustrate the difference between two research focuses: problem solving and conversation between people.

5.3 Iconic reasoning and languages

Typically visual reasoning models belong to one of two categories: logic-based "analogical" models and hybrid models [Frixione et al., 1997]:

- Logic-based "analogical" models use visual representations isomorphic to their logical models [Levesque 1988; Johnson-Laird 1983, Fauconnier 1985], and
- Hybrid models combine logical rule-based modules with diagrammatic representations [Forbus 1995; Myers, Konolige 1995, Barwise, Etchemendy, 1995].

Often these reasoning models have computational advantages over reasoning models based solely on propositional representations without visual components. Sometimes these models also are more expressive.

The mathematical base of visual reasoning comes from classical logic, probabilistic reasoning, fuzzy logic, possibilistic logic. Recently description logic (terminological logic) with fuzzy logic components was added [Lutz, 2003; Straccia, 2001].

Modern approaches in iconic languages start from Isotype [Neurath, 1978] and Semantography [Bliss, 1965] developed in 1920-1940s. A recent related bibliography is presented in [Camhy, Stubenrauch, 2003].

Below we comment on a few recent iconic languages. Computer Assisted Iconic Language System, CAILS [Champoux, 2001] produces "iconic message objects". It deals with visual/spatial concept representation with specific syntax. Visual references or "words" are classified in the following categories: Hands, Movements, Expression, and Pictures. CAILS's grammar contains six conjunctions: standard complementizer (that), implicative (if), antecessive (because), concessive (but) and connectives (and / with) shown in Figure 14.

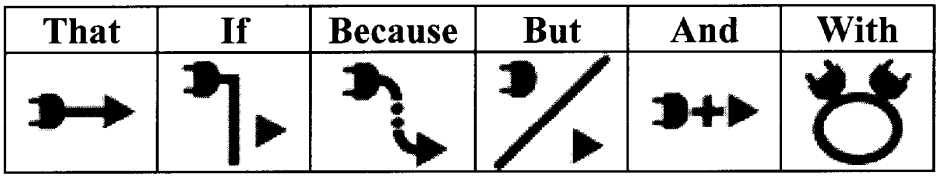


Figure 14. CAILS's conjunctions symbols [Champoux, Fujisawa, Inoue, Iwadate, 2000]

A system based upon a set of dynamic visuals with qualitative reasoning about information displayed within a document is known as Context Transport Mark up Language, CTML [Tonfoni, 1996,1998-2001].

An iconic communication system to assist a user to construct sentences, without typing them in words, i.e. solely relying on icons is called Visual Inter Language, VIL [Becker, 2000]. The goal of VIL is to make the system language independent so that it can be used universally.

5.4 Requirements for an efficient iconic system

It is widely believed that an **iconic language (IL)** and an iconic system can be successful if:

- IL is a specialized language for specific domain (such as music, math, traffic control, military, digital logic) with a built-in iconic reasoning mechanism.
- IL is a language naturally growing in communications of people who use and spread it (e.g., growth of natural spoken and hieroglyphic languages). This approach is advocated by Cruickshank and Barfiel [2000].
- Efficient learning procedures are established for IL (e.g., start from a small subset).
- IL is a small language with a very few graphical elements/icons with multiple meanings depending on their location relative to other icons.

For instance, in mathematics, line above word *lim* has one meaning and line below *lim* has another meaning. This polysemantic contextual approach is called **semantic compaction** in [Cruickshank & Barfiel, 2000].

Other requirements identified are [King, 1999]: (1) typographic convenience, (2) ability to draw *attention* and *interaction* to a point, (3) ability to encapsulate a *complex meaning* in a simple and "on the spot" message, and (4) ability to support creation of a community of effective *sign-users.*

Comprehensibility icons are still actively discussed, it is obviously desirable but often is not considered as a necessity. If other icons that are less comprehensible can be simple manipulated, then such icons can survive. The history of mathematics provides many examples in support of this point.

6. CONCLUSION

This chapter presented the iconic evidentiary reasoning (IER) architecture with iconic storytelling visualization and an overview of the related work. IER intends to convey complex multi-step analytical reasoning and decision making in a more efficient and condensed way than a traditional text is able. IER is defined as a *visual support mechanism* for the following general problem-solving step: defining the *problem*, generating initial alternatives for hypotheses that are called *pre-hypotheses*, linking pre-hypotheses with evidences, and generating hypotheses by evaluating pre-hypotheses against evidences.

The architecture is applicable for intelligence analysis, engineering and architectural design, market analysis, health care, and many other areas. IER approach was presented using an example of ongoing monitoring of the political situation in some fictitious country. As a result a compact single pic-

ture condenses all reasoning steps. This iconic picture combines indicates a conclusion, its status and a chain of evidences that support the conclusion. For presentation purposes the final part of the reasoning chain can be enlarged and icons can be replaced by actual maps, imagery and photographs of people involved.

Major components of IER architecture are: (1) Collecting and annotating analytical reports as *inputs* using a markup language, e.g., XML, DAML; (2) Providing iconic representation for *hypotheses*, *evidences, scenarios*, *implications*, *assessments*, and *interpretations* involved in analytical process; (3) Providing iconic representation for *confirmations*, and *beliefs* categorized by levels; (4) Providing iconic representation for evidentiary *reasoning* mechanisms (propositional, first-order logic, modal logic, probabilistic and fuzzy logics); (5) Providing scenario-based visualization and visual discovery of changing *patterns and relationships* and (6) Providing a condensed version of iconic representation of evidentiary *reasoning* mechanisms for *presentation and reporting.*

Iconic reasoning is much shorter and perceptually appearing than text. This is important for communicating analytical results (including underlying reasoning) to decision-makers and fellow analysts. With these advantages at some moment iconic reasoning can become a *major way of reasoning and communication* in general. In this chapter, the overview of iconic studies contrasted *iconic reasoning* and *iconic communication.* The application area for iconic reasoning is *problem solving* and the application area for iconic communication is unrestricted human communication. The chapter provided a short overview of iconic terminology started by Charles Pierce. It is also discussed *comprehensibility* of icons along with the *user-created icons* vs. a *fixed iconic vocabulary.* The overview indicated that visual reasoning models that use visual representations isomorphic to their logical models and hybrid models that combine visuals with logical representation often have computational advantages over reasoning models based solely on propositional representations without visual components. The overview also briefly presented a history of iconic languages and ideas of more recent iconic languages.

7. EXERCISES AND PROBLEMS

1. Develop visual reasoning rules similar to those presented in Figure 6. Tip: use OR and negation operations.

2. Build visual search-reasoning rules similar to those shown in section 1.8.

3. Develop your own integrated visual evidentiary reasoning description similar to the scheme presented in Figure 11. Tip: Select a text from a recent mass media report and attempt to present it as a visual reasoning.

4. Discuss requirements for an efficient iconic reasoning system based on considerations presented in section 3.4.

8. REFERENCES

AQUANT program, ARDA, Appendix C 1.1, http://www.ic-arda.org/InfoExploit/aquaint/index.html

Barwise, J. Etchemendy, J. Heterogeneous Logic, In Diagrammatic Reasoning: Cognitive and Computational Perspectives, J. Glasgow, N. H. Narayanan and B. Chandrasekaran, eds., Cambridge, Mass: The MIT Press and AAAI Press, 1995, pp. 211-234.

Becker, L., Leemans, P., VIL: A Visual Inter Lingua. In: Iconic Communication, Yazdani, M. and Barker, P. Eds. Iconic Communication, Intellect Books, Bristol, UK, 2000.

Bliss, C. K. Semantography, Semantography publications, Australia, 1965

Block, N. ed. Imagery. Cambridge, Mass.: MIT Press, 1981

Camhy, D., Stubenrauch, R., A Cross-Disciplinary Bibliography on Visual Languages for Information Sharing and Archiving, Journal of universal computer science, v. 9, 4, 2004, pp.369-389. http://www.jucs.org/jucs_9_4/a_cross_disciplinary_bibliography/paper.html

Champoux, B. Transmitting visual information: Icons to become words, 4th Iconic Communication Workshop, 2001.

Champoux, B.; Fujisawa, K.,Inoue, T.Iwadate, Y., Transmitting Visual Information. Icons Become Words. Proceedings of IEEE Symposium on Information Visualization 2000, p. 244, 2000. http://www.mic.atr.co.jp/organization/dept3/papers/Cails/cails_paper.html

Chandrasekaran, B., H. Simon, Eds. Reasoning with Diagrammatic Representations. AAAI Spring Symposium, Menlo Park, Calif.: AAAI Press, 1992.

Cruickshank, L., and Barfiel, L., The augmentation of textual communication with user-created icons, in: Iconic Communication, Intellect, Bistol, UK, Portland, OR, USA, 2000.

Domingue J. Visualizing knowledge based systems, In: Software visualization: programming as multimedia experience, Eds. Stasko, J., Domingue J., et al. , MIT Press, 1998

Dreyfuss, Henry: Symbol Sourcebook. An Authoritative Guide to International Graphic Symbols. McGraw-Hill, New York 1972; reprint: Van Nostrand Reinhold, New York 1984.

Fauconnier, G. Mental Spaces. Cambridge, Mass.: MIT Press, 1985

Forbus, K. Qualitative Spatial Reasoning: Framework and Frontiers. In Diagrammatic Reasoning: Cognitive and Computational Perspectives, J. Glasgow, N. H. Narayanan and B. Chandrasekaran, eds., Cambridge, Mass: The MIT Press and AAAI Press, 1995

Frixione, M., Gercelli, Zaccaria, R. Diagrammatic Reasoning about Actions Using Artificial Potential Fields, 1997, http://www.dif.unige.it/epi/hp/frixione/ IEEE_Control_Systems.pdf

Funt, B.V. Problem-Solving with Diagrammatic Representations. Artificial Intelligence 13(3): 201-230, 1980

Gardin, F., B. Meltzer, Analogical Representations of Naive Physics. Artificial Intelligence 38: 139-159, 1989

Gelerntner, H. Realization of a Geometry-Theorem Proving Machine. In E.A. Feigenbaum and J. Feldman eds., Computers and Thought, New York: McGraw

Gips J., Shape Grammars and Their Uses: Artificial Perception, Shape Generation and Computer Aesthetics, Birkhaüser, Basel:, Switzerland, 1975

Granger, E., Rubin, M, Grossberg, S., Lavoie, What-and-Where fusion neural network for recognition and tracking of multiple radar emitters, Neural Networks 14, 325-344, 2001

Johnson-Laird, P. , Mental Models. Cambridge: Cambridge University Press, 1983

King, A. Review: 3rd National Conference on Iconic Communication, University of the West of England, Bristol, 9-10th September 1999, http:/NNN/review3.html

Kosslyn, S.M., Images and Mind. Cambridge, Mass.: Harvard University Press. 1980

Kulpa, Z. Diagrammatic Representation and Reasoning. Machine Graphics & Vision 3(1/2): 77-103, 1994

Levesque, H., Logic and the Complexity of Reasoning. Journal of Philosophical Logic 17: 355-389, 1998

Lutz, C., Description logics, 2003, http://dl.kr.org/

Mikulin, L.,Elsaesser, D., Data Fusion and Correlation Technique Testbed (DFACTT): Analysis Tools for Emitter Fix Clustering and Doctrinal Template Matching) Defense Establishments, Ottawa, 1995. http://65.105.56.185/files/09/0912/A091293.html

Moriarty, S., Visual semiotics and the production of meaning in advertising, 1995 http://spot.colorado.edu/~moriarts/vissemiotics.html

Myers, K., K. Konolige 1995. Reasoning with Analogical Representations. In: Diagrammatic Reasoning, Glasgow et al. eds., MIT Press, 1995

Nadin, M., Küpper A., Semiotics for the HCI Community, 2003, http://www.code.uni-wuppertal. de/uk/hci/Applied_Semiotics/applied.html

Neurath, O International Picture Language, Department of Typography and Graphic Communication, University of Reading, England, 1978

Tonfoni G. The theory behind the icon: when, where and why should a system for text annotation actually be used, 4th Iconic Communication Workshop, 2001, Bournmouth. http://www.intellectbooks.com/authors/tonfoni/

Tonfoni G., Writing as a visual art. Intellect U.K., 2000

Tonfoni G., Communication Patterns and Textual Forms, Intellect, U.K., 1996

Tonfoni G., Intelligent control and monitoring of strategic documentation: a complex system for knowledge miniaturization and text iconization, In: Proceedings of the ISIC/ CIRA/ ISAS 98 Conference, pp. 869-874, NIST, Gaithersburg, MD, 1998

Tonfoni G., On augmenting documentation reliability through communicative context transport, In: Proceedings of the 1999 Symposium on Document Image Understanding Technology, pp.283-286, Annapolis, MD, 1999

Sanderson, D.W., Smileys, O'Reilley & Associates, Inc, Sebastopol, CA, USA., 1993

Shape Grammars, http://www.shapegrammar.org/

Sloman, A. Musings on the Roles of Logical and non-Logical Representations in Intelligence. In Diagrammatic Reasoning, Glasgow et al. eds. MIT Press, 1995

Spence, Information Visualization, ACM Press, 2001

Straccia, U., Reasoning within Fuzzy Description Logics, Journal of Artificial Intelligence Research 14, 2001

Vaillant P., Checler M., Intelligent Voice Prosthesis: converting icons into natural language sentences, http://xxx.lanl.gov/abs/cmp-lg/9506018., 1995

Vaillant, P., A Semantics-based Communication System for Dysphasic Subjects. Los Alamos Nat. lab paper archive http://xxx.lanl.gov/arXiv:cmp-lg/9703003 v1 12 Mar 1997

Yazdani, M , Barker, P.G. Iconic Communication, Intellect Books, Bristol, UK, 2000.

Chapter 7

TOWARD VISUAL REASONING AND DISCOVERY

Lessons from the early history of mathematics

Boris Kovalerchuk
Central Washington University, USA

Abstract: Currently computer visualization is moving from a pure illustration domain to visual reasoning, discovery, and decisions making. This trend is associated with new terms such as visual data mining, visual decision making, heterogeneous, iconic and diagrammatic reasoning. Beyond a new terminology, the trend itself is not new as the early history of mathematics clearly shows. In this chapter, we demonstrate that we can learn valuable lessons from the history of mathematics for visual reasoning and discovery.

Key words: Visualization, visual reasoning, visual discovery, history of mathematics.

1. INTRODUCTION

In Chapter 1, visuals were classified in three ways: (1) illustration, (2) reasoning, and (3) discovery. **Illustration** *demonstrates* the essence of *entities* involved and presents a *solution statement* without showing the underlying problem solving reasoning process. **Reasoning** sets up explanatory relevance of entities to each other and **discovery** finds relevant entities. These categories form the **creativity scale** shown in Figure 3 in Chapter 1 where illustration and discovery are the two extremes in this scale with many intermediate mixed cases. Reasoning occupies the middle of this scale. In Chapter 1, all three concepts have been illustrated with the Pythagoras Theorem: (1) visualization of the theorem statement, (2) visualization of the proof process for the theorem's statement, and (3) visualization of the discovery process that identifies the theorem's statement as a hypothesis.

In visual **decision making** the listed categories have their counterparts:

- visualization of a decision
- explanation of a decision
- visualization of the process of discovery of a decision.

Computer visualization is moving from being pure illustration to reasoning, discovery, and decision making. New terms such as visual data mining, visual decision making, visual, heterogeneous, iconic and diagrammatic reasoning clearly indicate this trend. Beyond a new terminology, the trend itself is not new as the early history of mathematics clearly shows. In this chapter, we demonstrate that we can learn valuable lessons from the history of mathematics. The first one is that all three aspects had been implemented in the ancient times without the modern power of computer graphics:

1) Egyptians and Babylonian had a well developed *illustration system* for visualizing numbers;
2) Egyptians and Babylonian had a well developed *reasoning system* for solving arithmetic, geometric and algebraic tasks using *visualized numbers* called **numerals**;
3) Ancient Egyptians were able to discover and test visually a non-trivial math relation, known now as the number π.

How can we learn lesson from this history? How can we accelerate the transition from illustration to decision-making and problem solving in new challenging tasks we face now using history lessons? At first that history should be described in terms of visual illustration, reasoning and discovery. This will give an empirical base for answering posed questions. Traditionally texts on the history of mathematics have different focus. This chapter could be viewed as an attempt to create such an empirical base for a few specific subjects.

The first lesson from this analysis is: *inappropriate results of illustration stage hinder and harm the next stages of visual reasoning and decision making.* Moreover, this can completely prevent visual reasoning and decision making, because these stages are based on visualization of entities provided in the illustration stage.

The most obvious example of such a case is exhibited by Roman numerals. These numerals perfectly fulfill the illustration and demonstration role, but have very limited abilities to support visual reasoning for arithmetic (summation, subtraction, multiplication and division). Hindu-Arabic numerals fit reasoning tasks much better.

The second lesson is that the most natural visualization that seems isomorphic to real world entities is not necessarily the best for reasoning and decision making. The Ptolemy Geocentric system was isomorphic to the observed rotation of the Sun around the Earth, but eventually it became clear that it does not provide advanced reasoning tools to compute the orbits of other planets.

The third lesson is that if we want to design (invent) visualization that will survive reasoning and decision making tests later we should be able to formulate future reasoning and decision making tasks explicitly at the time when design of visualization as illustration is started.

The forth lesson is that if we design (invent) visualization without a clear vision of future reasoning and decision making (problem solving) tasks the chance is no better than a flip of a coin that the visualization will survive reasoning and decision making tests later.

It seems that the history of mathematics points towards the conclusion that most initial visualizations of numerals were invented for illustration, description and recording purposes. Their usefulness for reasoning and problem solving was tested later. Those that survived, namely the Hindu-Arabic numerals, we use now. In essence, this history fits the idea of evolution with the survival of the fittest.

In this chapter, we analyze the history of Egyptian and Babylonian numerals that support lessons learned presented above.

2. VISUALIZATION AS ILLUSTRATION: LESSONS FROM HIEROGLYPHIC NUMERALS

To provide an illustration it is sufficient to *visualize concepts involved in the solution*. Let us consider a simple arithmetic example, 3535+1717=5252. There is a justified computational procedure for getting this solution 5252, but the expression 3535+1717=5252 does not show the reasoning steps that lead us to the solution.

In this example, concepts visualized are input, output, the summation operation and equality relation. Such visualization tasks were successfully solved in ancient Egypt, Greece, India and Mesopotamia by developing symbols for numbers and to some extent symbols for operations and relations too.

2.1. Egyptian numerals

Hieroglyphic numerals. Table 1 shows Egyptian hieroglyphic numerals and some of their ideographic meanings [Allen, 2001a, Williams, 2002c, Aleff, 2003, Bertin, 2003].

Table 1. Symbolic Egyptian Hieroglyphic numerals

1	10	10^2	10^3	10^4	10^5	10^6	10^7
							
vertical stroke	heal bone, vault	snare coil of rope	lotus flower	bent finger	burbot fish, Tadpole	kneeling figure with raised arms, Heh-god	Sun

Compare this with Roman and Hindu-Arabic numerals presented in Table 2. Romans had also other numerals: V for 5, L for 50, and D for 500.

Table 2. Roman and Hindu-Arabic numerals

1	10	10^2	10^3	10^4	10^5	10^6	10^7
1	10	100	1000	10000	100000	1000000	10000000
I	X	C	M	$\overline{\text{X}}$	$\overline{\text{C}}$	$\overline{\text{M}}$	

Table 3 shows some alternative design of hieroglyphs from sources listed above. Left and right forms were used on the left and right sides of the text to provide visual symmetric view of the wall. Several fonts have been developed for hieroglyphs. Table 3 uses Gardiner, Glyph basic, and Nahkt fonts [Bertin, 2003].

Table 3. Symmetric pairs and alternative glyphs

Number	*Left form*	*Right form*
100		
1,000		
10,000		
100,000		

Hieratic numerals. The Egyptian Hieratic numeral system is also 10 based. Hieratic symbols for 2 and 3 are repeated symbols for 1 (|), respectively (|| , |||) and the symbol for 8 (=) is a repeated symbol for 4 (–).

The same symbols || and ||| are used in Hieroglyphic and Roman systems for 2 and 3. Other unique symbols in the Hieratic system are [Friedman, 2003b]:

This system also has unique symbols for 20,30,40,50,60,70,80,90 and 100. The symbol for 20 ( ) is directly based on symbols for 10 (∩), and the symbol for 40 ( ) is based on symbol for 4 (–). Symbols for 200, 300, 400 and 500 are based on the symbol for 100 ( ). They are drawn by adding one, two, three or four dots (.) above, 200 ( ), 300 ( ), 400 ( ), and 500 ( ) [Friedman, 2003b].

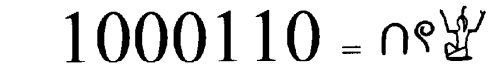
In Table 4, we show how numbers were composed using base glyphs presented in Figure 8 using the idea of Hindu-Arabic decimal positional system that is a modern standard for number visualization.

It is so common after several thousand years of use that we often miss the point that this is just *one of the possible number visualization systems.* It would be more transparent if we contrast it with a textual description of numbers. **The textual description at most visualizes the sequence of sounds**, e.g., the word "thousand" in phonetics based languages.

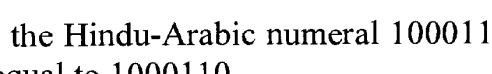
Table 4. Egyptian hieroglyphs and operations [font from Williams, 2002c]

Modern symbols	=	$1=10^0$	10	10^2	10^3
Egyptian hieroglyphs					
*Modern 3105=5+100+3*10³ (backward notation)*		5	0	100	$3*10^3$
Egyptian 3105					

In Egyptian numeral system, there is a flexible sequence and every decimal has its own symbol. Thus, some hieroglyphic numerals are shorter than in modern notation, for instance:

1000110 = 

The hieroglyphic system is less positional than the Hindu-Arabic system that we are using now. The change of the sequence of the components does not destroy the value of the number:

1000110 =  = 

but for the Hindu-Arabic numeral 1000110 the backward sequence 0110001 is not equal to 1000110.

The Roman system is an intermediate system between the Egyptian and the Hindu-Arabic systems. This system is more positional than the Egyptian system and less positional than the Hindu-Arabic system. At first glance, Tables 1 and 2 show that both Roman and Egyptian systems provide simpler visualization for numbers than the Hindu-Arabic system. However, only this system has survived as a reasoning and decision making tool for humans and was replaced by the Binary positional systems for computers only very recently. It is not clear which of these systems were developed first. It is most likely that all these systems were developed quite independently and then were tested for ease in solving mathematical tasks. It is possible that the Hindu-Arabic system is the oldest one.

Egyptians actually used the flexibility of their system to present numbers in a variety of ways including several lines. Their numerals are truly two-dimensional (see Figure 1). Thus, Egyptians had a well developed *illustration system* for visualizing numbers.

Figure 1 shows that Egyptians wrote a single number using 2-3 lines starting from larger digits. Alternatively, they used a single line, where larger digits are on the right because Egyptians wrote from right to left. The record for number 300003 shows also that Egyptians used a smaller size for smaller digits (all six ones "|" occupy the *same space* as a single glyph for 100,000). Writing for 1/25 in Figure 1 shows that they also used a format where digits are distributed in two columns. Writing for 3350 shows that Egyptians also used a "zipper" type of writing, where two of ∩ symbols are moved down. Sometimes Egyptians used a larger "font" for larger digits.

The symbol for 1000 is twice larger that the symbols for other digits in the Egyptian 1303 shown above. The same idea is implemented in writing 1010005, the digit for 1000000 is more than twice larger than digits for 10000 and 5. Thus, every **number** is represented as a *complex icon/glyph/numeral* that is combined from simpler icons (numerals) for basic digits.

This combining has its own **visual syntax**. The presentation for 19607 does not follow the pattern that larger digits are also larger in size. It shows digits for 1000 smaller because there are nine of them. Thus, a more general rule is that a *large number of equal digits is the main factor for drawing those digits smaller.*

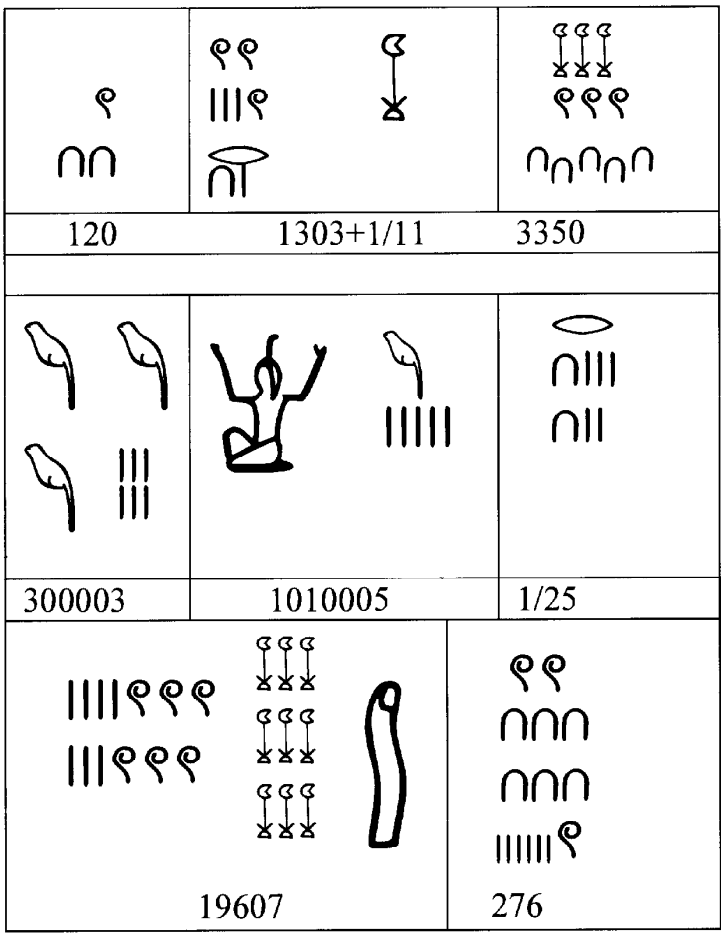


Figure 1. Free sequence of numeral components
(based on [Friedman, 2003c; Williams, 2002c; Arcavi, 2003; Allen,2001a])

2.2. Babylonian numerals

The Babylonians inherited the Sumerian style of writing on clay tablets. Their arithmetic was *positional* and *sexagesimal* **based** on 60 with symbols for 1, 10, 60, 600, 3600, 36000, and 216000. We follow a simplified notation from [Allen, 2001b] where V is used for 1 and ≺ for 10. In this notation

$$7341 = \text{VV VV} \prec\prec \text{V}$$

because $7341_{10} = 2221_{60} = 2*60^2 + 2*60 + 21$.

Base 60 has many advantages. One of them is that other systems can be converted to this one (2, 3, 5, 10, 12, 15, 20, and 30 all divide 60) .

The Babylonian system permitted some shortcuts. Without the shortcut, number 19 is represented as 𒐐VVVVVVVVV. Table 3 shows a shortcut for this number that uses a subtraction idea (19 = 20-1). This way the symbol for 1 that is V is not repeated nine times [Allen, 2001b].

Table 3. Babylonian numeral 19 (based on [Allen, 2001b])

𒐐𒐐V̄	19 is presented as 20-1, where 20 is 𒐐𒐐 and 1 is V. The negation symbol is over the symbol for 1.
𒐐 V V V V V V V V V	19 as 10+9, where symbol 𒐐 stands for 10 and 9 small symbols V stand for 1.

This brief description shows that the Babylonians had a well developed *illustration system* for visualizing numbers. It was not limited to integers; Babylonians also used fractions. Their abilities for *reasoning with numbers* included extracting square roots, solving linear systems, using Pythagorean triples such as (3, 4, 5): $3^2 + 4^2 = 5^2$, and solving cubic equations using tables. Several of these actions can be qualified as visual reasoning too.

2.3. Results of arithmetic operations

People in ancient Egypt knew how to visually represent results of arithmetic for both integers and fractions (see Figures 2 and 3).

𓎆𓎆𓏺𓏺𓏺𓏺𓏺	+	𓏺𓏺𓏺𓏺𓏺𓏺𓏺	=	𓎆𓎆𓏺𓏺𓏺𓏺𓏺𓏺𓏺
215	+	57	=	272

Figure 2. Visual adding integers

𓍢𓏺	𓍢𓏺 𓍢𓏺	𓍢𓏺𓏺𓏺 𓏺𓏺	𓍢𓏺𓏺𓏺 𓍢𓏺𓏺𓏺 𓏺𓏺 𓏺𓏺
1/10	1/10+1/10=2/10	1/25	1/25+1/25=2/25

Figure 3. Visual adding fractions

Egyptians also knew how to add numbers visually (see Figures 4, 5)

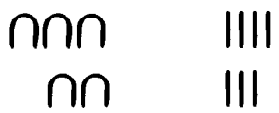
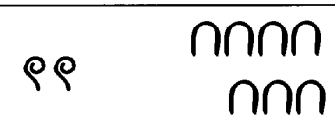
Adding visually by columns, 1700 BC, Egypt	Adding visually column by column	Adding using text (method does not exist yet)
	215	Two hundred fifteen
	57	Fifty seven
	272	two hundred seventy two

Figure 4. Adding numbers using symbols vs. text

Let us assume that we need to sum the numbers 215 and 57 written in words, two hundred fifteen plus fifty-seven. How can we do this using the words? A procedure to do this does not exist even after 4000 years of using numbers. The only method known now is converting verbal numbers to one of the symbolic (visual forms).

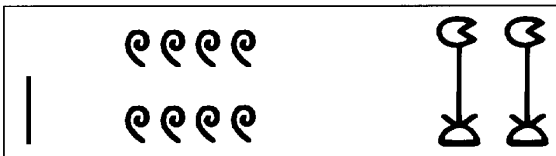
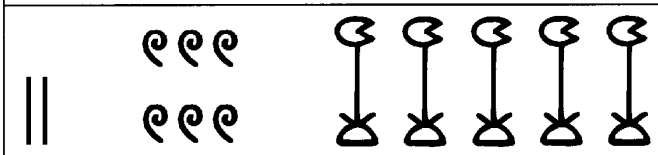
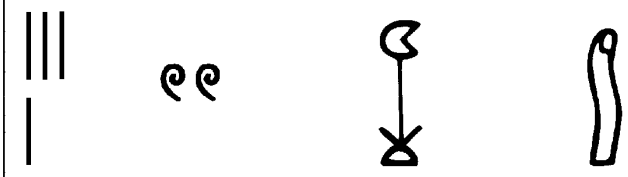
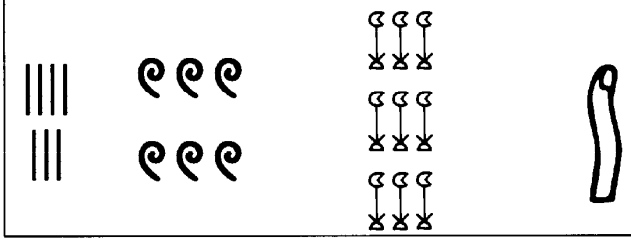
	2801
	5602 = 2801*2
	11204 = 5602*2
	Sum 19607 = 2801 + 5602 + 11204

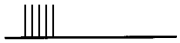
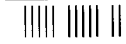
Figure 5. Visual summation (based on Arcavy, 2002)

In the next section, we analyze the visual reasoning in Egyptian mathematics that is representative of the level of sophistication reached in the Ancient world in visual reasoning.

3. VISUAL REASONING: LESSONS FROM HIEROGLYPHIC ARITHMETIC

Reasoning includes procedures for getting results from arithmetic operations. A proof that the operation is correct came from summation and multiplication tables. In early Egypt, addition and subtraction were simple visual processes using the counting glyphs. To add two numbers, we collect all symbols of the same type and replace ten of them by one of the next higher order. For example, adding and subtracting 5 and 7 is shown in Table 5, where ten symbols | are substituted by a single symbol ∩ that means 10.

Table 5. Visual arithmetic

Addition		*Subtraction*	
7		*7*	
5		*5*	
result		*result*	
compact result	|| ∩		

Addition and subtraction of 35 and 17 are shown in Table 6 where numbers are presented in backward sequence of decimal positions. These operations use the same visual techniques: *grouping* and *substitution*.

Table 6. Visual addition of 35 and 17

Number (in modern notation)	*Decimal position*	
	| (10^0)	∩ (10^1)
35	|| |||	∩∩∩
	5	30
17	|| ||| ||	∩
	7	10
Sum 52 = (12 + 40)	|| ||| || ||| ||	∩∩∩∩
	12	40
Sum 52 with use "carry"	||	∩∩∩∩∩
(after shifting 10^0 "|" to 10^1 position as ∩)	2	50

Similarly, Table 7 shows the addition of 70 and 10.

Table 7. Summation (with opposite sequence of decimal positions)

Number (in modern notation)	$\mid (10^0)$	$\cap\ (10^1)$
Sum 70 = 10 + 60	|| ||| || |||	∩∩ ∩∩∩
	10	6·10
Sum 70 = 10 + 60 (after shifting 10 "|" to 10^1 position as "∩")		∩∩ ∩∩∩∩
	0	7·10

Multiplication and division were done by using visuals and a lookup multiplication table for only $2n$, $4n$, $8n$ and so on for every n. Numbers in the table were generated by repeated visual summation such as $n + n = 2n$ then $2n + 2n = 4n$ and so on (see Table 7 for the number 25). For example, to multiply 25 by 11 the property $11 = 1 + 2 + 8$ is used along with the two times multiplication table (Table 8): $25 \cdot 11 = 25 \cdot (1+2+8) = 25 + 25 \cdot 2 + 25 \cdot 8$ (see Table 9).

Table 8. Two times multiplication table for number 25

1·25	|	|| ||| ∩ ∩	25
2·25	||	∩∩∩∩∩	50=25+25
4·25	||||	९	100=50+50
8·25	|||| ||||	९९	200=100+100

Table 9. Multiplication 25·11

1·25	|| ||| ∩ ∩	25
2·25	∩∩∩∩∩	50
8·25	९९९९	200
25·(1+2+8)=25·11	|| ||| ∩∩ ∩∩∩∩∩ ९९९९	275=25+50+200
	|| ||| ∩∩ ∩∩∩∩∩ ∩∩∩∩∩ ∩∩∩∩∩∩ ९९	275
	|| ||| ∩∩ ∩∩∩∩∩ ९९९	275

To get a feeling for the advantages of this visual process we just need to try to multiply 25 by 11 solely in a textual form. This simple task becomes very difficult to solve and even harder to prove that the result is correct. But the visual arithmetic computation process is not simple either if we try to record it completely. Below we show what happens when we add two numbers, 35 and 17, using standard arithmetic techniques. We use a spreadsheet visualization similar to the one used in MS Excel. In Table 10, number 35 occupies cells (1,3), (1,4) and number 17 occupies cells (2,3) and (2,4). The result should be located in cells (3,3) and (3,4). There are also cells (4,2) and (4,3) reserved to writing carries. We use symbols a_{ij} for a content of cell (i,j). In this notation, the algorithm for adding 35 and 17 consists of two steps:

Step 1: If $a_{14} + a_{24} > 9$ then $a_{34} := (a_{14} + a_{24}) - 10$ & $a_{43} := 1$
else $a_{34} := a_{14} + a_{24}$ & $a_{43} := 0$

Step 2: If $a_{13} + a_{23} + a_{43} > 9$ then $a_{33} := (a_{13} + a_{23}) - 10$ & $a_{42} := 1$
else $a_{33} := a_{13} + a_{23} + a_{43}$ & $a_{42} := 0$

Table 10. Visual summation

	j=1	*j=2*	*j=3*	*j=4*
i=1			3	5
i=2			1	7
i=3	sum		5	2
i=4	carry	0	1	

At first glance, this algorithm does not appear to be visual, but the placement of numbers is visual, similar to what users do everyday working with the Excel spreadsheet graphical user interface. People accomplish these steps easily visually, but it is hard to explain steps 1 and 2 without visuals, although it is almost a complete computer program.

4. VISUAL DISCOVERY: LESSONS FROM THE DISCOVERY OF π

Ancient Egyptians were able to discover and test visually a non-trivial mathematical relation between the diameter of the circle, D and its area, S. Now this relation is expressed using the number π, $S=(\pi/4)D^2$.

Egyptians discovered this relation for a specific diameter $D=9$ in the form $S=(D-1)^2=(9-1)^2$. It can be generalized to $S=(D-D/9)^2=(8D/9)^2=(64/81)D^2$.

We can compare the Egyptian coefficient $64/81=0.790123$ with $\pi/4=0.785398$ and notice that the formula discovered in Egypt is remarkably accurate. Below we discuss this discovery in more detail. Problem 50 from the Rhind papyrus (about 1550 BC) [Williams, 2002b] is about this relation between the diameter and the area of the circle:

"A circular field has diameter 9 khet. What is its area?"

One khet is about 50 meters. This papyrus is now in the British Museum. Its detailed description is presented in [Robins, Shute, 1998; Chance at al., 1927, 1929]. The papyrus was made by the scribe Ahmose and sometimes is called the Ahmose papyrus. The solution provided in the Rhind papyrus for problem 50 is [Friedman, 2003a] :

You are to subtract one ninth of it, namely 1: remainder 8.
You are to multiply 8 two times: it becomes 64. This is its area in land, 6 "thousands-of-land" and 4 setat.

In modern terms, it means as we already mentioned:

If diameter D=9 then circle area S=(9-1)(9-1)=8·8=64.

In accordance with this formula, π number would be 3.160494 which is quite close to the correct value 3.141593. To get this value we just need to notice that $S=(9-1)^2=64=\pi(9/2)^2$. Thus, $\pi=64/(9/2)^2 = 3.160494$.

How was it possible to discover such an accurate result 3500 years ago? We follow Williams' conjecture that it was done by means of visual discovery: "An alternate conjecture exhibiting the value of π is that the Egyptians easily observed that the area of a square 8 units on a side can be reformed to nearly yield a circle of diameter 9."

Figure 6 reconstructs how such a discovery could be made. It shows a variety of circles and squares with small circles inside. The square 8 by 8 has 64 small circles and the circle with diameter of nine small circles has 67 of those circles. These numbers 64 and 67 are similar with the difference less then 5% of each of them. The difference between any other pair of numbers in Figure 6 is greater, thus, it is discovered that the area of the square 8×8 closely interpolates the area of the circle with the diameter 9. This mental experiment could be conducted in ancient Egypt physically with small river rocks, apples, or seeds. Ancient Egyptians could easily collect hundreds of these items of almost the same size.

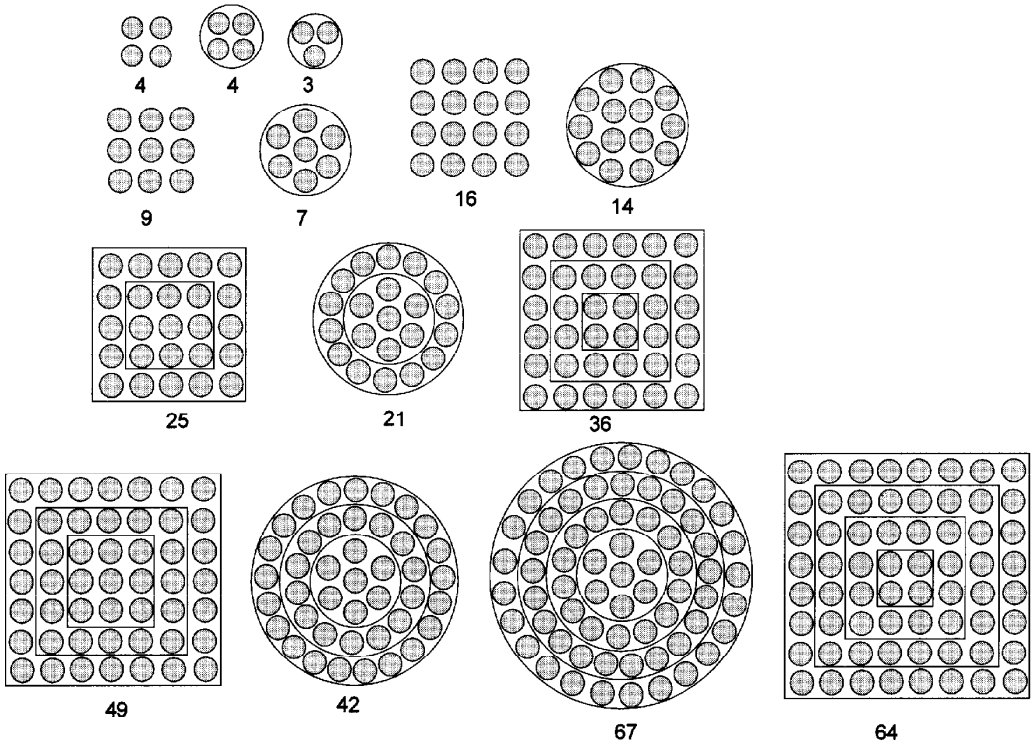


Figure 6. Visual experiment (adapted from [Williams, 2002a])

Now we need to analyze how after discovering the similarity of the number of rocks in some circle and square to discover the mathematical indicators to match the circle and the square areas. We already assumed that these indicators are the diameter (or radius) of the circle and the side of the square. It is a very realistic assumption. These indicators were already known to Egyptians as the main characteristics of a circle and a square. But we need to discover the relation between their squares, $S^2 = \pi R^2$ with an unknown coefficient π.

In essence, this is a *data mining task* in modern terms. It could be tried in ancient times visually again by experimenting. For instance, rocks can be counted in the square that contained in the circle and in the square that contains the circle getting $S_1<S<S_2$. Here, S_1 is the number of rocks in the contained circle and S_2 is the number of rocks in the circle that contains the circle with area S. Obviously this approach would give a very rough estimate of π. A more accurate result can be obtained by interpolating a circle by smaller squares and counting rocks that are contained in the circle. For instance, we can interpolate the circle area by subtracting from the total area of the surrounding square (9·9=81) the area that is not in the circle. This is about a half of four small square 3×3 in the corners of the 9×9 square in Figure 7, that is 4(3·3)/2, with the final result: 9·9 –2·3·3 = 81-18 = 63 that is close to 8·8=64. In essence, this is an octagon interpolation of the circle.

This visual approach has been used in problem 48 of the Rhind papyrus. In general, problems 41-43, 48, and 50 of that papyrus are devoted to the circle area computation. Problem 48 of the Rhind Papyrus states [Write, 2000]:

> The area of a circle of diameter 9 is the same as the area of a square of side 8. Where does this come from?

To justify statement (1) problem 48 contains a famous drawing of a square with an inscribed octagon:

```
A a b B
h     c
g     d
D f e C
```

where ABCD is a square and abcdefgh is an irregular octagon [Gnaedinger, 2001]. Problem 48 differs from problem 50 discussed above. Problem 48 is to justify of the result (**reasoning**, proof) of the already discovered statement

$$\text{AreaCircle}(9)=\text{AreaSquare}(8), \qquad (1)$$

where 9 is the length of the diameter of the circle and 8 is the length of the side of square. In contrast, problem 50 asks for **discovery** of the statement presented in (1). Figure 7 presents the inscribed octagon graphically.

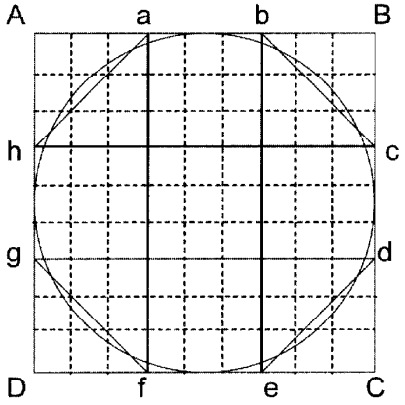


Figure 7. Illustration for the Rhind problem 48 (based on [Wright, 2000])

This simple method provides a reasonable approximation of π. As now known, this approximation idea permits one to get π with any desirable accuracy by using an *n*-gon with large *n*.

5. CONCLUSION

Examples in this chapter illustrate the power of *visual discovery combined with mathematical computations and reasoning*. Below we summarize the characteristics of ancient arithmetic based on the analysis in this chapter:

1. Ancient arithmetic was visual.
2. Ancient arithmetic involves explicit reasoning.
3. There are exact reasoning rules how to operate with visual entities to obtain the result of an operation.
4. Reasoning rules are specific for each arithmetic operation.
5. Rules of more complex operations are based on rules for simple operations (e.g., multiplication is based on addition and division is based on multiplication).

The goal of each visual operation was well defined.

Although formal models of the mathematical operations can be built, the actual Egyptian system is a *mixture* of visual intuitive procedures and formal manipulation with minimized double conversion between analytical and visual representation of the problem (see discussion of this subject in Chapter 1, Section 1.5). If we compare many modern visualization tools with the characteristics presented above, we see that we have not reached the level of sophistication known in ancient Egypt more than 3000 years ago. For instance, *visual data mining* does not go further then showing glyphs such as squares

and the rectangles from different viewpoints, but visual guidance how to experiment using visual tools for *pattern discovery* are not mature yet.

In Chapter 16, we present a new visual data mining technique based on Monotone Boolean functions that intends to fill this gap for the Boolean data type. The technique permits discovering patterns by analyzing structural interrelations between objects (cases) in the original visualization and by changing the visualization to its modifications that permits one to see a continuous border between patterns if it exists.

To build an advanced visual discovery system we need to start with a clearly stated goal, as was done in the examples analyzed in this chapter, such as the goal of discovering a formula to compute the area of the circle. In modern visual data mining, the goal is discovering patterns. At first glance, it looks that we also have a goal, but this goal is not that well stated as computing the area. The correctness of the area computation can be tested using well-stated and simple criteria. Like data mining, the goal in imagery conflation (see chapters 17-21) is to find matching features. However, there are no natural, well-stated formal criteria to test if the goal is reached, even if people are able to match features by visual inspection of two images using informal, tacit rules. There are two important questions:

(1) How can we know if a task can be solved by visual means?

(2) How do we select tasks to be solved by visual means?

The answer to both questions based on our analysis of early history of mathematics is: The task should have a goal and a formalized criteria to judge that the goal is reached, as was the case for these mathematical tasks. For less formal tasks, visual reasoning is still possible, as chapters (BK) and DB indicate, but the conclusions may be much less conclusive and the methods may be less sophisticated.

In Chapter 1 (Section 2.1) we discussed visualization, visual reasoning and visual discovery for the Pythagorean Theorem. The first two tasks were successfully solved visually many times (there are more than 300 different proofs of the theorem), but it is not the case for visual discovery of the theorem statement. It is difficult to formulate formal criteria for visual discovery. The goal can be formulated easily – to discover the theorem statement. However, we cannot assume that parameters and the types of relations (polynomial or other) between them are known if it is true discovery. Thus, the task should be formalized. This can be done in many different ways and the solutions can be quite different. The early history of mathematics clearly shows the trend from illustration to visual reasoning and discovery. This chapter demonstrates that we can learn valuable lessons from this history. The main lessons are:

- inappropriate results at the illustration stage harm the next stages of visual reasoning and decision making;

- to invent a visualization that will survive visual reasoning and decision making tests, reasoning and decision making tasks should be formulated explicitly when one designs visualization as illustration.

Future work will use additional empirical information about the use of visual reasoning and visual discovery in ancient mathematics to analyze how to solve visually modern problems.

6. EXERCISES AND PROBLEMS

1. Compute visually 71 + 71 in the Egyptian hieroglyphic system using Table 11 for 70+70 as a prototype.

Table 11. Hieroglyphic arithmetic for 70

Number (in modern notation)	*Decimal position*		
	| 10^0	∩ 10^1	୧ 10^2
70		∩∩∩∩∩∩∩	
	0	70	
70		∩∩∩∩∩∩∩	
		70	
Sum 140 = 10*14		∩∩∩∩∩∩∩ ∩∩∩∩∩∩∩	
	0	140	
Sum 140 = 100 + 40 (after shifting 10 ∩ to 10^2 position as ୧)		∩∩∩∩	୧
	0	40	100

2. Compute visually 145 + 145 in the Egyptian hieroglyph system using Table 12 for 140 + 140 as a prototype.

Table 12. Hieroglyphic arithmetic for 140

Number (in modern notation)	*Decimal position*		
	| 10^0	∩ 10^1	୧ 10^2
140		∩∩∩∩	୧
	0	40	100
140		∩∩∩∩	୧
		40	100
Sum 280 = 10*14		∩∩∩∩ ∩∩∩∩	୧୧
	0	80	200

3. Compute visually 37 + 74 + 280 in the Egyptian hieroglyph system using Table 13 for 35 + 74 + 280 as a prototype.

Table 13. Hieroglyphic arithmetic for 35, 70 and 280

Number (in modern notation)	*Decimal position*		
	$\vert\ 10^0$	$\cap\ 10^1$	$\text{𓎆}\ 10^2$
35	|||||	𓎆𓎆𓎆	
	5	3*10	
70		𓎆𓎆𓎆𓎆𓎆𓎆𓎆	
		7*10	
280		𓎆𓎆𓎆𓎆 𓎆𓎆𓎆𓎆	𓎆𓎆
	0	8*10	200
Sum 385 = 5 + 180 + 200	|||||	𓎆𓎆𓎆 𓎆𓎆𓎆𓎆𓎆𓎆𓎆 𓎆𓎆𓎆𓎆 𓎆𓎆𓎆𓎆 𓎆𓎆𓎆𓎆 𓎆𓎆𓎆𓎆	𓎆𓎆
	5	180	200
Sum 385 = 5 + 80 + 300 (after shifting 10 𓎆 to 10^2 position as 𓎆)	|||||	𓎆𓎆𓎆𓎆 𓎆𓎆𓎆𓎆 𓎆𓎆𓎆𓎆 𓎆𓎆𓎆𓎆	𓎆𓎆𓎆

4. Analyze efficiency of hieroglyphic visualization for arithmetic operations. Do you see any cases where the summation or multiplication using hieroglyphic numerals can be accomplished faster than using the Hindu-Arabic numerals? Tip: Start from the cases presented in exercise 1-3.

7. REFERENCES

Allen, D., (2001a) Counting and Arithmetic – basics, TAMU, 2001http://www.math.tamu.edu/~don. allen/history/egypt/node2.html

Allen, D., (2001b) Babylonian Mathematics, TAMU, 2001, http://www.math.tamu.edu/~dallen/masters/egypt_babylon/babylon.pdf

Aleff, H., Ancient Creation Stories told by the Numbers, 2003, http://www.recoveredscience.com/const105hehgods.htm

Arcavi, A., Using historical materials in the mathematics classroom,2002, http://www.artemisillustration.com/assets/text/Rhind%20Papyrus.htm.

Berlin, H., Foreign Font Archive, Hieroglyphic and Ancient Language Typefaces, 2003, http://user.dtcc.edu/~berlin/font/downloads/nahkt_tt.zip

Chance, A., Bull,L. .Manning,H., Archibald, R., The Rhind Mathematical Papyrus, vol. 1, Oberlin, Ohaio, MAA, 1927; vol. 2, Oberlin, Ohaio, MAA, 1929.

Friedman, B., (2003a)A Selection of Problems from the Rhind Mathematical Papyrus and the Moscow Mathematical Papyrus, 2003, http://www.chass.utoronto.ca/~ajones/cla203/egyptmath2.pdf]

Friedman, (2003b) B., Sample hieratic math answers, 2003, [http://www.mathorigins.com/Image%20grid/BRUCE%20OLD_007.htm]

Friedman, (2003c), B., CUBIT, 2003, http://www.mathorigins.com/image%20grid/ CUBIT_002.htm

Gnaedinger, F., Rhind Mathematical Papyrus, 2001, http://www.seshat.ch/home/rhind6.htm

Robins, G., Shute, C., The Rhind mathematical papyrus. An ancient Egyptian text, British Museum Publications, Ltd., London, *1998*

Williams, S., (2002a) Egyptian geometry determining the value of π the Pythagorean theorem, 2002, http://www.math.buffalo.edu/mad/Ancient-Africa/mad_ancient_egypt _geometry.html#ahmes10

Williams, S., (2002b) Rhind papyrus, 2002, http://www.math.buffalo.edu/ mad/Ancient-Africa/mad_ancient_egyptpapyrus.html#ahmes/rhind papyrus

Williams, S., (2002c) Egyptian counting with hieroglyphs, the Mathematics Department, SUNY, Buffalo, 2002, http://www.math.buffalo.edu/mad/ Ancient-Africa/mad_ ancient_egypt_arith.html

Wright, J., Lecture Notes for Egyptian Geometry, 2000, http://www2.sunysuffolk.edu/ wrightj/MA28/Egypt/geometry2.pdf

PART 3

VISUAL CORRELATION

Chapter 8

VISUAL CORRELATION METHODS AND MODELS

Boris Kovalerchuk
Central Washington University, USA

Abstract: This chapter introduces the concept of visual correlation and describes the essence of a generalized correlation to be used for multilevel and conflicting data. Several categories of visual correlation are presented accompanied by both numeric and non-numeric examples with three levels (high, medium and low) of coordination. We also present examples of multi-type visual correlations. Next, the chapter provides a classification of visual correlation methods with corresponding metaphors and criteria for visual correlation efficiency. Finally, the chapter finishes with a more formal treatment of visual correlation providing formal definitions, analysis, and theory.

Key words: Visual correlation, heterogeneous data, visual data mining, information visualization, glyph, metaphor, classification, guidance, distortion, formalization, homomorphism, relational structure.

1. INTRODUCTION

1.1 The motivation for a generalized correlation concept

The purpose of visual correlation (VC) is to represent and discover the correlation between objects and events (O/E) visually. It has its own value for many applications and has significant potential to support decision making. Several complex questions need to be answered if visual correlation is to be implemented successfully:

(1) How does one visually correlate non-numeric O/E data?

(2) How does one visually correlate O/Es with different levels of resolution?
(3) How does one visually correlate conflicting data for a given O/E?
(4) How does one visualize data for different categories of users?
(5) How can an O/E symbol be made "rich enough" to portray the differences between O/Es?

To illustrate the problem and various approaches to it, we start with the non-traditional problem of correlating non-numeric O/E data. One of the major challenges here is that often such O/Es are represented by non-structured or semi-structured **text**. One solution, a visual correlation system and visual language **BRUEGEL** described in Chapter 10, deals with this problem for text that is tagged with XML tags. The system and language are named after Pieter Bruegel the Elder (1525-1569). The naming is not accidental.

The visual correlation in BRUEGEL was inspired Bruegel's famous painting "Blue cloak" (1559) shown in Figure 1. This paining is named after the Netherlandish (Flemish) proverb "She hangs a blue cloak (lies) around her husband". This proverb is "visualized" in this painting and is marked by the center box in the picture below.

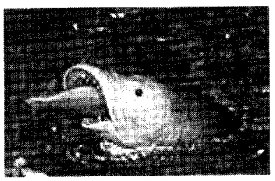
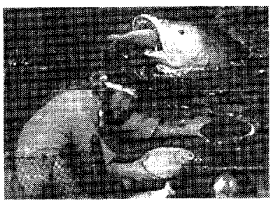
Figure 1. Pieter Bruegel's painting "Blue cloak" (1559), oil on oak panel, 117 x 163 cm. (with permission from Staatliche Museen zu Berlin - Gemaldegalerie, Berlin). See also color plates.

From the visual correlation viewpoint, the uniqueness of this painting is that Pieter Bruegel "compressed" and "visualized" a total of at least 78 Netherlandish proverbs, maxims, rhymes and symbols [Foote, 1968].

In addition, **visual correlation** is supported by locating related proverbs in the same area of the painting. In this way Pieter Bruegel "visualizes" and conveys more information than can be provided by each individual proverb. Two proverbs, marked by the right-hand box in Figure 1, are shown in more detail in Table 1. The first two rows show these proverbs separately and the third row shows them side-by-side as Pieter Bruegel painted them. It is similar to the modern concept of **side-by-side visual correlation** that we will discuss later in this chapter.

The visual correlation of two proverbs reveals the deeper meaning of each proverb and their combinations. In essence, each painted proverb fulfils the role of an **iconic summary** of a complex concept. Their combination, as noted in the last line of the table, is a prototype for **compound and composed icons** that are further discussed in chapter 9.

Table 1. Encoding text in art. See also color plates.

Raw text	***Compressed content of the text***	
	Text metaphor proverb	***Image metaphor icon***
A greater power controls a smaller power with a brute force and violence	Big fish eats little fish	
A greater power controls a smaller power in a smart way without using a brute force and violence	He catches fish with his hands	
A greater power controls a smaller power with a brute force while another greater power controls a smaller power in a smart way without using a brute force and violence	Big fish eats little fish while he catches fish with his hands	 Merged picture

Visual correlation is subject to two major interdependent challenges: (1) *distortion* of the actual relation and (2) *excessive guidance* necessary for the user to avoid distortion. Bruegel's proverb example above illustrates this point as we discuss below. Distortion of the actual relation *R(a,b)* between objects *a* and *b* may have several sources:

- inappropriate and/or insufficient guidance for extraction of a relation *R(a,b)* from visualized data,
- human misperception of the visualized data, and
- human visual expectations.

It is difficult to meet both challenges simultaneously. Less intensive guidance can increase misperception, specifically, the difference between an *actual relation R(a,b)* and a *perceived relation Q(a,b)* between objects *a* and *b*. In some cases, the actual relation *R(a,b)* can be lost completely because of misperception. Obviously, if we have no knowledge about the "Blue cloak" proverb and its meaning, we would not be able to recognize the relation *R(a,b)* = "*a* is lying to her husband *b*".

Textual guidance can explain the proverb, but if every visual symbol needs an excessive explanation then visualization is not serving its role – to convey information faster than the traditional text-based form of communication.

In the following sections, we define concepts, review current visual correlation studies, and provide examples, a classification of VC methods, and criteria to assess the quality of visual correlation.

1.2 Definitions of concepts

Current studies on visual correlation range from formally defined, classical linear correlation in statistics to very informally defined correlation between natural language statements and images.

Correlation can be defined as:

- a tool for combining multiple observations of the *same O/E* when data can be expressed in *mathematical statistical form precisely* or
- a tool for combining *measurements* involving single or multiple phenomenologies (e.g., Radar, Sonar) often in near-real time [Marsh, 2000].

Correlation can be contrasted with data fusion and semantic data integration as follows:

- **Fusion** – a tool for combining information of very different types, such as sensor data, imagery, or human reports. It is often non-real time.
- **Semantic Integration** – a tool to combining information where individual meanings and relationships can infer a *larger meaning*. It may involve some degree of *contextual reasoning* [Marsh, 2000].

In these terms, the Bruegel painting probably better fits into the semantic integration category.

Other definitions of correlation include: fusion and integration under correlation umbrella. The core term used in these definitions is *combining O/Es*.

From our viewpoint, the essence of correlation, fusion, and integration is *generating new knowledge in the form a clearly established link* or *relation* between different O/Es.

Classical correlation. We start our analysis with classical correlation and its visualization. Traditionally, correlation is expressed by using a *correlation coefficient* as a *measure of association* (interdependence) between two or more variables. A brief description of different correlation coefficients used for numeric and binary variables is shown in Table 2. Measures of interdependence among *several variables* include: multiple correlation, marginal correlation, conditional correlation, canonical correlation, and auto- and cross-correlation for ensembles of measurements.

Table 2. Correlation coefficients between numeric or binary variables

Correlation coefficient (CC)	*Used to evaluate the relationship between*
Simple CC	*Two numeric variables*
Auto CC	*The same numeric variable* at times X_i and X_{i+k}.
Cross CC	*Two different data sets at different lags* (a function of lag)
Rank CC	Two variables when the *distribution of variables is not normal*
Point biserial CC	*Continuous* variable and a *binary* variable
Tetrachoric CC	Two artificially *dichotomous* normally distributed variables
Contingency CC	Two *nominal* level variables.

Generalized correlation. Data that need to be correlated are not limited by numeric or binary variables. For instance, we may need to correlate roads or drainage systems on the maps with imagery from different sources that may not have a common scale, may have different rotations and may have no common reference points. This task is commonly known as a conflation and is discussed in Chapters 17-21.

There are also a variety of specialized correlations such as angular correlation and correlation for emitter error with confidence ellipses and time for identifying emitter locations using information from a radio direction-finding system [Mikulin & Elsaaesser, 1994].

The mathematical definition of correlation assumes that *variables* are specified in advance and a *procedure* for testing the *significance* of relationships between variables is also given in advance and expressed as a mathematical formula. In mathematical approach to correlation, we first observe (discover) some *relationship* between *variables.* The result is not the relationship itself but a conclusion about its statistical significance. That is, the relationship is reproducible.

Next, this approach can be applied to the situation where (i) objects and events are described by a few numeric variables with (ii) a large amount of values available. However, this situation is quite different from one where recorded O/Es represent social events such as terrorist activities where (iii)

objects and events are represented by a *large number of non-numeric descriptors* (textual or multimedia-based) and (iv) very *few records* for each individual descriptor are available.

Indeed, this difference makes the direct use of classical correlation methods *almost irrelevant* for tasks such as analysis of terrorist activities. To make classical correlation relevant, an intensive (and non-trivial) preprocessing is needed to generate a large number of values of a few numeric variables. Note that this process is very task specific and may not scale well. It is obvious that the success and generality of such an endeavor may be questionable. This consideration shows that a generalized concept of correlation that can handle complex objects and events is very desirable and not trivial.

1.3 Visual correlation categories

Current visual correlation practices and methods go beyond classical correlation and vary in their *level of the exact presentation* of correlation to the user. We distinguish the following presentation levels:

- **High level**. A system identifies exact and clear correlation between O/Es in advance as a part of the design stage.
- **Medium level.** A system does not correlate objects in advance, but provides a user with interactive tools such as a curve-matching cursor.
- **Low level**. A system mostly relies on human perception, providing similar graphical or multimedia presentations of correlated entities and some pointing mechanisms.

Another important feature of a VC environment is *level of knowledge* of correlation. Is the correlation already known or has it been previously discovered? In classical visualization, it is typically assumed that the correlation has already been discovered. In visual data mining, it is assumed that correlation needs to be discovered. Thus there are two major categories of visual correlation:

(VC1) **Classical visualization** -- visualization of existing correlations between objects and events and

(VC2) **Visual data mining** -- a process of finding correlations visually between O/Es.

For example, in VC1, we assume that a correlation such as $y = x - 100$ between human height in cm (x) and weight in kg (y) has already been found and is then visualized as a plot. On the other hand, in VC2, the correlation $y = x - 100$ should be discovered visually from a plot. Clearly, one can also imagine spectrum of possible combinations of VC1 and VC2. Both VC types can be combined with different levels of exact definition and presenta-

tion of correlation. The chance of building specific combinations are illustrated in Table 3 with more smilie faces, ☺, indicating a better chance. The best chance of building a successful visual correlation system arises when the relations to be visualized are known and the system relies mostly on a human perception and provides similar graphical or multimedia presentation of related O/Es along with some pointing mechanisms. This is depicted with a high value of five faces. The most challenging task is to build a system that would be able to present correlations visually that are not known yet and need to be discovered from given data. This option is shown as an empty cell in Table 3.

Table 3. Types of visual correlations

Level of correlation knowledge	***Level of exact presentation of correlation***		
	High	*Medium*	*Low*
Correlation is already known (Classical Visualization, VC1)	☺ ☺ ☺	☺ ☺ ☺ ☺	☺ ☺ ☺ ☺ ☺
Correlation is not known yet (Visual Data Mining, VC2)		☺	☺ ☺

The goal of *visualizing known relations* (VKR) is assisting *an unaware individual.* The goal of *discovering unknown relations* visually (DRV) is *assisting everybody*, because everybody is unaware in this case. For instance, if the mathematical linear relation $y = ax + b$ is known, then a specific ignorant individual will benefit from its visualization as a straight line plot. Otherwise, if this relation is not previously known, everybody would benefit.

The fundamental difference between VC1 and VC2 is in the *level of possible guidance.* For a known relation, it is possible to guide an ignorant person to see the relation through visualized data. The situation is quite different for unknown relations. Who can guide the understanding of the unknown relation? Thus, the classical visualization task, VC1 is more specific and should be analyzed first. The progress in technology for this task will also help form a solid base for visual data mining, VC2.

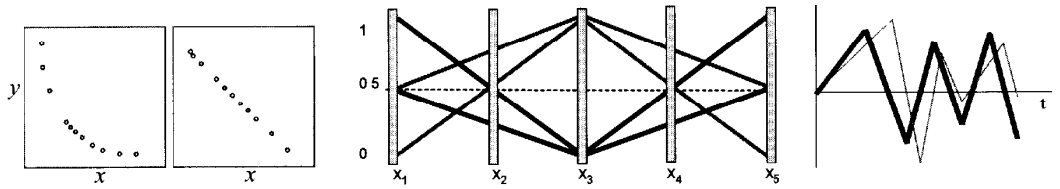
2. EXAMPLES OF NUMERIC VISUAL CORRELATIONS

2.1 High-level numeric visual correlation

In this section, we provide examples of high-level visual correlation for numeric data that require little guidance for a perceiver. In all examples, it is

assumed that the relation to be visualized is already known and available for the system. The software system computes and visualizes the relationship in a single VC panel based on user's data. The user's role is relatively passive and involves *evaluating the VC* without generating alternative visual correlations.

Our first example is classical Linear or Curvilinear correlation using **Cartesian coordinates** as shown in Figure 2(a). Initially, it may seem that the relation to need not be known in advance; that it can be discovered visually by observing the plot. While it is true that the plot can be used for discovery, that is a different role - not communicating a discovered relation to an ignorant person. In general, the same visualization may or may not serve both functions.



(a) Linear and Curvilinear Cartesian correlation plots of two variables.

(b) Parallel coordinate correlation plot of five variables.

(c)Cartesian correlation of two numeric variables vs. time.

Figure 2. Cartesian and Parallel correlation plots for homogeneous numeric variables

The next example is a **Parallel coordinate** plot [Inselberg, 1997], that uses Cartesian coordinates differently by locating them in parallel vertically. Figure 2(b) shows five parallel coordinates x_1, x_2, x_3, x_4, and x_5. In parallel coordinates, every 5-dimension O/E such as (0.5, 0.75, 1.0, 0.75, 0.5) is not a point in a 5-dimensional Cartesian coordinates but rather a line connecting values. Object (0.5, 0.75, 1.0, 0.75, 0.5) is shown in Figure 2(b) as the top dark line. Cartesian visualization is limited by 2-D and 3-D but parallel coordinate visualization in a truly multidimensional. Figure 2(b) shows 5 coordinates and we still have space to put five more coordinates. This visualization is efficient if O/Es have well-distinguished clusters such as the "convex" and "concave" clusters shown in Figure 2(b).

Figure 2(c) displays a typical correlation of **time series**. Each time series is a 2-dimensional O/E. The visual correlation of two 2-dimensional O/Es with a shared domain (usually time, t) is possible by overlaying two Cartesian. The time series correlation presented in Figure 2(c) can also be viewed as a parallel coordinate correlation. It is sufficient to consider $x(t)$ for $t = 1, 2, \ldots, n$ as a set of separate variables x_t: $x_1, x_2, \ldots, x_n$.

This differs from typical parallel coordinate use where m is much grater than n, m >> n, with m the number of O/Es. In time series correlation it is just the opposite, $n >> m$. That is, we correlate 2-3 time series with hundreds

of time measurements. This leads to another reason for visualizing differently. Having hundreds of parallel time coordinates x_t, the space between them is very small and often there is no need to draw lines connecting x_t and x_{t+1}.

Note that all thereof the visualizations presented could also used for discovering relations between O/Es.

2.2 Medium-level numeric visual correlations

With medium-level VC, an exact correlation match is not provided and, using the visualization, the user needs to be able to explore and discover an appropriate relation.

For example, a VC might present a user having many variables, x_1, x_2,..., x_n with many correlation plots of different x_j and x_j pairs. The user would then be able to identify potential correlations from the plots. These plots are typically organized on a grid. Figure 3 display an example where several of the plots exhibit clear correlation.

In the VC, some correlations would be left for the user's perception, recognition, and discovery, while other correlations would be pointed out explicitly. Clearly, the software would visualize n^2 pairs when n variables are used. The user's role in the VC is active as the user is responsible for selecting variables for the VC, performing evaluation in the VC, and selecting interesting relations out of those presented.

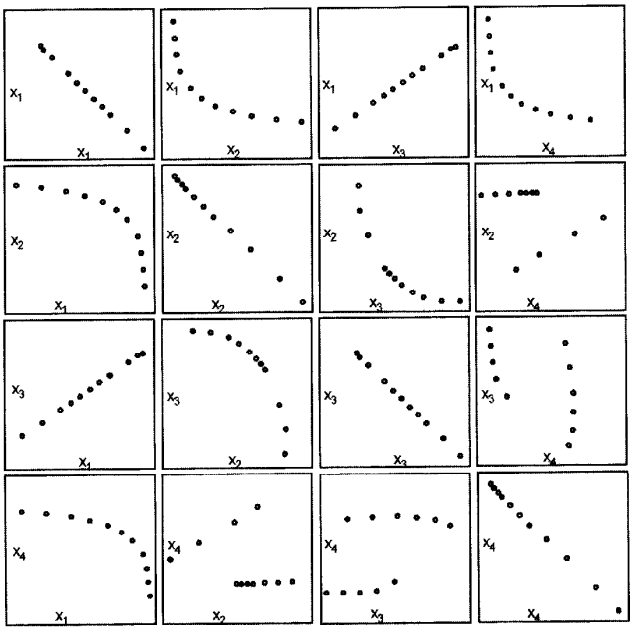


Figure 3. A grid panel of the pairwise correlation for four homogeneous numeric variables

Figure 4 shows another visual correlation method known as a **Table Lens** [Rao & Card, 1994, 1995; Pirolli & Rao, 1996]. This is a table of individual distributions of numeric variables shown side-by-side. Users are also active in this VC. In the Figure 4, the user can visually correlate distributions by noticing that first and forth distributions are similar.

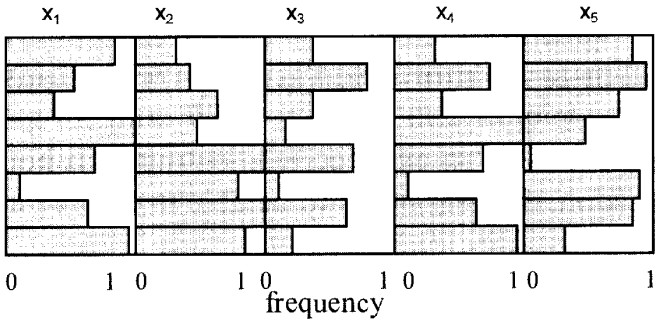


Figure 4. Table Lens: a table of individual distributions of numeric variables

The next visual correlation method we turn to is a **3-D Visualization Spreadsheet** with multiple visualizations. Here rows represent different objects or the same object at different times and columns represent alternative visualizations for each object. One way *alternative visualizations* can be produced is by showing the object from viewpoints that might reveal different aspects of the object and thus permit correlation. Indeed, if the object represents an actual 3-D object then the projections have a physical interpretation.

A more general correlation option arises when the object displayed is a glyph that encodes attributes of some other object. As an example, suppose the object to be represented has six numeric variables x_1, x_2, ..., x_6. These variables could be encoded as characteristics of a visual glyph, here a pyramid, as follows:

x_1 - height of the pyramid,
x_2 - width of the pyramid,
x_3 - color of the side 1,
x_4 - color of the side 2,
x_5 - color of the side 3, and
x_6 - color of the side 4.

In the VC, every cell, row, and column can be graphically manipulated simultaneously. Columns might also encode very different characteristics. For instance, columns 1 and 2 might present glyph while columns 3 and 4 can present bar charts of statistical distributions of the parameters.

This idea was used in [Chi, 1999] for finding interrelationships and dependencies among variables for a pattern recognition problem and for protein analysis.

Figure 5 demonstrates the 3-D spreadsheet with an object measured at three different times encoded as a glyph with 20 variables. Each column shows five of these variables.

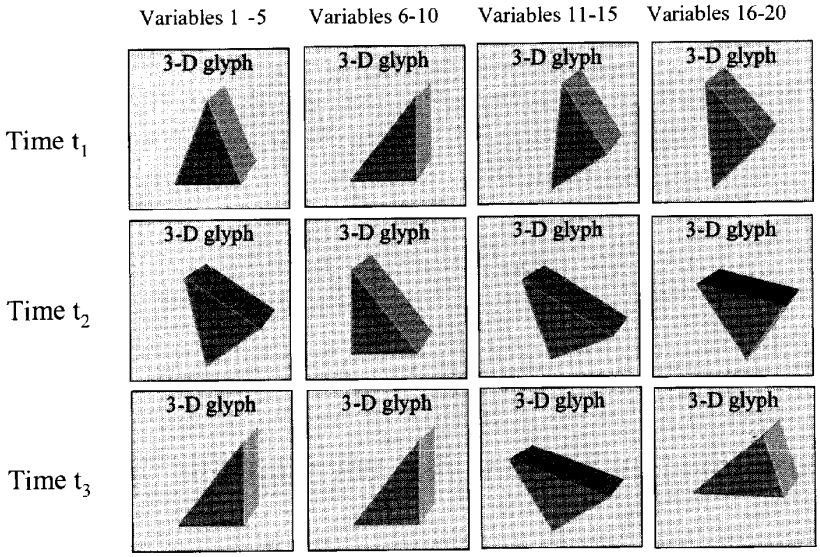


Figure 5. A 3-D multiview visual correlation spreadsheet

Dynamic Interactive Pointers. In this visual correlation method the system designer provides software for displaying and interactively linking two side-by-side panels. The user correlates these panels with a curve-matching cursor. Color strips between the links correlates objects (see Figure 6). This idea has been used in geology for spatial interwell correlation as an extension of the method of respective correlation in geology [Haites, 1963].

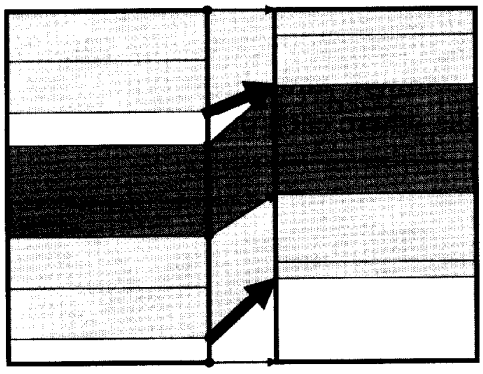


Figure 6. Dynamic Interactive Pointers

Dynamic object visual correlation. This VC permits a user to explore the dynamics of an object encoded as a shape (glyph) that represents n variables of the object at a given time t. Dynamic changes are presented using pointers for successive times such as t+1, t+2. Similar dynamics may be used for spatial movement of an object from location s_1 to successive locations such as s_2, s_3. Figure 7 depicts this VC.

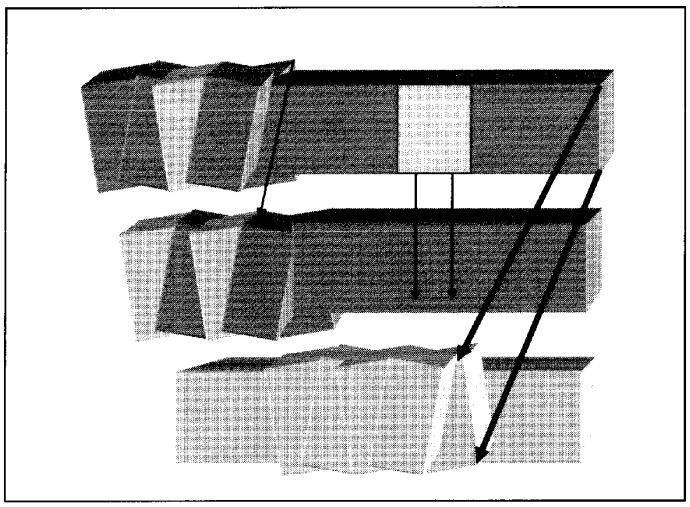


Figure 7. Dynamic object visual correlation

2.3 Low-level numeric visual correlations

In the low-level VC, a user visually correlates objects. Such correlation may be unstable because it depends on the scaling of numeric variables. Scaling may relocate and change visual objects used to represent and discover patterns. In 3-D Glyph correlation *each object is described as n-dimensional string of* numeric attributes that are mapped into 3-D boxes or glyphs in a single panel. Colors, sizes, orientations, and shapes of boxes are used to represent numeric attributes. The system *designer* provides software, which visualizes this data. The user selects variables for VC and visually correlates objects (boxes) using natural perception (see Figure 8).

Shape glyphs. Every shape encodes several parameters of the data via its color, height, width, rotation, and shape type. This approach has been used with a variety of tasks where each object can be represented by an individual shape glyph.

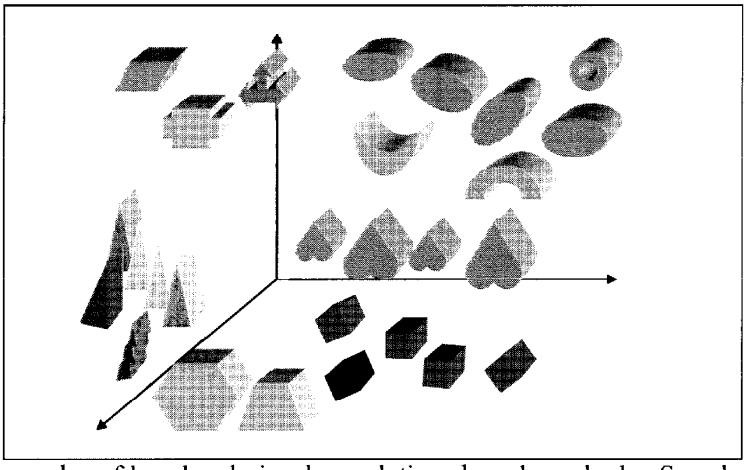


Figure 8. Examples of low-level visual correlations based on glyphs. See also color plates.

2.4 Examples of multi-type visual correlation

Pointer method. Multi-type and multi-source heterogeneous visual correlation is needed when we try to correlate objects with non-numeric attributes, with a mixture of numeric and non-numeric attributes, and with attributes from different layers of attribute hierarchies.

Figure 9 depicts multi-source heterogeneous example of high-level visual correlation, where image and text descriptors are correlated by pointers and commands are correlated by posting their list side-by-side with an image.

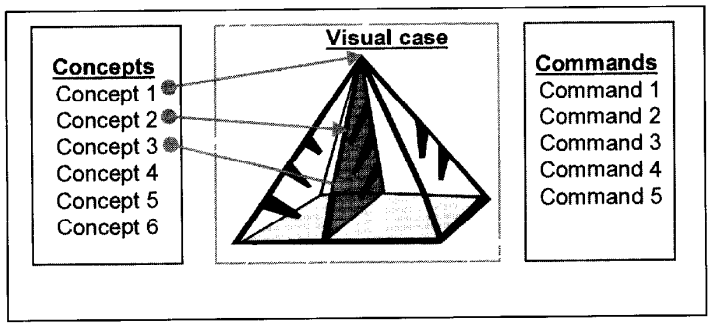


Figure 9. Pointer-based visual correlation for multi-type data

A system designer can link several heterogeneous panels in advance, for example: an image, text descriptors, and commands. A user would then apply pointers to correlate text concepts with the visual image. This idea was implemented in [Novak, 1995]

Rainbow correlation. Figure 10 demonstrates a multi-source heterogeneous high-level visual correlation. Entities (documents, people or concepts) are represented as small dots on a two-dimensional plane connected by colored arcs above and below the plane. Relationships are shown by the location of the dots, arcs and their colors [Hetzler, Harris, Havre & Whitney, 1998].

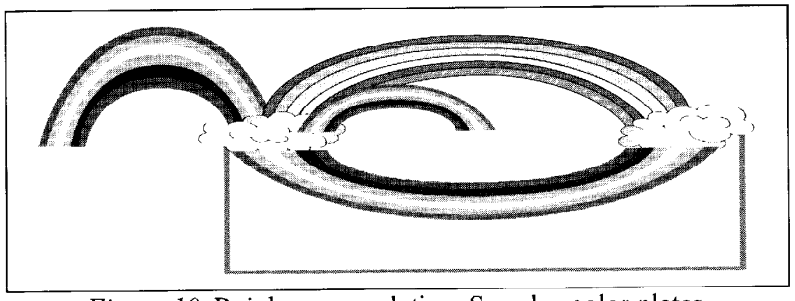


Figure 10. Rainbow correlation. See also color plates.

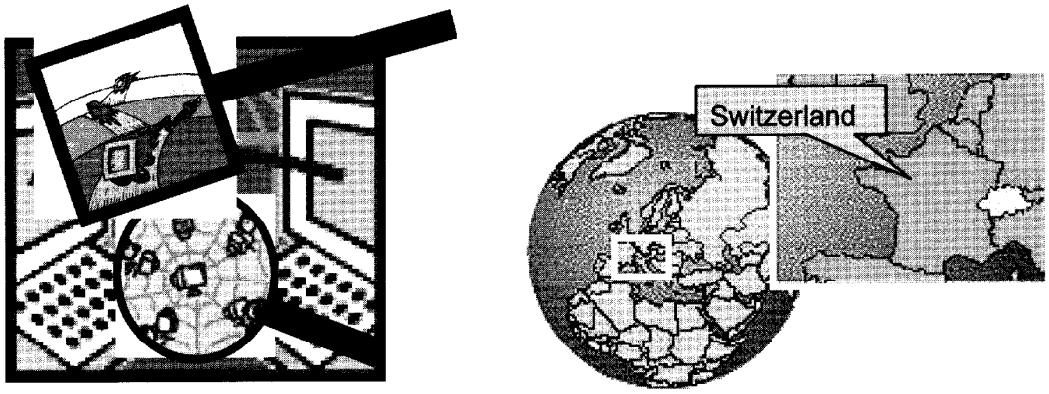
Metaphoric spatial visual correlation. In this correlation multidimensional data (objects, events) are represented in 2-D or 3-D as points in specific locations using variety of dimension reduction techniques. Related

O/Es are located nearby or as "galaxies." Users can list locations and events of interest and then use the *correlation tool* to quickly identify an events' location. This idea is implemented in *self-organized maps* (SOM) initiated by Kohonen via techniques called Galaxies and ThemeView [Hetzler & Miller, 1998]. Figure 11 depicts this correlation method.

Figure 11. Metaphoric spatial visual correlation methods for multi-source data

Geospatial visual correlation. Heterogeneous geospatial data are correlated using a variety of methods, one of them is known as the **Magic Lens.** In this method the user selects an area for magnification and the system reveals magnified objects using different display metaphors and layouts using query and text modes. See Figure 12(a). Typical magic lens systems support hierarchical views, see for example [Shaffer & Reed, 1999].

Linked panels is another widely used method of visual correlation. See Figure 12(b). Panels of different levels are linked by an inserted rectangular for a region and by a pointer for a country.

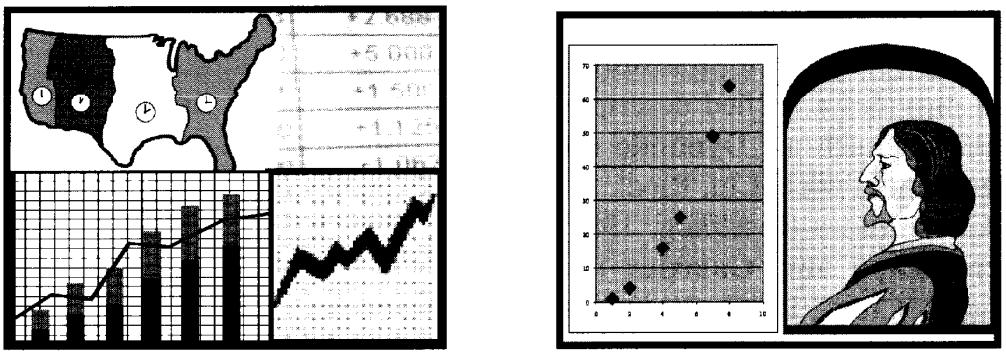


(a) Magic lens (b) Linked panels

Figure 12. High-level geospatial visual correlation for objects of different levels of resolution. See also color plates.

A variety of multi-source **medium- and low-level visual correlations** are also in use. Customizable, coordinated, side-by-side panels (Snap-Together Visualization) are among them. The user visually correlates contents of several panels. Users query their relational database and load results into a desired visualization. They then specify how to coordinate the various

visualizations when selecting, navigating, or re-querying. Some correlations are left to user's perception, recognition, and discovery. Some correlations are pointed out explicitly [North, 2000]. See Figure 13(a).



(a) (b)

Figure 13. Medium- and low-level visual correlations of heterogeneous data

Similarly, another **side-by-side visual correlation** for heterogeneous data has been implemented by combining graphics and images. Working with a setup such as Figure 13(b), a user could correlate an image of points with the portrait by recognizing the inventor of Cartesian coordinates with an image of those coordinates. The correlation is carried out by a comment such as "Descartes R., 1596-1650 and a visual correlation in Cartesian coordinates."

3. CLASSIFICATION OF VISUAL CORRELATION METHODS

Metaphors for visual correlation tasks. Appealing to common knowledge via metaphors for discovering and displaying correlations may both speed up and make visual correlation easier. Several examples of this type metaphor were shown in the previous section, recall rainbow correlations.

Table 4. Metaphors for visual correlation tasks

Task	*Metaphor*	*Familiar knowledge*
Visual correlation	Spreadsheet of icons	Table, Kids' puzzle cubes
	3-D trees of icons	Orchard work
	Room with floor, walls, and ceiling displaying parts of the task	Spatial structure of building
Learning visual correlation tools	Travel in iconic world	Tours, guides, navigation
Collaborative visual correlation work	Multi-agents	Travel agents

Table 4 lists more metaphors that can be used in *visual correlation tasks*. Two of them were implemented in Bruegel project (see chapter 10); these were *spreadsheet of icons* and *3-D trees of icons*.

Classification. Chi & Riedl [Chi & Riedl, 1998] offer a classification scheme for data visualization methods. This is an *"internal" classification* based on operations for transforming data into the visual form.

It provides a unified description of many well-known data visualization techniques such as: **Dynamic Querying** [Ahlberg & Shneiderman, 1994], **AlignmentViewer** [Chi, Riedl, Shoop, Carlis, Retzel & Barry, 1996], **Parallel Coordinates** [Inselberg, 1997], **SeeNet** [Becker, Eick & Wilks, 1995], **ThemeScape and Galaxies** [Wise, Thomas, Pennock, Lantrip, Pottier, Schur & Crow, 1995], **Hierarchical Techniques** : Cone tree [Robertson, Mackinlay & Card, 1991], Hyperbolic Browser [Lamping, Rao & Pirolli, 1995], TreeMap [Johnson & Shneiderman, 1991], Disk Tree [Chi et al., 1998], **Perspective Wall** [Mackinlay & Robertson, Card], **WebBook and WebForager** [Card, Robertson & York, 1996], **Table Lens** [Rao & Card, 1994, 1995], **Time Tube** [Chi et al., 1998], **Spreadsheet for Images** [Levoy, 1994], **FINESSE** [Varshney & Kaufman, 1996], **Spreadsheet for Information Visualization** [Chi, Barry, Riedl & Konstan, 1997].

Visual correlation is in need of an *"external" classification* scheme that reflects the goal of supporting the correlation of complex objects and events. This means that we would classify methods based on how they present correlated objects to a user and less on how the visualization has been obtained from original data.

Examples presented in the previous section served as the basis of the classification system shown in Tables 5 and 6. We distinguish types, subtypes and individual visual correlation methods. Here some types are the same as individual methods.

Table 5. Classification of visual correlation methods: simple structures

VC Type	*VC subtype*	*Visual Correlation method*
Single panel	Points and lines for a single dataset	Linear correlation plot
		Curvilinear correlation plot
	Points and lines for multiple datasets	*n* visualized entities in the single panel.
	Glyphs	2-D glyphs correlation
		3-D glyph correlation
		Mix of 2-D and 3-D glyphs
Line of panels Side-by-side	Static pointers	Panel contents linked by pointers
	Dynamic interactive pointers	User sets up links interactively
	n panels side-by-side	*n* abstract visualizations side-by-side.
		n real-world pictures side-by-side (e.g., *n* X-ray films side-by-side)

Table 6. Classification of visual correlation methods: complex structures

VC Type	*VC subtype*	*Visual Correlation method*
Tree of panels	Vertical tree of panels	Vertical tree of panels
	Horizontal tree of panels	Horizontal tree of panels
	Centered tree of panels (root in the center)	Centered tree of panels (root in the center)
Grid of panels (spreadsheet)	Table of n x n panels.	Grid of correlation and distribution plots
Network of panels	Static pointers	Static pointers
	Dynamic interactive pointers	Dynamic interactive pointers
	Side-by-side panels	Side-by-side panels
Nested panels	Nested panels for hierarchical views	Nested geographic maps and events
Panels in 3-D	Mountain panel	Mountain panel
	Fish eye	Fish eye
	Room	Room
	Gallery	Gallery
	Cone/disk tree	Cone/disk tree
Zooming and popping up panels	Standard zooming	Geographic map zooming (2D or 3D)
	Zooming with changing metaphors and layouts	Magic Lens (2D or 3D)
Panels spread over several monitors	Combinations	Combination of above listed methods

4. VISUAL CORRELATION EFFICIENCY

Visual correlation shares many efficiency criteria with visualization in general. Development of such measures is an important part of visualization theory. The problem of measuring information density score (IDS) for visual systems is highly nontrivial while the problem of measuring information density for text is well known and has been studied for a long time.

Claude Shannon described a measure of information, known as **information entropy**, applicable to transmission of information using communication channels. In visual correlation, we have a specific communication channel that of transmitting information from a computer to a human.

Shannon's approach was developed further by [Yang-Peláez , Flowers, 2000] at MIT as a measure of information content for quantifying the relative effectiveness of displays in different visualizations. We build on this approach by developing criteria that may be specifically applied to the evaluation of the visual correlation efficiency. The criteria are presented in Tables 7 and 8. Table 7 presents *time, speed and information density* characteristics and Table 8 considers their relative characteristics.

By comparison, the major characteristics of visual correlation are *text correlation time* (TCT) and *text correlation speed* (TCS), which evaluate the time and speed of capturing the correlation of O/Es presented in text form.

Table 7. Base characteristics of visual correlation efficiency

Characteristic	*Description*	*Comment*
Visual correlation time (VCT)	Time for catching correlation visually.	If VCT is relatively small then VC can be used in time-critical and/or information overloaded tasks
Information density score (IDS)	Amount of information presented visually. IDS is measured for each separate panel and screen and as a sum of them (integral measure). Amount of information can be measured in bits or bytes, Kb, Mb, and Gb.	If IDS is relatively large then VC can handle large applications. Visualization and visual correlation can be viewed as a specialized data compression method (only a human can work with this compressed information). IDS is an indicator of the efficiency of such compression.
Speed of VC (SVC)	SVC=IDS/VCT -- amount of information consumed per time unit.	If VCS is relatively high then VC can be used in time-critical and information overloaded applications

The source of visual correlation efficiency as compared to the correlation text is the higher speed of visual parallel information processing (PIP) when compared with the sequential nature textual analysis. Thus, the speed of visual correlation (SVC) ties in with the speed of parallel information processing (SPIP).

Computing SPIP may require measuring the amount of information processed per time unit in both visual and textual O/E correlation (that is indicated by information density, IDS). However, one can still measure relative time in experiments without explicitly measuring information density of the VC.

Table 8. Comparative criteria for visual correlation efficiency

Characteristic	*Description*	*Comment*
Relative VC time (RVCT)	Time of visual correlation (VC1) relative to another visual correlation: RVCT= VCT1/VCT2; RVCT=VCT1/TCT.	If RVCT is relatively low, then VC can be used in time-critical and information overloaded tasks.
Relative VC speed (RVCS)	Speed of visual correlation (VC1) relative to another visual correlation (VC2) or text TCS: VCS1/VCS2; VCS1/TCS	If RVCS is relatively high then VC can be used in time-critical and information overloaded tasks.

5. VISUAL CORRELATION: FORMAL DEFINITIONS, ANALYSIS, AND THEORY

Challenges. Earlier, we began the discussion of visual correlation challenges including (1) **distortion** of the actual relation R to be visualized and (2) **excessive guidance** required from the user to avoid distortion in Section 1 using by referring to the famous painting of Pieter Bruegel (see Figure 1). Often it is impossible or very difficult to meet both challenges simultaneously. Less intensive guidance can increase misperception, but intensive textual guidance may mean that visualization is not serving its role; namely conveying information more efficiently than using traditional text for communication. In what follows, we provide a more formal discussion of this subject.

5.1 Visualization of known relations

Definitions. Visualization of a known relation *R(a,b)* between O/Es *a* and *b* involves two components:

- a visual representation *V(R(a,b))* of relation *R(a,b)* and
- the relation *Q(a,b)=P(V(R(a,b)))* perceived by a person from the visual representation *V(R(a,b))* .

These components form a natural sequence from relation *R(a,b)* between O/Es *a* and *b* to visual representation of *R(a,b),* and to relation *Q(a,b)* that is percived by person in the process of observing a visual representation of *R(a,b)*. This sequence can be presented in a compact symbolic form:

$$R(a,b) \rightarrow_V V(R(a,b)) \rightarrow_P Q(a,b) \qquad (1)$$

where $\rightarrow_V$ means produced by a visualization design tool, $\rightarrow_P$ means produced by a visual perception mechanism and where *Q(a,b)=P(V(R(a,b)))*.

Ideally, we should have *Q(a,b)* = *R(a,b*). That is, the perceived relation *Q(a,b)* should be the same as relation *R(a,b):*

$$R(a,b) \rightarrow_V V(R(a,b)) \rightarrow_P R(a,b) \qquad (2)$$

Property (2) is the ultimate *goal of visualizing known relations.* Formulas (1) and (2) encode two steps:

(S1): Produce a visual representation of the relation *R*:

$$R(a,b) \rightarrow_V V(R(a,b))$$

(S2): Perceive the visual representation of the relation *R:*

$$V(R(a,b)) \rightarrow_P Q(a,b).$$

Both steps S1 and S2 can produce *corruptions of the relation R(a,b)* and finally produce a relation $Q(a,b)$ that differs significantly from $R(a,b)$; that is, $Q(a,b) \neq R(a,b)$.

Distortion. Below we provide an example of **relation distortion**. Let us consider the simple relation between two numbers a=2 and b=4: $R(2,4)$="the number 2 is two times smaller than the number 4." Figure 14(a) visualizes this relation by matching a and b with the radii of the circles so that $R_b=2R_a$. In this case, the relation Q_1 to be perceived is as follows:

$$Q_1(a,b)=\text{true} \Leftrightarrow r_b=2r_a.$$

Figure 14(b) visualizes the same relation by presenting two circles with areas S_a and S_b where the first area is half the second area, $S_b=2S_a$. In this case, the relation Q_2 is quite different:

$$Q_2(a,b)=\text{true} \Leftrightarrow S_b=2S_a.$$

that is $Q_2(a,b)=\text{true} \Leftrightarrow \pi (r_b)^2=2\pi(r_a)^2$

(a) (b)

Figure 14. Radius and area visualization metaphors

On the other hand, suppose we derive Q_2 from Q_1. Since $r_b=2r_a$ *in* Q_1, we would have area $S_a = \pi r_a^2$ and area $S_b = \pi (r_b)^2 = \pi(2r_a)^2 = 4\pi(r_a)^2 = 4S_a$. Thus, the relation $r_b=2r_a$ for radii is equivalent to the relation $S_b = 4S_a$ when converted to areas. This is double the relation originally expressed by Q_2.

Without guidance, a person does not know what relation to use, Q_1 or Q_2, for comparing alternatives. While the radius relation Q_1 is a correct visual representation for relation $R(a,b)$, without guidance, a person may compare areas Figure 14(a) even without consciously noticing it. As a result, the person might conclude that a is four times smaller than b. This is neither relation Q_1 or Q_2 but rather a third relation Q_3

$$Q_3(a,b)= \text{true} \Leftrightarrow S_b=4S_a$$

Translated into a relation this will produce an incorrect statement namely the number a is one quarter of the number b.

Surely then, Figures 14(a) requires guidance so that radii of the circles are compared. Without this guidance, areas might be compared in which case the actual relation $R(a,b)$ would not be discovered. Indeed, even with correct guidance, a person may still not be able to extract the relation $R(a,b)$ precisely because of misperception.

Misperception. The human *misperception of a characteristic* such as area or radius may cause that the corresponding relation, such as $S_b=2S_a$ between areas of two circles, not to be recognized. In fact, psychological studies [Tufte, 1983] have shown that the perceived area of a circle probably grows somewhat more slowly than the actual (physical, measured) area:

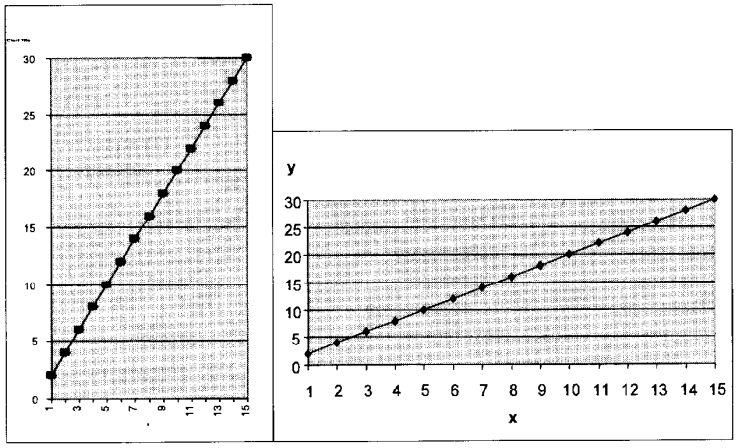
$$\textit{the reported perceived area} = (\textit{actual area})^{0.8\pm0.3}.$$

Line length perception is another known area of Human misperception. Perceived length depends on the context and what other people have already said about the lines [Tufte, 1983; Asch, 1956]. The concept, Lie Factor (LF), was introduced to measure misperception:

$$\text{LF} = (\textit{size of effect shown in graphics})/(\textit{size of effect in data})$$

with limits (LF< 0.95, LF>1.05) for substantial distortion [Tufte, 1983].

Next, we illustrate misleading visual expectations in terms of visualized relation $R(a,b)$ between two data sets $a=\{x\}$ and $b=\{y\}$ where for every x, $y=2x$. Figure 15(a) visualizes this relation while preserving proportion. Such a visualization permits the relation $y=2x$ be discovered. Alternately, Figure 15(b) shows the same relation but with inconsistent, disproportional axes since units on the y-axis are a quarter of the size of units on the x-axis.



(a) (b)

Figure 15. Visualization size effect

Thus, the plot might create a misleading visual expectation, $2y=x$, of a slowly growing of y instead of $2x=y$. Note that axes are correctly marked;

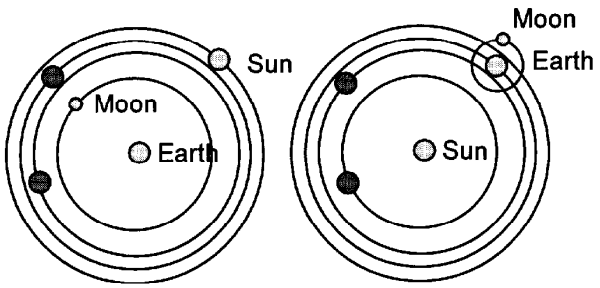
however, a user expecting a uniform scale perceives a much slower growth than is actually depicted.

Misleading visual expectations frequently arise from disproportional data sets in graphics [Tufte, 1983, pp.65-67]:

- a tall (vertical) shape of the plot of monetary spending emphasizes rapid growth;
- a short (horizontal) shape of the same plot suppresses a user's expectations of the rapid growth;
- visual objects located in front of the other objects are perceived as emphasized, towered and larger than others;
- horizontal arrows encourage impression of a stable base;
- arrows pointing vertically emphasize growth.

Visualization of incorrect relations. In the previous examples, guidance could help to avoid the misperception of relations between objects. However, guidance has its *limitations* – the correct relation *R(a,b)* should be known. Ptolemy's geocentric solar system, which depicts incorrect relations R_1*(Sun, Earth)* and R_2*(Sun, Moon)*, not only appeared on the visualization as relation between circles Q_1*(Sun, Earth)* and Q_2*(Sun, Moon)* (see Figure 16(a)) but also presented in the original form of relations R_1*(Sun, Earth)* and R_2*(Sun, Moon)*. It is important to notice that original relations are also visual. Ptolemy believed that he had seen Moon and Sun moving around the Earth every day. Thus, he depicted that visual relation in his geocentric system. This example shows that our steps 1 and 2 are applicable to original relations presented either visually or textually.

Figure 16 also serves as a side-by-side visual correlation of two models, quickly showing their differences (here Mercury and Venus are not named but are shown). This type of visual correlation emphasizes only differences.



(a) (b)

Figure 16. Ptolemy's Geocentric world (a) and Copernicus's Solar system (b)

Approaches. Several approaches have been generated to meet listed challenges [Tufte, 1983]:

- design different graphics for each perceiver in each context,
- design graphics that correct in average for many perceivers,

- use a table instead of graphics for data sets of 20 numbers or less,
- represent graphical objects in proportion to numbers, Lie Factor =1,
- label graphical objects to defeat graphical distortion and ambiguity.

The last three approaches are applicable only for numeric data, the first one requires a description of the context. Of course, identifying average perception requires a lot of psychological studies. An alternate approach is to use *active guidance* for the perceiver instead of passively relying on perceiver's choice.

5.2 A visual correlation model based on intermediate objects

5.2.1 The intermediate object concept

If objects A and B are given by their numeric attributes then they can be correlated by comparing the value of attributes, computing a measure of their closeness, and finally by evaluating that measure. If the measure is high enough then A and B are called correlated. This process can then be visualized in a variety of ways such as those presented in the previous sections.

However for many tasks, objects A and B are not represented directly by their attributes rather relations between A and B that are directly and explicitly recorded in a database. In such cases, *correlation may need to be discovered* from indirect data that can be spread in different records or even different databases. For such tasks, one approach to discovering the correlation between A and B is done by using an intermediate object B'.

This approach can be successful for tasks where discovering some relation R_1 between A and B' and another relation R_2 between B' and B would be simpler than discovering a single relation R between A and B, $R(A,B)$ directly. It is also expected that these two relations R_1 and R_2 can be combined into a single relation between A and B without significant difficulties. Thus, this approach requires the discovery of **an intermediate object** B'. Note that relation R_1 between A and B' and relation R_2 between B and B' can be quite different. This approach has a lot in common with link analysis. The DARPA EELD program [Senator, 2001] is the most intensive recent attempt in this area.

Example. Objects A and B are two terrorist attacks and the goal is to correlate them, that is to find what is common between the attacks. The straight comparison of attributes may not reveal any correlations *useful for decision making* -- preventing new attacks and punishing those who are responsible. More specifically, let a set of intermediate objects $\{B'\}$ be available and say that these objects constitute all communication intercepts in countries where

attacks A and B were committed. It is possible that one of intercepted messages B'_k indicates that a brother of the person X responsible for attack A called to the person responsible for the attack B.

This link permits one to set up a relation between the two attacks. Further surveillance of the brother's communication can potentially help prevent new attacks. The names of perpetrators involved in A and B can be used to search for links/relations in $\{B'\}$. If communication between perpetrators in A and B and their relatives is discovered then a correlation between A and B is established and can be visualized. A variety of visualizations have been developed to support link analysis. This visualization is typically carried out using individual attributes, not through the use of complex objects as we describe below.

5.2.2 Definitions

Definition 1. Two objects A and B from classes $\boldsymbol{A}$ and $\boldsymbol{B}$ are *exactly correlated objects* if there is a homomorphism between them.

Informally *homomorphism* means that relations in A have been matched to relations in B with the same properties, that is the structures of A and B and similar. For more formal definitions see section 5.3 and [Mal'cev, 1973; Kovalerchuk & Vityaev, 2000].

Definition 2. Two objects A and B are *correlated objects* if there is object B' (from class $\boldsymbol{B}$) exactly correlated to A, where B' is produced from B by some mapping F and $B'=F(B)$.

Definition 3. A function $\tau(B,B')$ is called the *difference* between B and B' if τ: $\{B,B'\} \rightarrow [0.1]$ and for every B and B' $\tau(B,B) = \tau(B',B') = 0$.

Definition 4. *Visual correlation* of two correlated objects A and B is a pair of visualizations VC1 and VC2, where VC1 is a visualization of the *similarity* (homomorphism) between A and B' and VC2 is a visualization of the *difference* between B' and B.

Figure 17 illustrates Definition 4.

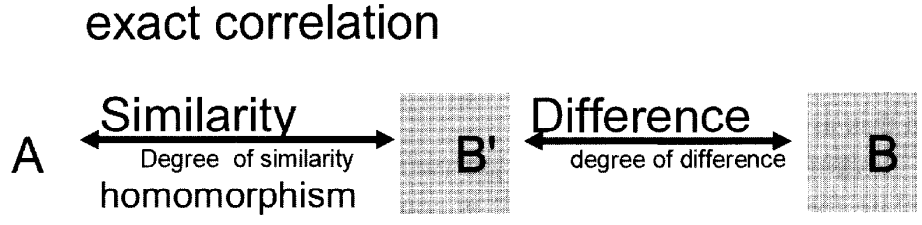


Figure 17. Visual correlation using intermediate object B'

5.2.3 The mathematical structures of objects

Objects A, B and the intermediate object B' may belong to a variety of mathematical structures, such as linear order, lattice, tree, planar graph, directed graph, general graph structure, and Zade's linguistic variable structure. For detailed definitions see [Birkhoff, 1979; Zadeh, 1977; Kovalerchuk & Vityaev, 2000].

According to the definitions in the previous section, objects B and B' should belong to the same mathematical structure $\boldsymbol{B}$ while object A can belong to another structure $\boldsymbol{A}$. In what follows, we give an example demonstrating how a lattice structure A can be correlated to a linear structure B via and intermediate linear structure B'.

Let B and B' be subsets of linear structure Z, $B \subset Z$, $B' \subset Z$, $Z=[0,1]$.

Let the lattice A be a subset of the Cartesian product, $X \times Y$, $A \subset X \times Y$, where $X=[0, 1]$, $Y=[0, 1]$. In other words, $A \subset \{(x,y): x \in X, y \in Y\}$. Further, note that A is *lattice*; that is, upper and low elements are defined for any pair of elements of A by operations $\wedge$ and $\vee$. For instance, for (x,y) and (y,x) if $x<y$ then (x,x) is a lower element,

$$(x,x) = (x,y) \wedge (y,x).$$

In a similar way, the upper element is $(y,y) = (x,y) \vee (y,x)$. In general, upper and low elements may not belong to A.

Let B be mapped to B' by a homomorphism F, $F(B) = B'$, that is B' may contain fewer elements than B. Also let A be mapped to B' by a mapping $M(A) = B'$ such that every (x,y) in A is mapped to z by its largest component, $z = \max(x,y)$. Thus, both A and B can be mapped to B':

$$A \rightarrow B' \leftarrow B$$

by using M and F: $M(A) = B' = F(B)$.

5.3 Algebraic relational approach for defining correlation

In section 5.2.2 we defined the exact correlation of two objects A and B from classes $\boldsymbol{A}$ and $\boldsymbol{B}$ based on the homomorphism between them. This generalized concept is available for describing the correlation of a variety of complex objects and it also permits the description of a classical linear correlation as we show below. The concept of classes $\boldsymbol{A}$ and $\boldsymbol{B}$ has not yet been formally defined. This was done deliberately because the class can be task specific. One of very general concepts of the class derives from abstract algebra that provides the concept of a *relation structure* also known as a *model* and the concept of the *algebraic system* [Mal'cev, 1973]. Another

and the concept of the *algebraic system* [Mal'cev, 1973]. Another even more abstract concept of class can be derived from the mathematics of **category theory** [Marquis, 2004]:

> Category theory now occupies a central position not only in contemporary mathematics, but also in theoretical computer science and even in mathematical physics. It can roughly be described as a general mathematical theory of structures and sytems of structures.

Below we will define and use the concepts of relational structure (model) and algebraic system.

Definition 5. A pair $A=<\boldsymbol{a}, \Omega_a>$ is called a *relational structure (model)*, if $\boldsymbol{a}$ is a set of objects and Ω_a is a set of relations (predicates) P_i on Cartesian products of $\boldsymbol{a}$, such that

$$P_i : \boldsymbol{a}\times\boldsymbol{a}\times\ldots\times\boldsymbol{a} \rightarrow \{0,1\}$$

that is for every vector $(a_1,a_2,\ldots,a_n)$, $P_i(a_1,a_2,\ldots,a_n)=0$ or $P_i(a_1,a_2,\ldots,a_n)=1$.

Definition 6. A pair $A=<\boldsymbol{a}, \Omega_a>$ is called an *algebraic system*, if $\boldsymbol{a}$ is a set of objects and Ω_a is a set of *predicates* P_i and *operators* F_i on Cartesian products of $\boldsymbol{a}$, such that $F_i(a_1,a_2,\ldots,a_n)=a_{n+1}$, that is

$$P_i : \boldsymbol{a}\times\boldsymbol{a}\times\ldots\times\boldsymbol{a} \rightarrow \{0,1\},\ F_i : \boldsymbol{a}\times\boldsymbol{a}\times\ldots\times\boldsymbol{a} \rightarrow \boldsymbol{a}.$$

Thus, a relational systems is a special case of an algebraic system; that is, one without operators. Set Ω_a is called a *signature* of A.

Definition 7. Let us given two algebraic systems $A=<\boldsymbol{a}, \Omega_a>$ and $B=<\boldsymbol{b}, \Omega_b>$, where $\Omega_a=<\{P_i\},\{F_i\}>$ and $\Omega_b=<\{Q_i\},\{G_i\}>$ where P_i, Q_i are predicates and F_i, G_i are operators. A mapping φ: $\boldsymbol{a} \rightarrow \boldsymbol{b}$, from A to B is called a **homomorphism** if for every vector $(a_1,a_2,\ldots,a_n)$

$$P_i(a_1,a_2,\ldots,a_n)=Q_i(\varphi(a_1),\ \varphi(a_2),\ldots,\ \varphi(a_n))$$

$$\varphi(F_i(a_1,a_2,\ldots,a_n))=G_i(\varphi(a_1),\ \varphi(a_2),\ldots,\ \varphi(a_n))$$

In classical linear regression suppose $f(x)$ correlates two ordered arrays $\{x_i\}$ and $\{y_i\}$ such that $|y_i - f(x_i)| < \varepsilon_i$ where value of ε_i can vary for different a_i and b_i. In classical correlation analysis, this function f is called a **regression function.**

The idea of correlation is that by using f we can judge relations in the $\{y_i\}$ by knowing relations in $\{x_i\}$. For instance, if $x_2 < x_5$ then we should be able to say that $y_2 < y_5$ with some level of confidence.

Correlating algebraic systems $A = \langle \boldsymbol{a}, \Omega_a \rangle$ and $B = \langle \boldsymbol{b}, \Omega_b \rangle$ has exactly the same goal. We build a function f from $\boldsymbol{a}$ to $\boldsymbol{b}$ with $|f(a_i) - b_i| < \varepsilon_i$, where ε_i is small or equal to zero and where knowing properties in A we use f to judge properties in B. In classical correlation, the set of such properties is not formulated explicitly. Algebraic systems permit the writing such properties explicitly. For instance, we may want to be sure that additivity in A is preserved in B; that is, we may postulate an additive operator $F(a_1,a_2)$ in A, F: $\boldsymbol{a}\times\boldsymbol{a} \rightarrow \boldsymbol{a}$,

$$F(a_1,a_2)=kF(a_1)+mF(a_2),$$

and an additive operator $\boldsymbol{G}(a_1,a_2)$ in B, G: $\boldsymbol{b}\times\boldsymbol{b} \rightarrow \boldsymbol{b}$,

$$G(b_1,b_2)=sG(b_1)+tG(b_2).$$

If we match a_1, a_2 and b_1, b_2 by a correlation function f this function should also match elements produced by combining these pairs using $F(a_1,a_2)$ and $G(b_1,b_2)$:

$$f(F(a_1,a_2))=G(f(a_1),f(a_2))=G(b_1,b_2)$$

This is exactly the property that is enforced by homomorphism. Thus, the following theorem can be formulated.

Theorem. If there is a homomorphism φ between A and B then $\varepsilon_i = 0$ and thus, A and B are exactly correlated (see definition in section 5.2.2).

If $|f(a_i)-\varphi(a_i)| \neq 0$ then the correlation mapping differs from the homomorphism φ. In this case, the correlation function f serves a role of pseudo-homomorphism. It is also possible that a homomorphism from A to B simply does not exist, that is any mapping from A to B will violate some properties of A. A formal definition of pseudo-homomorphism can depend on specific the properties of systems A and B. For instance, if the properties of A and B are only in a predicate true/false form, then we can measure the number of violations under mappings f and φ. If A and B contain some metric properties, then we can measure how significant the violation is in terms of the distance. The actual choice should depend on a particular application. Thus, pseudo-homomorphism formalizes a *less restricted concept of correlation.* The general concept of homomorphism and pseudo-homomorphism can be applied to a variety of structures with relations and operators.

Example. Suppose we want to correlate human height and weight using a dataset of heights of five people $\boldsymbol{a}=(a_1, a_2, a_3, a_4, a_5)$ and a dataset of weights of the same five people $\boldsymbol{b}=(b_1, b_2, b_3, b_4, b_5)$ to figure out if there is any correlation between weight and height. Let A be a relational structure

$$A= < \boldsymbol{a}, >, \geq_h, P_a(a_1,a_2,a_3,a_4) >,$$

where > indicates that subjects are ordered (indexed) and $\geq_h$ indicates a greater than or equal to relation for human *height*.

Let B be another relational structure

$$B=< \boldsymbol{b}, \geq_w, P_b(b_1, b_2, b_3, b_4) >,$$

where $\geq_w$ indicates a greater than or equal to relation for human *weight*. Here predicates P_a and P_b are the following:

$$P_a(a_i,a_j,a_k,a_m) \Leftrightarrow a_i\text{-}a_j \geq_h a_k\text{-}a_m,$$

$$P_b(b_i,b_j,b_k,b_m) \Leftrightarrow b_i\text{-}b_j \geq_w b_k\text{-}b_m.$$

Let φ map elements of *A* to *B* as follows: $a_i \rightarrow b_i$. We can test if *A* and *B* are homomorphic. Assume that it is true. We need to understand how this can help us in correlating human weight and height. According to homomorphism, if the difference between weights for two people *(i,j)* is smaller than for other two people *(k,m)* then the difference between heights for the first pair is also smaller. If persons *i* and *k* are the same person then we can judge the relation between persons *j* and *m* relative to person *i*. We can state that if *j* is taller than *m* (the difference in height *between j* and *i* is greater than between *m* and *i*), then j is heavier than *m* (the difference in weight between *j* and *i* is greater than between *m* and *i*). Thus, we are able to correlate weigh and height. This also can be converted to a more traditional numeric form, but for many non-numeric relations this is a natural form of correlation and visualization should be able to represent such relations. Visual correlation techniques based on an intermediate element is one potential approach to consider for this.

6. CONCLUSION

This chapter identified the concept of visual correlation. The challenges in visual correlation include how to correlate visually conflicting data and data with different levels of resolution and how to make a "rich" visual correlation for portraying the differences between objects. Visual correlation of objects and events has not yet defined itself as a separate field. It has a lot in common with visualization, visual data mining, statistics, and the general decision-making process. A variety of methods has been developed independently in different fields with little or no communication and without common terminology. A significant amount generalization work should be done. In this chapter, we provided a review, preliminary structure, classifica-

tion and formalizations for visual correlation methods and criteria to assess the quality of visual correlation.

7. ACKNOWLEDGEMENTS

This work has been supported by the US Intelligence Community via an ARDA/NGA grant that is gratefully acknowledged.

8. EXERCISES AND PROBLEMS

1. Discuss differences between concepts of Correlation, Fusion, and Semantic Integration defined in Section 1.2. Provide examples for each category.

2. Discuss differences between three levels of correlation (high, medium and low) defined in Section 1.3. Provide examples for each category. Tip: modify examples presented in Section 2.

3. Expand Table 4 from Section 3 with more metaphors for visual correlation tasks.

4. Provide your own example of relation distortion similar to the one presented in Figure 14 in Section 5.1.

5. Provide an example of visual correlation using intermediate object B'. Your example should be consistent with definitions given in Section 5.2.2-5.2.3 and illustrated in Figure 17.

Advanced

6. Try to visualize an algebraic form of the example presented in Section 5.3. Tip: start from visualization of classical linear regression and visualize algebraic relations (6) and (7) as a part of this exercise.

9. REFERENCES

Asch, S., Studies of independence and submission to group pressure, *Psychological monographs*, 1956

Ahlberg, C., Shneiderman, B., Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of ACM CHI'94 Conference on Human Fac-*

tors in Computing Systems, volume 1 of *Information Visualization*, pages 313–317, 1994. Color plates on pages 479-480.

Becker,R., Eick,S., Wilks, A. Visualizing network data. *IEEE Transaction on Visualization and Computer Graphics*, 1(1):16–28, 1995.

Birkhoff, G. *Lattice theory*, American Mathematical Society, Providence, R.I., 1979

Card, S., Robertson,G., and York, W., The webbook and the web forager: An information workspace for the world-wide web. In *Proceedings of ACM CHI'96 Conference on Human Factors in Computing Systems*, pages 111–117. ACM, ACM Press, 1996.

Chi,E., Riedl, J., An Operator Interaction Framework for Visualization Systems, In: Proceedings of INFOVIS'98, IEEE Symposium on Information Visualization (Research Triangle Park, NC, 19--20 October 1998), IEEE Computer Society Press, 63-70.

Chi, E., Barry, P., Riedl, J., and Konstan, J.A spreadsheet approach to information visualization. In *Proc. Information Visualization Symposium '97*, pp. 17–24,116. IEEE CS Press, 1997.

Chi, E., Pitkow, J. Mackinlay, J., Pirolli, P., Gossweiler, R., and Card, S., Visualizing the evolution of web ecologies. In *Conference on Human Factors in Computing Systems (CHI 98)*. ACM, ACM Press, April 1998.

Chi, E., Riedl, J., Shoop, E., Carlis, J., Retzel, E., and Barry, P., Flexible in-formation visualization of multivariate data from biological sequence similarity searches. In *IEEE Visualization '96*, pages 133–140, 477. IEEE CS Press, 1996.

Chi, H., A Framework for Information Visualization Spreadsheets, Ph.D. dissertation, 1999, University of Minnesota, http://www-users.cs.umn.edu/~echi/phd/chi-thesis.pdf

Foote, T., The world of Bruegel, c. 1525-1569, Time-Life Books, Time, NY, 1968

Haites, T., Some Geological Aspects of the Sydney Coalfield with Reference to the Influence on Mining Operations, Perspective correlation, Bulletin of the American Association of Petroleum Geologists, 47, no. 4 (1963): 553-574.

Hetzler, B., Harris, W., Havre, S., Whitney P., Visualizing the Full Spectrum of Document Relationships, In: Structures and Relations in Knowledge Organization. Proc. 5th Int. ISKO Conf. Wurzburg: ERGON Verlag, 1998. 168-175.

Hetzler, B., Miller, N., Four Critical Elements for Designing Information Exploration Systems. Presented at Information Exploration workshop for ACM SIGCHI '98. Los Angeles, CA. April 1998. http://www.pnl.gov/infoviz/technologies.html#correl

Inselberg, E., Multidimensional detective. In *Proc. Information Visualization Symposium (InfoVis '97)*, pp. 100–107. IEEE, IEEE CS Press, 1997.

Johnson, S., Shneiderman, B. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proc. IEEE Visualization '91*, pages 284–291, Piscataway, NJ, 1991. IEEE.

Kovalerchuk, B., Vityaev E., Data Mining in Finance: Advances in Relational and Hybrid Methods, Kluwer, Boston, 2000.

Loe Feijs, Roel de Jong, 3D visualization of software architectures, Communications of ACM, vol. 41, N. 12, pp. 73-78., 2000.

Mal'cev, A.I. Algebraic Systems, Springer, 1973.

Marquis,J-P. Category Theory, Stanford Encyclopedia of Philosophy, 2004, http://plato.stanford.edu/entries/category-theory, 2004

Lamping, J., Rao, R., and Pirolli, P., A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1 of *Papers: Information Visualization*, pages 401–408, 1995.

Levoy. M., Spreadsheet for images. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, volume 28, pages 139–146. SIGGRAPH, ACM Press, 1994.

Mackinlay, J., Robertson, G., and Card, S., The perspective wall: Detail and context smoothly integrated. In *Proceedings of ACMCHI'91 Conference on Human Factors in Computing Systems*, Information Visualization, pages 173–179, 1991.

Marsh, H.S., Decision Making and Information Technology, Office of Naval Research, 2000, http://www.onr.navy.mil/sci_tech/information/docs/dec_mak_it.pdf

Mikulin, L., Elsaaesser, D., The Data Fusion and Correlation Techniques Testbed (DFACTT) http://stinet.dtic.mil/ Report , 404576, Defense Research Establishment Ottawa, 1994

North, C., A User Interface for Coordinating Visualizations based on Relational Schemata: Snap-Together Visualization, University of Maryland Computer Science Dept. Doctoral Dissertation, May 2000.

Novak, G. Diagrams for Solving Physical Problems, in J. Glasgow, N. Narayanan, and B. Chandrasekaran, eds., *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, AAAI Press / MIT Press, 1995, pp. 753-774.

Pirolli, P., Rao: R., Table Lens as a Tool for Making Sense of Data, 1996, http://citeseer.nj.nec.com/cache/papers/cs/25475/http:zSzzSzwww.parc.xerox.comzSzistlz SzgroupszSzuirzSzpubszSzpdfzSzUIR-R-1996-06-Pirolli-AVI96-TableLens.pdf/pirolli96table.pdf

Rao, R., Card, S., The Table Lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 1 of *Information Visualization*, pages 318–322, 1994. Color plates on pages 481-482

Rao, R., Card, S., Exploring large tables with the table lens. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 2 of *Videos*, 403–404, 1995.

Robertson, G., Mackinlay, J., Card, S., Cone trees: Animated 3d visualizations of hierarchical information. In Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems, Information Visualization, pages 189–194, 1991.

Senator, T., 2001, EELD program, http://www.darpa.mil/ito/research/eeld/EELD_BAA.ppt

Shaffer E., Reed D., Whitmore S., Shaffer B., Virtue: Performance visualization of parallel and distributed applications, Computer, v. 12, 1999, 44-51.

Tufte, E.R., Visual Display of Quantitative Information, Graphics Press, 1983, 1992

Varshney, H., Kaufman, A., FINESSE: A financial information spreadsheet. In IEEE Information Visualization Symposium, 70–71, 125, 1996.

Wise, J., Thomas,J., Pennock,K., Lantrip, D., Pottier,M., Schur, A., and Crow, V., Visualizing the non-visual: Spatial analysis and interaction with in-formation from text documents. In Proc. Information Visualization Symposium (InfoVis '95), 51–58. IEEE, IEEE CS, 1995.

Yang-Peláez, J., Flowers, W., Information Content Measures of Visual Displays, Proceedings of the IEEE Symposium on Information Visualization 2000, IEEE, pp.99-104. http://www.computer.org/proceedings/infovis/0804/08040099abs.htm.

Chapter 9

ICONIC APPROACH FOR ANNOTATING, SEARCHING, AND CORRELATING

Boris Kovalerchuk
Central Washington University, USA

Abstract: This chapter presents the current state-of-the-art in iconic descriptive approaches to annotating, searching, and correlating that are based on the concepts of compound and composite icons, the iconic annotation process, and iconic queries. Specific iconic languages used for applications such as video annotation, military use and text annotation are discussed. Graphical coding principals are derived through the consideration of questions such as: How much information can a small icon convey? How many attributes can be displayed on a small icon either explicitly or implicitly? The chapter also summirizes impact of human perception on icon design.

Key words: iconic representation, compound icon, composite icon, iconic query, iconic sentence, graphical coding

1. INTRODUCTION

The application of iconic descriptive approaches and languages have proven useful for annotating, searching, and correlating traditional databases, and those containing images and multimedia [Chang at al., 1987; 1989, 1994; 1996; Davis, 1995]. In this section, we provide an overview of state-of-the-art in this arena.

1.1 An overview of Media Streams

We begin this review with a look at Media Steams, a system developed at MIT Medial Laboratory [Davis, 1995] for video annotation. This system contains about 3000 predefined icons. The icons are organized into a seman-

tic hierarchy (ontology). These icons serve as building blocks for **ordinary compound icons**, which are comprised by as many as *three icons placed side-by-side.* Table 1 shows icons that are similar but not identical to those used in Media Streams since our goal is only to present Media Streams conceptually. In essence, a user selects the desired icons from a semantic hierarchy of icons to builds an iconic sentence. Media Streams does not permit complex combinations of icons such as superimposing one icon on another with possible resizing or color change. Each concept is encoded in an icon and more complex concepts such as "two cars' are encoded using two icons "car" and 'two". Similarly "three blue birds" are encoded as "bird" and "blue three" and "two adult female dentists" are represented by three icons "adult female", "dentist" and "two".

Table 1. Examples of ordinary compound icons

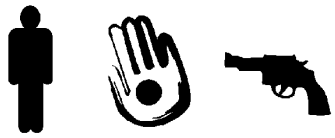
Ordinary compound icon	*Description*
	Two cars
	Three blue birds
	Two adult female dentists

An **annotation** is defined as a graphical descriptor of a segment of the video content, comprised of *a compound icon and an adjacent color bar* that indicated the end and the length of the segment. The color of the bar corresponds to the colors of the compound icon (see Figure 1).

Figure 1. Ordinary compound icons "two cars" and 'three birds: with their time lengths.

The **semantic hierarchy of icons** includes characters, objects, screen positions, relative positions, character actions, and object actions. **Glommed Icons** are a type of compound icons that combine up to *three ordinary compound icons across different descriptor hierarchies.* Table 2 provides an example of a glommed icon. In addition, **animated icons** can be used to express *character actions* or *object actions* dynamically.

Table 2. Examples of glommed compound icons

Gloomed compound icon	*Description*
	An adult male using his hand to operate a gun (black circle in hand indicates an object)

1.2 The iconic annotation process in Media Stream

The iconic annotation process in Media Streams consists of three major steps:

- *selecting* icons,
- *assembling* compound icons, and
- *filling* lines with selected compound icons.

The first step includes an analysis of an input shot. Table 3 shows an example illustrating this with the statue of an adult male using his hand to operate a gun.

Table 3. Selecting icons, assembling compound icons and filling lines

Selecting icons and assembling a compound icon	*Filling annotation lines*	*Icon description*
 video input		Location icon line: Earth ground icon outdoor icon
	 Jon	Character line: Adult man icon with attached descriptor "Jon"
	 Jon	Character's action icon line: Jon using his hand to operate a gun

A user can select appropriate icons from a hierarchical menu of icons in GUI interface. The first line in Table 3 show a location of the shot with two icons "Earth ground" and "outdoor" selected by the user.

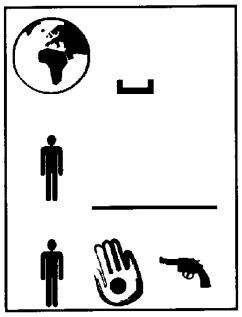
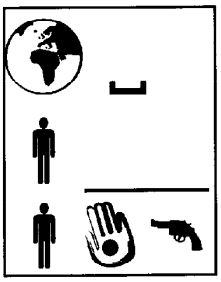
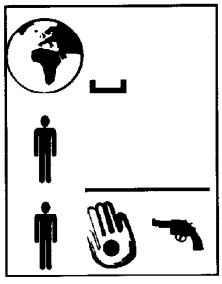
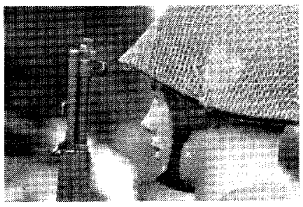
The second icon specifies the first one and sits below it in the ontology hierarchy. The Media Streams time line uses *logarithmic time scale* to shorten description. Filling time lines requires some experience with the system. Usability studies have shown that after two weeks people are comfortable enough to make annotations [Davis, 1995].

2. ICONIC QUERIES

An iconic query system built in [Narayanan & Shaman, 2002] implements a restricted subset of SQL commands for querying a database. The query language's terms (such as verbs, adjectives) are represented by icons. To construct a query, the user composes structured iconic expressions according to the grammar of the iconic language. The language and grammar include rules for iconic constructions based on combined icons.

The goal of such iconic queries is retrieving information from *traditional text-based databases*. Complex query design is often beyond the skills of ordinary users and users with disabilities who cannot write. Iconic queries can permit these users retrieve information from a database without the aid of a programmer. There is also hope that an experienced user can assemble and debug iconic queries faster than text-based SQL commands. The same reasons apply even more for multimedia databases.

Table 4. Example of iconic query and answer retrieval

Visual Query	*Descriptors*	*Best matched iconic result*	*Best matched video shots*
	Location: outdoor Character: adult man Action: man operates gun by hand	 	 shot 1  shot 2

An iconic query is a "native way" of querying multimedia databases. In multimedia, iconic queries support searching the space of iconic annotations for those icons annotating multimedia that satisfy the search condition.

Below we illustrate an iconic search process in the Media Streams and Bruegel systems. We begin with the Media Streams approach assuming that the compound icons shown in Table 4 represent an iconic query to find shots in an image database that show "An adult male using his hand to operate a gun."

The first column in Table 4 shows a sample iconic query: "Find a shot: (1) outdoor, (2) an adult male (3) using his hand to operate a gun." The best-matched iconic results shown in second the column, both are identical with the query, and the last column shows two best-matched video shots. There is a complete match for both shots with this iconic query. Probably a more sophisticated query would differentiate between a real male and a statue. Table 5 shows a sample query without outdoor requirement.

Table 5. Examples of iconic queries

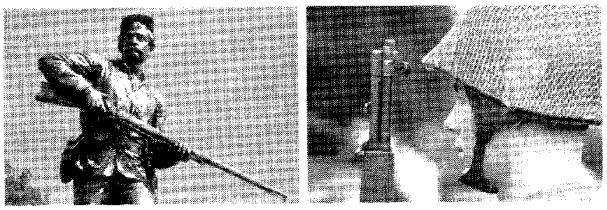
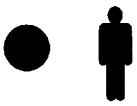
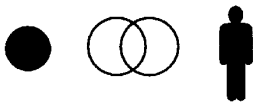
Text and iconic query	*Search results*
An *adult male* using his *hand* to operate a *gun*	

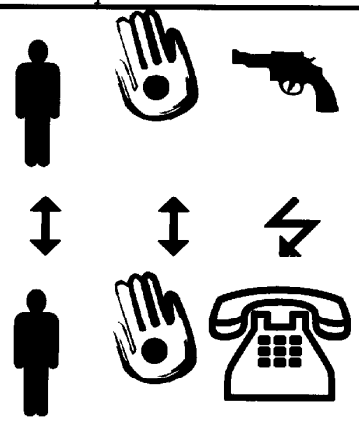
Table 6. Complex queries

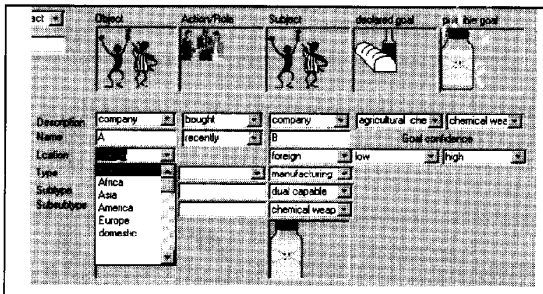
Visual query	*Content*
	OR query Find a compound icon that contains an object OR a character
	AND query Find a compound icon that contains an object AND a character
	Time overlap query Find a compound icon that contains the object, which is temporally-overlapping with a character

Media Streams permits complex queries produced by using AND, OR and Time-Overlapping operators. Table 6 shows some examples. Media Streams also supports queries with exceptions learned from previous retrievals. Table 7 shows a standard iconic query "Find an adult male using his hand to operate a gun" OR exception: "Find an adult male using his *hand* to operate a telephone." Pointer links are used to set up a query with exception.

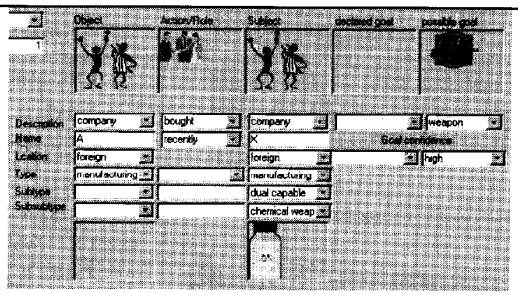
The Bruegel system presented in Chapter 10 specifically supports iconized records and queries that are designed as an extension of MS Access (see Figure 2). Further, this system intends to support both compound and composite icons in queries and description of actual records. We discuss the Bruegel system and its composite icons in the next section.

Table 7. Example of query with exception

Visual queries	*Comment*
	Standard query: Find an adult male using his hand to operate a gun OR Exception: "Find an adult male using his *hand* to operate a telephone". 



Iconized DB record about relations between companies A and B and their business profile



Iconic query: "Find company X that that was recently bought by company A and that has attributes depicted in icons and text"

Figure 2. Bruegel iconized records and queries

3. COMPOSITE ICONS

Table 8 illustrates the **composite icon** concept used in **Bruegel** system. **Base icons** are used to build composite icons by superimposing one base icon over another base icon. This is the way the "damaged truck" icon is composed. Next this composite icon and a base icon for "uncertainty" are combined to produce other composite icons for the concept "Truck damage questionable." There is a limit to how many concepts can be incorporated in a single composite icon. We discuss this issue below.

Table 8. Examples of Bruegel composite icons. See also color plates.

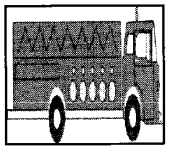
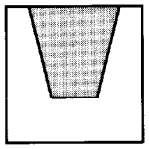
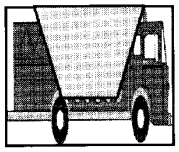
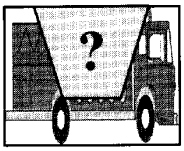
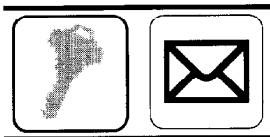
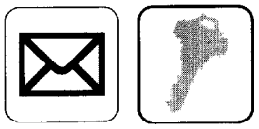
Base icons			*Composite icons*	
				
Truck	Damage	Uncertainty	Damaged truck.	Truck damage Questionable

Table 9 illustrates two composite icons that are generated by different sequences of superimposed icons. The meaning of the icon "the key over the envelope" can differ from the meaning of the icon "the envelope over the key." The first icon can mean a "secure message" and the second icon can mean a "protected security key."

Table 9. Composite vs. compound icons

compound icons	*contents*	*combined icons*
	Secure message	
	Message on security issue	

We use notation A↵B to indicate a composite icon, where icon B is posted on top of icon A, and notation B↵A is used for the composite icon where icon A is posted on top of icon B. Note that the **icon posting operation** ↵ is not commutative.

$$B\leftarrow\!\!\shortmid A \neq A\leftarrow\!\!\shortmid B$$

One of the difficulties of building a composite icon by superimposing one icon over another one is that a part of underlying icon can be obscured. We discuss this issue and the formal *Bruegel visual language* (BVL) below.

The difference between composite icons (Bruegel) and compound icons (Media Streams) is illustrated in Table 9. In Media Streams, "secure message" is iconized by a sequence of two icons. In Bruegel, just one icon is used that takes *less space*.

Table 10 illustrates how four base icons "message," "template," "security," and "ID" can be combined pair wise to produce 16 composite icons with different meanings.

Table 10. Composite icon generation. See also color plates

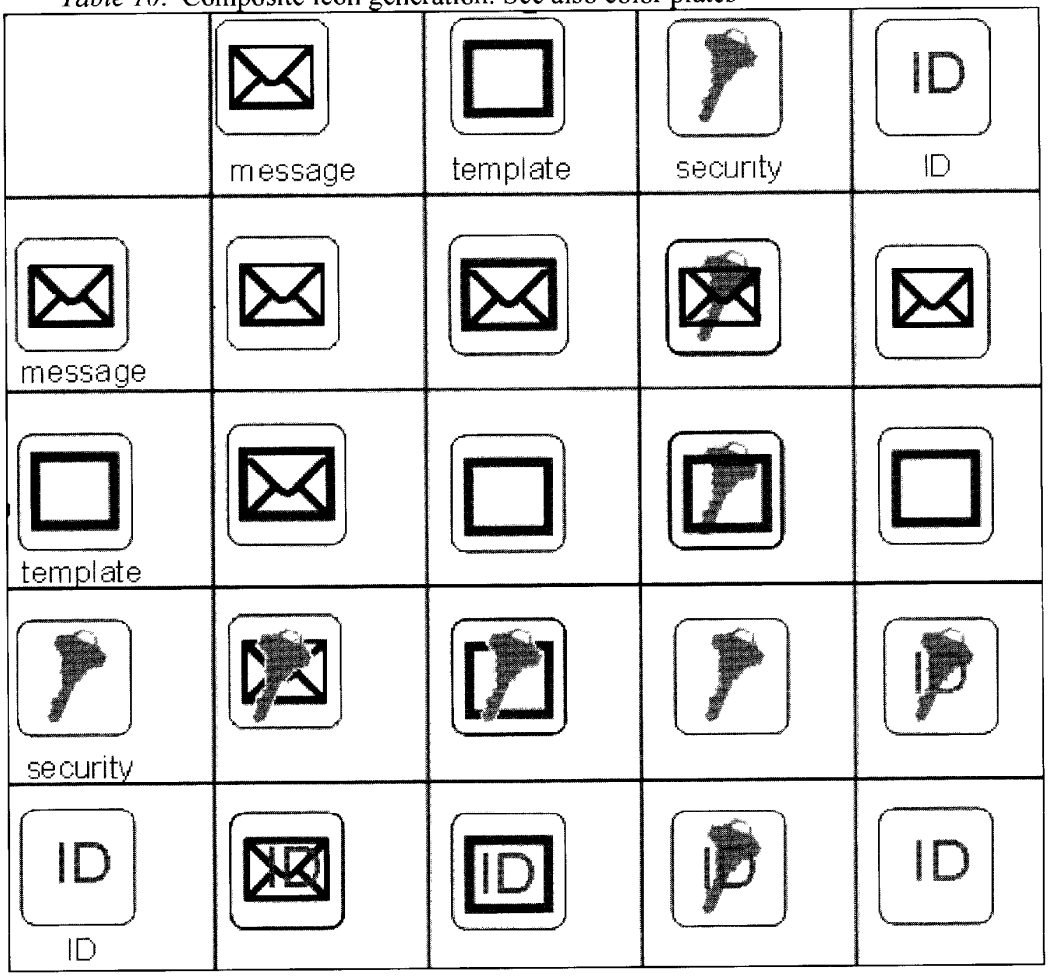


Figure 3 shows a fragment of the iconic language used in Microsoft Windows applications. These icons illustrate the current practice of posting iconic elements (**iconels**) over background icons. In Figure 3, iconels for different aspects of communication are posted on the base icon with two human profiles that represents a communication icon.

				
Request information	Request result unknown	Response positive	Response negative	Request canceled

Figure 3. Posting aspect iconels

The "shortcut" iconel (shown in Figure 4) posted over variety of base icons (appointments, contacts, journal entry, message, note, office document, and task) gives another example of posting iconels.

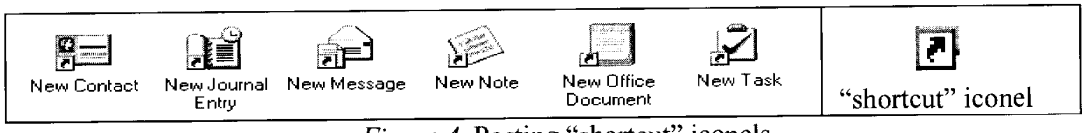
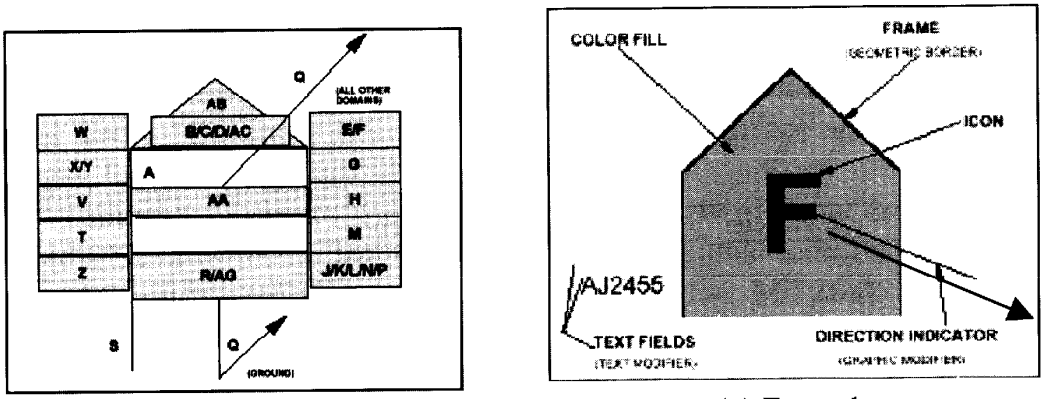


Figure 4. Posting "shortcut" iconels

A thorough analysis of the actual use of icons in Window's Office applications reveals that it is not very consistent in posting iconels. This consequently puts a higher load on a user's memory than it would if consistent options were used.

4. MILITARY ICONIC LANGUAGE

The Military Standard 25-25 [http://symbology.disa.mil/symbol/mil-std.htm] considers an **icon** to be *the innermost part of a (tactical) symbol* that provides a graphic representation of a *war fighting object* (see Figure 5). Below we will call this standard Mil 25-25. A tactical symbol contains more textual information than symbols in pure iconic languages. All right and left fields of a tactical symbol are textual. Thus, it would be more correct to call it a **mixed textual iconic language**.



(a) Structure of a tactical symbol (b) Example

Figure 5. Icon concept of Mil 25-25 [http://symbology.disa.mil/symbol/mil-std.htm]

Table 11 contains a summary of an upper level ontology of this language. War fighting objects have two major categories: equipment and military units. An object is characterized by its (1) attributes, (2) location, (3) time, (4) evaluation ratings and (5) actions. Some of these categories are highly elaborated having lower level ontologies that are described by more than 500 pages in Mil 25-25.

This ontology has two major differences from other iconic languages: (1) a highly elaborated hierarchy of the icons for objects and (2) much less elaborated icons for actions. Only movement, direction, and speed can be represented in the tactical symbol directly.

The language was designed to be used for tactical needs. A fighter's action in the battlefield is often derivable from its name – fighting. That is, a direct icon for such action can often be omitted. Similarly for visual correlation tasks, when only the result of an action is to be shown, direct use of icons for these actions can be avoided.

The example in Figure 5(b) represents a hostile fighter (ID AJ2455) moving in the air to the southeast. A hostile indicator is presented by the red color and duplicated by the use of the "hostile" shape in case the icon is used in a black and white presentation. A fixed wing fighter indicator is represented by letter "F" and "in the air" is represented by the shape with absence of the bottom frame. The southeastern movement is encoded by the arrow.

Table 11. Upper level ontology of the tactical symbology language

Concept	*Representation*	*Description*
Object category	Text and icon	Affiliation (friend, neutral, hostile…) Type: Equipment, Military units (infantry, motorized, reconnaissance, airborne, outpost, etc) Installation, e.g., military base
Object quantity/size	Text and icon	Equipment quantity, echelon (military unit size indicator)
Object ID	Text	Object unique ID
Combat effectiveness	Text	Completely effective/capable, almost fully effective/capable, fairly effective/ capable, effectiveness cannot be judged, effectiveness doubtful, ineffective.
Location	Text and icon	Battle dimension (air, space, ground, sea surface...) altitude/depth; degree, minute, and seconds Status: current, anticipated/planned (indicated by frame)
Action:	Text and icon	Movement: speed, direction
Date/time	Text	
Evaluation ratings	Text	reliability rating: completely reliable, usually reliable, not usually reliable… credibility rating: confirmed by other sources, probably true, improbable…

The major element of Mil 25-25 iconic language is the **shape**. Shapes are used to encode affiliations (friend, hostile, neutral, ...). In addition, shape information is duplicated through the use of color (for color blind people, black and white monitors, and drawing). The shape encodes not only affiliation but two more characteristics: battle dimension (air, space, ground, ...) and type (units, equipment, installations, ...). Figure 6 shows the shape for hostile *air track equipment* along with the icon hierarchy. The whole set of shapes includes $9 \times 9 = 81$ icons.

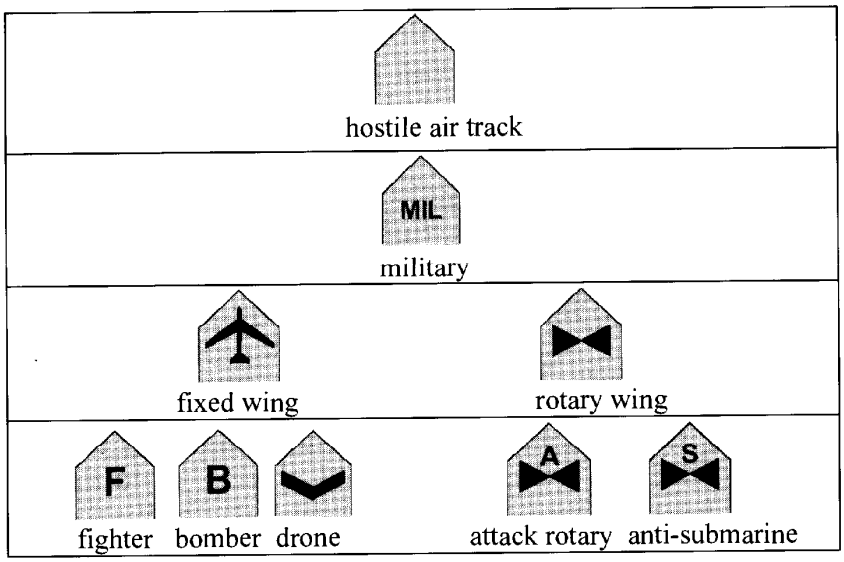


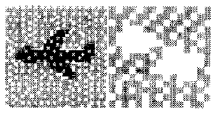
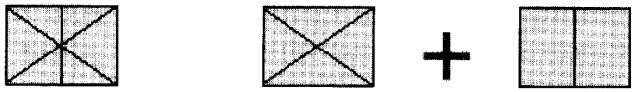
Figure 6. Fragment of the air track iconic hierarchy

In the Mil 25-25 standard, there is no independent fixed iconic element to represent location categories such as ground, air, space, sea surface, and submerged. These graphics are **context dependent**. Other characteristics such as object's affiliation (friend, hostile, unknown, ...) and type (unit, equipment, installation, ...) impact location graphics. This is a significant difference between these context-dependent icons and the context independent icons designed in Media Streams.

In Mil 25-25 global locations such as ground, air, and space are indicated by the icon frame. There is no bottom frame for air and space (see Figure 6 and Table 12). There is no upper frame for subsurface and there is a full frame for ground and sea surface.

Table 12 illustrates the *difference* between two iconic languages. The Media Streams iconic language sets up an individual icon for each concept "in air" and "plane" and has no icons for affiliation. Mil 25-25 is more compact it encodes three characteristics in a single icon, but it is a more complex language to learn. Also it has very elaborated set of icons for different types of planes (fighter, bombers, drone, ...) as shown in Figure 6.

Table 12. Comparison of syntax of Mil 25-25 and Media Streams

Icons	*Content*	*Iconic language system*
 composite icon	Affiliation: Hostile (shape and red) Object: plane Location: in air (no bottom frame)	Mil 25-25
 compound icon	Object: Plane Location: in air	Media Streams
 Composite icon "motorized infantry"	object "infantry" + modifier "motorized"	Mil 25-25

The Bruegel system incorporated the idea of utilizing shapes of icon frames from Mil 25-25 as a base for neutral objects. Note that Mil 25-25 uses only a **few colors.** These colors represent basic affiliation (thread) of war fighting objects: (1) yellow for unknown, (2) blue for friend, (3) green for neutral, and (4) red for hostile object. Two more colors (purple and brown) are used in the meteorological part of Mil 25-25. This is probably sufficient for tactical battlefield symbols on the map, but for more general database visualization and visual correlation of objects and events (O/E), the use of more colors can be beneficial. Use of similar colors for O/Es with similar attributes can help to facilitate visual correlation.

Next, Mil 25-25 uses a significant number of **textual indicators,** e.g., "FIRE". Such text must be read and mentally matched with a real-world object. It takes time. In contrast, an intuitive fire icon (used in the ISO standard for flammable materials) appeals directly to the real-world object -- fire. In addition, it is not easy to combine icons from Mil 25-25 into a composite icon such as truck fire and building fire. It may require a repositioning and resizing the text "FIRE" as well as the icons for truck and building as suggested in Mil 25-25. Without resizing and repositioning, icons for fire and vehicle will overlap and one of them will not be visible.

The Mil 25-25 language also uses a relatively small number of basic shapes. As we already mentioned shapes represent combinations of

- *location* (above surface, ground surface, sea surface, under surface, and unknown type of location),
- *type of object* (military unit, equipment, installation), and

- *basic affiliation/battle dimension* -- the threat posed by the war fighting object (friend, unknown, neutral and hostile). Recall also the colors for affiliations described above.

These shapes are relatively simple (square, rectangular, rectangular over rectangular, "diamond," "cloud," "arrow," "parabola") and often intuitive for users. A variety of modifiers is applied to these base icons (see Figures 5, 6 and Table 13).

The standard uses a variety of textual and graphical modifiers are to extend the number of alternatives covered. A **modifier** is defined as *optional text or graphics* that provides additional information about a symbol or tactical graphic. For instance, to include more threat affiliations (assumed friend, pending, joker, and faker) alphabetical symbols -- "J," "K," and "?" are attached to icons. The Mil 25-25 also includes an ontology for military operations other than war. Even a brief analysis of these icons shows that this ontology is very limited for visualizing such events as terrorist attacks. This limitation of Mil 25-25 explains our intention to develop an iconic language that will permit the visual correlation and analysis of complex events such as terrorist attacks (see Chapter 10 for more detail).

Table 13. Graphical modifiers

Characteristics	*Iconic indicator*
Direction	Arrow
Equipment	Squared dots, wave,
Feint or dummy	Dash lines
Task force	Rectangle
Headquarters staff	Flag
Installation	Small filled rectangle
Echelon (team/crew, ...army...)	|, ||, |||, X, XX, XXX, ..., XXXXXX, ...

5. ICONIC REPRESENTATIONS AS TRANSLATION INVARIANTS

Sophisticated machine translation typically requires that the deep content of a sentence be available. Below we illustrate how an iconic representation can encode this deep content. The icon shown in Figure 7 annotates the statement "A robber runs out quickly with a bag."

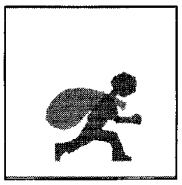


Figure 7. Icon "A robber runs out quickly with a bag."

A straight word-for-word Russian translation is a meaningful sentence "Grabitel' ubegaet bystro s meshkom." Now let us consider the Russian sentence "Ubegaet grabitel' na vseh parah s sumoi za pravym plechom." An English word-for-word translation will prodice the sentence "Runs out a robber on all pares with a bag agree right by shoulder," which is nonsence. The deep linguistic content cannot be translated word-for-word. Russian permits a free word order and this sentence starts with the verb. Next, the sentence contains two phrases "na vseh parah" and "s sumou za plechom." In addition, the word "za" has three English meanings: "behind," "agree," and "instead of" depending on context. Now assume that this Russian text is augmented with the icon shown in Figure 7. In this case, an Engish speaker almost does not need a translation thereby avoiding the misleading word-for-word translation or the significant effort associated with sophisticated translation. This is especially true when templates are used in both languages instead of free text. This idea was behind Blisssymbology designed by Bliss in 1940s [Bliss, 1978, 2000] but is still far from being fully utilized.

6. GRAPHICAL CODING PRINCIPLES

6.1 How much information can a small icon convey?

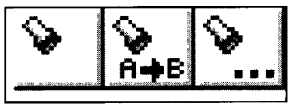
Below we analyze how much information can be reasonably conveyed in a single small icon. This is a critical issue for success of the entire enterprise of iconic language design. Specifically we are interested in clarifying answers to the questions:

(Q1) How many attributes can be *explicitly* encoded in a small icon?

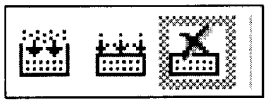
(Q2) How many attributes can be *implicitly* encoded in a small icon?

We start with question Q1 for the 32 x 32 pixels icons that are widely used in software design and provide us with empirical material for study. Figure 8(a) shows the icons "find," "find and replace," and "find again" from Borland C++ software. The first icon encodes one concept, "find," using a flashlight metaphor. The second icon encodes graphically two concepts: the first icon's concept and "replace" by adding the text "A→B." Similarly the third icon encodes two concepts: the first icon's concept and "again" by add-

ing an ellipsis "...". Figures 8(b) and 8(c) provide other examples of small icons that encode 1-2 attributes each. For instance, Figure 8(b) shows an "iconic sentence" that describes ways to manipulate the compiler and linker. This sentence differs structurally from traditional sentences in natural languages, but it is still the description of a complex object. All three parts of Figure 8 keep meaningful metaphors.



(a) Borland C++ icons "find"; "find and replace", and "find again"

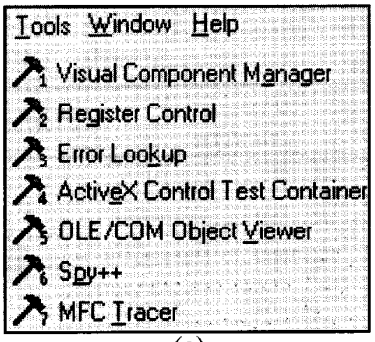


(b) Microsoft Visual C++, build, rebuild, stop to build.

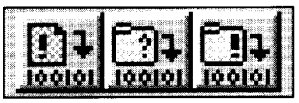
(c) Insert/remove a breakpoint; insert/ remove all breakpoints; enable/ disable a breakpoint; disable all breakpoints.

Figure 8. Icon analysis

Figure 9(a) shows Microsoft Visual C++ icons that accommodate two concepts by using a *combination of graphics and text* that includes a tool icon and a numeric index. Here a part of the *metaphoric component is lost.* The number identifies a specific tool (e.g., tool 6) and does not convey directly that the tool is spy++.



(a)



(b)

Figure 9. Combination of graphics and text

Figure 9(b) shows how three Borland C++ icons "compile unit," "make project," and "build project" convey more concepts -- *three attributes* are encoded in each icon. The last icon combines attributes of icons 1 and 2 taking symbol "!" from the first icon and yellow folder symbol from the second icon. The first one is the class of operations for the "steps of producing executable (binary) code" (binary sequence "100101"). In addition, each icon

conveys two specific attributes. The first icon conveys explicitly the attributes: (1) compile (!) and (2) unit (page and blue color); the second icon conveys two attributes: (3) make ("?") and (4) project (folder and yellow color); the last icon conveys attributes (4) "project" (folder) and attribute (5) "build" ("!"). Together these three icons form an "*iconic sentence*" that depicts the complex object "Borland compiler and linker". The use of color permits immediate recognition that the second and third operations deal with one entity and the first icon deals with a different entity. As you can see, the project concept is encoded by two graphical features (a folder and the color yellow). Similarly, the unit concept is encoded by two other graphical features (a page and the color blue). Thus, there is a redundancy in this encoding – features are doubled graphically resulting in clearer and more quickly distinguishable icons. These icons also permit one to see that the first icon indicates an action that produces executable code in one step. In contrast, the same result can be produced by using icons 2 and 3 together (the symbol "!").

Each icon presents three concepts using four graphical features. We can conclude that a simple 32 × 32 pixels icon is able to encode **3-4 independent concepts** without causing any perception difficulties.

Generally, an **iconic sentence** that contains three icons can depict 9-12 concepts explicitly and can also represent many relationships between the icons implicitly, such as: (i) to represent the same object or different objects (encoded by color and shape) and (ii) to represent a subset of operations (symbols "?" and "!"). Table 14 shows another example depicting 3-4 icons using **iconic metaphors**. The analysis above shows that a typical software icon *explicitly* presents 1-4 concepts per icon.

Table 14. Examples of Borland's iconic convention

Icon	*Vendor*	*Product type*	*Product subtype*	*Platform*
	Borland (background color pattern)	Debugger (bug iconels)		Win32 (iconel 32)
	Borland (background color pattern)	Debugger (bug iconels)	Installation program (disks iconel)	Win32 (iconel 32)

6.2 How many attributes can be implicitly encoded by a small icon?

At first glance, the observations above permit one to conclude that four independent concepts should be a practical maximum for the number of elements depicted in a small icon. This would then justify a compression ratio (from text to icons) of at most 4:1. That is, four icons (each depicts only one concept) can be combined into one small icon that would represent four text concepts.

The number of concepts **implicitly** encoded in an icon can be much larger than three or four concepts. This is done by *relaxing the requirement of a one-to-one match between an attribute and an iconic metaphor.* Below we describe an example from a Singapore executive job service company [http://www.liahona.com.sg/].

This company provided a long description of (1) the benefits that their client companies may provide to an employee, (2) application and resume requirements, and (3) expatriate bonuses. Each of these areas is represented by a single icon described in Table 15. The icons we use in table differ from original icons but encapsulate similar information.

Table 15. Example of complex concepts encoded in icons

Icon	*Description* [http://www.liahona.com.sg/icons+symbols.htm]
 sizable benefits	The company provides at least half of these benefits: medical, dental, accident insurance, low interest loans for car & housing, education assistance, transport allowance, technical & development training, holiday subsidy plan, recreational facilities, annual company function, compassionate, marriage, maternity, paternity, childcare, examination leave, stock options purchase plan, profit sharing, etc
	The company has a comprehensive expatriate package that includes most of these benefits higher basis salary, overseas premium, housing allowance, cost of living allowance, home leave, children's education,, spouse make-up salary compensation, company car, tax equalization, hospitalization & medical insurance, etc.
	Interested candidates are to apply with a detailed resume stating work experience, educational qualifications, full personal particulars of current & expected salary, starting data or resignation notice required, contact numbers (during & after office hours), address, age, nationality, marital status, language ability and driving license, a photograph and supporting documents.

It is hard to measure how many concepts are really depicted in the first three icons in Table 15, but obviously it is more than four concepts or attributes. It shows that a one-to-one mapping between text concepts and icon features is avoidable although a user may need assistance in learning an implicit iconic language without a steep learning curve. Table 16 presents an example of two iconic sentences that summarize two open positions. These iconic sentences are obviously shorter then original text and can be compared and correlated faster.

Table 16. Iconic sentences that summarize open positions

Due Date	*Transport nearby*	*Phone Fax*	*Pay package*	*Benefits*	*Expatriate package*	*Career growth*	*Resume requirements*
 June 1	 10 min walk	 781-1430	6FP 6-figure pay, > $100,000	 large	 large		 large
 July 9	 10 min walk	 980-1036	$$ depends on experience	 modest	 modest		 modest

7. PERCEPTION AND OPTIMAL NUMBER OF GRAPHICAL ELEMENTS

7.1 Perception and icon design

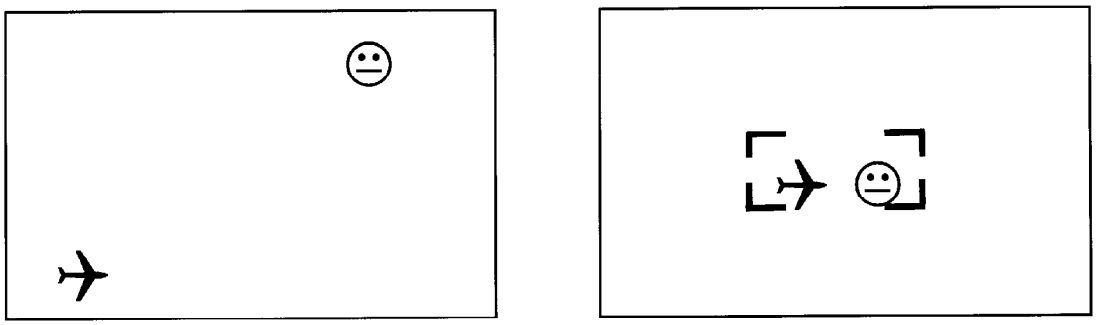
Context plays an important role in icon perception and should be reflected in icon design. Above we described the context dependence of military icons. Icon context has a variety of aspects. For instance, objects drawn nearby can change the meaning of the icon. General **gestalt laws** of perceptual organization [Preece, 1994] set up a perceptual framework for icon design:

- regions bounded by symmetrical borders tend to be perceived as coherent figures (see Figure 8(b)),
- elements of the same shape or color to tend to be seen as belonging together, and
- the boundary contrast is better than a linear boundary for making a shape stand out.

Another way to take into account a perceptual aspect in icon design is using the **ecological approach** [Gibson, 1979; Preece, 1994] -- to help a user to simply *detect information* rather than to *construct information* from the image. Detection is a single step process, but constructing may take two or more steps.

For instance, a user needs to analyze information on a victim. If victim's information is in two different spots (see Figure 10(a)) then the user needs to assemble/construct this information before analyzing it.

In contrast, Figure 10(b) provides victim's information already assembled as a single focus entity in the center of the window. The Bruegel iconic system supports the icon and an icon elements relocation mechanism to be viewed correlated.



(a) (b)

Figure 10. Examples of Gibson's ecological approach

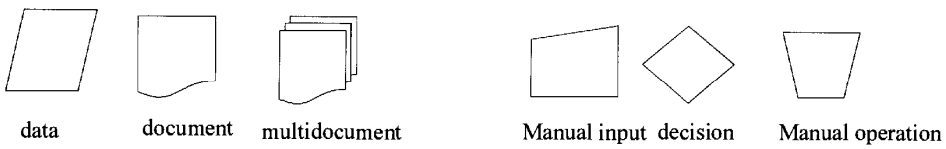
7.2 The optimal number of graphical elements

Table 17 shows a summary of the maximum number of effective codes (variations) of different graphical elements that can be used in visualization objects and actions based on [Preece, 1994]. Icon design would benefit from following the principles associated with these results.

The encoding principles described in Table 17 are used in many visualization applications including flowchart design. The basic idea is to represent objects and actions (see Figure 11) where the total number of object and/or action types encoded is determined in accordance with limits shown in Table 17.

Table 17. Principles of graphical coding in a single application based on [Preece, 1994]

Type of entity	*Graphical element*	*Examples of entity*	*Maximum number of effective codes*	*Perception comparison*
Any	**Alpha-numeric**	12", AK-47, high temperature, 35°	Practically unlimited (self-evident meaning)	Words are scanned longer than letters. Letters are scanned longer than digits.
Object	Abstract **shape**	Document, data, disk, ground, in air, in sea, target, victim.	10-20	Scanned longer than color
	Color	Red –hostile, warning sign; blue -- friend, green – neutral.	4-11	Scanned longer than digits, but faster than letters
Action/ operation	Abstract **shape**	Flowchart symbols: sort, delay, collate, decision, merge, manual operation.	10-20	Scanned longer than color
Direction	Line & angle	Wind direction, attack direction	8-11	
Numeric attribute	Line **length**	Percentage, temperature, confidence, size	3-4	
	Line **width**	Percentage, temperature, confidence, size.	2-3	
	Line **style** and fill		5-9	
Relation between attributes	**Ratio** of length and width	Correlation between person's weight and height.	3-5	Scanned longer than shape and color.



a) Shapes for objects b) Shapes for operations

Figure 11. Example of contrasting shapes for objects and actions

8. CONCLUSION

This chapter described the state-of-the-art in iconic descriptive approaches. These approaches are useful for annotating, searching and correlating traditional databases and those containing images and multimedia. They are compact and can provide a quick response since they are psychologically appealing for users.

Iconic annotations are based on concepts of base, compound and composite icons used to construct iconic sentences. For the iconic annotation process, grammars and ontologies were described using three representative systems. The first system considered was the MIT Media Streams system for annotating video. Another important iconic annotation system considered is known as military standard Mil 25-25, which provides a combination of icons, abbreviations, and short text to represent war fighting objects.

The third system considered was the Bruegel system for annotating text. Mil 25-25 and Bruegel are interesting from a conceptual viewpoint because both use context-dependent visual grammars to represent war fighting objects and text while Media Streams uses a context-free visual grammar to represent video content.

The concept of composite icon is employed in both Mil 25-25 and the Bruegel system, Composite icons are a natural way to introduce context to icon design and produce compact and informative icons. Language translation is another area that can benefit from the use of modern iconic annotations. This idea was behind of the development of the first such system, Blisssymbology, designed by Bliss in 1940s.

An iconic query is a "native way" of querying multimedia databases. In multimedia, iconic queries support the ability to search the space of iconic annotations for those icons annotating multimedia which satisfy a given search condition.

An analysis of the practical use of small icons in a software graphical user interface demonstrated that simple 32×32 pixels icons are able to encode **3-4 independent concepts** explicitly without any perceptual difficulties. We also provided an example where each icon conveys more than a dozen concepts implicitly.

The chapter concluded with a summary concerning the maximum number of effective variations of different graphical elements that can be used in visualization and icon design.

9. ACKNOWLEDGMENTS

This work has been sponsored by the US Intelligence Community via Advanced Research and Development Activity and National Geospatial-Intelligence Agency. This support is gratefully acknowledged.

10. EXERCISES AND PROBLEMS

1. Design ten base icons and construct four *compound icons* using them. Build four iconic queries using the ten initial icons and four compound icons. Each query should contain at least four icons.

2. Use the icons designed in exercise 1 and construct four *composite icons.* Build four iconic queries using the ten initial icons and four composite icons. Each query should contain at least four icons.

3. Build four composite icons that will encode six attributes each *explicitly.*

4. Build four composite icons that will encode at least ten attributes *implicitly.*

5. Construct five iconic sentences that will summarize a one-page text of your choice.

11. REFERENCES

Bliss, C K. Semantography-Blissymbolics. 3rd ed. N. S. W., Sydney. Semantography-Blissymbolics Publications, 1978.

Bliss for Windows, version 6.2, 2000, Handicom, NL, http://www.handicom.nl/DOWNLOAD/BLISSFW/BFW62_UK.EXE)

Chang, S. K , Icon Semantics - A Formal Approach to Icon System Design, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 1, No. 1, 103-120, 1987.

Chang, S. K , Tauber, M., Yu B., and Yu, J. S. , A Visual Language Compiler, IEEE Transactions on Software Engineering, May 1989, pp. 506-525.

Chang, S. K. Principles of Pictorial Information Systems Design, Prentice-Hall, 1989.

Chang, S. K. , Polese, G., Orefice, S., Tucci, M., A Methodology and Interactive Environment for Iconic Language Design, International Journal of Human-Computer Studies (IJHCS), 41, 1994, pp. 683-716. http://www.cs.pitt.edu/~chang/365/mins.html

Chang, S. K Symbolic Projection for Image Information Retrieval and Spatial Reasoning, Academic Press, 1996.

Dictionary of Blissymbolics, 2001, http://www.symbols.net/blissymbolics/dictionary.html

Davis, M., Media Streams, Ph.D. dissertation, MIT, Media Lab.,1995, http://garage.sims.berkeley.edu/pdfs/1995_Marc_Davis_Dissertation.pdf

Davis, M. Media streams: an iconic visual language for video annotation, 1995, http://www.w3.org/People/howcome/p/telektronikk-4-93/Davis_M.html

Davis, M. Media stream: an iconic visual language for video representation, In: Readings in Human-Computer Interaction toward the year 2000, Morgan Kaufman Publ., San Francisco, 1995, 854-866.

Gibson J.J., The ecological approach to visual perception, 1979, Boston: Houghton Mifflin.

Military Standard 25-25 [http://symbology.disa.mil/symbol/mil-std.htm]

Narayanan A. , Shaman T., Iconic SQL: Practical Issues in the Querying of Databases through Structured Iconic Expressions, Journal of Visual Languages & Computing Volume 13, Issue 6, December 2002, Pages 623-647, 2002.

Preece J. Human-Computer interaction, Addison-Wesley, 1994

Spence, R. Information Visualization, ACM Press, Addison-Wesley, 2001

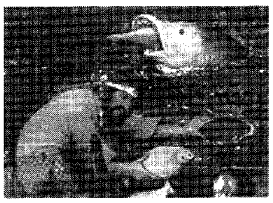
Chapter 10

BRUEGEL ICONIC CORRELATION SYSTEM

Boris Kovalerchuk, Jon Brown, and Michael Kovalerchuk
Central Washington University, USA;

Abstract: This chapter addresses the problem of visually correlating objects and events. A new Bruegel visual correlation system based on an iconographic language that permits a compact information representation is described. The description includes the Bruegel concept, functionality, the ability to compress information via iconic semantic zooming, and dynamic iconic sentences. The chapter provides a brief description of Bruegel architecture and tools. The formal Bruegel iconic language for automatic icon generation is outlined. The second part of the chapter is devoted to case studies that describe how Bruegel iconic architecture can be used for the visual correlation of terrorist events, for file system navigation, for the visual correlation of drug traffic and other criminal records, for the visual correlation of real estate and job markets offerings, and for the visual correlation of medical research, diagnosis, and treatment.

Key words: Visual correlation, iconographic language, semantic zooming, database visualization, iconic representation



P. Bruegel, "Blue cloak," 1559, fragment

1. INTRODUCTION

The Bruegel visual correlation system permits the compact visual annotation of information for objects and events along with their rapid comparison and correlation, search and summary presentation. The system was named after Flemish painter Pieter Bruegel and was inspired by his famous painting "Blue cloak" shown partially above (see Figure 1 and Table 1 in Chapters 8 for more detail). The main categories of possible visual correlation systems

are described in Chapter 9. The Bruegel iconic system supports three categories. To begin with, it is a *spreadsheet* category where each object and event (O/E) is represented as a spreadsheet of icons organized in rows. The second category is a *3-D tree* presentation where each O/E is represented as a 3-D tree of icons located in nodes. Icons located at the terminal nodes convey most detailed information about O/E while upper-level nodes convey more generalized information. The third representation is a *"planted" 3-D tree* representation that combines a geographic map or image with a 3-D tree "planted" at the location of O/E on the map/image. Each of these representations has its own *semantic zoomed form* which is conceptually described in section 2.2. This form permits *"semantic compression"* of icons to get a more compact representation.

The Bruegel visual correlation system includes several components:

- the **Bruegel graphical language** (BGL) that specifies the layering of iconographic elements into complex icons to represent textual content in a space efficient manner,
- **Dynico**, a supporting tool for BGL which aids in the generation of complex icons, iconic sentences, and spreadsheets of icons dynamically
- **3DGravTree**, a supporting tool for BGL to generate complex 3-D trees including "planted" trees of icons dynamically, and
- **other graphical tools** to support BGL for the visual correlation of complex objects and events.

2. THE MAIN CONCEPTS OF THE BRUEGEL ICONIC SYSTEM

2.1 Bruegel functionality

The dynamic iconic visual correlation system Bruegel enables users to create multi-layered, iconic annotations of events such as medical patient records (e.g. breast cancer patients), job advertisement, real estate, file system, criminal records, and terrorist attack records along with visual correlation of these events. Further, Bruegel provides a compact iconographic visual representation and correlation of information that extends military symbology standard (Mil Std 25-25). Currently the system contains an extension of Mil Std 25-25 intelligence symbology library for terrorist activities.

Experiments with a database on terrorist attacks in 1980s had shown that iconized data occupies *10 times less space* than text. This means that an analysts and decision makers can potentially spend *10 times less time* browsing and analyzing iconized data.

The **functionality** of the Bruegel Visual Correlation system includes:

- Facilities to create new icon libraries,
- Facilities to switch icon libraries for specific applications,
- Facilities to browse icon libraries,
- Multiple ways to represents and correlate objects and events as icon strings and icon trees with different levels of semantic resolution (macro view, middle–level view (list view), network view, clustering view),
- Facilities to correlate and sort icons strings and trees (sentences) in 2-D and 3-D trees,
- Facilities to assign weights to attributes for visual correlation;
- Facilities to merge icons, and
- Facilities for the 3-D correlation of iconized objects and events using flexible "gravitation" trees.

The system also aids in the navigation of databases, in searching records visually, in correlating records, in discovering useful patterns in databases, and in supporting decision-making. Figure 1 demonstrates how an analyst can utilize the system functionality.

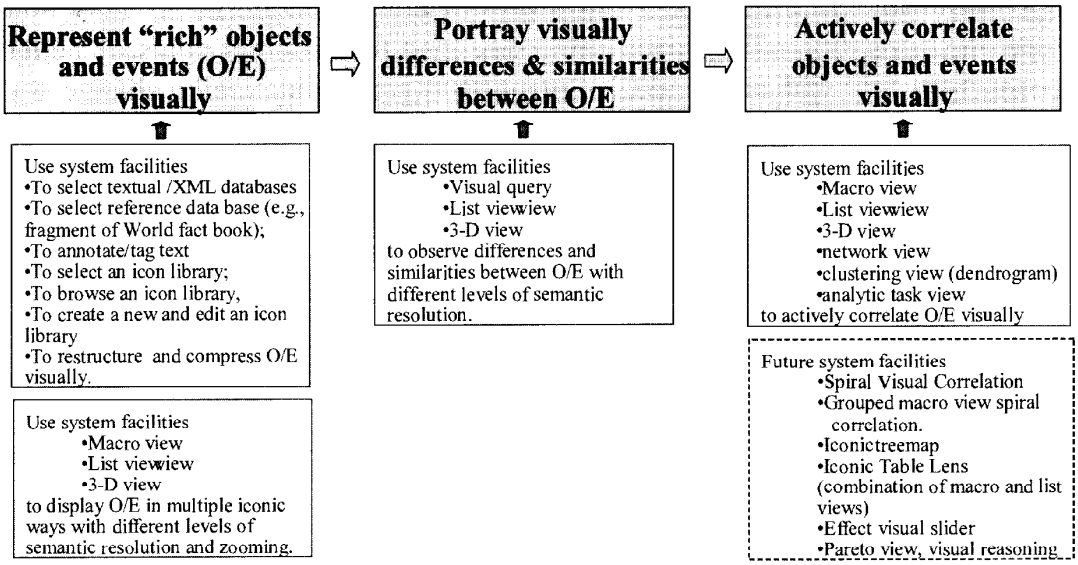


Figure 1. Sequence of analyst's work with the Bruegel system with supporting facilities

2.2 Dynamic iconic sentence compression

Iconic sentences have a significant potential to encode information efficiently. The number of icons in the sentence and their complexity are major factors that limit human abilities for making sense of the encoded information quickly.

Theoretically, all information can be compressed into a single icon. An example is "Blue cloak" painting by Peter Bruegel that compresses 78 Flem-

ish proverbs (see Figure 1 and Table 1 in Chapters 8 for more detail). This unique visual language provides the highest level of compression, but it is *not well structured for analysis and visual correlation.* Other iconic languages have a one-to-one mapping with text sentences. Consider for example Bliss's iconic language [Bliss for Windows, 2001] a sample of which can be found in Figure 4(a) below. In such a language, practically every text word is converted to an icon. Thus, an iconic sentence can take even more space than the corresponding text. Still the Bliss language fulfills its purpose – a global international communication tool - a visual Esperanto; however, it does not serve as a language for visual correlation.

Iconic languages such as Mil Std 25-25 and Media Streams described in Chapter 9 represent intermediate steps between "Blue cloak" and Bliss. Figure 2 depicts the relative compression provided by these languages.

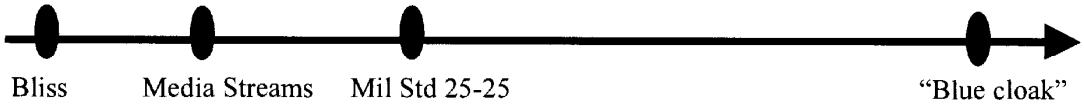


Figure 2. Levels of compression in iconic languages

Different goals dictate different levels of compression. Thus, more and more one finds iconic languages being designed for different goals. A better approach is to build an iconic language that can dynamically change the level of compression of an iconic sentence depending of the goal. Below we outline the design of such a language. It has been partially implemented in the form of our Bruegel iconic language.

Figure 3 shows how the concept of compression can be applied to iconic languages. It includes the concept of **spatial compression** as well as that of **semantic compression**.

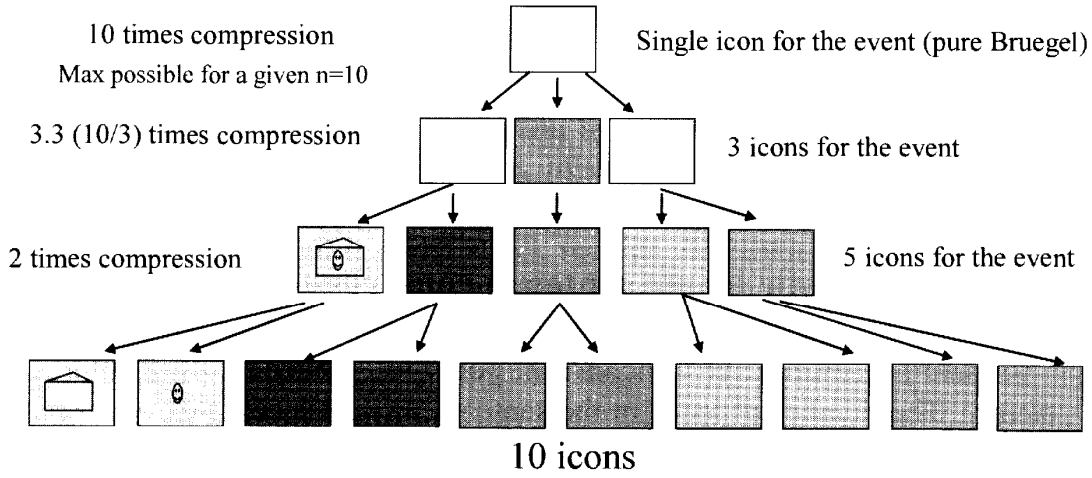


Figure 3. Dynamics of compression of iconic sentence. See also color plates.

Iconic compression can be lossless or it can permit some loss of information in a manner similar to the compression of a jpeg image compression. The difference is that we work on high-level compression. In contrast, jpeg compression is a low-level (pixel level) compression.

The bottom level on Figure 3 shows an iconic sentence that matches in a one-to-one manner icons to database fields for an individual record. This sentence has not been compressed yet. Indeed, this can also be a nearly one-to-one matching of icons to words in a natural language sentence. On the next level, each pair of low-level icons is combined to a single icon called **combined icon.** Figure 3 displays parent icons in the same color as their children since they represent the same information. This process can be repeated until the iconic sentence shrinks to a single icon.

An iconic sentence viewer would allow iconic sentences to be viewed at different levels of compression. A user can select the level of compression that best fits the specific analytical task the user is working on. This dynamic flexibility is not available in typical iconic languages.

We use the notation "**t-i compression**" to denote the compression ratio that occurs when text is substituted by an icon. Similarly, we will use the notation "**i-i compression**" to denote icon-to-icon compression, where several icons are compressed into a single icon. The compression ratio is measured by the ratio of the space occupied on the screen by the text and the icons.

Our experiments have shown that at the first stage, the Bruegel iconic visualization system reached a t-i compression ratio of two when all 24 attributes (slots) are mapped into 24 icons. Following this, the higher-level icon-to-icon compression mechanisms were applied as depicted in Figure 3. The result, these 24 icons were compressed into 6 icons yielding an i-i compression ratio of four and total t-i compression ratio of eight when two compressions are combined.

Gradual compression has several advantages for human perception by allowing the discovery of relationships between different levels of detail. The mechanism shown in Figure 3 has been developed to meet this challenge. The mechanism is known as gradual compression, which basically says that two icons can be compressed into one icon or some icons with important information can stay uncompressed. Gradual compression provides a smoother transition by first moving from 24 icons to 12 icons and then to six icons. In both cases, the ratio is appropriate for human perception.

This compression process can also be **animated.** Again, the user can select the level most appropriate for a specific task. We note that these levels are not predefined. It is a **dynamic and user-controlled process**. One of the objections typically raised against iconic and other visual languages is that a user needs to learn a new language.

To make this process easier, the Bruegel system can be adjusted to be a **learning game** that uses icons and text. The user can play with existing texts that already have a meaningful compression via the hierarchy of icons. The user needs to guess the meaning of the icons. Scores for a guess are calculated from the difference between that guess and the real text content encoded in the icons. The computer can play against the user showing only part of the icons. It also can be a network game with another person.

As final note in this section, the described compression mechanism supports **semantic zooming**. Semantic zooming differs from standard zooming in two ways. Standard zooming is completely pixel-based while semantic zooming provides an upper level understanding of the image content.

2.3 Iconic annotation for dynamic objects and events

Objects and events that change over the time and space are a special challenge to the iconic approach. Such dynamic objects and events can be represented using several approaches [Chang, Bottoni, Costabile, Levialdi & Mussio, 1998, 2002]. Table 1 contains a brief description of these three approaches (spatial-temporal, semantic explicit and semantic implicit).

Table 1. Approaches for representation of dynamic events (based on [Davis, 1995])

Approach	*Description*
Spatial-temporal	The normalization of temporal events by indexing temporal and spatial changes using some temporal and spatial scales and reference points
Semantic explicit	**Fixed semantics**: semantically relevant atomic units organized into various temporal patterns (repeated cycles, scripts, etc.)
Semantic implicit	**Unfixed semantics:** a class of possible semantics is identified implicitly by a **physically-based description**. A physical action is mapped to a set of possible semantics in concrete contexts. For an example see [Davis, 1995]: Physical description: two people shaking hands. Semantics 1:'greeting' if positioned at the beginning of a business meeting shown in the movie shot. Semantics 2:'agreeing' if positioned at the end of the same meeting (movie shot).

The Media Streams iconic system [Davis, 1995] uses the semantic implicit approach because its goal is to search for a video segment with a similar physical event. The Bruegel system follows all three approaches because when correlating events such as terrorist attacks each of these representations are needed.

3. DYNAMIC ICON GENERATION FOR VISUAL CORRELATION

3.1 Dynamic icon approach

Below we describe an approach for dynamically generating icons and Dynico, a software system (part of the Bruegel visual language framework) for visualizing complex events as aggregate iconographic images. The overall purpose of this visualization scheme is to move the process of recognizing semantic correlations and patterns from the textual information domain to the visual domain thus allowing faster correlation discovery and decision making. Dynico consists of basic syntactic and semantic conventions that the user can augment with domain specific information.

In the 1940s Bliss [Bliss, 1978, 2002] began a significant development of iconic communications for natural language sentences. Now Blissymbolics is supported by such software as *Bliss for Windows* [Bliss for Windows, 2002]. The basic elements of Blissymbolics reflect the technical capabilities of the 1940s. Icons are not always intuitive and the number of possible combinations is very small. This is probably one of the reasons for its limited acceptance. Currently it is feasible to implement more complex, realistic icons and generate new icons dynamically, on demand.

We classify **approaches** in this area into three categories.

1) **Static approach** – all icons are predefined and designed in advance using paint-type software (resizing and changing background colors is permitted).
2) **Static-dynamic approach** – complex icons can be generated automatically from predefined icon elements by simple combinations (on the right, left, top, or bottom) with possible resizing.
3) **Dynamic approach** – complex icons can be automatically synthesized using the content of the text to be visualized, structural template descriptions, and iconels with use of XML-type descriptions.

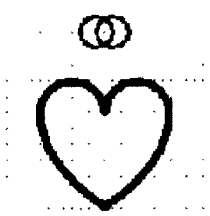
The first approach works in applications where only a limited number of visual features or icons are needed, for example, control icons in a specific software system or making painter-like visualizations [Healey, 2001]. Healey defines formally, what we call a static approach as a mapping:
ϕ: $A_j \rightarrow V_j$, where A_j is a data attribute from a set of data attributes A and V_j is a visual feature taken from a predefined set of visual features
$V = \{V_1, V_2, ..., V_m\}$.

Bliss' system is an example of the second approach. It maps the words presented in a sentence to individual icons and places the icons sequentially. For instance, the system can produce a new icon by placing one icon above

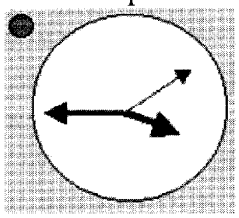
another to define a new concept. Consider the example in Figure 4(a) of the "Love in marriage" icon generated from the two words "Love" and "Marriage."

As an example of the complexity that can occur, consider the visualization of storytelling [Gershon & Page, 2001], massive texts and data often need more complex combinations of icons. Gershon and Page present several storytelling examples. For instance, the sentence "G-shape building is active between 8 and 10 a.m. and 4 and 6 p.m." might be part of a story that should be visualized.

For situations like this, we are developing a third approach. The first and second approaches are not scalable for the iconographic visualization of massive amounts of data where potentially thousands of icons are needed. It is not realistic that such a number of individual icons can be crafted in a static or static-dynamic manner. For instance, what if we want to visualize time information: "3:45 p.m. and 10 seconds." We may use a traditional watch icon with arrows for hours, minutes, seconds, and a colored dot indicating a.m./p.m. status as in Figure 4(b). The static approach for this visualization will require 432000 = 60 * 60 * 60 * 2 icons, assuming that each of three arrows can take any of 60 possible minute positions and two indications (a.m., p.m.). When on takes into account that some positions of the hands do not portray critical data, we can cut the number of alternatives to 86400 = 12 * 60 * 60 * 2 icons since we can assume that for any hour (out of 12), there are 60*60*2 combinations of the minute arrow, the second arrow and a.m./p.m. For such an application, the dynamic generation of visual features (icons) on demand is a natural way to solve the problem.



(a) Blissimbology: Love and marriage icon



(b). Dynico: Dynamic clock icon

Figure 4. Static and dynamic icons

The Dynico system addresses many shortcomings of static graphics in these contexts. A single dynamic icon can be used to convey many facets of a particular subject and can easily interact with other graphical components due to its vector graphics content. As we mentioned above, static and static-dynamic approaches do not attempt to incorporate the full dynamic potential in their systems design. To the best of our knowledge, the Dynico system is

the only dynamic system for creating aggregate iconographic images in this manner.

The dynamic approach includes not only the dynamic generation of *static icons* but also the dynamic generation of *dynamic icons* such as framed animation and key-framed interpolated animation. Since the goal of the current version is speed of visual interpretation of textual information and data, we have chosen to focus on static display states (static icons). This also avoids chronologically masking data that occurs in animated graphics.

3.2 System architecture

In the previous section, we motivated the need for a dynamic generation system. This section discusses different approaches to developing a dynamic iconic system. The problem is complex because: (1) the content of the text and data should be used to dynamically identify parameters of visual features in the icon and (2) a dynamic approach exhibits complex visual and textual content interactions that require strict control. This is in contrast with the minimal flexibility provided by standard and thus simpler operation with bitmaps and similar files available in a static visualization approach.

Two competing ideas drive this issue: (i) design-specialized and content-specific tools, and (ii) development of a relatively universal mechanism. The first alternative can be used for the watch example presented in Figure 4 in the following way:

- Generate five predefined iconels: H arrow, M arrow, S arrow, Watch base and Dot indicator for a.m./p.m.
- Compute the position of the M arrow on the Watch base as a function of parsed text content: "<hour> 3 </hour>: <minutes> 45 </minutes>: <seconds> 10 </seconds> <ampm> p.m. </ampm>".
- Compute the position of the H arrow on the Watch base as a function of the same parsed text and the position of the M arrow already computed.
- Compute a position of the S arrow and color fill for p.m.

Obviously, this design will work only for this watch icon and thus it is highly specialized. This motivates the development of a more general mechanism called the *template-based mechanism*. In this mechanism, possible locations of individual iconels are identified in a template including their mutual location (as we have seen for hour and minute arrows). Thus, the specifics are encapsulated in a template. Indeed, Dynico supports a higher level of parsing of text contents and rendering images. In this higher level, another set of templates is used to generate complex icons using lower-level templates, which are combined into a single icon.

3.2.1 Architecture details

Input information is mapped to a visual representation on the gross level, and then a particular state of the graphical content is chosen. This state is based on the specific qualities of the mapped content. The icon is modeled as a set of layers to be rendered sequentially in creating the complete icon. Each layer is associated with an icon element (iconel) that is a basic syntactic element of the icon. Sequences of iconels are not static and can be changed in a user-defined template. Each layer has: (1) a semantic content associated with an underlying text, and (2) a specified sub-region of the total icon area to which the display of icon element in that field is restricted.

Syntactic units. In Dynico, each abstract icon consists of a number of **iconels,** which are the basic graphical unit with which the system works. An iconel has an array of **dynels**, which maintains an array of *frames*, each containing **primitives** made up of *points*. Thus, there is a hierarchy of components, up to 5 - 6 levels depending on the level of an iconel or an icon. Dynico also uses dynels to support **animation** as another aspect of dynamic iconography. A sequence of frames operates like film frames or as a series of still images representing different states of the graphical data. The number of dynels can be large (50 or more) and vary for different icons. Naturally, dynels must be informed what to display. Designing a reasonably simple way to communicate these structures is a challenging task. **Semantic units**. A set of icons associated with a locality is called an Icon phrase. Icon phrases are combined into a "full thought" or an "event." Generally, the locality is a single icon used to reinforce the association of the content, however this is current convention rather than a restriction; other grouping schemes can be realized through the use of templating. An individual icon or a group of icons is associated with some locality. A fuzzy logic membership function

$$\mu_{\text{locality}j}(\text{Icon}_i)\text{: } \{\text{Icon}_i\} \rightarrow [0,1]$$

is used to associate Icon_i with locality_j if a strict categorization is not appropriate for the task in hand.

Iconic templates. Templates are used to accommodate aspects of a user configuration such as user weighting of data, potential interfaces for template design, dynamic placement, and dynamic merging of icons (variable iconographic data resolution).

Specification of syntactic units. All syntactic graphical units of Dynico are vector graphics rather than bitmaps. Transformations with vector graphics are usually simple. Iconels, dynels, frames, and primitives are specified in a file called an SRT file. This is an XML formatted file; where each unit is described with attributes regarding points such as coordinate data and color. Figure 5 presents an example of an iconel description.

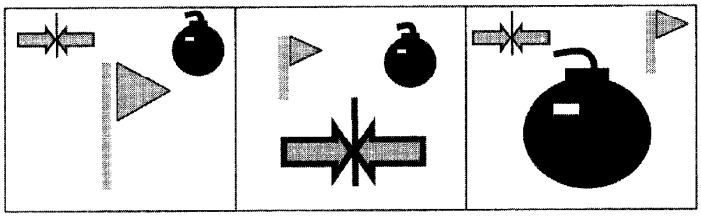
```
<?xml version="1.0"?>
<ICONEL id="human armed.SRT" dimension="600,600" dynels="1">
 <DYNEL id="New Dynel" frames="2">
  <FRAME id="New Frame" primitives="24">
   <PRIMITIVE id="New Primitive" type="LINE" width="4"
        stroke="44,200,44" fill="155,155,255" points="2">
        <POINT coordinates="585,513"/>
        <POINT coordinates="513,585"/>
      </PRIMITIVE>
   <PRIMITIVE id="New Primitive" type="LINE" width="4"
         stroke="44,200,44" fill="155,155,255" points="2">
         <POINT coordinates="585,450"/>
         <POINT coordinates="450,585"/>
       </PRIMITIVE>
       <PRIMITIVE id="Gun" type="POLYGON" width="2"
          stroke="1,1,1" fill="77,77,77" points="7">
         <POINT coordinates="387,315"/>
         <POINT coordinates="405,315"/>
         <POINT coordinates="405,225"/>
         <POINT coordinates="414,225"/>
         <POINT coordinates="396,180"/>
         <POINT coordinates="378,225"/>
         <POINT coordinates="387,225"/>
        </PRIMITIVE>
  </FRAME>
 </DYNEL>
```

Figure 5. XML representation of iconel data

3.2.2 Current tools and usage

Layering iconographic elements into complex composite icons for representing textual content in Bruegel is illustrated in Figure 6 for the text "Accomplished bomb attack" where the flag icon indicates "accomplished" and the icon with two opposing arrows indicates an attack. Figure 6(a) emphasizes accomplished through the location and size of iconel for accomplished. Similarly, Figures 6(b) and 6(c) emphasize attack and bomb respectively.

Below we describe a **dynamic placement schema** and the use of raw iconels. Dynico works by operating upon a relatively small number of data points. This property is exploited to further generalize the specification of graphical elements as they need only be designed to occupy the optimum space for their individual design. The system can then transform the iconels to the appropriate proportions and locations in order to represent the semantic content behind them. In addition, User Defined Weighting (UDW) can determine the iconels actual location and the size of the icon. This may more clearly emphasize the data that an analyst finds to be more important.



(a) (b) (c)

Figure 6. Icons with user defined weighting. See also color plates.

How user priority interacts with the placement schema. Interaction is controlled by merging rules. The ability to collapse and expand icon groups in order to easily provide a user-required variable data resolution implies that some automated merging mechanism must be implemented in the system.

To this end, a number of rules have been specified to guide the merging of icon elements. The rules are based on the geometrical attributes of graphical content rather than data that the visual elements might represent.

Templated Visualizations. With content encoded into icons, the use of a template allows a description of how icons are to be grouped. For example, either one might allow for simple left to right ordering of content or for a more complex aggregate super icons (see Figure 7).

```
<ICON_DEF id="LIST Icon">
        <LAYER_DEF id="Background" iconel="Background.srp"/>
        <LAYER_DEF id="Main" x="10" y="10" width="80" height="80"/>
        <LAYER_DEF id="IconFrame" iconel="IconFrame2.srp"/>
</ICON_DEF><!--LIST Icon-->

<STORYBOARD_DEF id="LIST Storyboard" using_def="Basic Dynico Icon" width="400"
height="120">
        <PLACE_ICON id="Perp" x="40" y="40" width="100" height="100"/>
        <PLACE_ICON id="Perp Org" x="10" y="10" width="40" height="40"/>

        <PLACE_ICON id="Wep 1" x="10" y="145" width="30" height="30"/>
        <PLACE_ICON id="Wep 2" x="60" y="145" width="30" height="30"/>
        <PLACE_ICON id="Wep 3" x="110" y="145" width="30" height="30"/>

        <PLACE_ICON id="Targ Effect" x="150" y="10" width="60" height="60"/>
        <PLACE_ICON id="Perp Effect" x="150" y="110" width="60" height="60"/>
</STORYBOARD_DEF><!--LIST Storyboard-->
```

Figure 7. The XML Template File data

Smurfico is a software tool developed to ease the burden of designing dynamic icons (see Figure 8). It is a simple utility that allows editing of an SRT file either visually or textually. Smurfico provides access to all of an SRT's data using two formats: graphical and XML/Textual. The graphic view allows editing by directly manipulating the shapes that comprise the data in the file. Icons are also stored in SVG format.

A number of tools are provided to control the various aspects of the file such as primitive type, color, ordering and shape. Things that would be hard to edit textually are included in this view. This includes scaling a primitive, changing the rendering order using a tree control, moving primitives, or moving entire frames.

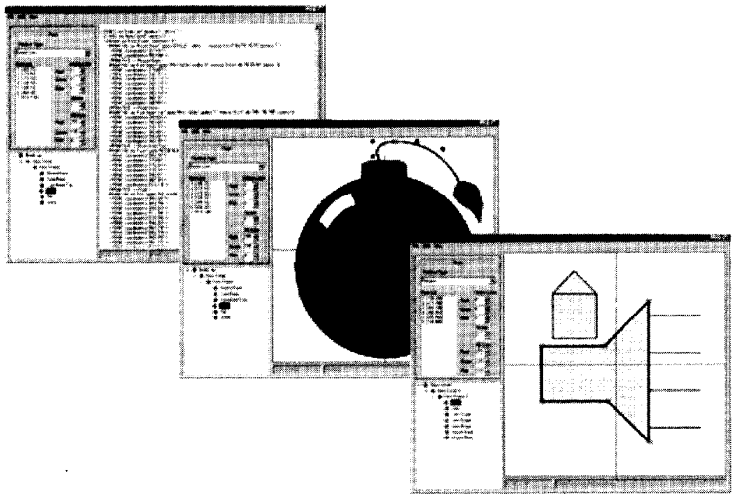


Figure 8. Bruegel object-oriented icon design tool, Smurfico

The XML View allows access to and editing of the textual details of the file. This provides an easy way to change aspects of the file such as reordering large groups of Dynamic Icon Objects, or copying/pasting of data. Rather than spending time designing interface features for every detail of a SRT file, some data are more easily accessed from this XML View.

4. THE BRUEGEL ICONIC LANGUAGE FOR AUTOMATIC ICON GENERATION

An advanced iconic system can not exist without extensive library of icons as we discussed above. Manual icon design is very time consuming. It is desirable to generate icons automatically and dynamically on demand. To be able to do this a formal language is needed. The main idea of such language is to represent icons to be generated as language expressions E_i with parameters $p_1, p_2, \ldots, p_k$, $E(p_1, p_2, \ldots, p_k)$ and to be able automatically generate icons for all values of parameters that satisfy both *graphic and semantic constraints*. Further, this then allows the selecting an icon from previously generated and stored parameterized icons or the generating of such an icon dynamically, on demand for specific parameter values $p_1, p_2, \ldots, p_k$. This motivates the following **operations** for producing combination icons for our formal Bruegel language:

1) *post a subicon* over another icon called a background icon,
2) *resolve the collision of subicons* and iconels by moving icons,
3) *post an iconel* over another iconel,
4) *resolve the collision of iconels* by moving iconels.

The main difference between subicons and iconels is semantic, typically an iconel is not used as an independent semantic entity; it is rather an attribute or a property of the entity. For instance, the MS Windows iconel "shortcut" needs to be posted on an icon that represents a shortcut to the actual file rather than the file itself.

In Bruegel each icon is modeled as an object in Object Oriented Programming terms. New icon objects are created from parent icon objects. When one icon is posted over another icon, the new icon inherits properties of both icon objects. Below we describe an **icon posting algebra** based on a **posting operation** that produces a **composite icon.**

Expression $O_1 \hookleftarrow O_2$ means that icon O_2 is *posted on* icon O_1 and icon $O_1 \hookleftarrow O_2$ is called a composite icon. The posting operation $\hookleftarrow$ is called an *associative operation* if expression $O_1 \hookleftarrow (O_2 \hookleftarrow O_3)$ is equivalent to expression $(O_1 \hookleftarrow O_2) \hookleftarrow O_3$, that is,

$$O_1 \hookleftarrow (O_2 \hookleftarrow O_3) = (O_1 \hookleftarrow O_2) \hookleftarrow O_3.$$

Statement. If $O_2 \cap O_3 = \varnothing$ then posting operation $\hookleftarrow$ is associative for O_1, O_2 and O_3.

The last statement means that icons O_2 and O_3 can be posted over the icon O_1 in any order without overlapping each other, because their intersection is empty. Thus resulting icons $O_1 \hookleftarrow (O_2 \hookleftarrow O_3)$ and $(O_1 \hookleftarrow O_2) \hookleftarrow O_3$ will be identical. For associative icons O_1, O_2 and O_3 we will write simply $O_1 \hookleftarrow O_2 \hookleftarrow O_3$ omitting parentheses.

It will be assumed by default that $O_1 \hookleftarrow O_2 \hookleftarrow O_3 = (O_1 \hookleftarrow O_2) \hookleftarrow O_3$ if the icons are not associative. The notation $O_1 \hookleftarrow (O_2 \hookleftarrow O_3)$ will also be read as *posting two icons over another icon.* The expression $O_2 \phi O_3$ denotes **icon collision**. **Resolving collision** is provided by using operations such as those shown below:

$10p \leftarrow O_2$ means to move icon O_2 10 pixels to the left.
$10p \uparrow O_2$ means to move icon O_2 10 pixels up.
$10p \downarrow O_2$ means to move icon O_2 10 pixels down.
$80\% O_2$ means to produce a new icon that has 80% of size of icon O_2.

More formally the first collision resolving rule can be written as written as a formal rule:

$$O_2 \phi O_3 \Rightarrow (10p \leftarrow O_2)$$

Below we show an example (in Bruegel) of a **syntactically correct expression:**

$$O_1 \hookleftarrow (10p \uparrow 80\% O_2,\ 80\%(O_3 \hookleftarrow 60\% O_4)).$$

This formula means that icon O_1 is used as a background, O_4 is scaled to 60% of its original size is posted on the O_3, then O_3 with O_4 is scaled to 80% and posted on O_1, Also O_2 scaled to 80% is moved 10 pixels up and posted on the O_1.

This dynamic iconic language has an advantage over *hardcoded icons*. For instance, with hardcoded icons if we need to encode in icons three attributes with ten values each then we need to produce 10 * 10 * 10 = 1000 icons manually. But formulas like those presented above will produce all of the required icons uniformly through the introduction of *parameters* such as *a, b, c* and *d*:

$$O_1 \hookleftarrow (a\ p\uparrow b\%O_2,\ c\%(O_3 \hookleftarrow d\%O_4)).$$

Parameter *a* might have 32 different values in a range [0, 31] pixels while parameters *b, c* and *d* might have 11 values 0%, 10%, 20%, ..., 90%, and 100%. In this way we generate **parametric icons**.

A collision in BGL can be automatically detected using standard clipping algorithms from computer graphics. For rectangular icons this is trivial and for rounded it remains simple. For more complex icons, it can be more challenge, but assuming vectorized icons presented as objects, we can use a polygon clipping algorithm to test for collision.

BGL uses rules to avoid and resolve collisions. Below we present examples of such rules:

Rule 1. If iconel *x* is a human target and iconel *y* is an armament/weapon then *y* should be posted on x and made 80% of x size:

$$x \hookleftarrow (80\%y).$$

Rule 2. If iconel *x* is a human target and iconel *y* is a target modifier then *y* should be posted on the predefined spot (m_1,m_2) on frame *f*:

$$f(m_1,m_2) \hookleftarrow (30\%y),$$

where m_1 and m_2 depend on *x*, $m_1=m_2(x)$; $m_2=m_2(x)$.

Rule 3. If iconel *x* is a human target in record #1 and iconel *y* is a human target in record #2 and $x \neq y$ then *y*'s color should differ from color of x:

$$f(x) \hookleftarrow \text{colorAlternate}(n\%y).$$

The last rule is an example of color alternation rules. To limit complexity of icons, BGL prohibits posting expressions (formulas) with more than four posting operators. Alignment of posted icons in the case of their discrepan-

cies is made by applying conflation algorithms. This dynamic iconic language has an advantage over hardcoded icons; formulas like those presented above can produce icons uniformly.

There has been related work conducted in the area of icon algebra [Chang, 1989; Chang, Polese, Orefice & Tucci, 1997]. The main differences between our work and these developments are in the goals. We are interested in automatic dynamic icon generation while Chang at al. are focused on deriving natural language concepts (from icons) by applying formal operators on the icons and their semantic meaning, which are limited to a set of predefined icons and meanings. This differences in focus produce *rich semantic operations* in Chang's icon algebra and rich *iconic operations* in our iconic algebra. These two sets of operations can be combined if some goal were to require both types of operations.

The motivation of Chang's icon algebra can be illustrated by the following situation. Assume that we have a person who can not speak, but can communicate by composing iconic sentences from a limited set of icons and their modifiers. Having a limited set of icons, much smaller than the total number of words in English, a person who cannot speak is forced to combine available icons and modifiers to produce other words.

For some words, the combinations can be acceptable, leading a good approximation, while for other words, the result can be almost arbitrary or perhaps a distant metaphor. To express the word "cold," a person who cannot speak could select two icons "icicle" and "thermometer." The problem we face is that we want to retrieve just the word "cold" and not the word "winter" or "solid body" from these two icons.

Our goal is a quite different. We assume that we are providing a system for a decision maker or an analyst who can speak, write, select, and analyze images including icons, but that the user does not have time to read a long text. That is, we assume such a person would prefer to read and "decipher" an iconic sentence that summarizes long text.

Further, we assume that the user can get the exact icon meaning by, say, placing the mouse over an icon causing the icon meaning to pop up and be read. Thus, the major problem of Chang-type of applications simply does not exist in our environment. In a Chang application, a person with a speech disability is very limited in his abilities to combine icons and to produce the exact icon that is needed. We have no such severe limitation; icons can be combined extensively and thus a relatively large set of concepts can be directly encoded in icons.

5. CASE STUDIES: CORRELATING TERRORISM EVENTS

Iconic annotation of database records means creating another database, where some textual attributes of records are augmented or substituted by icons. There are several reasons for an iconic annotation of databases. Iconic annotations can help navigate the database, search records visually, correlate records, discover useful patterns in databases, and support decision making. People usually process alphanumeric textual information sequentially, but can process images (icons) in parallel, which is much faster.

The Bruegel visual correlation system has several potential applications in homeland security, defense, intelligence and crime prevention through activities such as tracking terrorist activities, which includes identifying modus operandi and estimating future terrorist threats, tracking weapons of mass destruction, and drug trafficking.

5.1 MUC data description

In this case study, we use DARPA MUC-3 and MUC-4 data on terrorist activities in Latin America in 1980s. MUC data are now in the public domain at NIST and downloadable from [MUC Data Sets, NIST, 2001].

Table 2 presents a summary of the raw text corpus and Table 3 provides a sample of raw text message.

Table 2. MUC raw text corpus description

Characteristic	*Description*
1. Data sources	The Foreign Broadcast Information Service.
2. Text types	Newspaper and newswire stories, radio and TV broadcasts, interviews, and rebel communiqués summary reports, transcripts from speeches and interviews
3. Location	Latin America
4. Original language	Spanish
5. Text grammar	Well-formed sentences, all are in upper case
6. Number of texts s	1300 texts
7. Individual text size	Average size is 12 sentences (~0.5 a page), smallest text -one paragraph, largest text -two pages
8. Number of sentences	15,600 sentences
9.Number of unique lexical items	18,240 lexical units
10.Number of words	400,000 words
11.Number of events in a text	1-5 events per single text source
12.Average sentence length	27 words
13.Timeframe	1980s
14.Terrorism relevant texts, %	50%

DARPA has sponsored content extraction competitions based on this text corpus [Cardie, 1994; Hobbs, Appelt, Bear, Israel, Kameyama, Stickel, & Tyson, 1996; Lehnert, Sundheim, 1991; Lehnert, Cardie, Fisher, McCarthy, Riloff, & Soderland, 1992a,b; Lehnert, 1994; MUC-3, 1991; MUC-4, 1992]. Competitions are called Message Understanding Conferences (consider specifically MUC-3, 1991 and MUC-4, 1992).

Table 3. A sample of raw text

DEV-MUC3-0008 (NOSC)
BOGOTA, 9 JAN 90 (EFE) -- [TEXT] RICARDO ALFONSO CASTELLAR, MAYOR OF ACHI, IN THE NORTHERN DEPARTMENT OF BOLIVAR, WHO WAS KIDNAPPED ON 5 JANUARY, APPARENTLY BY ARMY OF NATIONAL LIBERATION (ELN) GUERRILLAS, WAS FOUND DEAD TODAY, ACCORDING TO AUTHORITIES.CASTELLAR WAS KIDNAPPED ON 5 JANUARY ON THE OUTSKIRTS OF ACHI, ABOUT 850 KM NORTH OF BOGOTA, BY A GROUP OF ARMED MEN, WHO FORCED HIM TO ACCOMPANY THEM TO AN UNDISCLOSED LOCATION. POLICE SOURCES IN CARTAGENA REPORTED THAT CASTELLAR'S BODY SHOWED SIGNS OF TORTURE AND SEVERAL BULLET WOUNDS. CASTELLAR WAS KIDNAPPED BY ELN GUERRILLAS WHILE HE WAS TRAVELING IN A BOAT DOWN THE CAUCA RIVER TO THE TENCHE AREA, A REGION WITHIN HIS JURISDICTION. IN CARTAGENA IT WAS REPORTED THAT CASTELLAR FACED A "REVOLUTIONARY TRIAL" BY THE ELN AND THAT HE WAS FOUND GUILTY AND EXECUTED. CASTELLAR IS THE SECOND MAYOR THAT HAS BEEN MURDERED IN COLOMBIA IN THE LAST 3 DAYS. ON 5 JANUARY, CARLOS JULIO TORRADO, MAYOR OF ABREGO IN THE NORTHEASTERN DEPARTMENT OF SANTANDER, WAS KILLED APPARENTLY BY ANOTHER GUERILLA COLUMN, ALSO BELONGING TO THE ELN. TORRADO'S SON, WILLIAM; GUSTAVO JACOME QUINTERO, THE DEPARTMENTAL GOVERNMENT SECRETARY; AND BODYGUARD JAIRO ORTEGA, WERE ALSO KILLED. THE GROUP WAS TRAVELING IN A 4-WHEEL DRIVE VEHICLE BETWEEN CUCUTA AND THE RURAL AREA KNOWN AS CAMPANARIO WHEN THEIR VEHICLE WAS BLOWN UP BY FOUR EXPLOSIVE CHARGES THAT DETONATED ON THE HIGHWAY.

The case study uses **structured data** called the development corpus that has been produced by 15 MUC teams from raw texts during MUC-3/MUC-4 using manual categorization and tagging. The MUC-3/MUC-4 task was to *automatically* extract information about terrorist incidents from raw **test texts** compiled for two years using the structured development corpus as training data. The goal was to determine when a given text contained relevant or irrelevant information. Each team stored their results in a **template format**, one template per event.

The template format contains with 24 attributes called slots. The text example above has been converted to two output templates because it describes two terrorist events; the kidnapping of Castellar and the bombing of Torrado's four-wheel drive vehicle.

Template attributes cover the *date and location* of the incident, the *type of incident* (24 types of violence), the *perpetrators*, *victims,* and *physical*

targets in the attack, and the *effect* on the target. If there is more than one value for a slot then options are separated by a "/" [Hobbs et al., 1996].

The twenty-four types of violence include eight basic types (such as murder, bombings, kidnappings, arson, and attacks) plus two variations on each (for threatened incidents and attempted incidents). There are also *judgmental attributes* such as *perpetrator confidence*, concerning the reliability of the perpetrator's identity.

The manual process of obtaining structured data as a development corpus was time consuming and non-trivial:

> ... it takes an experienced researcher at least three days to cover 100 texts and produce good quality template representations for those texts. This is an optimistic estimate, which assumes familiarity with a stable set of encoding guidelines [Lehnert & Sundheim, 1991]

Although automatic content extraction is not the focus of our research, MUC results set up a benchmark for the level of effort needed for obtaining structured data as input for Bruegel iconic visual correlation system.

5.2 What needs to be visualized in icons?

The algorithm for deciding what part of a textual message will become iconic and what will be placed into the "mouse over" pop-up uses three criteria:

- Terms included in an ontology (key-words) go into the icon,
- The most frequent terms go into the icon, and
- Personal names and individual organization names go into the "mouse over" pop-up.

To build an ontology, we analyzed the frequencies of MUC terms. The most frequent terms have been found in the following steps:

Step 1. Create a parser to dissect the MUC files, particularly the data fields where *three classes of phrases* were defined based on their context:
KEYWORDS such as accomplished and civilian.
PLAIN_TEXT_STRINGS such as names and extended text descriptions.
DESCRIPTORS such as further data about locations, e.g., city and town.

Step 2. Run the parser on MUC data

Step 3. Format and track the results with regard to their usage count and location within categories and sub-sections.

Step 4. Generate a list of the 100 most frequently occurring phrases in the MUC_TST_1 and MUC_TST_2 files.

Step 5. Design icons for the 100 most common words tracked, which constitute a large portion of the data contained in the MUC files.

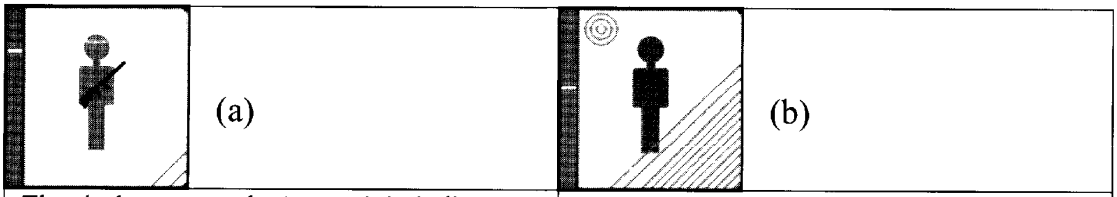
As the usage count drops below fifteen, the PLAIN_TEXT_STRINGS more often are proper names that would be better suited to "mouse over" data, than to visualization.

A large number of the words identified were KEYWORDS. These were the most likely candidates to be represented as fairly static icon elements often forming iconels. Some PLAIN_TEXT_STRINGS occurred often enough that they could be considered key words. Dynamite was a commonly occurring word of this type. Most other PLAIN_TEXT_STRINGS were good candidates for "mouse over" data (MOD) available on a lower level of information. Finally, descriptors are always associated with a location and thus can be represented as simple iconel identifiers as we will see below.

We defined the following characteristics for *location: name, geographical location (coordinates), importance/threat, classification (friend, hostile, neutral)* that needed to be conveyed by the system. Similarly, for people, we defined location characteristics with associated *nation, affiliation* and several others.

5.3 A Demonstration

Characteristics that take on graduated values are prevalent in the MUC files. They include: organizational confidence; damage (physical and human) and quantity (number and total number). We explored a variety of options for visualizing these values. Some options are shown in Figure 9. Both options in this figure use "slash" scales incorporated into the icons.



The slashes across the lower right indicate that this is a small band (blue scale) of armed rebels (or terrorists).	This icon describes a fairly large number of civilian targets (blue scale) were hurt pretty badly (red scale) by some action.

Figure 9. Possible icons for MUC concepts. See also color plates.

In some cases, it may be beneficial to encode more than one parameter in such scales. For instance, the number of targets and the level of damage caused to them can be encoded in the same icon. This can be accomplished by offsetting on two scales. Figure 9(b) shows offsetting of red scale (level of damage) and blue scales (the number of people) to avoid their overlapping. Usability studies have shown that users can easily read case 9(a) while users need some accommodation time for case 9(b).

The intent of the vertical scale on Figure 9 is to represent confidence level about the truth of the data. In particular, it can show low confidence in

the quantity or quality of the information displayed in the icon. Thus, the meaning of the vertical scale and "slash" scales is **content-dependent.**

The meaning depends on the main content of the icon. For example, a yellow mark in the green section of the vertical scale in Figure 9(a) can indicate a high confidence in the data depicted in this icon. It clearly can not represent confidence in other data not present in the icon.

The vertical scale also can be interpreted as two additional content-dependent scales that represent the relevance of the icon content to the evaluating party. The top green half can represent importance, the bottom red half can represent threat. Assigning values may require more information about an item than is available from the source being visualized. In this case there must be some database available that can supply additional information.

Figures 10-14 show the icons developed as a result of the analysis of the most common words in the MUC files. These icons are used in Bruegel's **listview** to represent records for visual correlation. Another view implemented in Bruegel is called **macroview.** It is mostly is based on simple filled rectangles possibly combined with simple texture. These simple icons allow the encoding of more records into a single screen than the rich listview icons allow, but macroview icons provide less detail.

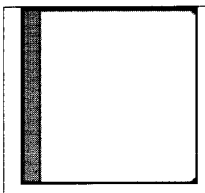
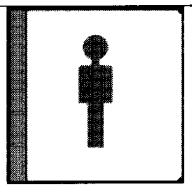
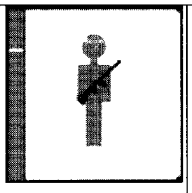
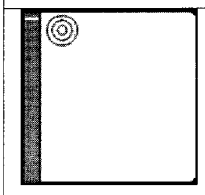
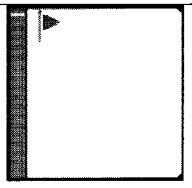
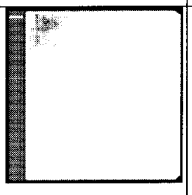
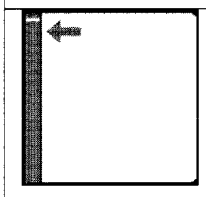
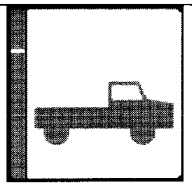
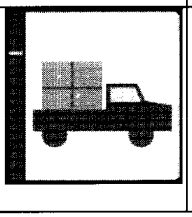
	the basic frame with red relevance scale and		civilian		terrorist (without the gun he can be just a rebel)
	target modifier (iconel)		stage of execution (accomplished)		stage of execution (threatened)
	perpetrator, highly relevant (yellow mark on green)		land vehicle		land transport vehicle

Figure 10. Bruegel icon examples: base icons. See also color plates.

Bruegel also allows complex combinations of icons and iconic sentences: some icons can represent information in a very detailed way, and some icons can be generalized as macroview icons.

These combinations serve as a tool to present information on **multiple levels simultaneously**. Records presented as iconic sentences and located on the same screen potentially generate a variety of **patterns** that can aid in their visual correlation. In addition, **contradiction** between records can be revealed in these patterns.

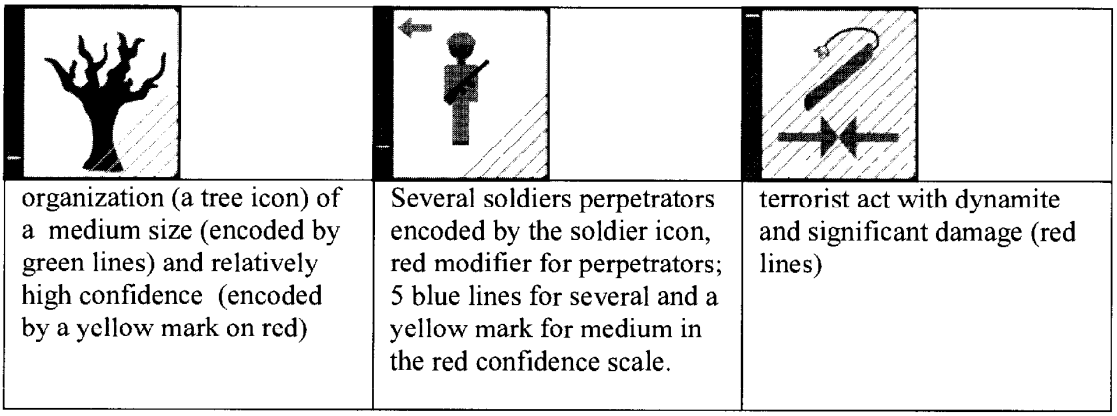


Figure 11. Bruegel composite icons

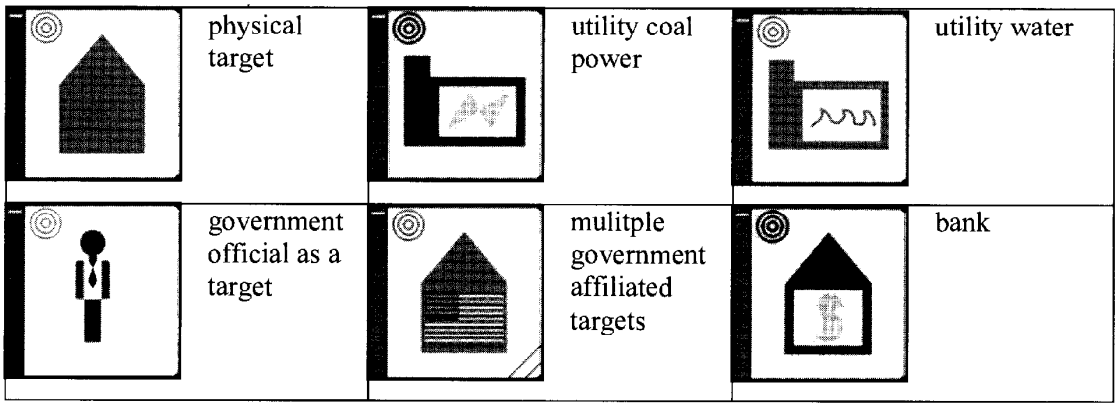


Figure 12. Bruegel icon examples: targets

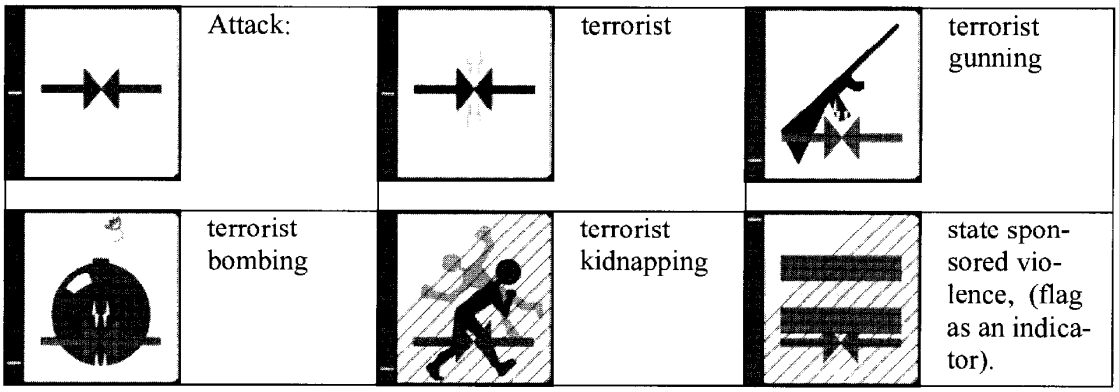


Figure 13. Bruegel icon examples: attacks

In macroview, we deliberately designed some icons to be able to produce large visible patterns although each individual icon or a spider's web.

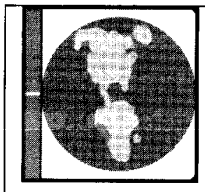
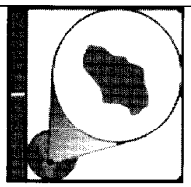
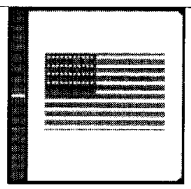
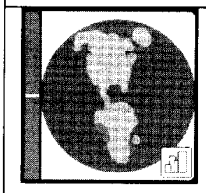
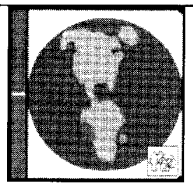
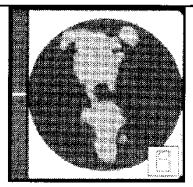
	Location: icon type 1. Red shows country's location.		Location: icon type 2. Lens shows a small country.		location: icon type 3.The flag indicates a country.
	Location icon with city modifier		Location icon with town modifier		Location icon with department modifier

Figure 14. Bruegel icon examples: location icon types. See also color plates.

The convention of BIL is to use a base icon and attached modifiers. For instance, the first icon in the second row of Figure 12 shows a government official as a target using the *target modifier*. This official was also pretty important as can be seen by the scale on the left. Similarly, the icon in the middle of the second row shows that a fair sized band of soldiers from a rather hostile government committed some act, most likely against something or someone important to the country. If more than one target is described it is suggested that, in order to not hinder the intuition, only the most important target be represented, and then the others implied with the Quantity scale shown as here.

The iconic language presented above provides one of the examples of languages that can be loaded to the Bruegel system along with the appropriate and icons matched to the ontology. Some implemented examples are presented in Figure 15. A time test for reading iconic sentences in Bruegel is available on line [Kovalerchuk, Brown & Kovalerchuk, 2001].

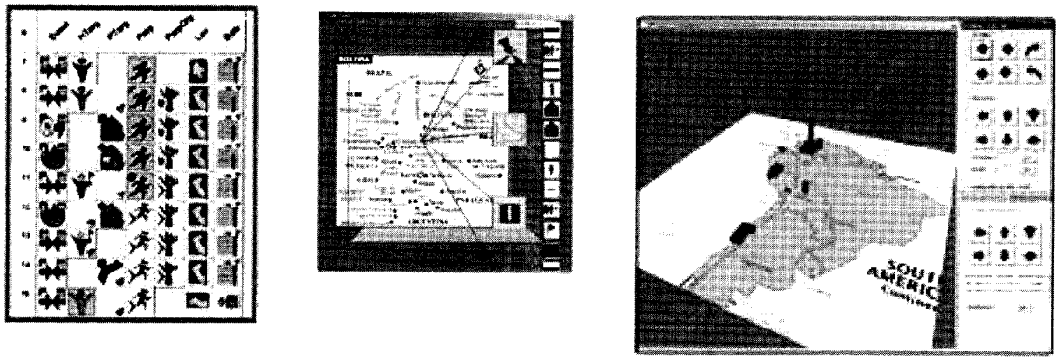


Figure 15. Bruegel case studies. See also color plates.

6. CASE STUDIES: CORRELATING FILES AND CRIMINAL EVENTS

6.1 Visualization for file system navigation

Computer users spend millions of hours navigating file systems with thousands of files using tools such as Windows Explorer. Iconic annotation of DB can make this task easier. Currently iconic database annotating is in its infancy stage. For instance, Windows Explorer shows 1-5 attributes of files and uses a single icon. To get more information about a file, a user needs to dig deeper; often it requires opening a file and browsing its contents, which can be time consuming.

Available search tools can help only if a useful keyword is known. In a number of circumstances, such a keyword is unknown. In cases like this, users might be more successful in their search by carrying out their navigation in an iconic file system. Consider first the information provided by Windows Explorer and similar tools. The "Large icons" option shows the file name, fie type, and an icon while the "Details" option additionally provides the file size, time of modification and attributes such as archive, read-only, compressed, hidden, and system. To get more information a user pops up a "properties" menu for each individual file. In this way a user will get more information such as when a file was created, modified and security details (permissions, auditing, and ownership).

Table 4. Iconic sentences that summarize file decryptions in a file system

Date created	*Name*	*Modified*	*Category*	*Security*	*Platform*	*Similar to X*	*Size*
	Xx1				**Win 32**		
	Yy1				XP		
	Zz1				**Win 16**		

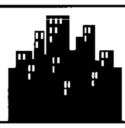
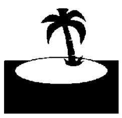
Such a traditional GUI interface does not help to compare/correlate these attributes for different files, because it shows one file at a time. Table 4 illustrates an iconic description of files. Here category can be represent attributes

such as archive, compressed, system, or hidden, while security can be represented by permissions, auditing, and ownership.The size of the icons in the "Size" field represent file size. The larger icon represents a larger file. We use a logarithmic scale to encode size. Similarly, a larger calendar icon represents more recent dates of file creation and modification. For instance if a user wants to free some space by looking for old large files that can be deleted, the size of the icons for dates and file sizes quickly reveal those files. In general, the whole ontology of file system concepts can be encoded in icons in our Bruegel system as XML files. Visual correlation and navigation in such visual sentences (visual databases) can be beneficial in a variety of applications some of them we illustrate below.

6.2 Visual correlation for drug trafficking

An iconic annotation of drug trafficking database can make explanatory analysis easier. In addition, the same general benefits of iconizing the DB are applicable here. These benefits include intuitive iconic queries and quickly browsing the DB contents. A traditional DB GUI interface was not designed to assist in comparing/correlating drug trafficking records. Table 5 illustrates an iconic description of drug trafficking records. The depicted categories include date, location, offender, delivery storage, transit point, value, witness, and legal aspects. Additional categories may include a record ID and security information.

Table 5. Iconic sentences that summarize drag trafficking records

Date	*Location*	*Offender*	*Delivery, storage*	*Transit point*	*Value*	*Witness*	*Legal*
					$$$ $		
					$$		
					$	--	

As before, the whole ontology of drug trafficking can be encoded in icons as XML files. Similar to file system navigation, the size of the icons also conveys information. For instance, the size of the "Legal" icon can represent the amount of legal information collected or level of potential legal

implications for the offender. Visual correlation and navigation in such visual sentences can reveal patterns and new trends in drug trafficking.

6.3 Visual correlation of criminal records

Much like the iconization of a drug traffic DB, an iconic annotation for criminal records can improve the ease with which explanatory studies are conducted. Table 6 illustrates an iconic description of criminal records that includes date, location, offender, tools, victim, harm, witness, and legal aspects.

Visual correlation of displayed records quickly reveals the variety of locations, tools, victims, and witnesses of the criminal acts on the same day and similar categories of offenders.

Table 6. Iconic sentences that summarize criminal records

Date	*Location*	*Offender*	*Tools*	*Victim*	*Harm*	*Witness*	*Legal*
					--		
							
					?	--	

7. CASE STUDIES: MARKET AND HEALTH CARE

7.1 Visual correlation of the real estate market

A real estate market provides another example where an iconic annotation of a DB provides a simpler explanatory analysis. Consider for example, comparing several offerings at once and selecting the most appropriate one.

Table 7 illustrates an iconic description of real estate offerings that includes date, available transport nearby, contact information (phone, fax), price range, number of floor and rooms, quality of schools nearby, and legal

aspects. Visual correlation of displayed records quickly reveals the variety of prices, sizes, and other parameters of offerings to select from.

Table 7. Iconic sentences that summarize houses for sale

Date	*Trans-port*	*Phone Fax*	*Price*	*Number of floors*	*Rooms*	*School*	*Legal*
			$$$$				
			$$				
			$$$				

7.2 Visual correlation for job search

Job market records can be naturally iconized again enhancing explanatory analysis. Here the activities include comparing available jobs and candidates. Table 8 illustrates an iconic description of jobs available that includes due date, available transport nearby, contact information (phone, fax), pay package, benefits, relocation package, career growth opportunities, and job requirements. A larger size of icons is indicative to a larger benefits and requirements. An icon "6FP" stands for "six figure pay". There is a natural order between pay icons: 6FP>$$$>$$.

Visual correlation of the displayed records quickly reveals that there is one highly paid job with significant benefits and requirements, another job is more modest in these parameters and the third job opening is a well paid but with lower benefits and career growth opportunities. Such a visual comparison demonstrates an easier method for multicriteria comparison along with decreasing the possibility of memory overload. The Bruegel system supports the identification of otherwise non-comparable offerings that are closest to a given job description (query).

In more formal terms these offerings can be represented as a set of n-dimensional vectors $\boldsymbol{x}_i=(x_{i1},x_{i2},\ldots,x_{in})$, where each vector corresponds to a row in Table 8 or its extension that contains more rows. Parameters such as pay package, benefits, relocation package have a natural order of their values, e.g., for pay packages we have \$50000< \$60000. However, the two vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ might not be ordered, e.g., rows 2 and 3 in Table 8 are not ordered – row 2 indicates a lower pay package, but higher benefits than row 3. In general, there is only a partial order between vectors. Vector $\boldsymbol{x}_i$ is called a

Pareto optimal vector in the set of vectors X if there is no other vector in X that has all parameters equal or greater than x. A set X can have several Pareto optimal vectors. These vectors form a *Pareto optimal border*.

Table 8. Iconic sentences that summarize open positions

Due date	*Trans-port*	*Phone Fax*	*Pay package*	*Benefits*	*Relocation package*	*Career growth*	*Require-ments*
			6FP				
			$$				
			$$$				

7.3 Visual correlation for medical research

Our investigations have shown that medical databases can specifically benefit from iconic visualization, e.g., breast cancer databases. Figure 16 illustrates such an application.

It is also interesting to note that here we provide not only an annotation of alphanumeric data but also an annotation of the image database associated with the mammography X-ray images. Image information is especially appropriate for iconization.

The shape of tumor can be sketched in an icon more easily and precisely than it can be described in text. Figure 16 shows records of several patients, where the 12 columns represent features of a mammogram such as tissue density, tumor shape, tumor size, and calcification. The last column indicates a type of tumor, benign or malignant, where the red shape indicates cancer and the white shape indicates benign tumor.

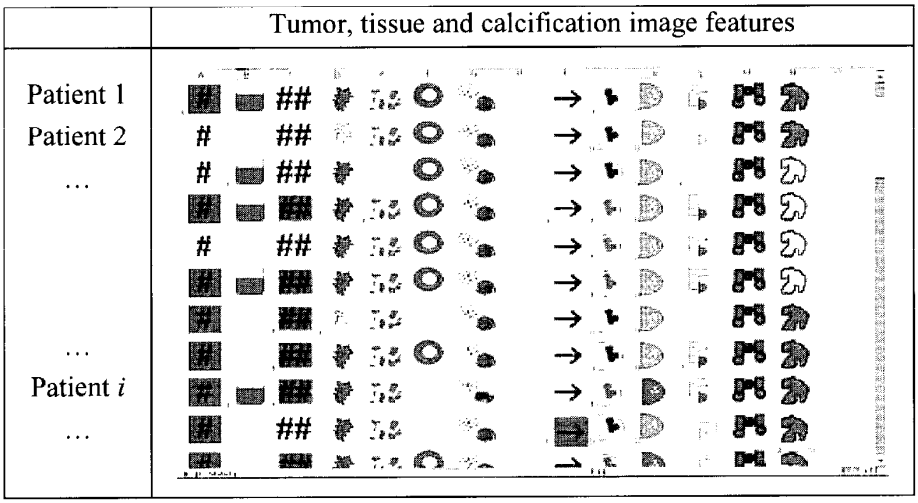


Figure 16. Iconic representation of breast cancer X-ray images

Numeric attribute values are also shown in this representation next to icons. However, numeric values are much less helpful for insight about feature patterns. For instance, the red/white tumor icons in the last column are immediately identified as cancer positive and cancer negative cases while encoding, say, 1 for cancer cases and 0 for benign cases is purely arbitrary.

There are many more reasons for an iconic annotation of medical image and databases; these include the fact that iconic annotations aid navigation through a set of patients' records, aid searching for patients with specific features visually, aid in correlating patients, aid in discovering useful cancer and benign patterns in a database, and aid in supporting diagnosis and treatment. All these advantages are based on unique the human ability of parallel image processing and the quick discovering visual patterns that were developed over millions of years of evolution.

8. CONCLUSIONS

This chapter described the basic concepts of the Bruegel iconic visualization and visual correlation system. The description includes Bruegel functionality and Bruegel's ability to compress information via iconic, semantic zooming, and dynamic iconic sentences. The chapter provides a brief description of Bruegel's architecture and tools.

The formal Bruegel iconic language for automatic icon generation is also outlined. BGL specifies the layering of iconographic elements into complex icons in order to represent textual content in a space efficient manner. *Dynico*, is a supporting tool for the BGL and aids in the generation complex

icons dynamically. *3DGravTree*, described briefly in section 1, is another a supporting tool for the BGL which generates complex 3-D trees of icons dynamically. Other graphical tools in Bruegel support BGL by correlating visually complex objects and events.

Several case studies describe the capabilities of the Bruegel iconic architecture which can be used for the visual correlation of a variety of tasks from terrorist events, to file system navigation, to drug trafficking, to criminal record presentation, to the real estate and job markets, and finally to medical research, diagnosis and treatment.

Future research will look at the development of an iconographic algebra that fits the visual correlation tasks. It will also address scriptable interactivity and the incorporation of XML-based data representation conventions such as Darpa Agent Markup Language (DAML). The architecture will be expanded to accommodate more tasks such as pattern matching, iconic optimization and decision making.

9. ACKNOWLEDGMENTS

This research has been sponsored by the US Intelligence Community via its Advanced Research and Development Activity (ARDA) and National Geospatial-Intelligence Agency (NGA). This support is gratefully acknowledged along with acknowledging assistance of CWU students Jamie Powers, Chris Watson, Bea Koempel-Thomas, Ping Jiang, Paul Martinez, and Suzanne DeBusschere provided throughout this research.

10. EXERCISES AND PROBLEMS

1. Design a hierarchy of concepts for job market as an iconized ontology with at least three levels, where each concept has its icon. This ontology can permit you to build semantic zooming. To do this describe with icons three job offerings using this iconized ontology at each level. You can animate your semantic zooming in PowerPoint.

Advanced

2. Design a set of iconels for exercise 1 that will permit you to build your icons dynamically on demand. Build a formal language for such iconel combinations including iconel repositioning and resizing.

3. Offer a new application domain for iconized ontology and workout exercises 1 and 2 for this domain.

4. Suggest an advanced visual correlation architecture based on Bruegel architecture that will permit to discover patterns between visual sentences recorded as iconic sequences.

11. REFERENCES

Bliss, C K. Semantography-Blissymbolics. 3rd ed. N. S. W., Sydney. Semantography-Blissymbolics Publications, 1978.

Bliss for Windows, version 6.2, 2002, Handicom, NL. http://www.handicom.nl/DOWNLOAD/BLISSFW/BFW62_UK.EXE

Chang S.K., Principles of Pictorial Information System Design, Prentice-Hall, 1989

Chang S. K., Polese G., Orefice S., Tucci M. A Methodology and Interactive Environment for Iconic Language Design, University of Pittsburgh, Pittsburgh, http://www.cs.pitt.edu/~chang/365/mins.html)

Chang S. K [0]. Bottoni, M. F. Costabile, S. Levialdi, Mussio, P., On the Specification of Dynamic Visual Languages, Proceedings of IEEE Symposium on Visual Languages, Sept 1-4, 1998, Halifax, Canada, 14-21.

Chang S. K [0]. Bottoni, M. F. Costabile, S. Levialdi, Mussio, P., Modeling Visual Interaction Systems through Dynamic Visual Languages, IEEE Transactions on Systems, Man and Cybernetics, Vol. 32, No. 6, November 2002, 654-669.

Cardie, C., Domain-specific Knowledge Acquisition for Conceptual Sentence Analysis, PhD Thesis. Dept. of Computer Science Technical Report, 1994. http://www-nlp.cs.umass.edu/ciir-pubs/UM-CS-1994-074.pdf

Gershon, N., Page, W. What storytelling can do for information visualization? ACM Communications, vol. 4, No 8., 2001, pp. 31-37.

Healey, C. Formalizing artistic techniques and scientific visualization for painting renditions of complex information spaces. IJCAI-01, Seattle, 2001, pp. 371-376

Davis, M. Media streams: an iconic visual language for video annotation, 1995 http://www.w3.org/People/howcome/p/telektronikk-4-93/Davis_M.html#KREF7

Dictionary of Blissymbolics, 2001, http://www.symbols.net/blissymbolics/dictionary.html

Hobbs,J.R., Appelt, D., Bear, J., Israel, D., Kameyama,M., Stickel, M., and Tyson, M. FASTUS: Extracting Information from Natural-Language Texts, Artificial Intelligence Center, SRI International, CA, 1996, http://www.ai.sri.com/~appelt/fastus-schabes.pdf

Kovalerchuk, B, Brown, J., Kovalerchuk, M. Usability study webpage http://www.cwu.edu/~borisk/timetest, 2001

Lehnert, W, Sundheim, B., A Performance Evaluation of Text-Analysis Technologies, AI Magazine, 81-94. 1991, http://www-nlp.cs.umass.edu/ciir-pubs/aimag2.pdf

Lehnert, W.G., C. Cardie, D. Fisher, J. McCarthy, E. Riloff and S. Soderland. University of Massachusetts: Description of the CIRCUS System as Used for MUC-4, in The Proceedings of the Fourth Message Understanding Conference. 1992, pp. 282-288.

Lehnert, W.G., C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. University of Massachusetts: MUC-4 Test Results and Analysis, In: The Proceedings of the Fourth Message Understanding Conference. 1992, pp. 151-158.

Lehnert, W. Natural Language Processing Overview", In: 1993 Research Brochure for the Department of Computer Science at the University of Massachusetts, Amherst. http://www-nlp.cs.umass.edu/ciir-pubs/NLP_overview.pdf

Lehnert, W.G. Cognition, Computers and Car Bombs: How Yale Prepared Me for the 90's, in Beliefs, Reasoning, and Decision Making: Psychologic in Honor of Bob Abelson (eds: Schank & Langer), Lawrence Erlbaum Associates, Hillsdale, NJ. pp. 143-173., 1994, http://www-nlp.cs.umass.edu/ciir-pubs/cognition3.pdf

[MUC-3] Proceedings of the Third Message Understanding Conference (MUC-3). San Diego, CA: Third Message Understanding Conference (Ed. Sundheim, B.M. 1991).

[MUC-4] Proceedings of the Forth Message Understanding Conferences (MUC-4), Morgan Kaufmann Publ., 1992.

MUC Data Sets, NIST, 2001, http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

Spence, R. Information Visualization, ACM Press, Addison-Wesley, 2001

PART 4

VISUAL AND SPATIAL DATA MINING

Chapter 11

VISUALIZING DATA STREAMS

Pak Chung Wong, Harlan Foote, Dan Adams, Wendy Cowley,
L. Ruby Leung, and Jim Thomas
Pacific Northwest National Laboratory, USA

Abstract: We introduce two dynamic visualization techniques using multi-dimensional scaling to analyze transient *data streams* such as newswires and remote sensing imagery. While the time-sensitive nature of these data streams requires immediate attention in many applications, the unpredictable and unbounded characteristics of this information can potentially overwhelm many scaling algorithms that require a full re-computation for every update. We present an adaptive visualization technique based on data stratification to ingest stream information adaptively when influx rate exceeds processing rate. We also describe an incremental visualization technique based on data fusion to project new information directly onto a visualization subspace spanned by the singular vectors of the previously processed neighboring data. The ultimate goal is to leverage the value of legacy and new information and minimize re-processing of the entire dataset in full resolution. We demonstrate these dynamic visualization results using a newswire corpus, a remote sensing imagery sequence, and a hydroclimate dataset.

Key words: Dynamic Visualization, Text Visualization, Remote Sensing Imagery, Hydroclimate Dataset, Transient Data Stream

1. INTRODUCTION

Advancements in telecommunications and high-speed networks have recently created a new category of digital information known as *data streams* [Babcock, Babu, Datar, Motwani & Widom, 2002]. This time-varying in-

formation has the unique characteristic of arriving continuously, unpredictably, and unboundedly without any persistent patterns. Data stream examples include newswires, Internet click streams, network resource measurements, phone call records, and remote sensing imagery. The increasing demands of immediate analyses and actions on these transient data streams in many time-sensitive applications such as Homeland Security have spawned a series of investigations [Babu & Widom, 2001; Cortes, Fisher, Pregibon, Rogers & Smith, 2000; Domingos & Hulten, 2000; O'Callaghan, Mishra, Meyerson, Guba & Motwani, 2002] to query, mine, and model the information through nontraditional approaches. This chapter focuses on finding a visual-based solution to the fast-growing research area with demonstrations using text, imagery, and climate streams.

Generally, visualizing transient data streams requires fusing a large amount of previously analyzed information with a smaller amount of new information. This new information is at least as important as its larger counterpart because the *resultant visualization* is entirely dependent on the *dataset* and the *user parameters* applied to it. Thus, the whole dataset must be reprocessed in full resolution in order to capture the finest details. In reality, this is a challenging task given the unbounded and unpredictable nature of the streams.

Our first objective is to develop an adaptive visualization technique that allows one to get the best understanding of the transient data streams adaptively during critical moments when influx rate exceeds processing rate. Our approach is built on the concept of *data stratification* that intelligently reduces the data size in exchange for substantial reduction of processing time. Although we only use classical multidimensional scaling (MDS) [Cox & Cox, 1994] in our investigation, our adaptive technique will work well with other scaling methods because we only modify the data, not the scaling algorithms that process it.

Our second objective is to develop an incremental visualization technique that allows one to project a certain amount of new information incrementally onto an orthogonal subspace spanned by the most important singular vectors of the previously processed data. The design is based on a *multiple sliding window* concept that uses dominant Eigenvectors obtained from a large data window to accommodate the information from a smaller data window without reprocessing the entire dataset.

The primary visualization output of an MDS process is a low-dimensional scatterplot in which pairwise distances between any points reflect the similarities of the items represented by the points. Because a large part of our work is based on progressive approximation and adaptation, error-tracking plays a vital role in showing the viability of our work. We use

both visual- and computational-based means extensively to compare multiple scatterplots simultaneously and report their discrepancies.

Our ultimate goal is to leverage the value of legacy and new information and minimize re-processing the entire dataset in full resolution. Our approach is to turn a prevailing and widespread visualization tool (i.e., MDS), which was initially developed for analyzing static data collections, into a dynamic data analysis tool for transient data streams. This chapter explains the motivations and provides critical technologies to accomplish such a dynamic analysis environment.

2. RELATED WORK

Although the study of data streams is relatively new to the visualization community, it gradually has become one of the most pressing and difficult data problems in today's information technology (IT) world. Recently, [Babcock, Babu, Datar, Motwani & Widom, 2002] presented an overview paper that defines the topic, describes the background, and introduces challenging issues of this young research area. Among the hot research topics are dynamic query [Babu & Widom, 2001; O'Callaghan, Mishra, Meyerson, Guba & Motwani, 2002] and data mining [Cortes, Fisher, Pregibon, Rogers & Smith, 2000; Domingos & Hulten, 2000] of the transient streams.

MDS has always been an important part of information visualization studies. [Wise, Thomas, Pennock, Lantrip, Pottier, Schur & Crow, 1995] and [Wise, 1999] used MDS to analyze large corpora. [Bentley & Ward, 1996] studied the stress property of a class of MDS known as non-metric multidimensional scaling [Cox & Cox, 1994]. [Wong & Bergeron, 1997] applied MDS in a high-dimensional brushing and linking visualization environment. And more recently, [Morrison, Ross & Chalmers, 2002] reported a new MDS technique that algorithmically outperforms all previous implementations. Many of these MDS-based applications can potentially take advantage of our adaptive technique and further improve performance.

This chapter uses text, imagery, and climate streams for demonstrations. The text visualization community has largely been influenced by two groundbreaking works: 1) Salton and McGill's Vector Space Model (VSM) [Salton & McGill, 1983], which represents the documents as numerical vectors, and 2) Deerwester et al.'s Latent Semantic Indexing (LSI) [Deerwester, Dumais, Furnas, Landauer & Harshman, 1990], which effectively projects the vectors into an Eigenspace based on a classical MDS design. Many successful commercial text visualization systems today—such as Aureka [Aureka, 2003], OmniViz [OmniViz, 2003], SPIRE [Wise, 1999; Wise, Thomas, Pennock, Lantrip, Pottier, Schur & Crow, 1995], Starlight [Risch,

Rex, Dowson, Walters, May & Moon, 1997], and VxInsight [VxInsight, 2003]—were developed based on these two models.

Although MDS is frequently applied to text analysis, researchers also use MDS to visualize images and climate modeling information. For example, [Rodden, Basalaj, Sinclair & Wood, 1999] discuss a novel image scatterplot without overlapping and [Wong, Foote, Leung, Adams & Thomas, 2000] present a data signature scheme to study the trend of climate modeling datasets. Of particular note is that the prior work presented here has assumed a static data collection, which is very different from the dynamic data streams discussed in this chapter.

3. DEMONSTRATION DATASET AND PREPROCESSING

We use a newswire corpus, a remote imagery sequence, and an observed hydroclimate dataset for demonstrations. We describe the datasets and preprocessing steps to generate the required *vectors* (which represent individual units) and *matrices* (which represent the whole dataset) for MDS analysis.

3.1 Document Corpus

The demo corpus has 3,298 news articles collected from open sources during April 20-26, 1995. It has a strong theme associated with the bombing of the U.S. Federal Building in Oklahoma, the O.J. Simpson trial, and the French elections.

The first step in processing the corpus is to identify a set of content-bearing words [Bookstein, Klein & Raita, 1998] from the documents. Words separated by white spaces in a corpus are evaluated within the context of the corpus to assess whether a word is interesting enough to be a topic. The co-occurrence or lack of co-occurrence of these words in documents is used to evaluate the strengths of the words.

The second step is to construct the *document vectors* for the corpus. A document vector, which is an array of real numbers, contains the weighted strengths of the interesting words found in the corresponding document. These vectors are normalized and the result is a *document matrix* that represents the corpus. In our example, a document vector contains 200 numbers. Because there are 3,298 documents, the dimensions of the document matrix are 3,298×200. Details of our text engine can be found in [Wise, Thomas, Pennock, Lantrip, Pottier, Schur & Crow, 1995; Wise, 1999].

3.2 Remote Sensing Imagery

The remote sensing imagery sequence used in our demonstration was taken by an aircraft over the semi-desert area in Eastern Washington. The aircraft was equipped with a hyperspectral sensor that could take multiple images of the same locations simultaneously in different spectral bands. Figure 1a illustrates the basic concept of the scanning operation.

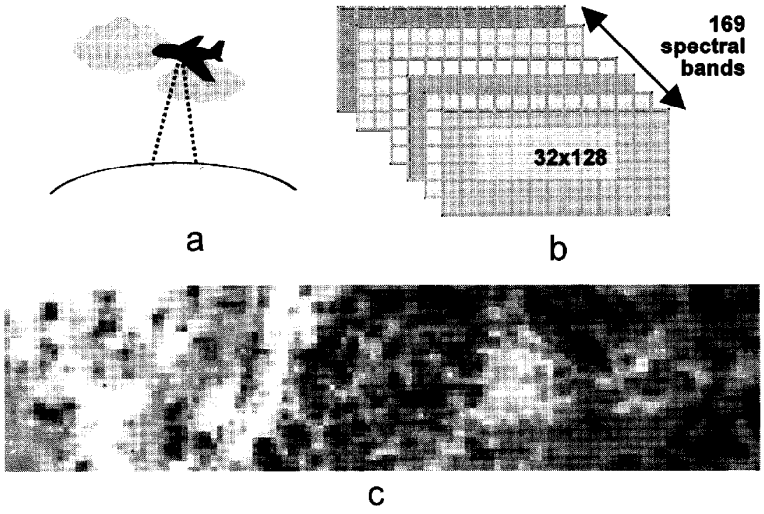


Figure 1. a) An illustration of the operation. b) A sketch of a hyperspectral image set with 169 spectral bands ranging from very short to very long bands. c) A color infrared (CIR) imagery of the semi-desert area in Eastern Washington. See also color plates.

The processing of the remote imagery information is straightforward. We want to extract features of the area by analyzing the similarities of the image pixels from different spectral bands. In our examples in Sections 6.4 and 8, the image in each spectral band (or layer) has 32×128=4096 pixels, as illustrated in Figure 1b. Figure 1c depicts a color infrared (CIR) image shot in infrared band. A *pixel vector*, in this case, contains image information of the same pixel position across the 169 spectral bands. In other words, each pixel position establishes a pixel vector. Because there are 4,096 pixels in each image, the dimensions of the *pixel matrix* are 4,096×169. For larger images we can sub-sample the image and control the size of the matrix.

3.3 Hydroclimate Dataset

The hydroclimate dataset used in our study consists of daily maximum and minimum surface temperature and precipitation gridded at one-eight degree of the western US from 1949 to 2000. Gridded data were produced following the methodology outlined by Maurer et al. based on daily observa-

tions made at the National Oceanic and Atmospheric Administration (NOAA) Cooperative Observer (Co-op) stations. A study using this observation dataset with a second simulated dataset was conducted in [Leung, Qian & Bian, 2003; Leung, Qian, Bian & Hunt, 2003].

In our study, 6,155 sampling points are selected regularly from the gridded dataset that covers the entire western US. The surface temperature and precipitation information is combined using a series of Gaussian-weighted spatial smoothing, Fast Fourier Transform, and histogram processes to form a hydro-climate vector with 24 real numbers, which represents the climate characteristics and properties of that location. In other words, the dimensions of the hydro-climate matrix in our study is 6155×24.

4. MULTIDIMENSIONAL SCALING

MDS is a prevailing technique used to visualize high-dimensional datasets. There are a variety of proven MDS algorithms designed for different analytical purposes. (Readers are directed to [Cox & Cox, 1994; MDS, 2003] for a comprehensive overview.) Our investigation focuses on a class of MDS known as classical scaling [Cox & Cox, 1994]. To show the flexibility of our design approach, we also include an example in Section 6.3 using a least-squares-based scaling technique known as Sammon Projection [Sammon, 1969].

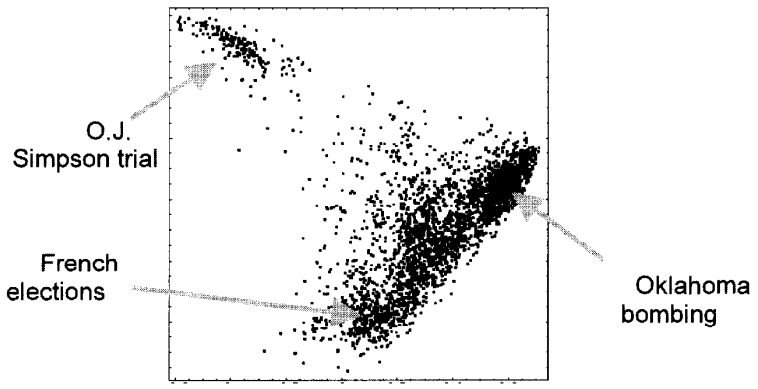


Figure 2. A document scatterplot generated by MDS

Given a high-dimensional dataset (a set of similar data objects represented by numerical vectors), MDS generates a low-dimensional configuration—like a 2-D scatterplot—such that the pairwise distances between any points in the low-dimensional space approximate the similarities between the vectors that represent the points. For example, Figure 2 shows a scatterplot with 3,298 points (each point represents a document vector) generated by a

classical MDS process using the corpus described in Section 3.1. In this example, documents with similar themes are clustered together as annotated.

5. ADAPTIVE VISUALIZATION USING STRATIFICATION

We present an adaptive visualization technique based on data stratification to substantially reduce the processing time of the streams and yet maintain the overall integrity of the MDS visualization output. The data ingest scheme that supports the technique is illustrated in Figure 3. If the primary data processing route (the bottom pipe) has overflowed, the data are redirected to a secondary one (the middle pipe), which generates a coarser version of visualization but at a much faster processing rate. Also when the middle pipe has overflowed, too, the data go to upper red one and so on. The two data stratification strategies presented in this chapter are *vector dimension reduction* and *vector sampling.*

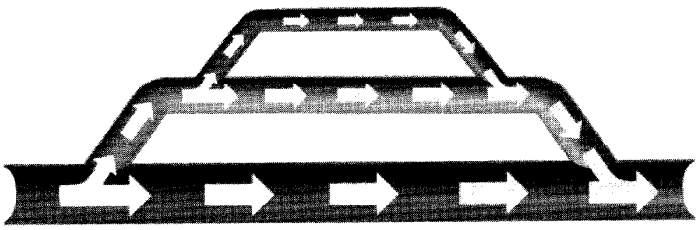


Figure 3. An adaptive ingest scheme for stream visualization

5.1 Vector Dimension Reduction

Our first stratification strategy is to cut down the dimensions of the data vectors. The biggest challenge is to reduce the physical size of the vectors but maintain most of their important contents. We accomplish this by applying dyadic wavelets [Mallat 1998; Strang & Nguyen, 1997] to decompose individual vectors (and thus compress them) progressively. While the theory of wavelets is extensive, our experiments show that the rectangular (piecewise-constant) Haar wavelets perform well in all our visualizations. Not surprisingly, the basic Haar also outperforms all the other wavelet candidates in processing time, which is considered a top priority for analyzing data streams [Gilbert, Kotidis, Muthukrishnan & Strauss, 2001].

Figure 4 shows an example of two consecutive wavelet decompositions on a document vector randomly selected from the demo corpus (described in Section 3.1.) Figure 4a is the original vector with 200 terms. Figure 4b is the

result of the first wavelet decomposition with 100 terms. Figure 4c is the result of the second wavelet decomposition with 50 terms. Because Haar belongs to the dyadic wavelet family, one wavelet application will reduce the vector dimension by 50%.

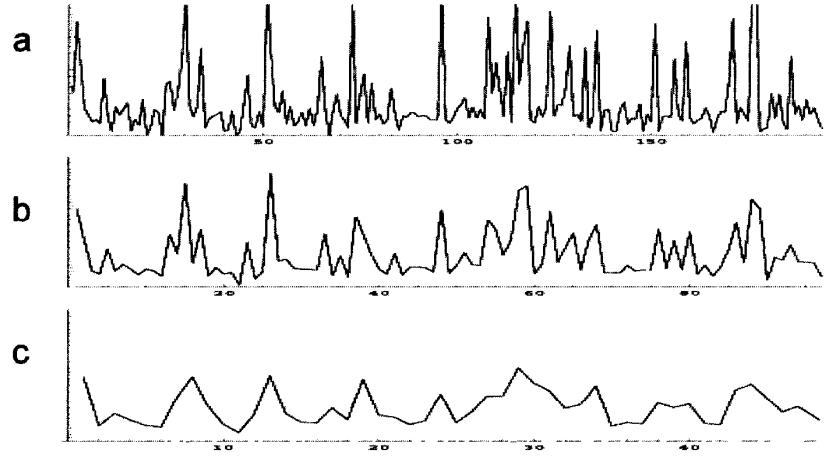


Figure 4. a) A document vector with 200 terms. b) Result of the first wavelet decomposition with 100 terms. c) Result of the second decomposition with 50 terms.

While the example in Figure 4 shows the feature-preserving property of wavelets on individual vectors, the next example demonstrates the accuracy of the resultant vectors in generating visualizations using MDS. We start with the same scatterplot shown in Figure 2, which is generated using a full-resolution matrix of the demo corpus. In order to give visual identities to the scatter points, we apply a K-mean [Seber, 1984] clustering process to the 2D scatterplot and subdivide the points into four clusters. A K-mean clustering process tries to minimize the sum of Euclidean distance from each data point to its cluster centroid. Each cluster receives a unique color (shown in Figure 5a as magenta, cyan, grey, and yellow) for point identification.

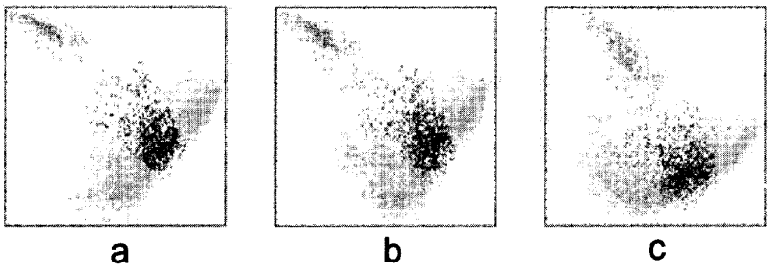


Figure 5. Scatterplots generated by MDS using document vectors with sizes equal to a) 200, b) 100, and c) 50 terms. See also color plates.

Using wavelets, we then progressively reduce the dimensions of the document vectors from 200, to 100, and then to 50. Each reduction is followed by an MDS process; the visualization results are shown in Figures 5b and 5c. Although the orientations and spreads of the scatter points vary slightly in Figures 5b and 5c, major features such as clustering and separation remain.

5.2 Vector Sampling

Our second stratification strategy is to reduce the number of data vectors based on sampling. We use a regular sampling technique to obtain an even data distribution in our example. Other sampling options such as a uniform random sampling function can also be applied.

We first repeat the initial two steps (i.e., generate scatterplot using all 3,298 document vectors and assign color identities to each scatter point) of the last example. Instead of reducing the dimensions of the vectors, this time we progressively reduce the number of document vectors by 50% every time using a regular sampling method. Each sampling process is followed by another MDS to project a scatterplot based on the sampled input. The results are shown in Figure 6.

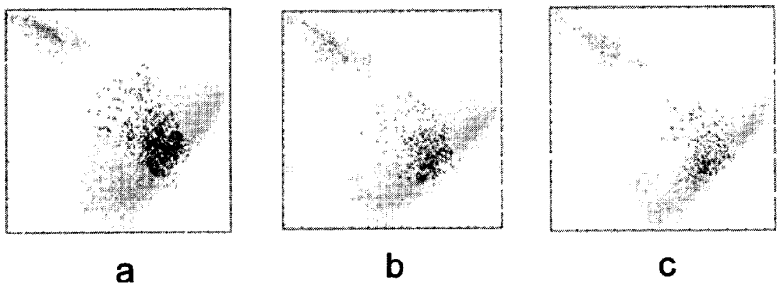


Figure 6. Scatterplots generated by MDS using a) 3298, b) 1649, and c) 824 document vectors. See also color plates.

The three visualizations in Figure 6 demonstrate that even though we substantially reduce the number of vectors for the MDS process, the shape or spread of the scatterpoints remains more or less the same. This phenomenon can be explained by the stability of the two most important Eigenvectors generated by the highly related document vectors. This property helps simplify the structures of a complex dataset for visual analytics and potentially speeds up the time requirement for critical decision making. It also lays the foundation of the data fusion method discussed later in this chapter.

6. DATA STRATIFICATION OPTIONS AND RESULTS

In this section, we discuss the results of the two stratification strategies and their impacts on the resultant visualizations. Our discussion focuses on computation performance and the flexibility in dealing with different data types and different scaling methods.

6.1 Scatterplot Matrix

To improve the visualization for comparison and evaluation, we progressively combine the two approaches and concatenate their results into a scatterplot matrix. The scatterplot matrix in Figure 7 shows the consequences of reducing document vectors (row) versus reducing vector dimensions (column).

The results indicate that although the shape of the point distribution changes to some extent, the overall integrity of the visualizations such as clustering and separation remain intact. The fact that the cluster borders remain clear and crisp in all nine matrix panes indicates a very positive result of our approach. (We will revisit the fidelity issue in Section 7.)

6.2 Computation Performance

Perhaps a more important goal of the stratification effort is to substantially reduce the computation time of the MDS process. While all our numerical programs are implemented locally using C++ on a SUN Ultra 10, we choose to use a commercial package—Mathematica 4.2 [Mathematica, 2003] running on a Macintosh G4 with 1 GB memory—to report the computation performances. The Eigenvalue function used in Mathematica is algorithmically compatible to those championed by Netlib [Netlib, 2003].

Table 1 shows the benchmark results of our study. The top row (in blue) shows the number of dimensions in the document vectors. The left column (in green) shows the number of document vectors included in the computation. The other nine cells are computation time measured in wall clock seconds. The corresponding scatterplot of each cell is shown in Figure 7.

Table 1. Execution times measured in wall clock seconds

	200 Dimensions	100 Dimensions	50 Dimensions
All (3268) Documents	34.90s	9.50s	2.62s
½ (1649) Documents	14.80s	4.78s	1.52s
¼ (824) Documents	8.83s	2.58s	0.89s

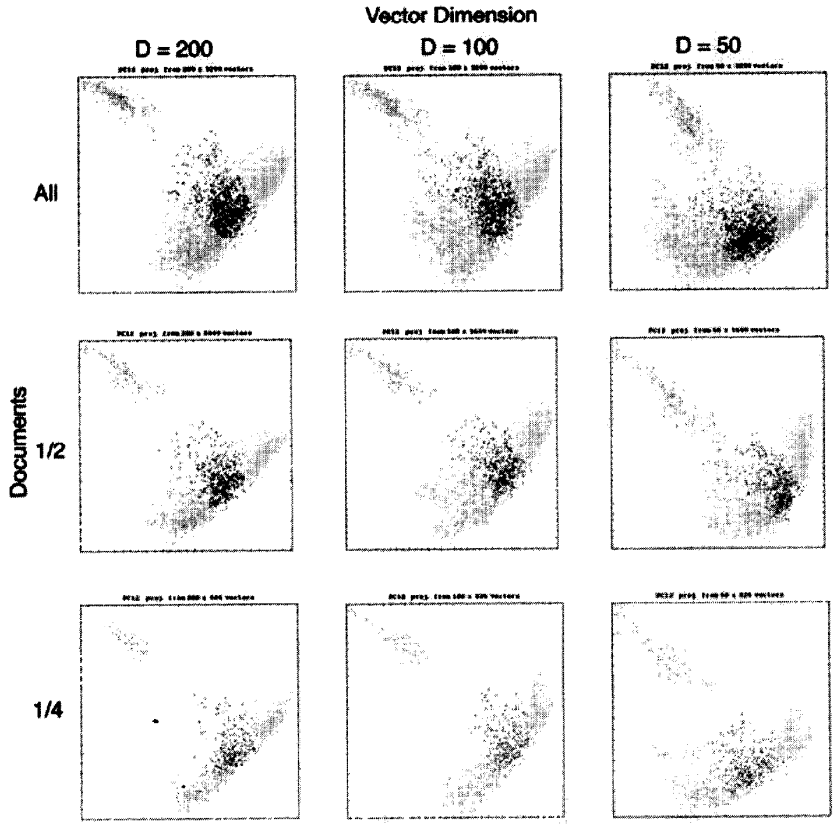


Figure 7. A scatterplot-matrix demonstrates the impact of reducing document vectors (row) versus reducing vector dimensions (column) using classical scaling technique. See also color plates.

6.3 Sammon Projection

So far we have focused only on the classical MDS [Cox & Cox, 1994] in our investigation. To show the flexibility of our adaptive visualization technique, we demonstrate a second scaling example using a least-square MDS technique known as Sammon Projection [Sammon, 1969].

Classical MDS treats similarity between two vectors directly as Euclidean distances whereas least-square MDS takes it as the least squares of a continuous monotonic function. A particular strength of a Sammon Projection over a classical MDS projection in visualization is that the former usually has fewer overlapping clusters due largely to its non-linear mapping approach.

Figure 8 shows a re-execution of Figure 7 using the Sammon Projection technique. Although the visualization results look very different from those in Figure 7 because of the preservation of higher dimensional distances, the impact on the stratification and their results are very much like those in Figure 7. Most of the scatter points are able to maintain their original positions and orientations. The four point colors (red, green, blue, and orange), which

are assigned after a K-mean clustering process, clearly show the integrity of the visualization after substantial dimension and vector reductions.

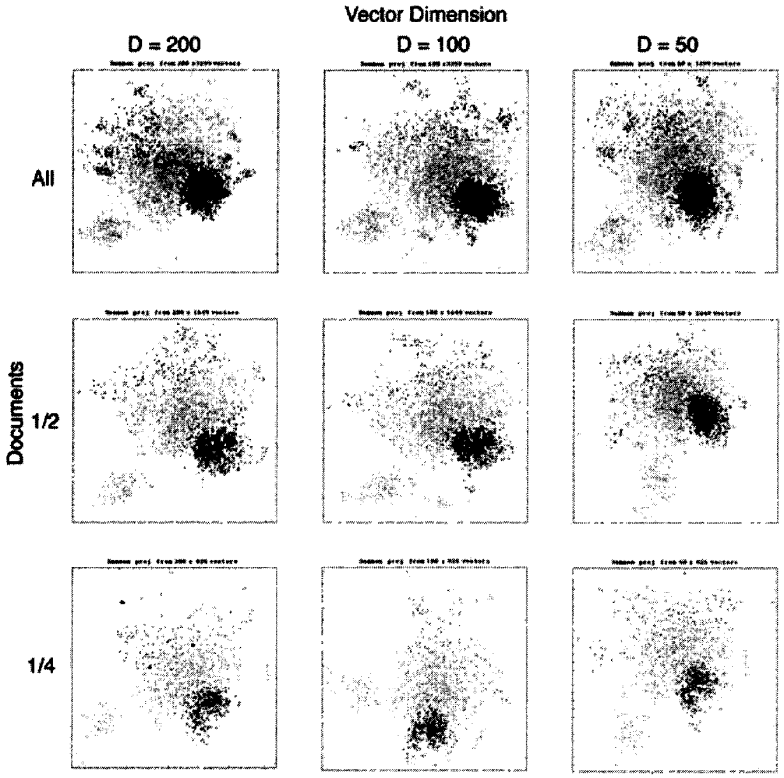


Figure 8. A scatterplot matrix demonstrates the impact of reducing document vectors (row) versus reducing vector dimensions (column) using the Sammon Projection technique. See also color plates.

6.4 Remote Sensing Imagery

Our adaptive visualization technique can also be used to visualize image streams such as hyperspectral remote sensing imagery. A major motivation to include all spectral bands in the image analysis is that subjects that appear identical in one spectral band (like visible color) may be very different from each other if we look into all possible spectral bands of the images. The goal of this example is to show that we can apply the same stratification approaches to analyze imagery streams.

Our next example uses the hyperspectral image set discussed in Section 3.2. We first apply classical MDS to scaling the pixel vectors followed by a K-mean process to assign unique colors to eight scatter point clusters. We then stratify the vectors progressively and generate the MDS visualization after each reduction. Figure 9 shows the scatterplot matrix with results from different degrees of stratifications.

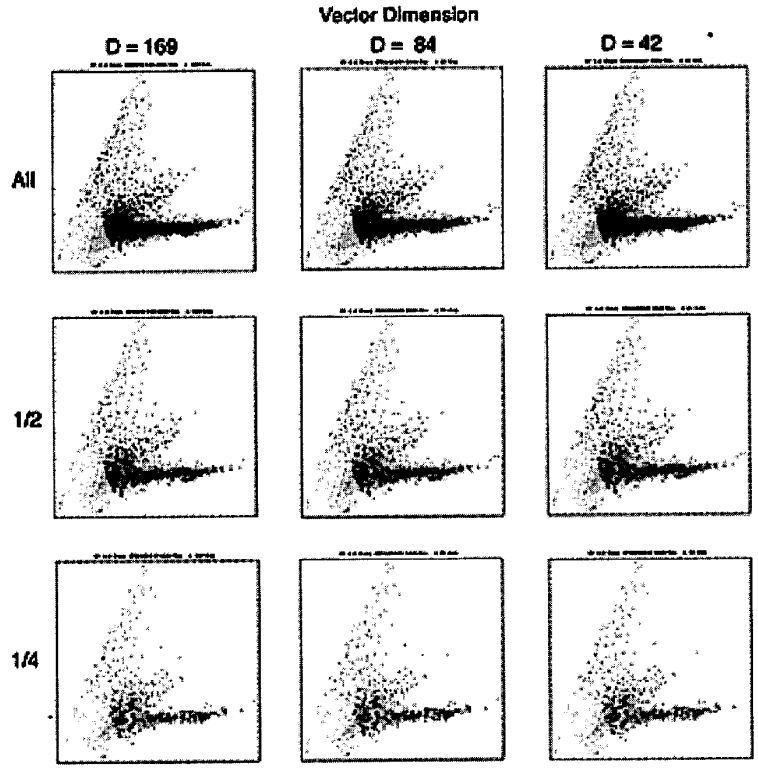


Figure 9. A scatterplot matrix demonstrates the effects of reducing pixel vectors (row) versus reducing vector dimensions (column) using remote sensing imagery. See also color plates.

In addition to the close proximity among the nine scatterplots as shown in previous examples, this time we can use a different approach to evaluate the accuracy of the results. If we map the colors of individual pixels from Figure 9 back to Figure 1c, we obtain the image shown in Figure 10. By comparing features in Figure 1c to those in Figure 10, we see that all nine scatterplots correctly identify different features of the original image and separate them into different clusters.

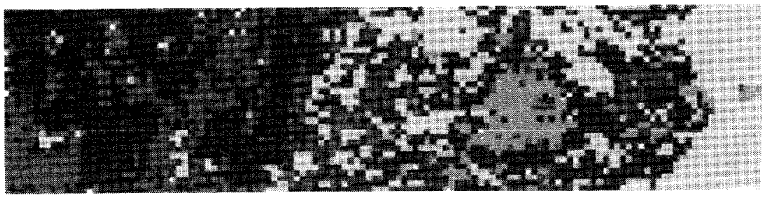


Figure 10. Colors generated by the scatterplot clusters clearly identify different features of the images shown in Figure 1c. See also color plates.

7. SCATTERPLOT SIMILARITY MATCHING

So far our discussion on scatterplot comparison has been based on visual means. Visual-based comparisons are fast and convincing. However, they need human attention and interpretation and thus are not practical in some applications. Besides, visual-based pairwise comparison is not always reliable if the scatterplots do not reflect any visible patterns. For example, Figure 11 shows two scatterplots filled with white noise. As we will reveal later, these two scatterplots are very similar. But our eyes are fooled by the lack of visible patterns that are required to correlate the images.

We want to find a reliable computation method to evaluate the similarity between two scatterplots so that we have a metric to measure the fidelity of our approximations generated by stratification and later our fusion technique. In statistics studies, the class of techniques of matching two similar n-D configurations and producing a measure of the match is commonly known as *Procrustes* analysis [Cox & Cox, 1994].

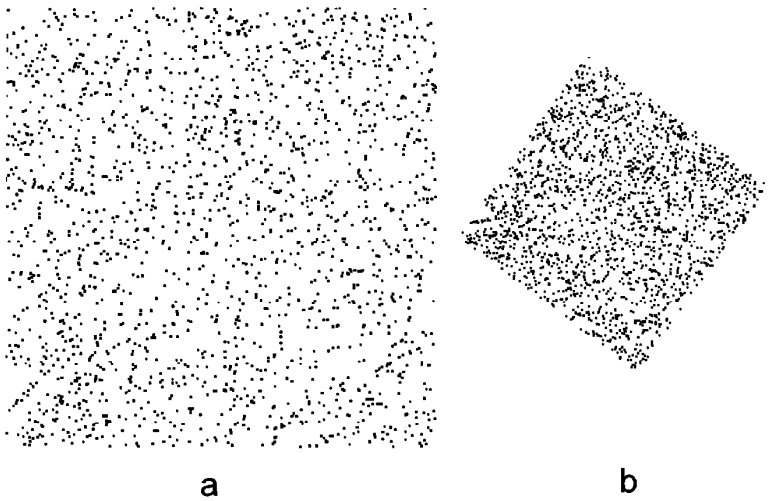


Figure 11. Two scatterplots filled with white noise

7.1 Procrustes Analysis

We implemented a Procrustes program that can match scatterplots in any number of dimensions. We also assume that the one-to-one correspondence information among the scatter points is known. While the theory behind the analysis is beyond the scope of this chapter, we can comfortably summarize our implementation in four basic steps. Given two 2-D scatterplots $\boldsymbol{X}$ and $\boldsymbol{Y}$ where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are $(n \times 2)$ matrices, the steps to match $\boldsymbol{X}$ to $\boldsymbol{Y}$ and report a measure of the match are:

1. *Translate* the two scatterplots so that they both have their centroids at the origin—by subtracting each point with its mean coordinates of the scatterplot.
2. *Rotate* $\boldsymbol{X}$ to match $\boldsymbol{Y}$ by multiplying $\boldsymbol{X}$ with $(\boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{X})^{½}(\boldsymbol{Y}^T\boldsymbol{X})^{-1}$.
3. *Dilate* scatter points in $\boldsymbol{X}$ by multiplying each of them with $\mathrm{tr}(\boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{X})^{½}/\mathrm{tr}(\boldsymbol{X}^T\boldsymbol{X})^T$.
4. The matching index between $\boldsymbol{X}$ and $\boldsymbol{Y}$: $1 - \{\mathrm{tr}(\boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{X})^{½}\}^2/\{\mathrm{tr}(\boldsymbol{X}^T\boldsymbol{X})\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{Y})\}$.

The goal is to seek the isotropic dilation and the rigid translation, reflection, and rotation required to match one scatterplot with the other. The matching index calculated in Step 4 ranges from zero (best) to one (worst).

For example, we can match the scatterplot in Figure 11b to the one in Figure 11a by a rotation of -36 degree (Step 2) followed by a scaling of 2 (Step 3). The matching index (Step 4) of this Procrustes analysis is 2.21008×10^{-30}—which indicates the two scatterplots are nearly identical.

Bear in mind that we use Procrustes analysis to measure the similarity between two 2-D scatterplots, not the original high-dimensional datasets. In other words, we merely use Procrustes analysis as a means to remove much of the human subjectivity when we peruse the scatterplot patterns.

7.2 Procrustes Analysis Results

Table 2 shows the results of Procrustes analyses that were carried out on the corpus scatterplots in Figure 7. The very low index values (from 0.016 to 0.14) in Table 2 indicate that all eight scatterplots generated by stratified vectors are extremely similar to the full resolution one using all 3,268 vectors with all 200 dimensions. These highly accurate results and the notable 97.5% time reduction in generating one of them (reported in Table 1) are strong evidence that the two demonstrated stratification approaches are viable solutions in visualizing transient data streams.

Table 2. Matching indices between the eight document corpus scatterplots and the original full resolution one shown in Figure 8

	200	100	50
All (3268)	0.0 (SELF)	0.0224058	0.0841326
1/2 (1649)	0.0162034	0.0513420	0.1114290
1/4 (824)	0.0329420	0.0620215	0.1417580

To further support our argument, we provide the matching results of the remote sensing imagery scatterplots shown in Figure 9 in Table 3. The matching indices listed in Table 3 are even lower than those listed in Table 2. Even the worst case (1/4 dimension, 1/4 vectors) accomplishes an identical matching index up to four significant figures.

Table 3. Matching indices between the eight remote sensing imagery scatterplots and the full resolution one shown in Figure 10

	169	84	42
All (4096)	0.0 (SELF)	0.000004106	0.0000565361
1/2 (2048)	0.000000279	0.000004136	0.0000567618
1/4 (1024)	0.000004299	0.000007314	0.0000577721

8. INCREMENTAL VISUALIZATION USING FUSION

Our first visualization technique focuses primarily on the use of stratified vectors, which substitute the full-resolution ones to generate fast and accurate MDS scatterplots. The stratification effort alone, however, does not eliminate the requirement to re-process the entire dataset whenever new items arrive. Our next visualization technique, which is developed based on a data-fusion concept, addresses the re-processing issue by projecting new items directly onto an existing visualization without frequently re-processing the entire dataset.

8.1 Robust Eigenvectors

In Section 5.2, we observed that the visualization subspaces spanned by the two dominant Eigenvectors of our demo datasets are extremely resilient for changes. All the follow-on examples shown in Figures 6 to 9 indicate only minor distortions even after a substantial amount of the input data is removed.

To further explore this characteristic, we use the hyperspectral imagery (see Section 3.2 and Figure 1c) to study the similarity between the Eigenvectors (and the corresponding scatterplots) generated from local regions versus the entire dataset. To provide identities to individual pixels, we use the image representation shown in Figure 10 to represent the hyperspectral imagery instead of the one in Figure 1c throughout this discussion.

Our first step is to generate a MDS scatterplot (Figure 12a) using the pixel vectors from the entire hyperspectral imagery. We then select and crop three smaller regions and generate three MDS plots (Figures 12b to 12d) using only the pixel vectors from the corresponding regions. These local regions are selected because they contain diverse image features (i.e., purple, pink, and blue on left; yellow, green, and red on right; and a mixture of everything in the middle) as reflected by the pixel colors.

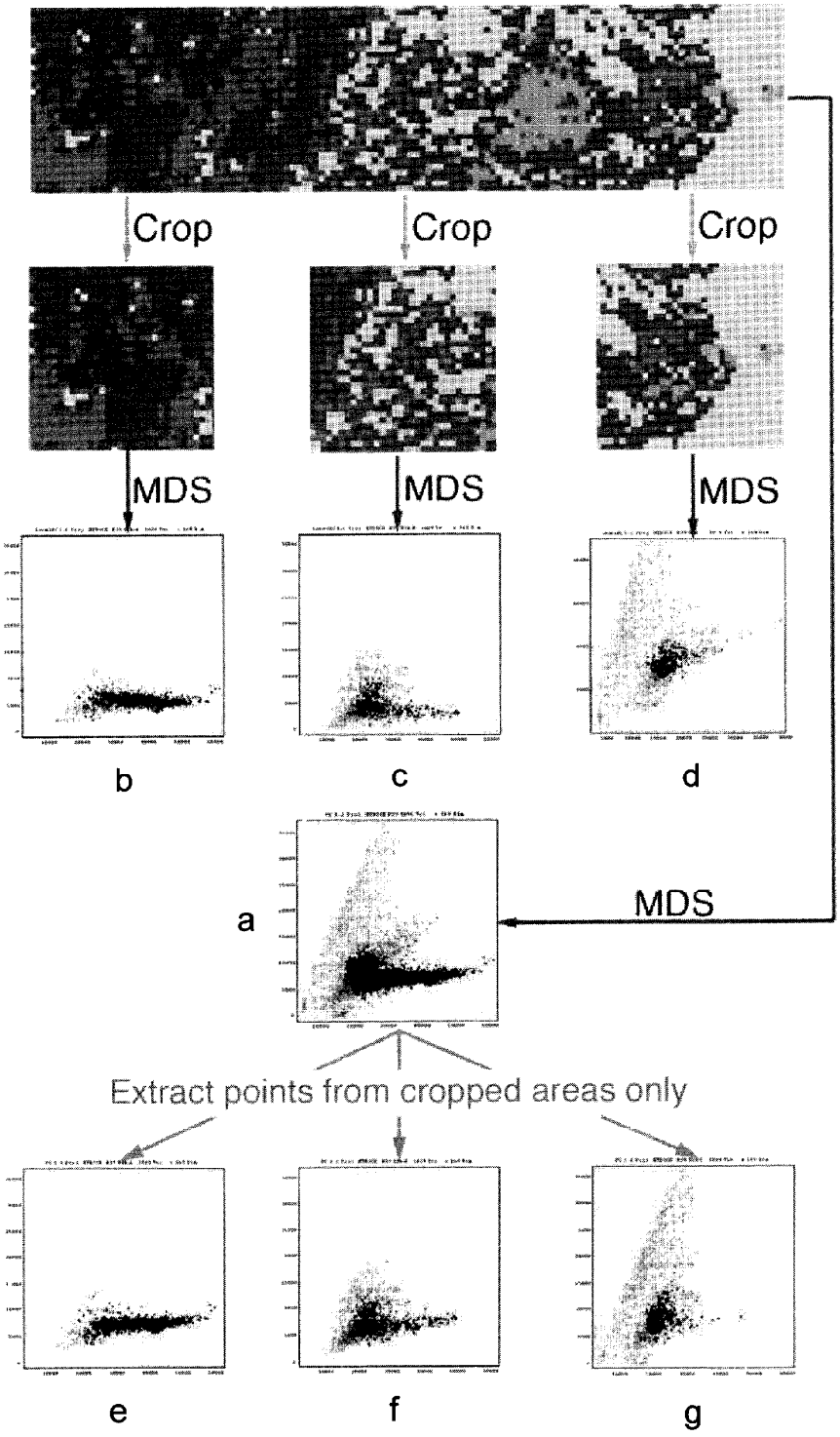


Figure 12. Scatterplot a) (top right) is generated from the demo imagery (top left). Scatterplots b) to d) (bottom left) are generated from the corresponding cropped areas. Scatterplots e) to g) are generated by extracting the scatter points from a) that are found in the corresponding cropping windows which generate scatterplots b) to d). See also color plates.

Our next step is to generate three more scatterplots (Figures 12e to 12g) using the corresponding pixel vectors found in Figures 12b to 12d. This time we use the Eigenvectors computed from the entire hyperspectral imagery (instead of the local cropped windows) to construct all three scatterplots. This can be done by reusing the coordinates of the selected scatterpoints from Figure 12a, which is constructed using Eigenvectors from the entire imagery. In other words, Figures 12b and 12e are generated using same pixel vectors, as are Figures 12c and 12f and Figures 12d and 12g. However, Figures 12b to 12d use local Eigenvectors of the cropped regions whereas the ones in Figures 12e to 12g use global Eigenvectors of the entire imagery.

The resultant scatterplots in Figures 12b to 12g show that the three corresponding pairs (i.e., Figures 12b and 12e, Figures 12c and 12f, and Figures 12d and 12g) closely resemble each other. This visual-based conclusion is consistent with the near zero Procrustes matching indices shown in Table 4, which imply a close similarity among the pairs.

Table 4. Procrustes matching indices of the three scatterplot pairs shown in Figure 12

Figures	12b vs. 12e	12c vs. 12f	12d vs. 12g
Matching Index	0.000718745	0.0000381942	0.000683066

Because the first Eigenvector is the line though the centroid of the scatter points along which the variance of the projections is greatest (not necessarily the direction of the greatest ranges or extent of the data) and the second Eigenvector is orthogonal to the first one, these Eigenvectors tend to be very robust for changes unless a substantial amount of disparate information is added. The property is particularly noteworthy because of the frequently high similarities among neighboring data streams. This remarkable combination becomes the foundation of our next visualization technique on data streams.

8.2 Multiple Sliding Windows

Figure 13 illustrates an overview of our incremental visualization technique using multiple time frames, which we call *sliding windows* in our discussion, in chronological order. After the Eigenvectors of a long (blue) window are determined and a 2-D scatterplot is generated, newly arrived individual vectors from the short (red) window are projected directly onto the visualization subspace by simply computing the dot-products between the incoming data vectors with the Eigenvectors of the long window. So instead of repeatedly processing the rather expensive ($O(n^3)$) classical MDS function or even the faster ($O(n\sqrt{n})$) version [Morrison, Ross & Chalmers, 2002] whenever new information arrives, one can now obtain an almost instant visualization update by carrying out a simple ($O(m)$, m = vector dimension,

$m << n$) dot-product operation to determine the point location of the new information in the scatterplot.

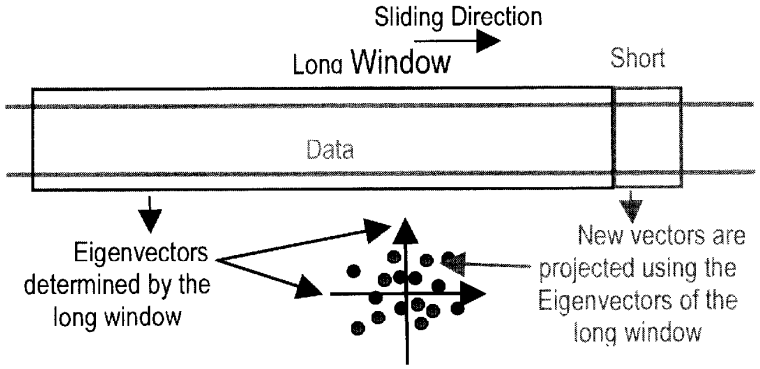


Figure 13. An illustration of our multiple sliding window design in visualizing data streams. See also color plates.

8.2.1 Hyperspectral Imagery

We use the same hyperspectral imagery to demonstrate the concept of our visualization technique. Figure 14a shows an ideal case when 100% of the pixel vectors are used to generate the scatterplot by MDS. In Figure 14b, only 75% of the pixel vectors are projected onto the scatterplot by MDS. The other 25% are projected by a dot-product function using the Eigenvectors of the first 75%. Finally, only 50% of the pixel vectors are projected by MDS. The other 50% are projected according to the Eigenvectors of the first 50%.

While the scatterplots in Figures 14b to 14c look similar to the full resolution one in Figure 14a, the low Procrustes indices in Table 5 confirm that the three scatterplots are indeed close to each other. These near-zero matching indices also validate our argument that we can obtain a fast and accurate overview of the entire dataset without the requirement of re-processing the entire dataset.

Table 5. Procrustes matching indices of the three scatterplots shown in Figure 14

Figures	14a vs. 14b	14a vs. 14c
Matching Index	000123405	0.00233882

8.2.2 Hydroclimate Dataset

We pre-process the hydroclimate matrix, which is defined in Section 3.3, similar to the other two datasets. MDS is first applied to the 6,155 hydroclimate vectors to form a scatterplot. A K-mean process is then used to subdivide the 10 most important components into 10 clusters. Random colors are applied to each cluster for identification throughout the demonstration as

depicted in Figure 15a. Finally, the colors of individual vectors are mapped to the corresponding co-ordinates of the western US map as shown in Figure 15b.

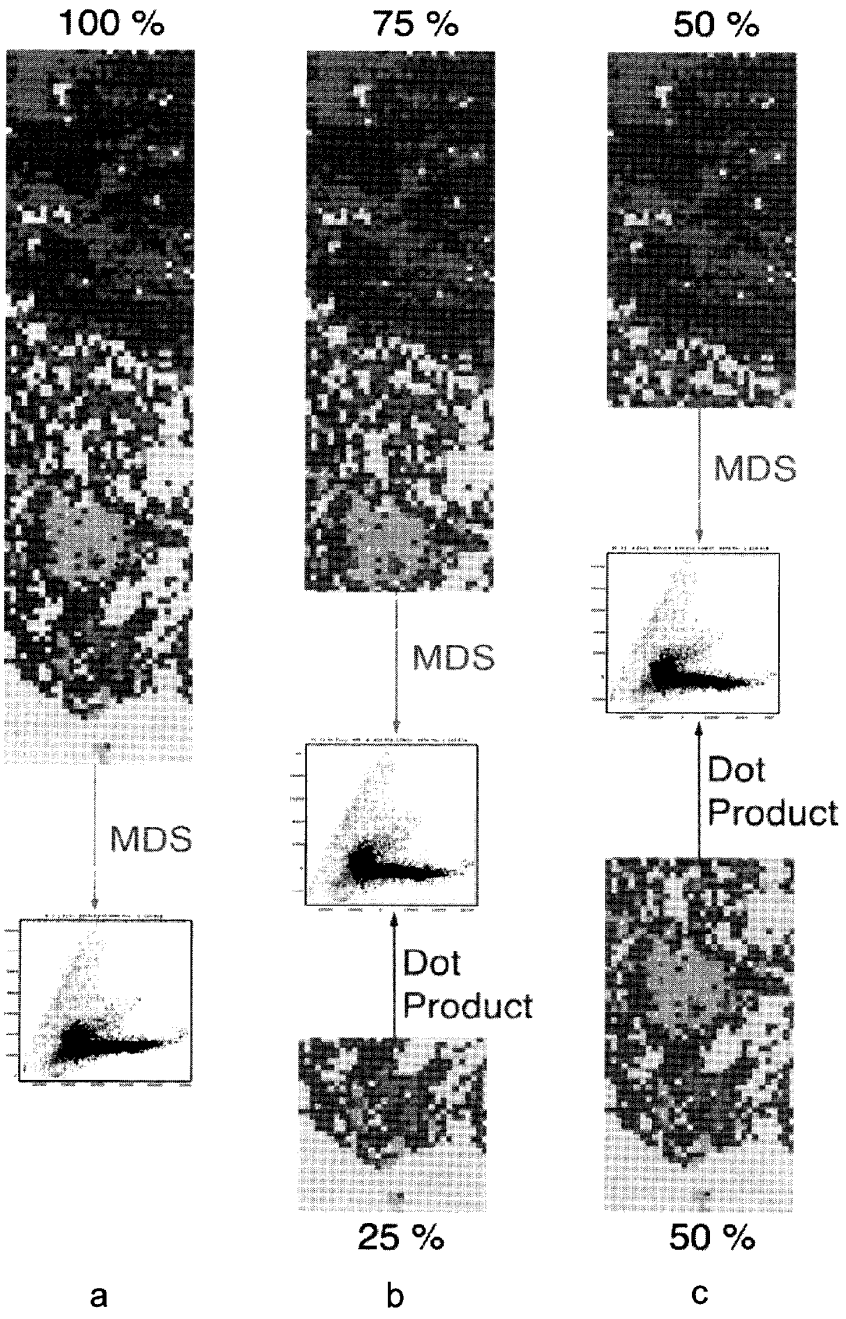


Figure 14. a) The Eigenvectors of the scatterplot are computed using 100% of the pixel vectors. b) The Eigenvectors of the scatterplot are computed from 75% of the pixel vectors. The other 25% are projected onto the scatterplot by dot-product. c) The Eigenvectors of the scatterplot are computed from 50% of the pixel vectors. The other 50% are projected by dot-product. See also color plates.

Similar to the previous CIR example in Figure 14, we start with a scatterplot generated using 100% of the vectors as shown in Figure 16a. We then use 75% of the vectors to determine their two major Eigenvectors. The rest of the 25% are inserted into the scatterplot using dot-products. The result is shown in Figure 16b. Finally, 50% of the vectors (on the left side) are used to determine the Eigenvectors of the scatterplot and the rest of the vectors (on the right side) are inserted into the scatterplot by dot-products.

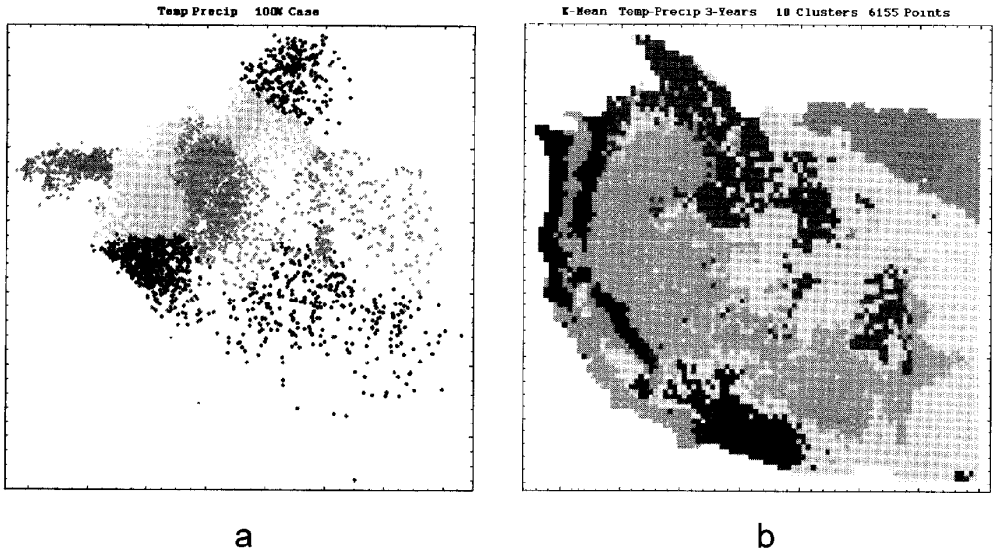


Figure 15. a) A scatterplot of 6,155 hydroclimate vectors divided into 10 clusters. Each cluster is represented by a unique random color. b) Corresponding cluster colors are projected to the map position. See also color plates.

Based on visual analysis, Figures 16a and 16b are almost identical. The orientation of the scatterplot in Figure 16c rotates slightly in counter-clockwise direction. Nevertheless, the shape and the integrity of the scatterplot remain intact and look very similar to Figure 16a. The near-zero Procrustes analysis indices shown in Table 6 prove that the three scatterplots are indeed extremely similar.

Table 6. Procrustes matching indices of the three scatterplots shown in Figure 14

Figures	16a vs. 16b	16a vs. 16c
Matching Index	0.00363075	0.0103674

9. COMBINED VISUALIZATION TECHNIQUE

Although the two visualization techniques presented in this chapter are intended to operate independently, we can combine them together and get the best of both worlds. The newly combined technique is capable of processing both individual items one at a time (by data fusion) and large amounts of items all together (by data stratification) efficiently and effectively.

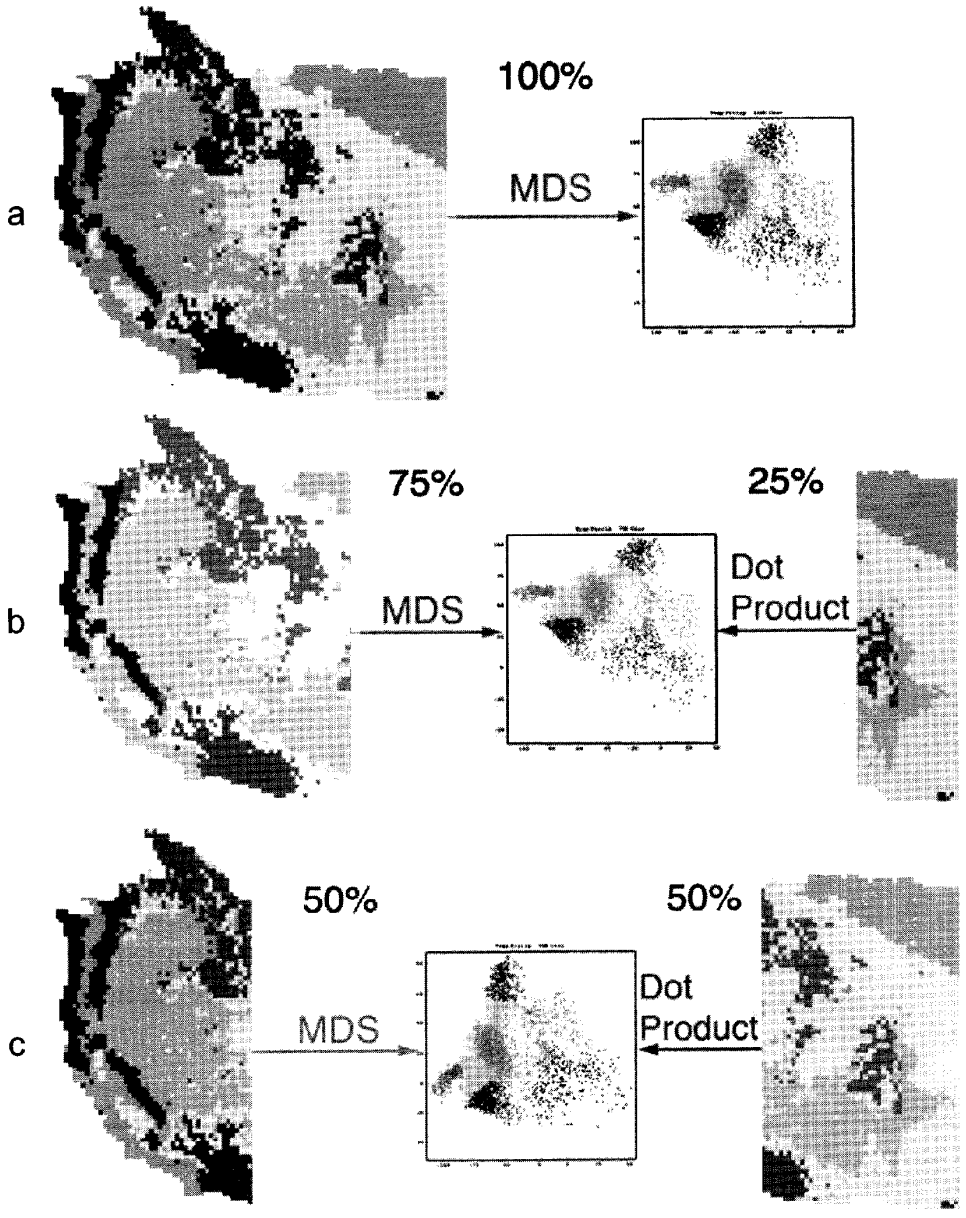


Figure 16. a) The Eigenvectors of the scatterplot are computed using 100% of the hydorclimate vectors. b) The Eigenvectors of the scatterplot are computed from 75% of the vectors. The other 25% are projected onto the scatterplot by dot-product. c) The Eigenvectors of the scatterplot are computed from 50% of the pixel vectors. The other 50% are projected by dot-product. See also color plates.

9.1 Error-Tracking Optimization

We have used Procrustes analysis as the primary computational method to evaluate the errors between a full-resolution standard scatterplot (e.g., Figure 14a) and ones based on multiple sliding windows approach (Figures 14b and 14c.) A noticeable enhancement to speed up the error-tracking process (and thus the visualization process) is to replace the full-resolution standard scatterplot with a fast and accurate substitute like the one using stratified vectors as presented in Section 5.1.

The results in Table 1 show that we can save up to 92% of computation time (from 34.9s to 2.62s) if we compress the dimensions of 3,268 vectors by 75%. And the results in Table 3 show that a 75%-reduced data matrix (dimension reduced from 169 to 42) can still be as accurate as the full-resolution one. Because of this faster error-checking process, we can now afford to carry out error estimation more frequently and thus improve the overall quality of the analysis.

Although the pre-processing steps of different data streams may vary, we can summarize the operations of the combined visualization technique in six major steps as follows:

1. When influx rate < processing rate, use MDS to reprocess the entire dataset when new information arrives.
2. When influx rate > processing rate, halt the MDS process.
3. Use the multiple sliding windows approach to update the existing scatterplot with the new information. Repeat Step 3 for a predefined number of updates.
4. Use the stratification approach to come up with a fast overview of the entire dataset.
5. Use the stratified overview to evaluate the accumulated error generated by the multiple sliding windows method using Procrustes analysis.
6. If error threshold is reached, go to Step 1. Otherwise, go to Step 3.

10. DISCUSSION AND FUTURE WORK

The general concept of scaling has been applied extensively to visualize the similarities of high-dimensional information for many years. In the case of corpus visualization, the combination of representing individual documents as mathematical vectors and projecting them onto a low-dimensional space using certain scaling techniques has becomes the *de facto* approach for many corpora visualization packages. Unfortunately, these packages are designed to analyze static corpora only and thus cannot handle dynamic text

streams proficiently. The work presented in this chapter provides critical technology to tackle such a clear and present problem.

Decades of MDS research and development by biologists, psychologists, statisticians, and now computational scientists have successfully reduced the time complexity of MDS from $O(n^3)$ at the beginning, to $O(n^2)$ [Chalmers, 1996] in 1996, and to $O(n\sqrt{n})$ [Morrison, Ross & Chalmers, 2002] in 2002. However, these significant results still require the reprocessing of at least a portion of the dataset for every update. When an individual item arrives, it is far better to use our data fusion method, which takes only $O(m)$ (m = vector dimension, $m << n$) time to obtain an instant scatterplot update.

While our dynamic visualization approach appears to work well on text, image, and hyperclimate streams, we have not included other data streams in our study. Until we conduct a full investigation, we cannot claim our approach is not without limitations.

We are in the process of integrating part of our work into an ongoing system development effort focusing on text stream visualization. We plan to evaluate the performance in a real-life environment using live text streams and present the results in the near future.

11. CONCLUSIONS

We have presented two dynamic visualization techniques to analyze transient data streams. Using a newswire corpus and a remote sensing imagery sequence, we demonstrate that our data stratification approach can substantially speed up the visualization process and yet maintain the high fidelity of the graphic results. We also show that our data fusion approach can provide instant updates of an MDS scatterplot without the requirement of reprocessing all the information in full resolution. All our approximation results have been validated by both visual and computational tests based on Procrustes analysis.

For information visualization applications similar to our examples, we believe our two analytic concepts will play an important role in many future data stream analysis tool designs. And for the other applications involving knowledge discovery and decision making, we believe our approach will work well with the established non-visual approaches and complement each other.

12. ACKNOWLEDGMENTS

This research has been sponsored by the US Intelligence Community and Advanced Research and Development Activity (ARDA), with additional support from the Homeland Security Initiative at the Pacific Northwest National Laboratory.

Special thanks go to George Chin, Kris Cook, Ted Divine, Sharon Eaton, Mark Goodwin, Beth Hetzler, Doug Lemon, Russ Rose, Bob Silva, and Alan Turner who provided assistance of many forms throughout this research.

The Pacific Northwest National Laboratory is managed for the U.S. Department of Energy by Battelle Memorial Institute under Contract DE-AC06-76RL1830.

13. EXERCISES AND PROBLEMS

1. The term Multidimensional Scaling (MDS) covers a variety of techniques to generate a low-dimensional plot from a high-dimensional dataset. This chapter shows only two of them that perform well in our applications. Find yet another MDS technique and see if it works as effectively as the two described in Section 6.

2. The Procrustes analysis algorithm described in Section 7.1 can handle two datasets (i.e., pairwise comparison) at a time. Redesign the algorithm so that it can compare more than two scatterplots simultaneously.

3. Design a new visualization technique that combines both visual and non-visual matching results described in Section 7.

14. REFERENCES

Aureka (2003). Retrieved Dec 2003 from: http://www.aurigin.com/aureka.html.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. Models and Issues in Data Stream Systems. *Proceedings of the 2002 ACM Symposium on Principles of Database Systems (PODS 2002)*, ACM Press, 1-16, 2002.

Babu, S. and Widom, J. Continuous Queries over Data Streams. *SIGMOD Record,* ACM Press, 109-120, 2001.

Bentley, C. L., Ward, M. O. Animating Multidimensional Scaling to Visualize N-Dimensional Data Sets. *Proceedings IEEE Symposium on Information Visualization '96*, IEEE CS Press, 72-73, 1996.

Bookstein, A., Klein, S. T., and Raita, T. Clumping Properties or Content-Bearing Words. *Journal of the American Society for Information Science and Technology*, 49, 2, 102-114, 1998.

Chalmers, M. A Linear Iteration Time Layout Algorithm for Visualising High-Dimensional Data. *Proceedings IEEE Visualization '96*, ACM Press, 127-132, 1996.

Cortes, C., Fisher, K., Pregibon, D., Rogers, A., and Smith, F. Hancock: A Language for Extrating Signatures from Data Streams. *KDD 2000 Proceedings*, ACM Press, 9-17, 2000.

Cox, T. F. and Cox, M. A. A. *Multidimensional Scaling*. Chapman & Hall, 1994.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science and Technology*, 41, 6, 391-407, 1990.

Domingos, P. and Hulten, G. Mining High-Speed Data Streams. *KDD 2000 Proceedings*, ACM Press, 71-80, 2000.

Gilbert, A. C., Kotidis, Y., Muthukrishnan, S., and Strauss, M. T. Surfing Wavelets on Streams: One-Pass Summaries for Approximation Aggregate Queries. *Proceedings of 27th Very Large Data Bases (VLDB) Conference*, 541-554, 2001.

Leung, L. R., Qian, Y., and Bian, X. Hydroclimate of the Western United States Based on Observations and Regional Climate Simulation of 1981-2000. Part I: Seasonal Statistics. *Journal of Climate*, 16, 12, 1892-1911, 2003.

Leung, L. R., Qian, Y., Bian, X., and Hunt, A. Hydroclimate of the Western United States Based on Observations and Regional Climate Simulation of 1911-2000. Part II: Mesoscale ENSO Anomalies. *Journal of Climate*, 16, 12, 1912-1928, 2003.

Mallat, S. *A Wavelet Tour of Signal Processing*. Academic Press, 1998.

Maurer, E.P., G.M. O'Donnell, D.P. Lettenmaier, and J.O. Roads. Evaluation of the land surface water budget in NCEP/NCAR and NCEP/DOE Reanalyses using an off-line hydrologic model. *J. Geophys. Res.*, 106, D16, 17,841-17,862, 2001.

Mathematica (2003). Retrieved Dec 2003 from: http://www.wolfram.com/.

MDS (2003). Retrieved Dec 2003 from: http://www.ncl.ac.uk/mds/.

Morrison, A., Ross, G., and Chalmers, M. A Hybrid Layout Algorithm for Sub-Quadratic Multidimensional Scaling, *Proceedings IEEE Symposium on Information Visualization 2002*, IEEE CS Press, 152-158, 2002.

Netlib (2003). Retrieved Dec 2003 from: http://www.netlib.org/lapack/index.html.

OmniViz (2003). Retrieved Dec 2003 from: http://www.omniviz.com/.

O'Callaghan, L., Mishra, N., Meyerson, A., Guba, S., and Motwani, R. Streaming-Data Algorithms for High-Quality Clustering, *Proceedings of the 2002 International Conference on Data Engineering (ICDE 2002)*, IEEE CS Press, 685-696, 2002.

Rodden, K., Basalaj, W., Sinclair, D., and Wood, K. Evaluating a Visualisation of Image Similarity as a Tool for Image Browsing, *Proceedings IEEE Symposium on Information Visualization 99*, 36-43, 1999.

Risch, J. S., Rex, D. B., Dowson, S. T., Walters, T. B., May, R. A., and Moon, B. D. The STARLIGHT Information Visualization System. *Proceedings IEEE Conference on Information Visualization*, 42-49, IEEE CS Press, 42-49, 1997.

Salton, G. and McGill, M. J. *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.

Sammon, J. W. A Non-Linear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, C-18, 5, IEEE CS Press, 401-409, 1969.

Seber, G. A. F. *Multivariate Observations*. John Wiley & Sons, 1984.

Strang, G. and Nguyen, T. *Wavelets and Filter Banks*, Wellesley-Cambridge Press, 1997.

VxInsight(2003). Retrieved Dec 03 from: http://www.cs.sandia.gov/projects/VxInsight.html.

Wise, J. A. The Ecological Approach to Text Visualization. *Journal of the American Society for Information Science and Technology*, 50, 9, 814-835, 1999.

Wise, J. A., Thomas, J. J., Pennock, K., Lantrip, d., Pottier, M., Schur, A., and Crow, V. Visualizing the Non-Visual: Spatial Analysis and Interaction with Information from Text Documents. *Proceedings IEEE Symposium on Information Visualization '95*, 51-58, 1995.

Wong, P. C. and Bergeron, R. D. Multivariate Visualization Using Metric Scaling. *Proceedings IEEE Visualization '97*, IEEE CS Press, 111-118, 1997.

Wong, P. C., Foote, H., Adams, D., Cowley, W., and Thomas, J. Dynamic Visualization of Transient Data Streams. *Proceedings IEEE Symposium on Information Visualization 2003*, IEEE CS Press, 97-104, 2003.

Wong, P. C., Foote, H., Leung, L. R., Adams, D., and Thomas, J. Data Signatures and Visualization of Very Large Datasets. *IEEE Computer Graphics and Applications*, 20, 2, IEEE CS Press, 12-15, 2000.

Chapter 12

SPIN! — AN ENTERPRISE ARCHITECTURE FOR DATA MINING AND VISUAL ANALYSIS OF SPATIAL DATA

Michael May and Alexandr Savinov
Fraunhofer Institute for Autonomous Intelligent Systems, Germany

Abstract: The rapidly expanding market for Spatial Data Mining systems and technologies is driven by pressure from the public sector, environmental agencies and industry to provide innovative solutions to a wide range of different problems. The main objective of the described spatial data mining platform is to provide an open, highly extensible, n-tier system architecture based on the Java 2 Platform, Enterprise Edition (J2EE). The data mining functionality is distributed among (i) Java client application for visualization and workspace management, (ii) application server with Enterprise Java Bean (EJB) container for running data mining algorithms and workspace management, and (iii) spatial database for storing data and spatial query execution. In the SPIN! system visual problem solving involves displaying data mining results, using visual data analysis tools, and finally producing a solution based on linked interactive displays with different visualizations of various types of knowledge and data.

Key words: Spatial data mining, Interactive visual analysis, Enterprise architecture

1. INTRODUCTION

Data mining is the partially automated search for hidden patterns in typically large and multi-dimensional databases. It draws on results in machine learning, statistics and database theory [Klösgen & Zytkow, 2002]. Data mining methods have been packaged in data mining platforms, which are software environments providing support for the application of one or more data mining algorithms. So far, data mining and Geographic

Information Systems (GIS) have existed as two separate technologies, each with its own methods, traditions and approaches to visualization and data analysis. Recently, the task of integrating these two technologies has become highly attractive [Ester, Frommelt, Kriegel & Sander, 2000; Koperski, Adhikary & Han, 1996; Koperski & Han, 1997] especially as various public and private sector organizations possessing huge databases with thematic and geographically referenced data began to realize the huge potential of information hidden there.

In response to this demand a prototype was developed [Andrienko, Andrienko, Savinov & Wettschereck, 2000; Andrienko, Andrienko, Savinov, Voss & Wettschereck, 2001] which demonstrated the potential of combining data mining and GIS. The results of this initial prototype encouraged the development of the SPIN! system [European IST SPIN! project web site; May, 2000; May & Savinov, 2002]. The overall objective of SPIN! system consists in developing a spatial data mining platform by integrating state-of-the-art Geographic Information System (GIS) and data mining functionality in a closely coupled, open, and extensible system architecture.

This chapter describes the SPIN! architecture and pays special attention to such features as scalability, security, multi-user access, robustness, platform independence and adherence to standards. It integrates Geographic Information System for interactive visual data exploration and Data Mining functionality specially adapted for spatial data. The system is built on the Java 2 Enterprise Edition (J2EE) architecture and particularly uses Enterprise Java Bean (EJB) technology for implementing remote object functionality. The flexibility and scalability of the J2EE platform has made it the platform of choice for building different multitiered enterprise applications so using it as a basis for a spatial data mining platform in SPIN! project is a natural extension.

EJB is a server-side component architecture, which cleanly separates the "business logic" (the analysis tools, in our case) from server issues, shielding the method developers from many technicalities involved in client-server programming. This choice also allows us to meet typical requirements found in business applications such as security, scalability, and platform independence in a principled manner.

The system is tightly integrated with a relational database and can serve as a data access and transformation tool for both spatial and non-spatial data. Analysis tools can be integrated either as stand-alone modules or coupled more tightly by distributing the analysis functionality between the database and the core algorithm. A spatial database is used to execute the complex spatial queries generated by the analysis algorithms. The final system integrates several data mining methods that have been adapted to the analysis of spatial data such as multi-relational subgroup discovery, rule

induction and spatial cluster analysis. The system then combines these methods with the rich interactive functionality of visual data exploration, thus offering an integrated, distributed environment for spatial data analysis.

2. THE SYSTEM OVERVIEW

The SPIN! spatial data mining system has a component architecture. This means that it provides the infrastructure and environment while all the system functionality is provided by separate software modules called components. Components can be easily added allowing for the expansion of system capabilities.

Each component is developed as an independent module for solving a limited number of tasks. For example, there may be components for data access, analysis or visualization. In order to solve complex problems, components need to communicate with and utilize the capabilities of each other. For example, when an algorithm needs data to be loaded, it asks another component to do this task.

To support interactions among components we developed a Common Connectivity Framework called CoCon — a generic library written in Java consisting of a number of interfaces and classes. The idea is that components can be connected by means of different types of connections. Currently, there are three connections available: visual, hierarchical and user-defined. Visual connections are used to link a component with its view (similar to the Model-View-Controller architecture). Hierarchical connections are used to compose parent-child relationships among components in the workspace. An example of this connection is the linking of a workspace folder with its elements. User-defined connections are the principal type used in the system. These connections allow for the arbitrary linking of components in the workspace as required by the task to be solved. Using such a connection, we could visualize a data mining result on a map by connecting the two corresponding components.

All components are implemented on the basis of the CoCon common connectivity framework what allows them to communicate within one workspace. It is important that components explicitly declare connectivity capabilities. This includes how to connect to a given component and with what other components a given component can work. These capabilities can be described either statically or dynamically. A static description consists of listing the necessary descriptors such as the ability of a component to accept an incoming connection from another component of some class. On the other hand, a dynamic determination can be made at run time by asking each

component if it can be connected. The static connectivity information is simple to use while the dynamic is important for the cases where components may change their behavior at run time. As an example for the need of such dynamic support, consider that an algorithm might not be interested in establishing or removing connections while it is processing data or when its requirements depend on the current parameters.

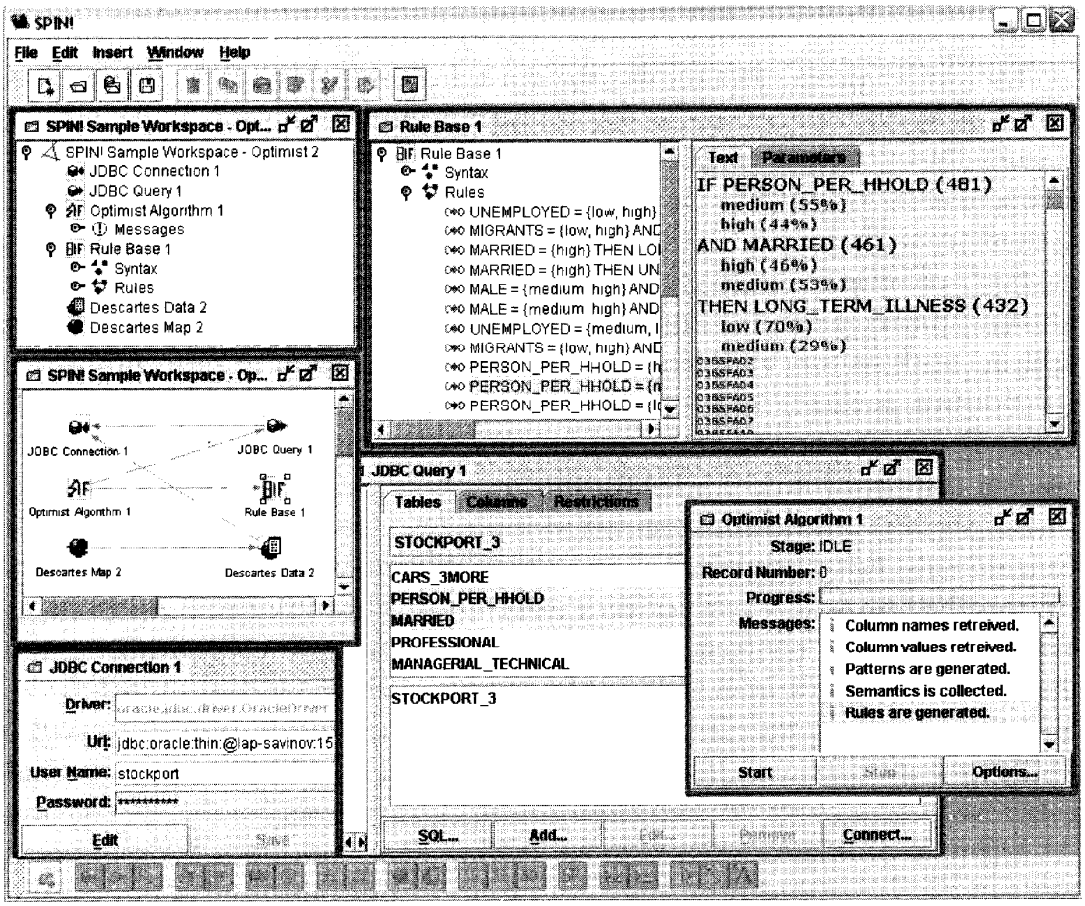


Figure 1. The SPIN! client provides views for its components: rule base (upper right window), database connection (lower left window), database query and algorithm (lower right windows). The workspace is visualized in the form of tree view and graph view.

The SPIN! spatial data mining system (Figure 1) provides an integrated environment based on component architecture. If the system is configured to include a certain set of components, then it will automatically update its main menu, tool bar and other functions. The repository of components is available via an Insert menu where components are broken in groups. Essentially the system core provides common facilities and services similar to those implemented in such platforms as Eclipse and NetBeans. It is also quite general extendable platform, which is intended for everything and for

nothing simultaneously because its functionality is determined by the current set of components.

A workspace is a set of components and corresponding connections. A workspace can be stored in or retrieved from a persistent storage (currently file or database). Once opened, a workspace appears in two views: a tree view and a graph view. In the tree view, the hierarchical structure of the workspace is visualized where components are individual tree nodes that can be expanded or collapsed. User-defined connections are not visible in the tree view but can be edited by means of the connection editor. In the graph view, components are visualized as nodes of a graph while user-defined connections are the graph edges.

Components can be added to the workspace by selecting them either from the menu or the tool bar where all repository components are shown. A new component appears both in the tree view (Figure 1, left top) and in the graph view (Figure 1, left middle). After the component has been added it should be connected with other relevant components. This can be done by means of the Connection Editor dialog where we choose the target component for the currently selected source component. An alternative easy and friendly way to establish new connections among components is through the workspace graph view where connections are explicitly displayed as arrows between nodes. In this view, connections are created by drawing arrows between the corresponding graph nodes designating the source and the target components. It is also very important that components can be arranged within views into visually expressive diagrams. While adding connections between components, the environment uses information about their connectivity so that connections are allowed only for components that can cooperate. That is, the link sticks to the target if the connection is possible. It is possible to move components in the graph view; their positions are remembered in the workspace.

Each component has an appropriate view. In fact, the views are also connectable components. However, such view connections are not persistent. Each component can be opened in a separate window so that the user can access its functions. When a workspace component is opened, the system automatically creates a view, connects it with the model and then displays it in a separate internal window within the main system frame.

Typical data mining tasks include data preprocessing, analysis and visualization. The SPIN! system includes Database Connection and Database Query components for accessing data. The Database Connection is intended for storing information about the database in which the data is stored. To use the identified database, the Database Connection component needs to be connected with some other component via, say, the graph view. The

Database Query component describes a typical database query, for example, how a result set is generated from the tables in the database. Essentially, this component is a SQL query design tool, which allows a user to describe a result set by choosing appropriate items such as tables, columns, restrictions, and functions. Since the system is intended to be used for spatial data mining this component includes spatial functions and predicates that are specific for an Oracle database. An example would be a RELATE predicate, which is true only if certain type of relationship between geometric objects exists. The Database Query component does not include information about the database and thus must be linked to the Database Connection component. Alternately, the query can be described manually in SQL. Notice also that neither the Database Connection nor the Database Query component work alone, other components must make use of them. Such encapsulation of functionality and use of user-defined connections to compose various aggregates has been one of the main design goals of the SPIN! component architecture.

Any knowledge discovery task includes a data analysis step where the dataset obtained from preprocessing step is processed by an appropriate data mining algorithm. The SPIN! system currently includes several components that implement different approaches to mining both ordinary and spatial data: spatial clustering, subgroup discovery [Klösgen & May, 2002; Klösgen & Zytkow, 2002], spatial association rules [Lisi & Malerba, 2002], Bayesian analysis, and rule induction based on finding largest empty intervals in data [Savinov, 2000a; Savinov, 2003]. Subgroup discovery and rule induction will be described in Section 4 of this chapter.

3. THE SYSTEM ARCHITECTURE

3.1 N-tier EJB-based Architecture

The general SPIN! architecture is shown in Figure 2. It is an *n*-tier Client/Server-architecture based on *Enterprise Java Beans* for the server side components. A major advantage of using Enterprise Java Beans is that such tasks as controlling and maintaining user access rights, handling multi-user access, pooling of database connections, caching, handling persistency and transaction management are delegated to the *EJB container*. The architecture has the following major subsystems: *client*, *application servers* each with one or more EJB containers, one or more *database servers* and optional *compute servers*.

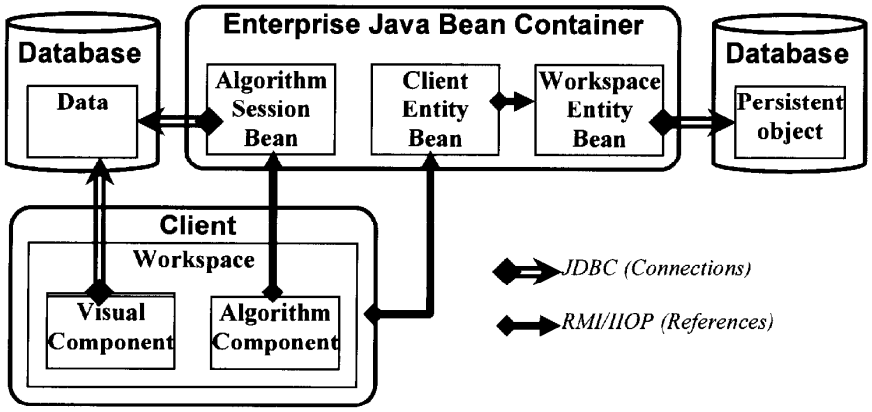


Figure 2. SPIN! platform architecture. The main components are a Java-based client, an Enterprise Java Beans Container and one or more databases serving spatial and non-spatial data.

The SPIN! *client* is a standalone Java application. It always creates one server side representative in the form of session bean. The methods of the session bean are accessed through the corresponding remote reference via Java RMI or CORBA IIOP protocol. The client session bean executes various server side tasks on behalf of the client. In particular, workspace objects may be loaded from or saved to its persistent state. The client is based on component connectivity framework, which is implemented in Java as connectivity library (CoCon). The idea is that the workspace consists of components each of which is considered a storage for a set of parameters and pieces of functionality such as algorithms. The system functionality is determined by a set of available components.

The *application server* is an Enterprise Java Bean container. It manages the client workspace, analysis tasks, data access and persistency. There may be more multiple containers running simultaneously on one or more servers. Among other things, this means that different algorithms and alternate tasks can be executed on different computers under different restrictions. The SPIN! system uses an EJB container for making workspaces persistent in the database and for remote computations. For the first task the client creates a special session bean, which is responsible on the server side for workspace persistence and access. Specifically, if the client needs to load or save a workspace it delegates this task to this session bean. The client creates one remote object for each analysis task that is to be run so that data can be transferred directly from the database to the algorithm. After the analysis is finished, the result is transferred to the client for visualization.

User data are stored in primary *data storage*, which is a relational database system accessed via JDBC protocol, which is a part of J2EE standard. The database can reside on the same machine as the application

server, it can reside on the client machine, or it can reside on a separate dedicated computer. Optionally, there may be one or more *secondary databases*. In addition, data can be loaded from other sources such as databases, ASCII files in the file system or Excel files. For remote computations in the application, it is important that server data be transferred directly into the remote algorithm bypassing the client. It is only a set of components (subgraph of the workspace) that is transferred between application server and client. In enterprise applications the amount of data processed may be quite large so it is very important to avoid unnecessary network traffic. In particular, if the data is going to be processed remotely in application server then it should be transferred directly from the storage to this computing server. This is precisely what is done in the SPIN! system where the workspace stores only data description. This data description is transferred to the application server, which has already loaded the data itself directly from the specified database for processing.

3.2 Remote Algorithm Management

The developed architecture supposes that all algorithms are executed on compute servers. For each running algorithm a separate session bean is created which implements high-level methods for controlling the algorithm behavior particularly starting and stopping the execution, getting and setting parameters, setting the data to process, and getting the result. The session bean then is responsible for the method's implementation. There are several ways how it can be done.

- A clean and convenient but in some cases inefficient approach is using Java for implementing the complete algorithm directly within the corresponding EJB along with loading all data via JDBC into the workspace.
- A second approach divides the labor between the EJB container and the relational database. We have implemented a multi-relational spatial subgroup-mining algorithm [Klösgen & May, 2002] that does most of the analysis work (especially the spatial analysis) directly in the database. The EJB part retrieves summary statistics, manages hypotheses and controls the search.
- A third approach consists in implementing computationally intensive methods in native code wrapped into shared library by means of Java Native Interface (JNI). A rule induction algorithm based on finding largest empty intervals in data [Savinov, 2000a; Savinov, 2003] has been implemented in this way, namely, as a dynamically linked library the functions of which are called by the EJB algorithm.

- A fourth option is that the algorithm session bean directly calls an external executable module. This approach has been used to run SPADA algorithm [Lisi & Malerba, 2002].
- And finally other remote objects (e.g. CORBA) can be used to execute the task.

The algorithm parameters are formed in the client and transferred to the EJB algorithm as a workspace component before the execution. In particular, data to be processed by the algorithm has to be specified. It is important to note that only a data *description* is specified and transferred not the complete data set. In other words, the EJB algorithm gets information concerning where and how to find the data and what kind of restrictions to use. Thus when the algorithm starts, the data is directly retrieved by the EJB algorithm rather than passing this information through the client.

For example, assume that we need to find interesting subgroups in spatially referenced data [Klösgen & May, 2002]. The data is characterized by both thematic attributes such as population and spatial attributes such as proximity to a highway or the percentage of forests in the area. The data to be analyzed is specified in the corresponding component where we can choose tables, columns, and join and restriction conditions including spatial operators supported by the underlying database system. The algorithm component is connected to the data component and the subgroup pattern component. The algorithm component creates a remote algorithm object in the EJB container as a session bean and transfers all necessary components to it such as the data description. The remote EJB object starts computations while its local counterpart periodically checks the remote object state until the process is finished. During computations the remote object retrieves data, analyzes it and stores the result in the result component. Note that each client may start several local and remote analysis algorithms simultaneously. When this happens, each algorithm is created as a separate thread. Once interesting subgroups have been discovered and stored in a component, they can be visualized in a special view, which provides a list of all subgroups with all parameters as well as a two-dimensional chart where each subgroup is represented by a point according to its coverage and strength. Additionally, the data analyzed by the subgroup discovery data mining algorithm can be viewed in a geographic information system and analyzed by visual analysis methods.

3.3 Workspace Management

One task, which is very important in distributed environment, is workspace management. During any given session, the user loads a

workspace from a central store into the client begins working. When the work is finished, the workspace is stored back in its initial location or perhaps a new one. A persistent workspace can be implemented in two alternate ways:

1. the whole workspace can be serialized and stored in one object like a local/remote file or a database record, or
2. the workspace components and connections can be stored separately in different database records.

The first approach is much simpler but it is difficult to share workspaces. The second approach allows us to treat workspace components as individual objects even within persistent storage; that is, the whole workspace graph structure is represented in the storage.

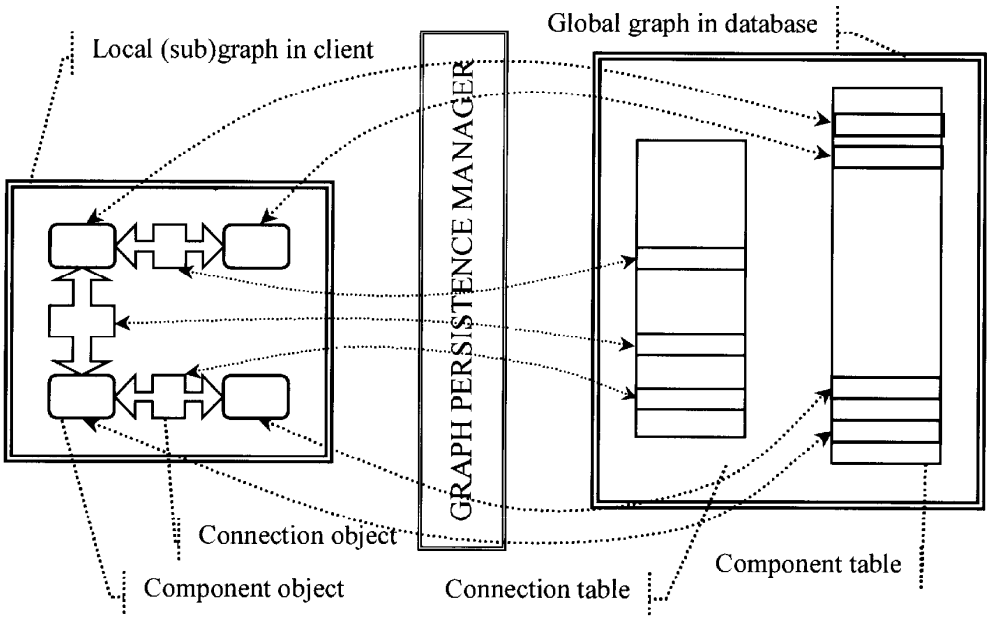


Figure 3. A workspace is a graph where nodes are components and edges are connections between them. All workspaces are stored in a database and retrieving a workspace means finding its component and connection objects. The persistent workspace management functionality is implemented as a session bean, which manipulates two types of entity beans: workspace components and workspace connections.

We have implemented both approaches and in both cases the workspace is represented as a special graph object: a set of its nodes (workspace components) and a set of its edges (workspace connections). These graphs can be created from existing run-time workspace objects by specifying constraints on its nodes and connections. For example, for loading and storing workspaces, view connections are ignored. The selected subgraph is passed to the persistence manager. If it needs to be stored as one object then the whole graph is serialized. Otherwise, individual node and edge objects

are serialized. We used XML for serialization, that is, any object state is represented as XML text.

The functionality of remote workspace management is implemented by a special session bean. This EJB has functions for loading and storing workspaces. If the workspace is stored as a set of its constituents then the session bean uses entity beans that correspond to the workspace components. The state of such workspaces is stored in two tables: one for nodes and one for edges. There exist two classes of entity beans, which are used to manipulate these two tables. The workspace management architecture for this case is shown in Figure 3.

4. ANALYSIS OF SPATIAL DATA

4.1 Mining Interesting Spatial Rules

The conventional approach to rule induction consists in finding anomalies by searching for intervals with surprisingly high values of probability distribution (the larger the interval the better) representing the data semantics. For instance, in association rule mining, patterns are generated in the form of itemsets and their interestingness is measured by support (the number of objects satisfying both condition and conclusion) and confidence (the number of objects satisfying the rule consequent among those satisfying the antecedent). For example, we might infer a rule where some item, such as high long-term illness, under certain conditions has 99% confidence. Implicitly this means that other (mutually exclusive) items such as medium and low long-term illness have a much lower probability very close to 0. Thus, the rule semantics can be reformulated as the incompatibility of some target values with items in the condition. Interesting rules then can be generated by finding item combinations that never occur in the data. The goal is still finding some kind of anomalous behavior but the main distinction from traditional approaches is that we are trying to find empty areas instead of high frequency areas in the data. A related approach to mining association rules based on this principle is described in [Liu, Ku & Hsu, 1997; Liu, Wang, Mun & Qi, 1998; Ku, Liu & Hsu, 1997; Edmonds, Gryz, Liang & Miller, 2001] where empty intervals among numeric attributes are called holes in data. The holes are found by using an algorithm [Orlowski, 1990; Chazelle, Drysdale & Lee, 1986] from computational geometry. In the SPIN! system, we apply an original rule induction algorithm, called Optimist [Savinov, 1999a; Savinov, 1999b; Savinov,

2000b], which works with finite value attributes and generates rules with one pass through the data set by using a method of sectioned vectors.

Let us assume that *attributes* $x_1, x_2, \ldots, x_n$ take a finite number of *values,* n_i from their domains $A_i = \{a_{i1}, a_{i2}, \ldots, a_{in_i}\}$. All combinations of values $\omega = \langle x_1, x_2, \ldots, x_n \rangle \in \Omega = A_1 \times A_2 \times \ldots \times A_n$ form the *state space* or *universe of discourse.* Each record from a data set corresponds to one combination of attribute values or a point. If, for a combination of values, a record in the data set exists then it is said to be possible. Otherwise, the point is impossible. To represent the data semantics as Boolean distribution over the universe of discourse we use the method of sectioned vectors and matrices [Savinov, 1999b; Savinov, 2000b]. The main idea of the method is that one vector can represent a multidimensional interval of possible or impossible points (called also positive and negative internal, respectively). Each vector consists of 0s and 1s that are grouped into sections separated by dots and corresponding to all attributes. A section consists of n_i components corresponding to all attribute values. For example, 01.010.0101 is a sectioned vector for three attributes taking 2, 3 and 4 values. Each component corresponds to one attribute value so that the number of components in the vector is equal to the total number of attribute values. A sectioned vector associates n components with each point from Ω (one from each section). The position of these components in the vector corresponds to the point coordinates. To represent negative intervals we use a disjunctive interpretation of sectioned vector. This means that the point is assigned 0 if all its components in the vector are 0s, and it is assigned 1 if at least one the components is 1. For example, the point $\langle a_{11}, a_{21}, a_{31} \rangle$ is impossible according to the semantics of vector $\underline{0}1.\underline{0}10.\underline{0}101$ because all three components corresponding to its coordinates in the vector $\underline{a_{11}}a_{12}.\underline{a_{21}}a_{22}a_{23}.\underline{a_{31}}a_{32}a_{33}$ are zeros. Yet the point $\langle a_{11}, a_{22}, a_{33} \rangle$ is possible because the component corresponding to a_{22} is 1 in the vector $\underline{0}1.0\underline{1}0.010\underline{1}$.

The main idea behind the algorithm for finding largest empty intervals consists in representing data semantics by a set of negative sectioned vectors and updating it for each record. Initially the data is represented only by the empty interval consisting of all 0s and making all points impossible. After the first record is added it is split into several smaller negative intervals so that the point corresponding to this record becomes possible. For example, addition of the record $0\underline{1}.00\underline{1}.000\underline{1}$ (where 1s correspond to its values) to the interval 00.010.0100 splits it into three new intervals: $0\underline{1}$.010.0100, 00.01$\underline{1}$.0100, and 00.010.010$\underline{1}$ (changed components are underlined). During this procedure very small intervals with a lot of 1s are removed since they generate very specific rules leaving only the top set of the largest intervals.

Once largest empty intervals have been found, they can be easily transformed into rules by negating sections that should be in the antecedent. For example, the vector $\{0,1\} \vee \{0,1,0\} \vee \{0,1,0,1\}$ can be transformed into the implication $\{1,0\} \wedge \{1,0,1\} \rightarrow \{0,1,0,1\}$ interpreted as the rule IF $x_1 = \{a_{11}\}$ AND $x_2 = \{a_{21}, a_{23}\}$ THEN $x_3 = \{a_{32}, a_{34}\}$. The rules are filled in by statistical information in the form of the target value frequencies within the rule condition interval (for one additional pass through the data set). In other words, each value in conclusion is assigned its frequency within the condition interval, for example, IF $x_1 = \{a_{11}\}$ AND $x_2 = \{a_{21}, a_{23}\}$ THEN $x_3 = \{a_{32} : 145, a_{34} : 178\}$, which is obviously more expressive. Here 145 means that the value a_{32} occurs 145 times within the selected interval.

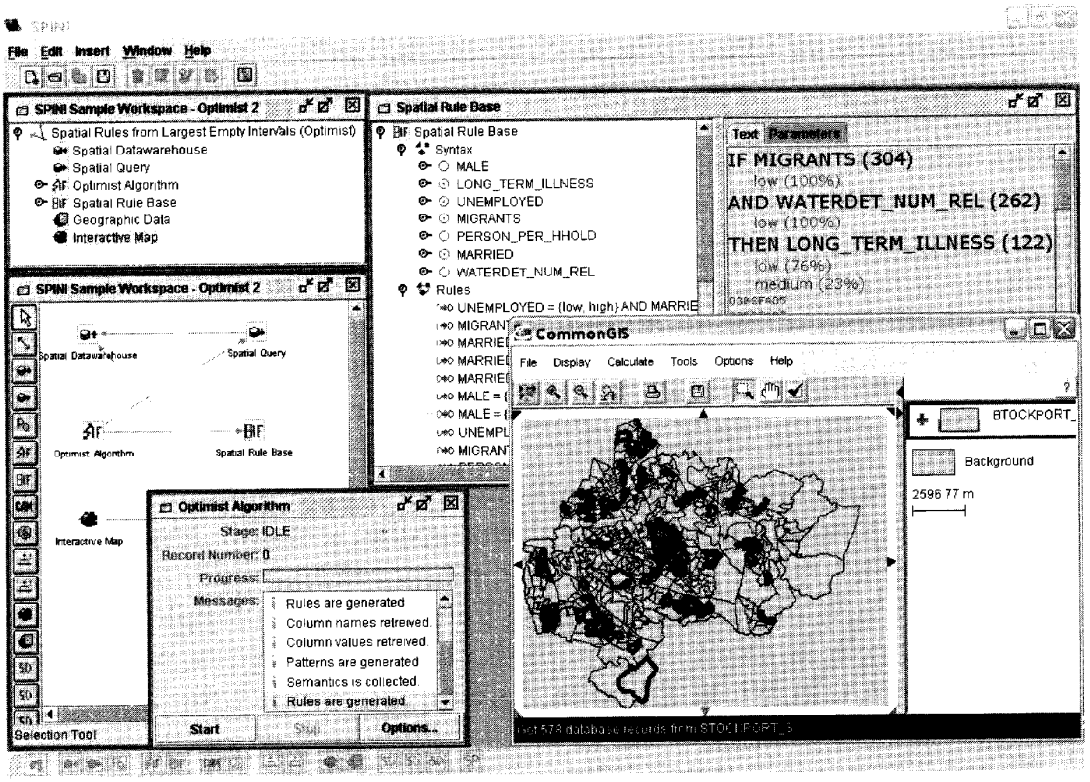


Figure 4. Visualization of spatial rules simultaneously and interactively with the map and other views in the SPIN! system. As one rule is selected in the upper right view all objects satisfying its condition are dynamically highlighted on the map in the lower right window.

The Optimist algorithm has been implemented as one of the SPIN! spatial data mining system components (Figure 4). It is tuned by a set of algorithm parameters such as maximal number of patterns (empty intervals) and executed either on the client or on the server. Input data for the

algorithm is specified by a standard SPIN! query component, which uses a separate connection component to access a database. The spatial rules generated by the algorithm are stored in rule base component. When appropriately connected, this minimal set of components implements the conventional knowledge discovery cycle. The analysis starts from specifying database and query, which can produce data in the necessary format. In our case, we need data columns to take only a finite number of values. Since most of the source data had continuous attributes we applied the SPIN! optimal discretization algorithm [Andrienko, Andrienko & Savinov, 2001]. Once columns had been discretized it was necessary to generate spatial attributes. For this purpose we used the spatial functionality of the Oracle 9i database where objects are represented by means of special built-in geometry type. Using such a representation, a query can combine spatial information with thematic data describing objects located in space. It is important that various spatial properties can be automatically generated by the database with the help of spatial predicates and relationships.

We used UK 1991 census data for Stockport, one of the ten districts in Greater Manchester, UK. The analysis was carried out at the level of enumeration districts (the lowest level of aggregation) characterized by such attributes as person per household, cars per household, migration, long-term illness, unemployment and other census statistics. Spatial information was available as coordinates and borders of objects such as enumeration districts, water, roads, streets, railways, and bus stops. For a typical analysis, we might be interested in finding dependencies among different thematic and spatial attributes, for example, what spatial and non-spatial factors influence long-term illness. As a spatial characteristic we define an attribute that counts the number of water resources belonging to each enumeration district calculated by means of SQL statement with spatial join. The final result set produced by SQL query is a normal table, which can be directly analyzed by the Optimist rule induction algorithm. The following example has been generated by such an analysis where MARRIED is the percentage of married people and WATER_NUM_REL is a characteristic of water resources in the enumeration district:

IF MARRIED (461) {high (46%) OR medium (53%)}
AND WATER_NUM_REL (447) {low (58%) OR medium (41%)}
THEN LONG_TERM_ILLNESS (358) {low (68%) OR medium (31%)}

The produced rules can be shown in their own window where the rules can be studied in detail. However, the SPIN! system provides a much more powerful method by using linked displays and interactive visualization [Andrienko et al., 2001]. The idea is that objects described in one view can be simultaneously visualized in other views. In our case, the rules describe enumeration districts and these very districts can be simultaneously shown

on a map. Moreover, if we select a certain rule, all objects that satisfy its left-hand side are dynamically highlighted on the map so that we can easily see how they are spatially distributed (Figure 4). For example, we might find that enumeration districts satisfying a certain rule and thus having interesting characteristics in terms of the target attribute form a cluster or have a more complex spatial configuration with respect to other geographic objects such as roads and cities.

4.2 Spatial Subgroup Mining

In this section, we focus on spatial patterns from the perspective of the subgroup-mining paradigm. Subgroup Mining [Klösgen, 1991, 1996, 2002] is used to analyze dependencies between a target variable and a large number of explanatory variables. Interesting subgroups are searched that show some type of deviation, for example, subgroups with an over proportionally high target share for a value of a discrete target variable, or a high mean for a continuous target variable. An advanced subgroup-mining algorithm has been implemented in the SPIN! system as one of its components. It supports multirelational hypotheses, efficient data base integration, discovery of causal subgroup structures, and visualization-based interaction options. The goal is to provide a spatial mining tool applicable in a wide range of circumstances.

Spatial subgroups are represented using an object-relational query language by embedding part of the search algorithm in a spatial database (SDBS). Thus the data mining and the visualization in a GIS share the same data. While this approach embraces the full complexity and richness of the spatial domain, most approaches to Spatial Data Mining export and pre-process the data from a SDBS. Our approach results in significant improvements for all stages of the knowledge discovery cycle:

1. Data Access: Subgroup Mining is partially embedded in a spatial database, where analysis is performed. No data transformation is necessary and the same data is used for analysis and mapping in a GIS. This is important for the applicability of the system since pre-processing of spatial data is error-prone and complex.
2. Pre-processing and analysis: SubgroupMiner handles both numeric and nominal target attributes. For numeric explanatory variables on-the-fly discretization is performed. Spatial and non-spatial joins are executed dynamically.
3. Post-processing and Interpretation: Similar subgroups are clustered according to degree of overlap of instances to identify multicollinearities. A Bayesian network between subgroups can be inferred to support causal analysis.

4. Visualization: SubgroupMiner is dynamically linked to a GIS, so that spatial subgroups are visualized on a map. This allows the user to bring in background knowledge into the exploratory process, to perform several forms of interactive sensitivity analysis and to explore the relation to further variables and spatial features.

Subgroups are subsets of analysis objects described by selection expressions of a query language such as simple conjunctional attributive selections, or multirelational selections joining several tables. *Spatial subgroups* are described by a spatial query language that includes operations on the spatial references of objects. For instance, a spatial subgroup may consist of the enumeration districts of Manchester that are intersected by a certain river. A spatial predicate (*intersects*) operates on the coordinates of the spatially referenced objects *enumeration districts* and *rivers*.

Subgroups are described via a hypothesis language. The *domain* is an object-relational database schema $S = \{R_1, ..., R_n\}$ where each R_i can have at most one geometry attribute G_i. Multirelational subgroups are represented by a concept set $C = \{C_i\}$, where each C_i consists of a set of conjunctive attribute-value-pairs $\{C_i.A_1=v_1,..., C_i.A_n=v_n\}$ from a relation in S, a set of links $L=\{L_i\}$ between two concepts C_j, C_k in C via their attributes A_m, A_k, where the link has the form $C_i.A_m\ \theta\ C_k.A_m$, and θ can be '=', a distance, or topological predicate (*disjoint, meet, equal, inside, contains, covered by, covers, overlap, or interacts*).

For example, the subgroup "districts with high rate of migration and unemployment crossed by the M60" can be represented as

C={{district.migration=high,district.unemplyoment=high},
{road.name= 'M60'}}

L= {{ spatially_interact(district.geometry, road.geometry)}}

Existential quantifiers of the links are problematic when many objects are linked, for example, many persons living in a city or many measurements of a person. The condition that one of these objects has a special value combination will often not result in a useful subgroup. In this case, conditions based on *aggregates* such as counts, shares or averages will be more useful [Knobbe, de Haas & Siebes, 2001; Krogel & Wrobel, 2001].

These aggregation conditions are included by aggregation operations (*avg*, *count*, *share*, *min*, *max*, *sum*) for an attribute of a selector. An average operation on a numerical attribute additionally needs labeled intervals to be specified.

C = (district.migration = high; building.count(id) = high)

L = (spatially_interact(district.geometry, building.geometry))

Extension: Districts with many buildings.

For *buildings.sum* the labels *low, normal, high* and their intervals are specified.

Multirelational subgroups were described in [Wrobel, 1997] in an Inductive Logic Programming (ILP) setting. Our hypothesis language is more powerful due to numeric target variables, aggregations, and spatial links. Moreover, all combinations of numeric and nominal variables in the independent and dependent variables are permitted in the problem description (Table 1). Numeric independent variables are discretized on the fly. This increases applicability of subgroup mining.

Table 1. All combinations of numeric and nominal variables are permitted

	Numeric Target	*Nominal Target*
Numeric input variables	Yes	Yes
Nominal input variables	Yes	Yes
Mixed numeric/nominal	Yes	Yes

Our approach is based on an *object-relational* representation. The formulation of queries depends on non-atomic data-types for the geometry, spatial operators based on computational geometry, grouping and aggregation. None of these features is present in basic relational algebra or Datalog. An interesting theoretical framework for the study of spatial databases are constraint databases [Kuper, Libkin & Paredaens, 2000], which can be formulated as (non-trivial) extensions of relational algebra or Datalog. However, using SQL is more direct and much more practical for our purposes. The price to pay is that SQL extended by object-relational features is less amenable to theoretical analysis (but see [Libkin, 2001]). For calculating spatial relationships, spatial extensions of DBMS-like *Oracle Spatial* can be used.

For database integration, it is necessary to express a multirelational subgroup as defined above as a query of a database system. The result of the query is a table representing the extension of the subgroup description. One part of this query defines the subset of the product space according to the l concepts and l-1 link conditions. The *from* part includes the l (not necessarily different) tables and the *where* part includes the l-1 link conditions (they are given as strings or default options in the link specification; spatial extensions of SQL apply a special syntax for the spatial operations). Additionally, the *where* part includes the conditions associated to the definition of selectors of concepts. Then the aggregation conditions are applied and finally the product space is projected to the target table (using the DISTINCT feature of SQL).

The complexity of the SQL statement is low for a single relational subgroup. Only the attributive selectors must be included in the *where* part of the query. For multirelational subgroups without aggregates and no distinction of multiple instances, the *from* part must manage possible

duplicate uses of tables, and the *where* part includes the link conditions (transformed from the link specification) and the attributive selectors. For aggregation queries, a nested two-level select statement is necessary, first constructing the multirelational attributive part and then generating the aggregations. Multiple instances of objects of one table are treated by including the table in the *from* part several times and the distinction predicate in the *where* part.

The space of subgroups to be explored within a search depends on the specification of a relation graph, which includes tables (object classes) and links. For spatial links the system can automatically identify geometry attributes by which spatial objects are linked, since there is at most one such attribute. A relation graph constrains the multi-relational hypothesis space in a similar way as attribute selection constrains it for single relations.

The search for interesting subgroups is arranged as an iterated *general to specific, generate and test procedure*. In each iteration, a number of parent subgroups is expanded in all possible ways, the resulting specialized subgroups are evaluated, and the subgroups are selected that are used as parent subgroups for the next iteration step, until a pre-specified iteration depth is achieved or no further significant subgroup can be found. There is a natural partial ordering of subgroup descriptions. According to the partial ordering, a specialization of a subgroup either includes a further selector to any of the concepts of the description or introduces an additional link to a further table.

The statistical significance of a subgroup is evaluated by a quality function. As a standard quality function, SubgroupMiner uses the classical binomial test to verify if the target share is significantly different in a subgroup:

$$\frac{p-p_0}{\sqrt{p_0(1-p_0)}}\sqrt{n}\sqrt{\frac{N}{N-n}}$$

This z-score quality function based on comparing the target group share in the subgroup (p) with the share in its complementary subset balances four criteria: size of subgroup (n), relative size of subgroup with respect to total population size (N), difference of the target shares (p-p_0), and the level of the target share in the total population (p_0). The quality function is symmetric with respect to the complementary subgroup. It is equivalent to the χ^2-test of dependence between subgroup S and target group T, and the correlation coefficient for the (binary) subgroup and target group variables. For continuous target variables and the deviating mean pattern, the quality function is similar, using mean and variance instead of share p and binary case variance $p_0(1\text{-}p_0)$.

To evaluate a subgroup description, a contingency table is statistically analyzed (Table 2). It is computed for the extension of the subgroup description in the target object class. To get these numbers, a multirelational query is forwarded to the database. Contingency tables must be calculated in an efficient way for the very many subgroups evaluated during a search task.

Table 2. Contingency table for target migration=high vs. unemployment=high

		Target		
		migration = high	¬migration = high	
Subgroup	unemployment=high,	16	19	35
	¬unemployment=high	47	496	543
		63	515	578

We use a two-layer implementation, where evaluation of contingency tables is done in SQL, while the search manager is implemented in Java. A sufficient statistics approach is applied by which a single query provides the aggregates that are sufficient to evaluate the whole bunch of successor subgroups. In the data server layer, within *one pass* over the database all contingency tables are calculated that are needed for a next search level. Thus not each single hypothesis queries the database, but a (next) population of hypotheses is treated concurrently to optimize data access and aggregation needed by these hypotheses. The search manager receives only aggregated data from the database so that network traffic is reduced. Besides offering scaling potential, such an approach includes the advantage of development ease, portability, and parallelization possibilities.

The central component of the query is the selection of the multirelational parent subgroup. This is why representation of multirelational spatial subgroup in SQL is required. To generate the aggregations (cross tables) for a parent subgroup, a nested select-expression is applied for multirelational parents. From the product table, first the expansion attribute(s), key-attribute for the primary table and target attribute are projected and aggregates calculated for the projection. Then the cross tables (target versus expansion attribute) are calculated. Efficient calculation of several cross tables, however, is difficult in SQL-implementations. An obvious solution could be based on building the union of several group-by operations (of target and expansion attributes). Although, in principle, several parallel aggregations could be calculated in one scan over the database, this is not optimised in SQL implementations. Indeed each union operation unnecessarily performs an own scan over the database. Therefore, to achieve a scalable implementation (at least for single relational and some subtypes of multirelational or spatial applications), the group-by operation has been replaced by explicit sum operations including case statements combining the

different value combinations. Thus for each parent, only one scan over the database (or one joined product table) is executed. Further optimisations are achieved by combining those parents that are in the same joined product space (to eliminate unnecessary duplicate joins).

As in our prior example, we applied this algorithm within the SPIN! system to UK 1991 census data for Stockport, one of the ten districts in Greater Manchester, UK. Census data provide aggregated information on demographic attributes such as person per household, cars per household, unemployment, migration, and long-term-illness. Their lowest level of aggregation is so called *enumeration districts*. Also available are detailed geographical layers, among them streets, rivers, buildings, railway lines, shopping areas. Data are provided to the project by the project partners Manchester University and Manchester Metropolitan University.

Assume we are interested in enumeration districts with a high migration rate. We want to find out how those enumeration districts are characterized, and especially what distinguishes them from other enumeration districts not having a high migration rate. Spatial subgroup discovery helps to answer this question by searching the hypothesis space for interesting deviation patterns with respect to the target attribute.

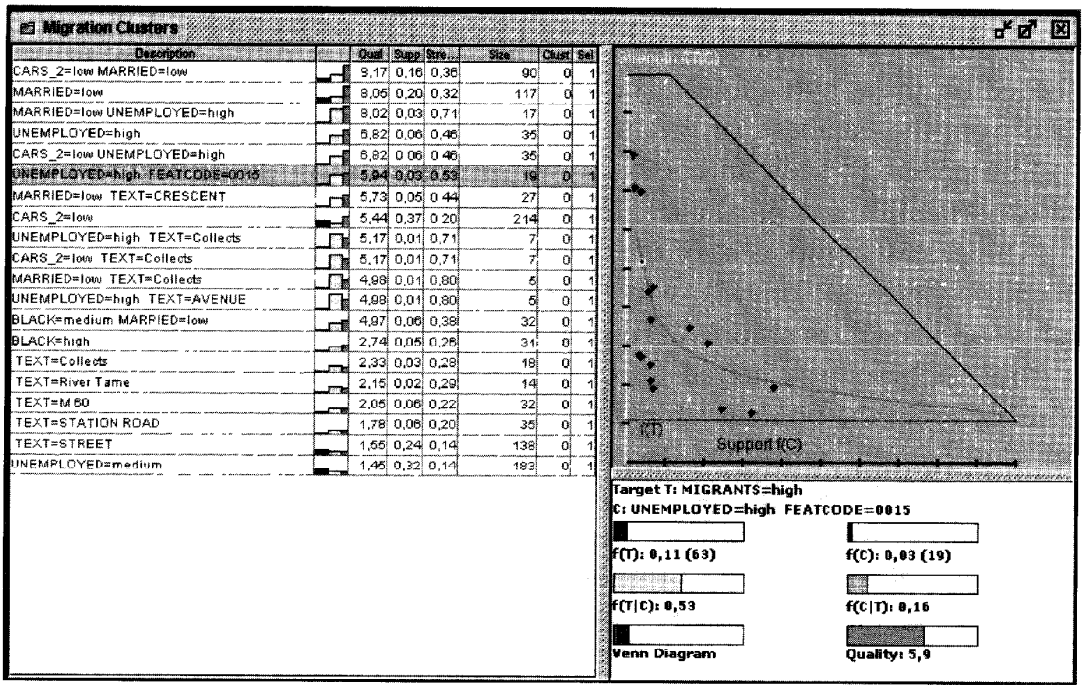


Figure 5. Overview on subgroups found showing the subgroup description (left). Bottom right side shows a detail view for the overlap of the concept C (e.g. located near a railway line) and the target attribute T (high unemployment rate). The window on the right top plots $p(T|C)$ against $p(C)$ for the subgroup selected on the left and shows *isolines* as theoretically discussed in [Klösgen, 1996].

The target attribute T is then *high migration rate.* A concept C found in the search is *Enumeration districts with high unemployment crossed by a railway line.* Note that this subgroup combines spatial and non-spatial features. The deviation pattern is that the proportion of districts satisfying the target T is higher in districts that satisfy pattern C than in the overall population ($p(T \mid C) > p(T)$). In other words if a district is characterized by high unemployment (census data) and at the same time is crossed by a railway line (geographic data) then the migration rate is expected to be higher than normally.

Another – this time purely spatial – subgroup found is *Enumeration district crossed by motorway M60.* This spatial subgroup induces a homogenous cluster taking the form of a physical spatial object. Spatial objects can often act as causal proxies for causally relevant attributes not part of the search space.

A third – this time non-spatial – subgroup found is *Enumeration districts with low rate of households with 2 cars and low rate of married people.* By spotting the subgroup on the map we note that is a spatially inhomogeneous group, but with its center of gravity directed towards the center of Stockport.

Figure 6. Enumeration districts satisfying the subgroup description C (high unemployment rate and crossed by a railway line) are highlighted with a thicker black line. Enumeration districts also satisfying the target (high migration rate) are displayed in a lighter color.

The way data mining results are presented to the user is essential for their appropriate interpretation. We use a combination of cartographic and non-cartographic displays linked together through simultaneous dynamic highlighting of the corresponding parts. The user navigates in the list of subgroups (Figure 5), which are dynamically highlighted in the map window (Figure 6). As a mapping tool, the SPIN!-platform integrates the CommonGIS system [Andrienko & Andrienko, 1999], whose strengths lies in the dynamic manipulation of spatial statistical data. Figure 6 shows an example for the migrant scenario, where the subgroup discovery method reports a relation between districts with high migration rate and high-unemployment.

Such an analysis, where results of data mining and interactive data analysis are visualized simultaneously, can be used to make non-trivial decisions in very different and diverse application areas including decision making that takes place in public and private sector organizations. In particular, this approach offers a great potential to improve decisions made by statistical analysts, urban planners, environmental decision makers, people in geomarketing, the management of natural and industrial hazards, nuclear safety and radiation protection and many other domains. Combining the strengths of GIS and Data Mining in a Spatial Mining tool helps the decision maker to back up her intuitive insights by sound statistics, and to automatically explore patterns in the data that are invisible to the eye because they live in high-dimensional spaces.

5. CONCLUSION

We have described the general architecture of the SPIN! spatial data mining platform. It integrates GIS and data mining algorithms that have been adapted to spatial data. The choice of J2EE technology allows us to meet requirements such as security, scalability, platform independence, in a principled manner. The system is tightly integrated with a RDBMS and can serve as data access and transformation tool for spatial and non-spatial data. The client has been implemented in Java using Swing components for its visual interface. Jboss 3.0 has been used as an application server [JBoss Application Server web site]. An Oracle 9i database has been used for spatial data and workspace storage. In the future it would be very interesting to add the following features to this architecture: persistent algorithms running with no client, a web interface to data mining algorithms via a conventional browser, data mining functionality as web services via XML-based SOAP protocol, and shared workspaces where components can belong to more than one workspace.

6. ACKNOWLEDGEMENTS

Work on this chapter has been partially funded by the European Commission under IST-1999-10536 SPIN! – Spatial Mining for Data of Public Interest. We would like to thank Jim Petch, Keith Cole and Mohammed Islam from Manchester University and Chrissie Gibson from Manchester Metropolitan University for making available the census data.

7. EXERCISES AND PROBLEMS

1. Assume a dataset includes a set of wards (voting districts) represented by polygons and a set of roads represented by lines. What numeric attributes could be generated from these geographic data characterizing each district? How can these attributes be generated by using the functions of an object-relational database?

2. Assume we have found a subgroup of wards with an above-average value for the attribute *low_social* (*low_social=high*), a below-average value for the percentage of married people (*married=low*) and an above-average value for unemployed men (*unempl_male=high*). This subgroup is characterized by an average value for the Carstairs deprivation index of 6.24 compared to the overall average of 0.94. What conclusions can be drawn from this subgroup by such authorities as departments of national and local government or providers of health and education services?

3. Under the assumptions of the previous exercise, suppose that we have additional geographic data such as roads, cities, rivers, railway lines, bus stops. These objects can be visualized on the map where the Carstairs deprivation index is represented by color for each ward. In addition, suppose that wards from the selected interesting subgroup are highlighted. What kind of interesting relationships could be visually discovered?

4. Suppose that by analyzing census data we found that such characteristics as high long-term illness (LONG_TERN_ILLNESS=High), high person per household (PERSON_PER_HHOLD=High) and low relative density of water resources (WATERDET_NUM_REL=Low) are incompatible, i.e., there exist no ward with such characteristics. What rule could be generated from this information?

8. REFERENCES

Andrienko G., Andrienko N. Interactive Maps for Visual Data Exploration. International Journal of Geographical Information Science 13(5), 355-374, 1999.

Andrienko G., Andrienko N., Savinov A. Choropleth Maps: Classification revisited. Proceedings ICA 2001, Beijing, China, Vol. 2, 1209-1219.

Andrienko N., Andrienko G., Savinov A., Voss H., Wettschereck D. Exploratory Analysis of Spatial Data Using Interactive Maps and Data Mining. Cartography and Geographic Information Science 28(3), July 2001, 151-165.

Andrienko N., Andrienko G., Savinov A., Wettschereck D. Descartes and Kepler for Spatial Data Mining. ERCIM News, No. 40, January 2000, 44–45.

Chazelle B., Drysdale R.L., Lee D.T. Computing the largest empty rectangle. SIAM J. Comput., 15:300-315, 1986.

Edmonds J., Gryz J., Liang D., Miller R.J. Mining for Empty Rectangles in Large Data Sets. Proceedings of the 8th International Conference on Database Theory (ICDT), London, UK, January 2001, 174-188.

Ester M., Frommelt A., Kriegel H.P., Sander J. Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support. Data Mining and Knowledge Discovery 4(2/3), 2000, 193-216.

European IST SPIN! project web site. http://www.ccg.leeds.ac.uk/spin/

JBoss Application Server web site. http://www.jboss.org.

Klösgen W. Explora: A Multipattern and Multistrategy Discovery Assistant. Advances in Knowledge Discovery and Data Mining, eds. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Cambridge, MA: MIT Press, 249–271, 1996.

Klösgen W. Subgroup Discovery. In Handbook of Data Mining and Knowledge Discovery (Chapter 16.3), Klösgen, W., Zytkow, J., eds. Oxford University Press, New York, 2002.

Klösgen W. Visualization and Adaptivity in the Statistics Interpreter EXPLORA. Proceedings of the 1991 Workshop on KDD, ed. Piatetsky-Shapiro, G., 25-34, 1991.

Klösgen W., May M. Spatial Subgroup Mining Integrated in an Object-Relational Spatial Database. PKDD 2002, Helsinki, Finland, August 2002, 275-286.

Klösgen W., Zytkow J. (eds.) Handbook of Data Mining and Knowledge Discovery. Oxford University Press, 2002.

Knobbe A.J., de Haas M., Siebes A. Propositionalisation and Aggregates. Proc. PKDD 2001, eds. De Raedt, L., Siebes, A., Berlin:Springer, 277-288, 2001.

Koperski K., Adhikary J., Han J. Spatial Data Mining, Progress and Challenges. Technical Report, Vancouver, Canada, 1996.

Koperski K., Han J. GeoMiner: A System Prototype for Spatial Mining. Proceedings ACM-SIGMOD, Arizona, 1997, 553-556.

Krogel M., Wrobel S. Transformation-Based Learning Using Multirelational Aggregation. Proc. ILP 2001, eds. Rouveirol, C., Sebag, M., Springer, 142-155, 2001.

Ku L.-P., Liu B., Hsu W. Discovering Large Empty Maximal Hyper-rectangles in Multi-dimensional Space. Technical Report, Department of Information Systems and Computer Science (DCOMP), National University of Singapore, 1997.

Kuper G.M., Libkin L., Paredaens J. (eds.) Constraint Databases. Berlin:Springer, 2000.

Libkin L. Expressive Power of SQL. Proc. of the 8th International Conference on Database Theory (ICDT01), eds. Bussche, J, Vianu, V., Berlin:Springer, 1-21, 2001.

Lisi F.A., Malerba D. SPADA: A Spatial Association Discovery System. In A. Zanasi, C.A. Brebbia, N.F.F. Ebecken and P. Melli (Eds.), *Data Mining III*, Series: Management Information Systems, Vol. 6, 157-166, WIT Press, 2002.

Liu B., Ku L.-P., Hsu W. Discovering Interesting Holes in Data. Proceedings of Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97), pp. 930-935, August 23-29, 1997, Nagoya, Japan.

Liu B., Wang K., Mun L.-F., Qi X.-Z. Using Decision Tree Induction for Discovering Holes in Data. Pacific Rim International Conference on Artificial Intelligence (PRICAI-98), 182-193, 1998.

May M. Spatial Knowledge Discovery: The SPIN! System. Fullerton, K. (ed.) Proceedings of the 6th EC-GIS Workshop, Lyon, 28-30th June, European Commission, JRC, Ispra, 2000.

May M., Savinov A. An integrated platform for spatial data mining and interactive visual analysis. Data Mining 2002, Third International Conference on Data Mining Methods and Databases for Engineering, Finance and Other Fields, 25-27 September 2002, Bologna, Italy, 51-60.

Orlowski M. A New Algorithm for the Largest Empty Rectangle Problem. Algorithmica, 5(1):65-73, 1990.

Savinov A. Application of multi-dimensional fuzzy analysis to decision making. In Advances in Soft Computing — Engineering Design and Manufacturing, R. Roy, T. Furuhashi and P.K. Chawdhry, eds. Springer-Verlag, London, 1999b.

Savinov A. Mining possibilistic set-valued rules by generating prime disjunctions. Proc. 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases — PKDD'99, Prague, Czech Republic, September 15-18, 1999a, 536-541.

Savinov A. Mining Interesting Possibilistic Set-Valued Rules. In Fuzzy If-Then Rules in Computational Intelligence: Theory and Applications, Da Ruan and Etienne E. Kerre, eds. Kluwer, 2000a, 107-133.

Savinov A. An algorithm for induction of possibilistic set-valued rules by finding prime disjunctions. In Soft computing in industrial applications, Suzuki, Y., Ovaska, S.J., Furuhashi, T., Roy, R., Dote, Y., eds. Springer-Verlag, London, 2000b.

Savinov A. Mining Spatial Rules by Finding Empty Intervals in Data. Proc. of the 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'03), 3-5 September 2003, Oxford, UK, 1058-1063.

Wrobel S. An Algorithm for Multi-relational Discovery of Subgroups. Proc. of First PKDD, eds. Komorowski, J., Zytkow, J., Berlin:Springer, 78-87, 1997.

Chapter 13

XML-BASED VISUALIZATION AND EVALUATION OF DATA MINING RESULTS

Dietrich Wettschereck
The Robert Gordon University, UK

Abstract: The Predictive Model Markup Language (PMML) is an XML-based industrial standard for the platform- and system-independent representation of data mining models. It is currently supported by a number of knowledge discovery systems. The primary purpose of the PMML standard is to separate model generation from model storage in order to enable users to view, post-process, and utilize data mining models independently of the tool that generated the model. In this chapter, a short introduction to PMML is followed by the presentation of VizWiz. VizWiz is a tool for the visualization and evaluation of data mining models that are specified in PMML. This tool allows for the highly interactive visual exploration of a variety of data mining result types such as decision trees, classification and association rules or subgroups. A noteworthy contribution of this work is that most of these result types can be presented to the user in the same manner, thus reducing the learning rate for the user and removing some of the jargon that often prevents application experts from using knowledge discovery tools.

Key words: Data Mining, Knowledge Discovery, PMML, XML, Visualization

1. INTRODUCTION

The Predictive Model Markup Language (PMML) aims at defining a common representational language for data mining models. Data mining models are typically the results of the modeling phase of the knowledge

discovery process.[1] These results (a.k.a. models) are then evaluated and, if found valuable by the user, incorporated into operational systems via SQL statements or C-programs. The focus of this chapter is the phase between the generation of the model and its deployment. During this phase, typically numerous models are evaluated visually and experimentally, most are discarded, some are modified (manually or automatically) and then re-evaluated and, finally, very few models are deployed. This phase can be seen as an exploratory phase with the difference that in the field of information visualization data are typically explored while in this case models are explored.

Shneiderman [Shneiderman, 2002] makes four recommendations regarding the development of discovery tools and the thesis of this chapter is that the tool, called VizWiz, that is described below follows these recommendations:

1. **Integrate data mining and information visualization**: VizWiz is not a data mining tool, but rather an information visualization tool. However, in a certain sense it does combine these two techniques since it visualizes data mining results and as such uses information visualization techniques as a post-processing mechanism for data mining.
2. **Allow users to specify what they are seeking and what they find interesting**: The highly interactive graphical user interface of VizWiz enables users to quickly zoom in on those (parts of) models that are most interesting to them. Overview plots showing multiple models can be utilized to identify those models best suited for the purpose. For example, a user may prefer models with high coverage over extremely accurate models or vice versa.
3. **Support collaboration**: The input and output[2] format of VizWiz is an XML-based standard that is already supported by a variety of other tools such as IBM's Intelligent Miner or Clementine from SPSS. Users working jointly on a specific analysis problem, for example in a medical application of high public interest, can easily exchange their (preliminary) models. Furthermore, analysis experts using these rather complex systems can utilize VizWiz to present their results to application experts that may not have access to these complex knowledge discovery tools. Finally, Java technology in VizWiz allows for the presentation of data mining models in the Internet thus enabling wide dissemination of interesting findings.

[1] The CRISP-DM process model [Chapman et al., 2000] divides the knowledge discovery process into six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment.

[2] VizWiz can write modified or enhanced PMML files.

4. **Respect human responsibility**: VizWiz is a powerful post-processing tool that combines visualization, evaluation and editing features in order to give the human user maximal influence on the final outcome of the analysis.

VizWiz was therefore developed as a highly interactive tool for the visualization and evaluation of data mining results. This tool utilizes information visualization techniques to enable application experts who are not necessarily experts in knowledge discovery, to explore, evaluate and select those data mining results that are most suited for their purpose. The primary arguments for this enabling technology are that the number and complexity of data mining methods is significantly higher than the number of distinct model types and that the model *generation* process is much more complex than the model *understanding* process, especially when models are properly visualized.

This chapter is organized as follows: Section 2 provides a short introduction to PMML with the aid of an example. The relevance of PMML for visualization tools is highlighted in Section 3 with the introduction of VizWiz. Related work is discussed in Section 4. The chapter concludes with a discussion on the (potential) impact of PMML and VizWiz in Section 5.

2. THE PREDICTIVE MODEL MARKUP LANGUAGE

The Predictive Model Markup Language (PMML) is an XML mark up language that can be used to describe statistical and data mining models. The most recent version (2.1) has been developed by the Data Mining Group [Data Mining Group, 2003], a group of more than 20 vendors of data mining software packages. Prominent members of the group are for example IBM, SPSS, SAS, and NCR. A large variety of model types is already supported (decision and regression trees, neural networks, clustering, regression, naive Bayes, and association and sequence rules), further developments are under way.

Figure 1 lists the XML source code for a simple decision tree. Detailed documentation on the syntax of PMML can be downloaded from the DMG web site [Data Mining Group, 2003] and shall not be the subject of this chapter. The PMML example shown in Figure 1 contains no information about the performance of this decision tree on any compatible data set. It simply provides for each node, what class this node would predict. A visualization of this decision tree is shown in Figure 2.

PMML separates model generation from model visualization and evaluation which should lead to simpler, cheaper, and more robust tools for producing data mining models, and for their subsequent processing. For the purposes of this chapter, PMML should be seen as the vehicle that bridges the gap between data mining tools and (information) visualization. The potential of this bridge is that advanced visualization techniques can be applied for the visualization of data mining results without requiring the developers of such techniques to incorporate their software into any specific data mining tool.

```
<?xml version="1.0" ?> <PMML version="2.1" >
<Header copyright="www.dmg.org" description="A very small
binary tree model to show structure."/>
<DataDictionary numberOfFields="5" >
  <DataField name="temp." optype="continuous"/>
  <DataField name="humidity" optype="continuous"/>
  <DataField name="windy" optype="categorical" >
    <Value value="true"/> <Value value="false"/> </DataField>
  <DataField name="outlook" optype="categorical" >
    <Value value="sunny"/> <Value value="overcast"/>
    <Value value="rain"/> </DataField>
  <DataField name="whatIdo" optype="categorical" >
    <Value value="will play"/> <Value value="may play"/>
    <Value value="no play"/> </DataField>
</DataDictionary>
<TreeModel modelName="golfing" functionName="classification">
 <MiningSchema>
  <MiningField name="temp."/> <MiningField name="humidity"/>
  <MiningField name="windy"/> <MiningField name="outlook"/>
  <MiningField name="whatIdo" usageType="predicted"/>
 </MiningSchema>
 <Node score="will play"> <True/>
  <Node score="will play">
    <SimplePredicate field="outlook" operator="equal"
    value="sunny"/>
   <Node score="will play">
    <CompoundPredicate booleanOperator="and" >
      <SimplePredicate field="temp." operator="lessThan"
      value="90"/>
      <SimplePredicate field="temp." operator="greaterThan"
      value="50"/>
    </CompoundPredicate>
    <Node score="will play" >
      <SimplePredicate field="humidity" operator="lessThan"
      value="80"/></Node>
    <Node score="no play" >
      <SimplePredicate field="humidity"
      operator="greaterOrEqual" value="80"/> </Node> </Node>
   <Node score="no play" >
      <CompoundPredicate booleanOperator="or" >
        <SimplePredicate field="temp."
        operator="greaterOrEqual" value="90"/>
```

```
        <SimplePredicate field="temp." operator="lessOrEqual"
        value="50"/>
      </CompoundPredicate> </Node> </Node>
    <Node score="may play" >
      <CompoundPredicate booleanOperator="or" >
        <SimplePredicate field="outlook" operator="equal"
        value="overcast" />
        <SimplePredicate field="outlook" operator="equal"
        value="rain" />
      </CompoundPredicate>
      <Node score="may play" >
        <CompoundPredicate booleanOperator="and" >
          <SimplePredicate field="temp."
          operator="greaterThan" value="60"/>
          <SimplePredicate field="temp." operator="lessThan"
          value="100"/>
          <SimplePredicate field="outlook" operator="equal"
          value="overcast" />
      <SimplePredicate field="humidity" operator="lessThan"
      value="70" />
      <SimplePredicate field="windy" operator="equal"
      value="false" />
    </CompoundPredicate> </Node>
  <Node score="no play" >
    <CompoundPredicate booleanOperator="and" >
      <SimplePredicate field="outlook" operator="equal"
      value="rain" />
      <SimplePredicate field="humidity" operator="lessThan"
      value="70" />
    </CompoundPredicate>
  </Node></Node></Node></TreeModel></PMML>
```

Figure 1. A PMML example defining a decision tree for the "play/don't play" classification task first defined in [Quinlan, 1993]. This example has been taken from the DMG web site [Data Mining Group, 2003].

In the field of information visualization, PMML should motivate researchers to advance the state of the art in model visualization as opposed to the visualization of (abstracted) data that is typically the subject of this field. The primary difference is that data have already been abstracted by data mining, but in most cases, further abstraction will be needed to support users in picking out the right model(s) for their tasks at hand. A further relevant aspect of PMML is that the XML code representing data mining models can be extended with visualization specific information in order to store user preferences and/or information that is relevant for user adaptation. Especially the use of XML style sheets can support different visualization modes for different user groups. For example, technical users may be shown more enriched visualizations while business users may be presented with standard business charts that show less detail.

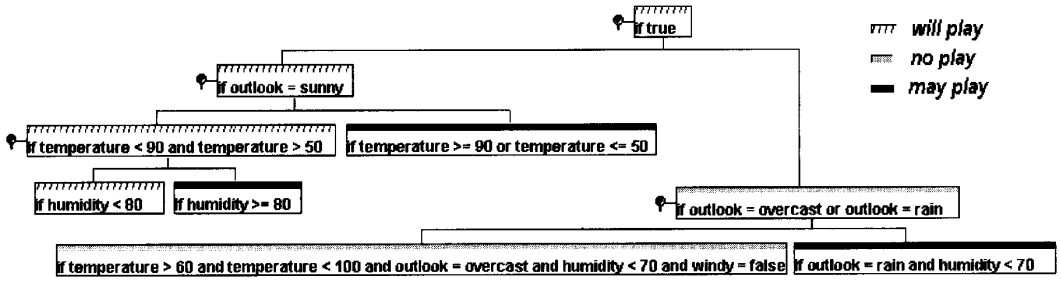


Figure 2. VizWiz visualization of a decision tree, based on the DMG example shown in Figure 1. The bars on top of each node indicate the class predicted by this node and the text in each node lists the conditions that must be satisfied to reach this node.

3. VIZWIZ: INTERACTIVE VISUALIZATION AND EVALUATION

VizWiz is a tool for the visualization and evaluation of data mining models. It is written in Java and can be run as an Applet with the Java Runtime Environment, version 1.4. VizWiz reads and writes modified models in PMML format. The graphical user interface of VizWiz offers two primary viewing options to the user: one can either view a plot showing the relative performance of each model on a Receiver Operator Characteristics (ROC) [Provost & Fawcett, 2001] plot (Figure 3) or one can view a detailed graphical rendering of each model (Figure 4). The first version can thus serve as an overview window that supports the user in quickly zooming in on those models that are most interesting to him/her. The second viewing option can be used to learn more about each model, to edit the model and to test the model on selected or all data records of a given test data set. VizWiz currently offers interactive visualizations for the following model types:

- Linear regression
- Decision- and regression trees
- Association rules
- Propositional and first-order rules[3]
- Subgroups

The last two model types (rules and subgroups) are not explicitly supported by PMML and were therefore encoded in the format of multi-variate decision trees with minor extensions. The output of a multitude of

[3] Propositional rules are typically of the form "if variable_a = value_1 and variable_b = value_2 then class = class_1", more complicated conditions are of course possible and supported. First-order rules are typically of the form "if pred_1(A,B) and pred_2(B,C) then class_1(A)" where pred_1 and pred_2 are predicates such as "father_of" and A, B, C, and D are variables.

data mining and machine learning algorithms is therefore represented and visualized in a coherent manner. This did not only significantly reduce the coding effort for VizWiz, but also removed the burden of understanding the output of such diverse systems as decision tree learners and ILP-based[4] rule learners from the user.

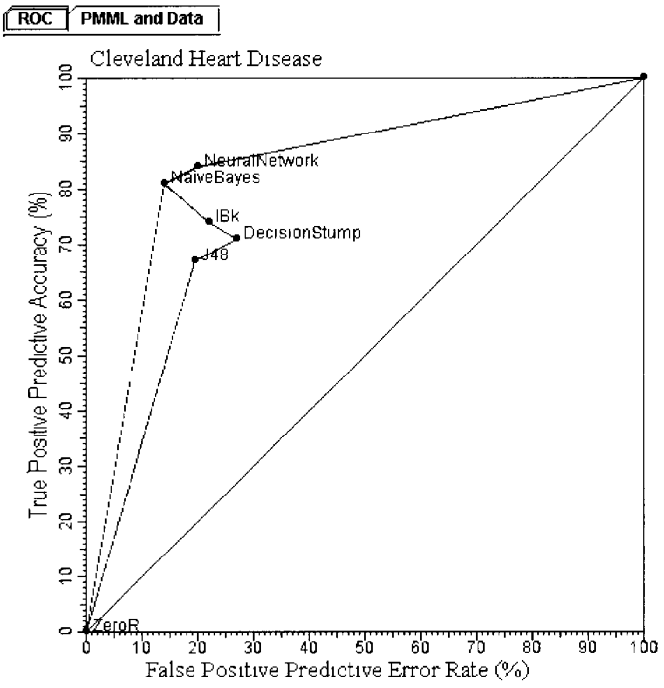


Figure 3. The ROC curve for six models generated from the Cleveland Heart Disease domain [Blake, Keogh & Merz, 2002]

Figure 3 summarizes the predictive performance of six data mining algorithms on the Cleveland Heart Disease domain [Blake, Keogh & Merz, 2002] with the help of a ROC [Provost & Fawcett, 2001] plot. The results were produced using the machine learning toolbox WEKA [Witten & Frank, 1999][5]; default parameters were used in all cases. Plotted are the "true positive" vs. the "false positive" rates. The ideal point on this plot is the upper left corner. A model reaching this point would correctly classify all positive instances, but would not classify any negative instances as belonging to that class. The points ("model performance") lying on or near the outer hull of the curve connecting the points (0, 0) and (100, 100) ("Naive Bayes" and "Neural Network" in the case of Figure 3) indicate the best models. Depending on the characteristics of the particular task at hand, a user should use one of these two models or a combination thereof.

[4] ILP: inductive logic programming
[5] Software that converts WEKA output to PMML is currently under development.

Ongoing research investigates how this technique can be extended to problems with more than 2 classes. Currently, VizWiz plots the performance of all evaluated models for one particular class against all other classes and shows multiple overview graphs, one for each class.

The PMML file used to generate the visualization shown in Figure 4 contains information about the performance of the decision tree on a specific data set and the visualization shows this additional information in the differently patterned[6] bars on top of each tree node. The bar on the top left of each node shows the number of instances of the class predicted by this node that are covered by this node. The bar (or bars in case of more than two classes) on the right hand side indicates the number of exceptions or misclassifications performed by this node. The user can thus quickly discover relatively pure nodes that cover a large number of instances and are therefore of most interest. The user cans interactively open and close tree nodes by clicking on the symbol to the left of each internal node. The tree can also be viewed in vertical mode which is more convenient for very large trees. Further interactive features are supported for node editing and viewing: change conditions, remove or add internal node or subtree, show more details, print node conditions as SQL. When a compatible data set is loaded into VizWiz (as in Figure 4), then the user can also click on a node to receive a table of all instances that are covered by this node. Alternatively, the user can click on a specific data record and the path to the tree node that classifies this instance will be highlighted.

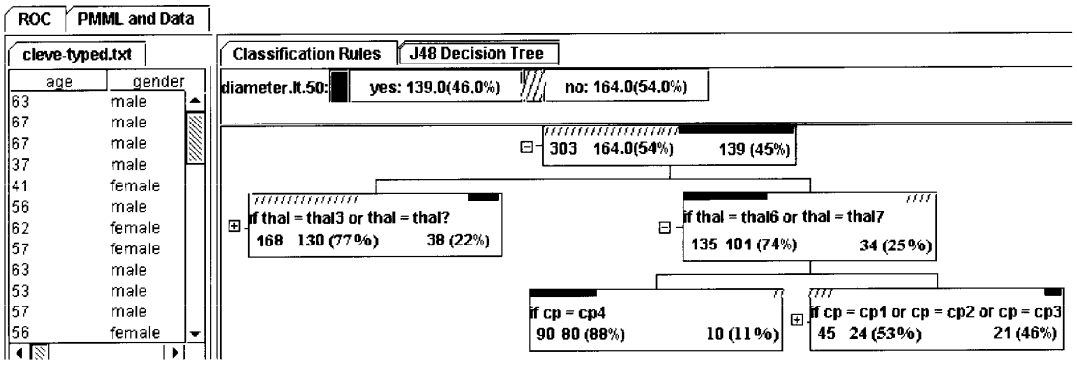


Figure 4. VizWiz detail view of the decision tree model for the Cleveland domain. The panel on the left hand side shows selected records of the test data set that was used to generate the ROC curve shown in Figure 3.

[6] These bars are colored in reality, but have been changed to patterns for easier viewing in the printed version.

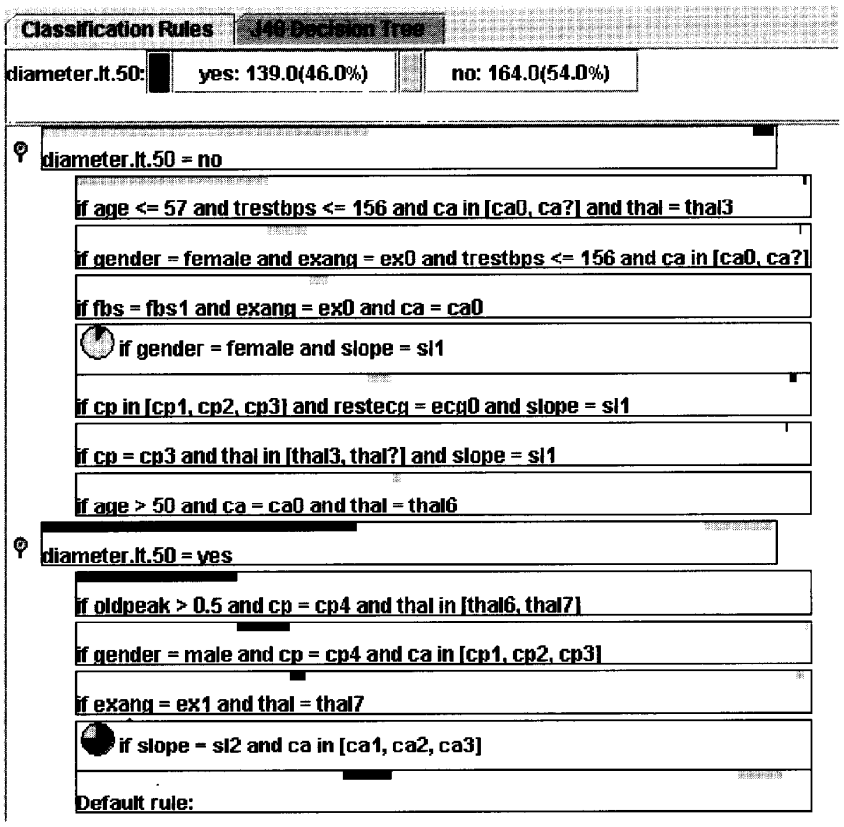


Figure 5. VizWiz visualization of a set of propositional rules

Figure 5 shows a set of ordered rules that were produced by C4.5-rules, a combined decision tree / rule learner that first creates a decision tree and then generates a set of ordered rules from the tree. The rule set is visualized as a horizontal tree of depth two. C4.5-rules orders rules by classes such that they can be visualized in separate sub-trees. The first branch in Figure 5 shows all rules that predict the class "no," while the second branch predicts the class "yes." A further interactive feature of VizWiz is also shown in this figure: the display of rule / node statistics can be changed from bars to pie charts. Bars convey more information since they also indicate the number of instances covered by the rule, but pie charts are more expressive in showing the relative impurity of a rule, i.e. the ratio between correctly predicted and incorrectly predicted instances. The fact that the rules are ordered is indicated by the indentation of the bars for each rule. Only the instances that are not covered by previous rules may be covered by subsequent rules.

An extension of the standard PMML format for decision trees supports the display of first-order rules as shown in Figure 6. Especially first-order rules are sometimes hard to read for users that are not accustomed to Prolog notation. VizWiz was therefore extended with a facility to provide pretty print representation for rules. In Figure 6, some of the conditions are still in their original form (for example, "gender(Actor, female)") while others have

been replaced by natural language ("Actor beams down to planet with kirk"). The mapping is defined in the PMML file and variable and constants are replaced automatically by their proper values.

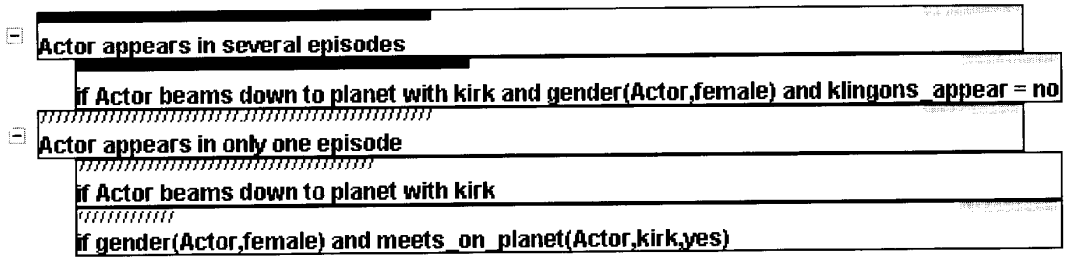


Figure 6. VizWiz visualization of a set of hand-crafted first-order rules predicting whether an actor will appear more than once in the original star trek series

Association rules are explicitly supported by their own PMML format that was one of the first formats to be agreed upon by the DMG [Data Mining Group, 2003]. The confidence and support values of each rule are of foremost interest when inspecting association rules and this is reflected by the slightly modified (as compared to trees and classification rules) visualization of association rules shown in Figure. There is now only a single bar on top of each rule that denotes the number of instances that support this rule. The shade of the bar denotes the confidence value, with a lighter shade indicating a higher confidence value. A number of controls (not shown) are available to filter rules such that the user can restrict the visualization to those rules that are within a specific range of confidence and support values. Such interaction features are essential when working with association rules since most association rule learners typically output a very large number of rules.

The last model type to be discussed in this chapter that is visualized by VizWiz are subgroups, also known as deviation patterns. A typical subgroup discovery algorithm is MIDOS [Wrobel, 1997]. Subgroups may be used as classification systems, but typically they are used in tasks with highly skewed distributions such as the analysis of the return rate of mailing campaigns. The quality of a subgroup is essentially determined by its deviation from the norm and not by its predictive accuracy. Typically, a higher deviation is desired, since it denotes subgroups with very different behavior. The visualization of the subgroups shown in Figure 8 therefore shows the distribution of the target value within each group. Each group (with the exception of the left-most group that shows the entire data set) is represented by a pie chart that is nested inside another pie chart that displays the distribution of the target value in the entire data set. The area of the pie chart corresponds to the size of the subgroup. This visualization method has

been chosen despite the inherent danger associated with all pie chart visualizations that the user incorrectly perceives exact group sizes. Feedback from a variety of technical and non-technical users has indicated that the intuitive appeal of pie charts supersedes the danger of misinterpretation and users inspect the actual numbers once they have gained an overview through the pie charts.

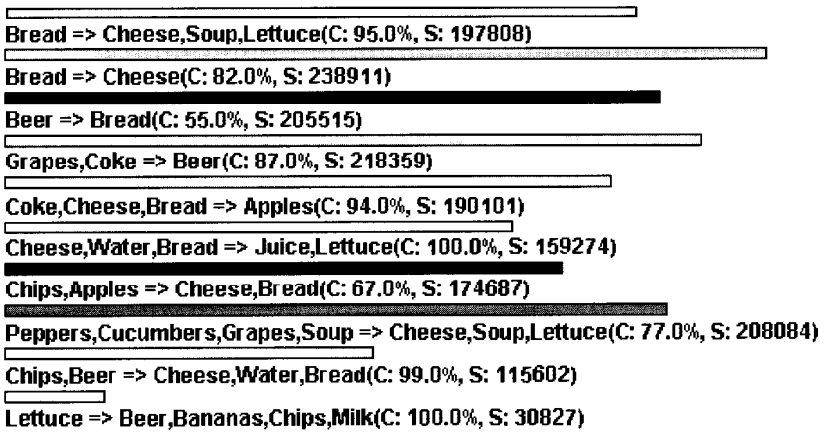


Figure 7. VizWiz visualization of a set of hand crafted association rules

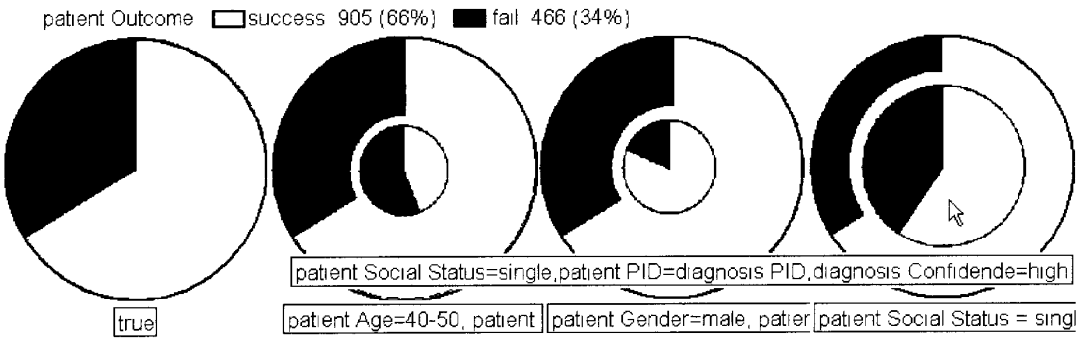


Figure 8. Visualization of three subgroups in a multi-relational medical application. Shown are the distributions for the entire data set (left most chart) and three selected subgroups. The legend on top shows the size of the entire data set and the distribution of the two target values 'success' and 'fail'. The scrollable text below each subgroup shows the actual description of this group. For example the right most subgroup contains all single patients where the doctor gave his diagnosis with a high confidence (as shown by the pop-up text).

VizWiz is quite efficient in processing and displaying data mining models and is capable of handling relatively large PMML files (i.e., decision trees with thousands of nodes or association rules sets with hundreds of rules). The time required for processing of the actual PMML file can be neglected in comparison to the time required to plot the visualization on screen. VizWiz has therefore been designed to initially display only small

portions of the model: in the case of decision trees, only the root is initially shown; in the case of association rules, only those rules with the highest confidence and support values are shown. The user may then select to display additional information which typically is realized in real time.

4. RELATED WORK

The complexity of data mining results and the explorative nature of knowledge discovery tasks have lead to a number of visualization methods for various data mining models [Thearling, Becker, DeCoste, Mawby, Pilote & Sommerfield, 2001]. Most data mining packages contain visualizations for decision trees. For example, WEKA [Witten & Frank, 1999] contains a simple non-interactive tree-viewer while SGI's MineSet contains a highly interactive 2.5-dimensional tree visualizer. The SNNS [Stuttgart Neural Network Simulator, 2003] visualizes neural networks with particular focus on the visualization of the training process. IBM's Intelligent Miner features a highly expressive visualization of their clustering method. It is common to all of these systems that visualization is only one aspect of the workbench and not its primary focus. Very few systems exist that specialize on visualization and evaluation alone. This is primarily due to the lack of representational standards for data mining results that is now addressed by PMML. The PEAR [Jorg, Pocas & Azevedo, 2002] system is such a system that was specifically designed as a web-based system for post-processing PMML association rules. It offers further selection and interaction mechanisms that go beyond VizWiz's capabilities, but is limited to association rules.

Gamberger, Lavrac, and Wettschereck [Gamberger, Lavrac & Wettschereck, 2002] have proposed an alternative method for the visualization of subgroups that may be implemented in VizWiz in the future.

An alternative to the visualization of actual data mining models and a means for visualizing the outcome of model types that cannot be visualized is the visualization of class probability estimates as proposed by [Rheingans & desJardins, 2000; Frank & Hall, 2003]. This method visualizes the class predictions of classifiers in a two-dimensional space where the user can interactively select the two dimensions to be displayed. Each class is assigned a certain color and each pixel in the feature space is colored according to a mixture of these colors depending on the probability of each class at this pixel's location.

5. DISCUSSION

This chapter pursued two goals: (1) to introduce PMML to the information visualization community and to motivate its use as bridge between the fields of data mining and information visualization and (2) to introduce a visualization tool that fully utilizes PMML and highlights the benefits that can be obtained when separating model generation from model processing.

VizWiz has a clear focus on the visualization and evaluation of classification models such as decision trees and classification rules. This focus reflects the fact that these methods are among the most powerful methods developed to date in data mining and that most commercial and non-commercial knowledge discovery software packages incorporate such methods. VizWiz is therefore potentially capable of handling the output of a huge variety of algorithms, as long as this output is converted to PMML. This implies a huge benefit for algorithm developers and users: developers need not worry about visualization; they can simply produce PMML output and then use a tool such as VizWiz to visualize their results. Users ("viewers") of models can work with one single post-processing tool that is not restricted to a given set of analysis methods and that produces coherent visualizations, evaluation and interaction methods which is especially beneficial to the technically less skilled user.

The impact of PMML on the field of data mining is clear: standardized interfaces for data input such as JDBC or ODBC and for output (PMML) enable researchers and developers to plug their tools with less effort into larger software packages thereby significantly increasing the potential user community. The field of information visualization can benefit from PMML as developers need not worry about understanding the output of specific analysis methods, but can concentrate on developing new methods for model browsing, selection, and editing. As such, visualization tools such as VizWiz can serve as powerful decision support tools that may help to bridge the gap between highly skilled knowledge discovery experts and application experts or general information seekers.

6. ACKNOWLEDGEMENTS

The ROC plotting software has been developed by J. Farrand from the University of Bristol. Comments on earlier drafts of this chapter and suggestions for the improvement of the software have been made by various members of the SolEuNet consortium, most notably Steve Moyle. I am also

indebted to B. Noble and other colleagues from RGU for their helpful comments.

This work has been supported in part by the EU funded project SolEuNet – Data Mining and Decision Support for Business Competitiveness: A European Virtual Enterprise (IST-1999-11495) and the Research Development Initiative (RDI) at The Robert Gordon University.

7. EXERCISES AND PROBLEMS

1. Rewrite the PMML example in Figure 1 using the "SimpleSetPredicate" (see http://www.dmg.org/v2-0/TreeModel.html) for the node:

```
<CompoundPredicate booleanOperator="or" >
        <SimplePredicate field="outlook" operator="equal"
         value="overcast" />
        <SimplePredicate field="outlook" operator="equal"
         value="rain" />
</CompoundPredicate>
```

2. Reconstruct the PMML code for Figure 7.

Advanced

3. Figure 5 assumes an implicit ordering of the rules. Is it possible to encode this ordering in PMML explicitly and if so, how?

4. What are the most useful (filter) controls a graphical user interface visualizing association rules should offer? Would these controls also be sensible for other model types? Does this assume the availability of additional data that is not part of the minimal PMML format?

8. REFERENCES

Blake C., Keogh E., Merz CJ. UCI repository of Machine Learning databases (machine readable data repository). Irvine, CA: Department of Information and Computer Science, University of California at Irvine. http://www.cs.uci.edu/mlearn/MLRepository.html (accessed 15 November 2002).

Blockeel H., Moyle S., Centralized model evaluation for collaborative data mining. In M. Grobelnik, D. Mladenic, M. Bohanec, and M. Gams, editors, Proceedings A of the 5th International Multi-Conference Information Society, 2002: Data Mining and Data Warehousing/Intelligent Systems, pages 100–103. Jozef Stefan Institute, Ljubljana, Slovenia.

Chapman P., Clinton J., Kerber R., Khabaza T., Reinartz T., Shearer C., Wirth R., CRISPDM 1.0: step-by-step data mining guide, 2000.

Data Mining Group (DMG), PMML specification [WWW document] http://www.dmg.org and http://sourceforge.net/projects/pmml/ (both accessed 09.05.2003).

Farrand J., ROCON, 2002, [WWW document] http://www.cs.bris.ac.uk/ farrand/rocon/ (accessed 22.05.2003).

Frank E., Hall M., Weka Boundary Visualizer, 2003, [WWW document] http://www.cs.waikato.ac.nz/ml/weka/bvis/ (accessed 15.05.2003).

Gamberger D., Lavrac N., Wettschereck D., Subgroup Visualization: A Method and Application in Population Screening. ECAI 2002 Workshop on Intelligent Data Analysis in Medicine and Pharmacology, 2002

Jorge A., Poc¸as J, Azevedo P., Post-processing operators for browsing large sets of association rules, in Proceedings of Discovery Science 02, Luebeck, Germany, LNCS 2534, Eds. Steffen Lange, Ken Satoh, Carl H. Smith, Springer-Verlag, 2002

Mladenic D., Lavrac N., Bohanec M., Moyle S., editors, Data Mining and Decision Support: Integration and Collaboration, Kluwer Academic Publishers, 2003

Provost F., Fawcett T., Robust Classification for Imprecise Environments. Machine Learning 42(3): 203-231, 2001

Quinlan J., C4.5: Programs for Machine Learning. Machine Learning. Morgan Kaufmann, San Mateo, CA, USA, 1993

Rheingans P., desJardins M., Visualizing high-dimensional predictive model quality. In Proceedings of IEEE Visualization 2000, pages 493–496, 2000

Shneiderman B., Inventing discovery tools: combining information visualization with data mining. Information Visualization 1:5-12. Palgrave Macmillan Ltd, 2002

Stuttgart Neural Network Simulator [WWW document] http://www-ra.informatik. unituebingen.de/SNNS/ (accessed 18.05.2003).

Thearling K, Becker B, DeCoste D, Mawby B, Pilote M, Sommerfield D (2001) Information Visualization in Data Mining and Knowledge Discovery, Chapter Visualizing Data Mining Models. Morgan Kaufmann.

Witten I., Frank E., Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, 1999

Wrobel S., An algorithm for multi-relational discovery of subgroups. Proc. First European Symposium on Principles of Data Mining and Knowledge Discovery, 78–87, Springer, 1997

Chapter 14

NEURAL-NETWORK TECHNIQUES FOR VISUAL MINING CLINICAL ELECTROENCEPHALOGRAMS

Vitaly Schetinin[1], Joachim Schult[2], and Anatoly Brazhnikov[3]
[1]*University of Exeter, UK;* [2]*University of Jena, Germany;* [3]*Ansvazer Consulting, Canada*

Abstract: In this chapter, we describe new neural-network techniques developed for visual mining clinical electroencephalograms (EEGs), the weak electrical potentials invoked by brain activity. These techniques exploit the fruitful ideas of Group Method of Data Handling (GMDH). Section 2 briefly describes the standard neural-network techniques that are able to learn well-suited classification modes from data presented by relevant features. Section 3 introduces an evolving cascade neural network technique that adds new input nodes as well as new neurons to the network while the training error decreases. This algorithm is applied to recognize artifacts in the clinical EEGs. Section 4 presents the GMDH-type polynomial networks trained from data. We applied this technique to distinguish the EEGs recorded from an Alzheimer and a healthy patient as well as recognize EEG artifacts. Section 5 describes the new neural-network technique developed to derive multi-class concepts from data. We used this technique for deriving a 16-class concept from the large-scale clinical EEG data. Finally, we discuss perspectives of applying the neural-network techniques to clinical EEGs

Key words: Classification model, pattern visualization, neural network, cascade architecture, feature selection, polynomial, electroencephalogram, decision tree

1. INTRODUCTION

Data mining as a process of discovering interesting patterns and relations in data presented by *labeled examples* can be referred to deriving *classification models* or *classifiers* that assign an unknown example to one of the given classes with acceptable accuracy. A typical classification problem

is presented by a *data set* of labeled examples that are characterized by *variables* or *features*. Experts assume such features make a distinct contribution to the classification problem. Such features are called *relevant*. However, among these features, some may be *irrelevant* and/or *redundant*: the first can seriously hurt the classification accuracy whereas the second are useless for the classification and can obstruct understanding how decisions are derived. Both the irrelevant and redundant features have to be discarded.

In solving the classification problem, a user has to derive or learn a classification model from a *training data* set and test its performance on a *testing data* set of the labeled examples. These data must be disjoint in order to objectively evaluate how well the classification model can classify unseen examples.

Besides that, users such as medical experts need to both classify unseen examples and verify decisions by analyzing the underlying causal relations between the involved features and the model outcome. Such an analysis can be comprehensively done by visualizing a discovered model and/or discovered patterns [Kovalerchuk & Vityaev, 2000]. Some data mining methods can provide the visualization of a classification model as well as associated patterns. For example, we may visualize an derived decision tree model where each branch visualizes an individual pattern. However using neural-network techniques, we can visualize an derived network but cannot visualize the interesting patterns. This issue is critical for medical experts who need to interpret data mining results using a visual form in addition to a textual description.

In this chapter, we describe new neural-network techniques developed to provide both the visualization of classification models and the visualization of patterns. The advantages of these techniques are illustrated by mining medical data such as electroencephalograms (EEGs), the weak electrical potentials invoked by brain activity, whose spectral characteristics are taken as visual features. Within this chapter, we compare some existing data mining techniques with the new techniques in the respect to the above aspects of visualization. The results show that in addition to a textual presentation, EEG-experts can visually interpret discovered classification models and patterns.

When applying data mining techniques, EEG-experts often cannot properly assume relevant features and avoid irrelevant and redundant ones. Besides that, some features become relevant after being taken into account in the combination with other features. In such cases data mining techniques exploit a special learning strategy capable of *selecting* relevant features during the derivation of a classification model [Duda & Hart, 2000; Farlow, 1984; Madala & Ivakhnenko, 1994; Müller & Lemke, 2003]. Such a strategy

allows experts to learn classification models more accurately than strategies that select features before learning.

Surveying data mining methods, we see that most of them aimed at extracting comprehensible models imply a *trade-off* between classification accuracy and representation complexity [Avilo Garcez, Broda & Gabbay, 2001; Setiono, 2000; Towell & Shavlik, 1993]. Less work has been undertaken to study methods capable of discovering the comprehensible models without decreasing their classification accuracy.

Below we describe new neural-network techniques developed for the visual mining clinical EEGs. By exploiting the fruitful ideas of Group Method of Data Handling (GMDH) of Ivakhnenko [Madala & Ivakhnenko, 1994; Müller & Lemke, 2003], these techniques are able to derive comprehensible classification models while in the meantime keeping their classification error down.

In section 2, we briefly describe standard neural-network techniques, including cascade-correlation architecture that are able to learn well-suited classification modes from data. These methods however cannot generalize well in the presence of irrelevant and/or noisy features.

Section 3 introduces an evolving cascade neural-network technique that adds new input nodes as well as new neurons to the network while the training error decreases. The resultant networks have a near minimal number of input variables and hidden neurons, which allow classifying new examples well. We apply this algorithm to recognize artifacts in the clinical EEGs.

Section 4 presents the GMDH-type polynomial networks that learn from data. These networks are represented as concise sets of short-term polynomials and can be presented in visual form. Moreover, the GMDH-type neural networks can generalize better than the standard fully connected neural networks. We apply this technique to distinguish the EEGs recorded from an Alzheimer patient and from a healthy patient as well as to recognize EEG artifacts.

Section 5 describes the new decision tree neural-network technique developed to derive multi-class concepts from data. We use this technique for deriving a 16-class concept from the large-scale clinical EEG data recorded from sleeping newborns. This concept assists clinicians to predict some brain development pathologies of newborns. Finally, we discuss perspectives of applying the neural-network techniques to clinical EEGs

2. NEURAL NETWORK BASED TECHNIQUES

In this section, we briefly describe a standard technique used in our experiments for training feed-forward neural networks (FNN) with a back-propagation algorithm. Then we describe the cascade-correlation architecture and end by discussing the shortcomings and advantages of these techniques.

2.1 A Standard Neural-Network Technique

A standard neural-network technique exploits a *feed-forward* fully connected network consisting of the *input nodes*, *hidden* and *output* neurons that are connected each other by the adjustable *synaptic weights* [Bishop, 1995]. This technique implies that a *structure* of neural network has to be predefined properly. This means that users must preset an appropriate number of the input nodes and hidden neurons and apply a suitable *activation function*. For example, the user may apply a *sigmoid* activation function described as

$$y = f(\mathbf{x}, \mathbf{w}) = 1/(1 + \exp(- w_0 - \Sigma_i^m w_i x_i)), \quad (1)$$

where $\mathbf{x} = (x_1, ..., x_m)^T$ is a $m \times 1$ input vector, $\mathbf{w} = (w_1, \ldots, w_m)^T$ is a $m \times 1$ synaptic weight vector, w_0 is a bias term and m is the number of input variables.

Then the user has to select a suitable learning algorithm and then properly set its parameters such as the *learning rate* and the number of the *training epochs*. Note that when the neural networks include at least two hidden neurons, the learning algorithms with *error back-propagation* usually provide the best performance in the term of the classification accuracy [Bishop, 1995].

Within the standard technique, first the learning algorithm initializes the synaptic weights **w**. The values of **w** are updated while the *training error* decreases for a given number of the training epochs. The resultant classification error is dependant on the given learning parameters as well as on the initial values $\mathbf{w}^0$ of neuron weights. For these reasons, neural networks are trained several times with random values of initial weights and different learning parameters. This allows the user to avoid local minima and find a neural network with a near minimal classification error.

After training, the user expects that the neural network can classify new inputs well and that its classification accuracy is acceptable. However the learning algorithm may fit the neuron weights to specifics of training data that are absent in new data. In this case, neural networks become to be *over-fitted* and do not generalize well. Within the standard technique, the

generalization ability of the trained network is evaluated on a *validation* subset of the labeled examples that have not been used for training the network.

Figure 1 depicts a case when after k^* training epochs the validation error starts to increase while the training error continues to decrease. This means that after k^* training epochs the neural network has become over-fitted. To prevent over-fitting, we can update the neuron weights while the validation error decreases.

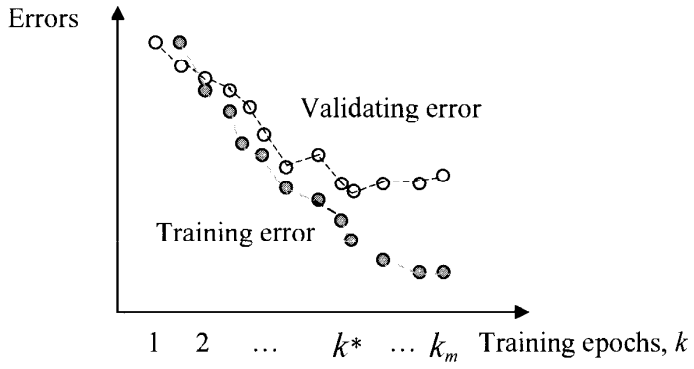


Figure 1. Learning curves for the training and validating sets

When classification problems are characterized in the m-dimensional space of input variables, the performance of neural networks may be radically improved by applying the Principal Component Analysis (PCA) to training data [Bishop, 1995]. The PCA may significantly reduce the number of the input variables and consequently the number of synaptic weights, which are updated during learning. A basic idea behind the PCA is to turn the initial variables so that the classification problem might be resolved in a reduced input space.

Figure 2 depicts an example of a classification problem resolved in a two-dimensional space with input variables x_1 and x_2 by using a separating function $f_1(x_1, x_2)$. However we can turn x_1 and x_2 so that this problem might be solved in one-dimensional input space of a principal component $z_1 = a_1x_1 + a_2x_2$, where a_1 and a_2 are the coefficients of a linear transformation. In this case a new separating function is $f_2(z_1)$ that is equal to 0 if $z_1 < \vartheta_1$ and equal to 1 if $z_1 \geq \vartheta_1$, where ϑ_1 is a threshold learned from the training data represented by the new variable z_1.

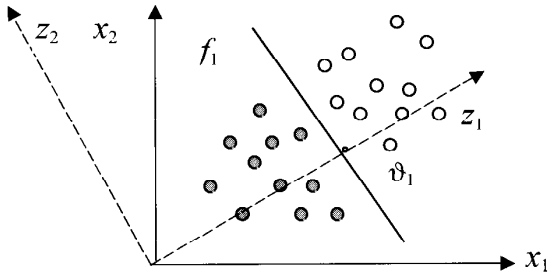


Figure 2. An example of two-dimensional classification problem

As we see, the new components z_1 and z_2 make a different contribution to the variance of the training data: the first component contributes much more than the second does. This example demonstrates how the PCA can rationally reduce the input space. However, users using PCA must properly define a variance level and the number of components making a contribution to the classification.

Thus by using the standard technique, we may find a suitable neural-network structure and then fit its weights to the training data while the validation error decreases. Each neural network with a given number of input nodes and hidden neurons should be trained several times, say 100 times.

Thus, we can see that the standard technique is computationally expensive. For this reason, users use fast learning algorithms such as the back-propagation algorithm by Levenberg-Marquardt [Bishop, 1995].

2.2 A Cascade-Correlation Architecture

To solve classification and pattern recognition problems, [Fahlman & Lebiere, 1990] proposed a *cascade-correlation architecture* of neural networks. The neural networks with the cascade-correlation architecture differ from the above networks with a predefined structure. In contrast to the last section, cascade networks start learning with only one neuron. Then the algorithm adds and trains new neurons creating a multi-layer structure. The new neurons are added to the networks as long as the residual classification error decreases. Thus, the cascade-correlation learning algorithm allows growing neural networks to a *near optimal size* required for good generalization [Farlow, 1984; Iba, deGaris & Sato, 1994; Madala & Ivakhnenko, 1994; Müller & Lemke, 2003].

Figure 3 depicts an example of cascade-correlation architecture consisting of four input nodes x_1, ..., x_4, two hidden neurons z_1 and z_2, and one output neuron y. The first hidden neuron is connected to all the input nodes, and the output neuron is connected to all the input nodes as well as to the hidden neurons z_1 and z_2.

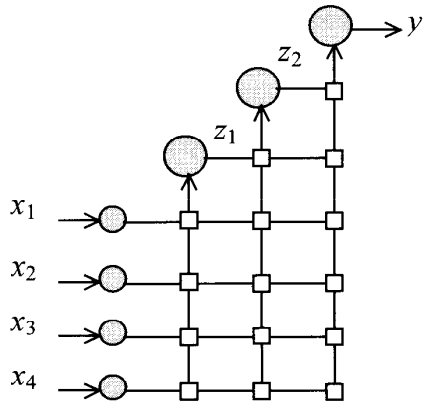


Figure 3. An example of cascade-correlation architecture

The learning of a cascade-correlation architecture is based on the following ideas. The first idea is to build up the cascade architecture by adding new neurons connected to all the input nodes and previous hidden neurons. The second idea is that the learning algorithm attempts to reduce the residual error by updating weights of the new neuron, that is, each time only the output neuron is trained. The third idea is to add one-by-one new neurons to the network while its residual error decreases.

The main steps of the learning cascade-correlation algorithm are described below.

```
nnet = []; % initializing
error = n;
new-error = error - 1;
while new-error < error
  nnet = add-new-neuron(nnet);
  nnet = train-neuron(nnet, X, Y);
  new-error = calc-error(net(nnet, X) - Y);
  error = new-error;
end
nnet = cut(nnet);
```

Here n, X, and Y are the number of training examples, the input data and a target vector, respectively. The procedure `cut` excludes the last neuron from the trained cascade network and then returns the result to the `nnet`.

There are two advantages of cascade neural networks. First, no size and connectivity of neural networks are predefined, that is, the network is automatically built up. Second, the cascade network learns fast because each of its neurons is trained independently from other neurons.

However, the algorithm can train cascade network well only if all the input variables are relevant to the classification problem. In the next section,

we will describe a new algorithm, which can train cascade neural networks in the presence of irrelevant features.

3. EVOLVING CASCADE NEURAL NETWORKS

In this section we describe an evolving cascade neural network technique, which adds new input nodes as well as new neurons to the network while the training error is decreased. This algorithm is used to recognize artifacts in the clinical EEGs.

3.1 An Evolving Cascade Neural Network

Let us assume a classification problem is represented by m input variables $x_1, \ldots, x_m$ some of which may be irrelevant or noisy. In this case, a standard pre-processing technique used for selecting relevant variables may fail because this technique does not consider useful combinations of the input variables. A more suitable strategy is to select relevant features during learning. To do so, let us define the neurons in which the number of inputs, p, increases as follows $p = r + 1$, where $r = 0, 1, 2, \ldots$ is the number of layer in the cascade network. So for $r = 0$, there are m neurons with one input variable. For the first layer, there are neurons with $p = 2$ inputs and so on.

Let us now fit all the neurons for $r = 0$. Then among these neurons, one can be found that provides the best performance on the validation data set. Fix an input variable of this neuron, x_{i1}, in order to connect it with all the neurons that will be added to the network.

At the first layer, the algorithm trains the candidate-neurons with two inputs: the variable x_{i1} and one of the remaining input variables. The neuron with the best performance is added to the network.

Each following neuron is connected with the variable x_{i1} and the outputs of all the previous neurons. For the second layer, the candidate-neurons have three inputs: the first is connected with the output of the previous neuron, the second input with the input x_{i1}, and the third with one of the input variables $x_1, \ldots, x_m$.

Defining a sigmoid activation function of the neurons, we can write the output z_r for the rth neuron as follows:

$$z_r = f(\mathbf{u}, \mathbf{w}) = 1/(1 + \exp(-w_0 - \Sigma_t^p u_t w_t)), \quad (2)$$

where $\mathbf{u} = (u_1, \ldots, u_p)$ is a $p \times 1$ input vector of the rth neuron, $\mathbf{w} = (w_1, \ldots, w_m)$ is a $m \times 1$ vector of synaptic weights, and w_1 is a bias term.

The idea behind our learning method is that the relevance of an input variable connected to the candidate-neuron can be estimated in *ad hoc*

manner. The learning algorithm starts to train the candidate-neurons with one input variable and then step-by-step adds new inputs and new neurons to the network. As a result, the internal connections of the cascade neural network are built according to the best performance achievable in each new layer. Therefore, in building the cascade network our algorithm exploits a greedy search heuristic.

Within our technique, the performance of the network, C_r, is evaluated for each candidate-neuron at the rth layer. The value of C_r is dependent on the generalization ability of the trained candidate-neuron with the given connections. Clearly, a neuron connected to the irrelevant connections cannot properly classify the validating examples and subsequently its value of C_r is high.

If value C_r calculated for the rth neuron is less than value C_{r-1} calculated for the previous $(r-1)$st neuron, the connections selected for the rth neuron are relevant, otherwise they are irrelevant. Formally, this heuristic can be described by the following inequality:

$$\begin{aligned} &\textit{if } C_r < C_{r-1}, \textit{ then} \text{ the connections are relevant,} \\ &\textit{else} \text{ the connections are irrelevant.} \end{aligned} \tag{3}$$

If the inequality (3) is met, the rth neuron is added to the network. If no neurons satisfy this inequality, the algorithm stops. As a result, an rth neuron providing a minimal validation error is assigned to be an output neuron for the cascade network.

3.2 An Algorithm for Evolving Cascade Neural Networks

By adding new features and neurons as they are required, the cascade neural network evolves during learning. The main steps of the evolving algorithm are described below.

```
X = [x1, ..., xm]; % a pool of m input variables
P = 1;       % the number of neuron inputs
% Train single-input neurons and calculate errors
for i = 1:m
  N1 = create-neuron(p, X(i));
  N1 = fit-weight(N1);
  E(i) = calc-error(N1);
end
[E1,F] = sort-ascend(E);
h = 1;       % the position of the variable in F
C0 = E1(h);
% Create a cascade network NN
```

```
NN = [];
r = 0;       % the number of hidden neurons
p = 2;
while h < m
  h := h + 1;
  V = [X(F(1)), X(F(h))];
  % Add links to the hidden neurons
  for j = 1:r
    V = [V, NN(j)];
  end
  % Create a candidate-neuron N1
  N1 = create-neuron(p, V];
  N1 = fit-weight(N1);
  C1 = calc-error(N1);
  if C1 < C0
    r := r + 1;
    p := r + 2;
    NN(r) = add-neuron(N1);
  end
end
```

The algorithm starts to learn the candidate-neurons with one input and then saves their validating errors in a pool E. The procedure `sort-ascend` arranges pool E in ascending order and saves the indexes of the input variables in a pool F. The first component of the F is an index of the input variable providing a minimal classification error C_0.

At the following steps, the algorithm adds new features as well as new neurons to the network while the validation error C_1 calculated for the candidate-neuron N_1 decreases. The weights of candidate-neurons are updated until condition (3) is satisfied.

As a result, the cascade neural network consisting of the r neurons is placed in the pool NN. The size of this network is nearly minimal because the stopping rule is met for a minimal number of neurons.

Below we describe an application of this algorithm for recognizing artifacts in clinical EEGs. These EEGs have characteristics such as noise and features that are redundant or irrelevant to the classification problem.

3.3 An Evolving Cascade Neural Network

In our experiment, we used the clinical EEGs recorded via the standard EEG channels C2 and C4 from two newborns during sleeping hours. Following [Breidbach, Holthausen, Scheidt & Frenzel, 1998] these EEGs were represented by spectral features calculated in 10-second segments for 6 frequency bands: subdelta (0-1.5 Hz), delta (1.5-3.5 Hz), theta (3.5-7.5 Hz),

alpha (7.5-13.5 Hz), beta 1 (13.5-19.5 Hz), and beta 2 (19.5-25 Hz). Additionally for each band, the values of relative powers and their variances were calculated for channels C3 and C4 and their sum, C3+C4. The total number of the features was 72. Values of these features were normalized to have a zero mean and a unit variance.

The normal segments and artifacts in the EEGs were manually labeled by an EEG-viewer that analyzed muscle and cardiac activities of patients recorded from additional channels. As an example of normal segments and artifacts, Figure 4 depicts the fragment of EEG containing 500 segments presented by 36 features. In this fragment the EEG-expert recognized segments 15, 22, 24, 84, and 85 as artifacts and the remaining as normal.

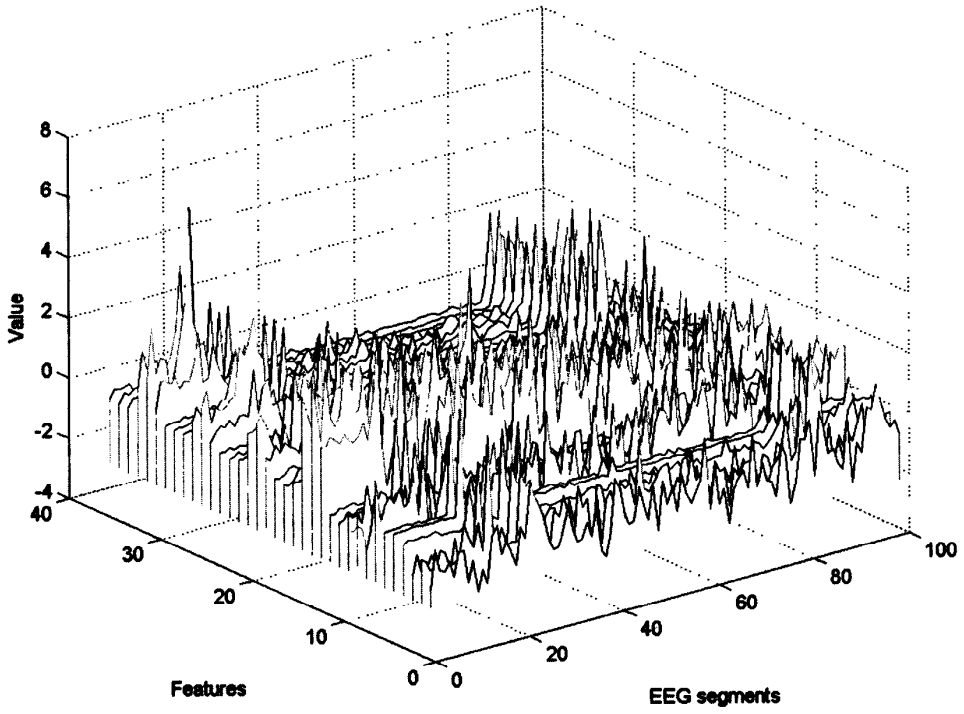


Figure 4. Fragment of EEG containing 100 segments presented by 36 features in which the EEG-viewer recognized five artifacts. See also color plates.

The patterns of EEG artifacts and normal segments can be visualized in a space of two principal components as depicted in Figure 5. Here artifacts and normal segments marked by the stars and the points, respectively.

Observing these patterns, we see that the artifacts are located far away from the normal segments and therefore the statistical characteristics of these patterns should be different. The labeled EEG segments were merged and divided into the training and testing subsets containing 2244 and 1210

randomly selected segments including 209 and 99 artifacts, respectively. We used one-third of the training data for validation and two-thirds for fitting the neuron weights.

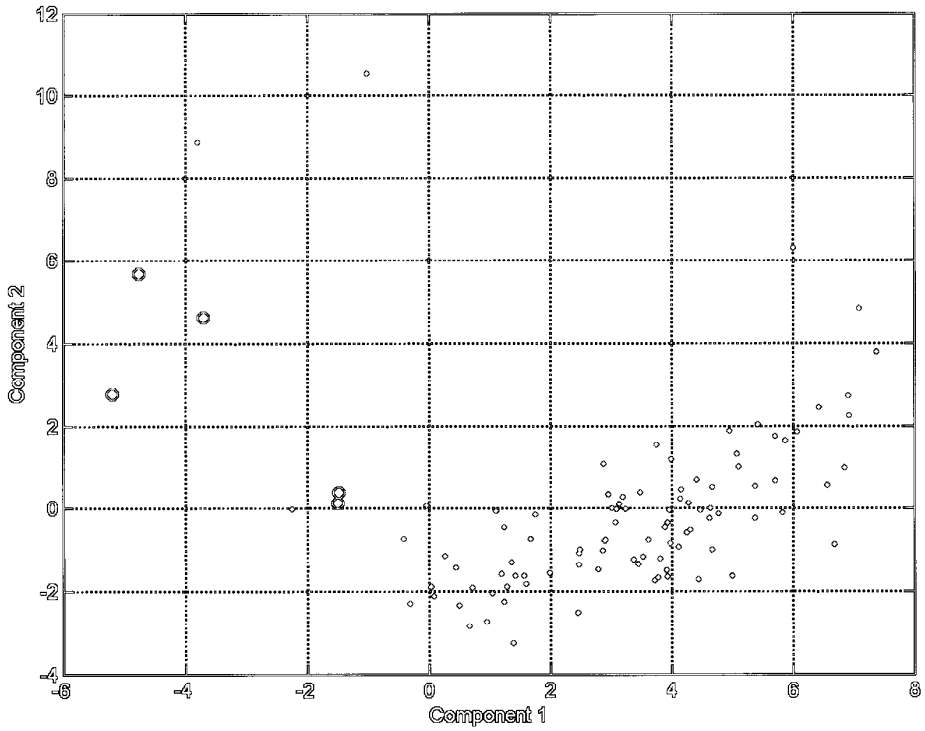


Figure 5. Pattern of EEG artifacts and normal segments in a space of two principal components. Here artifacts are marked by circles and normal segments by points.

Having trained 100 evolving cascade networks, we selected one which provides a minimal training error equal to 3.92%. This network misclassified 3.31% out of the testing examples.

Figure 6 depicts the structure of this network containing four input nodes, three hidden neurons and one output neuron. From the given 72 initial features, the training algorithm has selected only four features *AbsPowBeta2*, *AbsPowAlphaC4*, *AbsPowDeltaC3*, and *AbsVarDelta* that represent the absolute power of beta2 summed over C3 and C3, the absolute power of alpha in C4, the absolute power of delta in C3, and the absolute variance of delta summed over C3 and C4, respectively.

The inputs of the first neuron are connected to *AbsPowBeta2* and *AbsPowAlphaC4*. The output neuron is connected to *AbsPowBeta2*, *AbsPowAlphaC4* and *AbsVarDelta* as well as to the outputs of the hidden neurons, the hidden variables, z_1, z_2, and z_3.

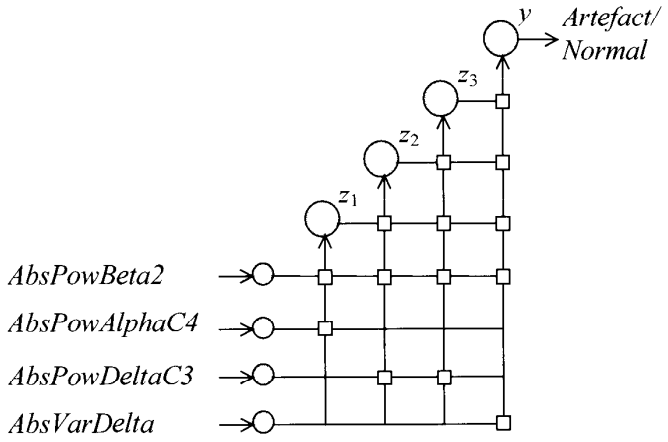


Figure 6. A cascade neural network trained for recognizing artifacts and normal segments in clinical EEGs. The squares represent synaptic connections.

EEG-expert observing this model can conclude the following. First, there are four features that make the most important contribution to the classification. These features are involved in the order of their significance – we can see that the most important feature is *AbsPowBeta2* and the less important is *AbsVarDelta*. So the most important contribution to the artifact recognition in EEG of sleeping newborns is made by *AbsPowBeta2* which is calculated for a high frequency band. This fact directly corresponds to a rule used for recognizing muscle artifacts in sleep EEG of adults [Brunner, Vasko, Detka, Monahan, Reynolds, & Kupfer, 1996].

Second, the discovered model shows the combinations between the selected features and hidden variables in the order of their classification accuracy. The EEG-expert can see that the maximal gain in the accuracy is achieved if the feature *AbsPowAlphaC4* is combined with *AbsPowBeta2*. Further improvement is achieved by combining the hidden variable z_1, which is a function of the above two features, and the new feature *AbsPowDeltaC3*. So the EEG expert can see the four combinations of the selected features and hidden variables z_1, ..., z_3 listed in the order of increasing classification accuracy, $p_1 < \ldots < p_4$, as follows

z_1: *AbsPowBet2* & *AbsPowAlphaC4* $\rightarrow p_1$,
z_2: z_1 & *AbsPowBet2* & *AbsPowDeltaC3* $\rightarrow p_2$,
z_3: z_2 & z_1 & *AbsPowBet2* & *AbsPowDeltaC3* $\rightarrow p_3$,
z_4: z_3 & z_2 & z_1 & *AbsPowBet2* & *AbsPowDelta* $\rightarrow p_4$,

where $z_4 = y$ is the outcome of the classification model.

The third useful issue is that the synaptic connections in the discovered model are characterized by the real-valued coefficients, which can be interpreted as the strength of relations between features and hidden variables.

The larger value of the coefficient, the stronger relation between the feature and hidden variable is.

In general, such models can assist EEG-experts to present the underlying casual relations between the features and outcomes in a visual form. The visualization of the discovered models can be useful for understanding the nature of EEG artifacts.

In our experiments, we compared the performance of the above classification model and an FNN trained on the same data. Using a sigmoid activation function and a standard neural-network technique, we found that a FNN with four hidden neurons and 11 input nodes provides a minimal training error. The training and testing errors were 2.97% and 5.54%, respectively.

Comparing the performances, we conclude that the discovered cascade network slightly outperforms the FNN on the testing EEG data. The improved performance is achieved because the cascade network is gradually built up by adding new hidden neurons and new connections. Each new neuron in the cascade network makes the most significant contribution to the artifact recognition among the all-possible combinations of the allowed number of features. This allows for avoiding the contribution of the noise features and discovering most significant relations which can then be visualized.

In this experiment, the FNN has misclassified more testing examples than the classification model described above. Therefore, we conclude that our cascade neural-network technique can more successfully recognize artifacts in clinical EEGs. At the same time the discovered classification model allows EEG-experts to present the basic relations between features and outcomes in visual form.

4. GMDH-TYPE NEURAL NETWORKS

In this section, we describe GMDH-type algorithms, which allow deriving polynomial neural networks from data. The derived networks generalize well because their size or complexity is near minimal. The derived networks are comprehensively described by concise sets of short-term polynomials (polynomials with few, simple terms), which are comprehensible for medical experts.

4.1 A GMDH Technique

GMDH-type neural networks are multi-layered, feed-forward networks consisting of the so-called *supporting neurons* [Farlow, 1984; Madala &

Ivakhnenko, 1994; Müller & Lemke, 2003]. The supporting neurons have at least two inputs v_1 and v_2. A transfer function g of these neurons may be described by short-term polynomials, for example, by a linear or non-linear polynomial:

$$y = g(v_1, v_2) = w_0 + w_1 v_1 + w_2 v_2, \tag{4}$$

$$y = g(v_1, v_2) = w_0 + w_1 v_1 + w_2 v_2 + w_3 v_1 v_2, \tag{5}$$

where w_0, w_1, w_2, ... are the polynomial coefficients or synaptic weights of the supporting neuron.

The idea behind GMDH-type algorithms is based on an evolution principle, which implies the *generation* and *selection* of the *candidate-neurons*. In the first layer, the neurons are connected to the input nodes, and in the second layer, they are connected to the previous neurons selected. For selecting the candidate-neurons, which provide the best classification accuracy, GMDH exploits the *exterior* criteria that are capable of evaluating the generalization ability of neurons on the validation data set.

The user must properly define the number F of the selected neurons providing the best classification accuracy. For example, the GMDH algorithm may combine the m input variables by 2 in order to generate the first, $r = 1$, layer of candidate-neurons $y_1^{(1)}, \ldots, y_{L1}^{(1)}$, where $L_1 = m(m-1)/2$ is the number of the neurons which is $0(m^2)$. The algorithm trains these candidate-neurons and then selects F best of them in order to generate the next layer. When generating the second layer, it is combined the outputs $y_1^{(1)}, \ldots, y_F^{(1)}$ of these F selected neurons. Here the best performance of the algorithm it is achieved for $F = 0.4L_1$ [Farlow, 1984; Madala & Ivakhnenko, 1994].

In Figure 7, we depict an example of a three-layer, GMDH-type network.

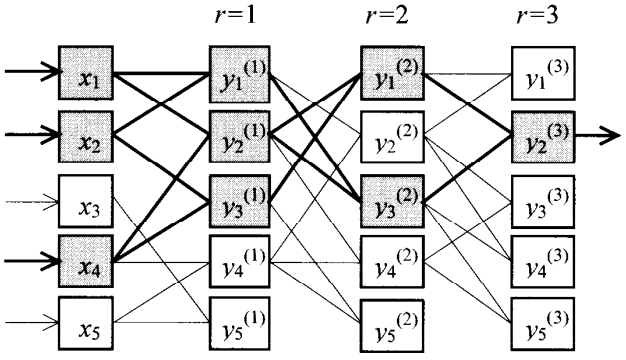


Figure 7. The structure of neural network grown by GMDH algorithm

The neuron-candidates that were selected for each of the layers are depicted here as the gray boxes. Here the neuron $y_2^{(3)}$ that provides the best classification accuracy and is assigned to be the output neuron. The resulting

polynomial network, as we can see, is the three-layer network consisting of six neurons and three input nodes. This network is described by a set of the following polynomials:

$$y_1^{(1)} = g_1(x_1, x_2),$$
$$y_2^{(1)} = g_2(x_1, x_4),$$
$$y_3^{(1)} = g_3(x_2, x_4),$$
$$y_2^{(3)} = g_4(y_1^{(2)}, y_3^{(2)}),$$
$$y_1^{(2)} = g_5(y_2^{(1)}, y_3^{(1)}),$$
$$y_3^{(2)} = g_6(y_1^{(1)}, y_2^{(1)}).$$

Thus, for the kth training example, we can calculate the output y of the neuron as

$$y = g(\mathbf{w}, \mathbf{v}^{(k)}),\ k = 1, \ldots, n,$$

where **w** is a weight vector, **v** is an input vector and n is the number of training examples.

For selecting F best neurons, the exterior criterion is calculated on the unseen examples of the validation set that have not been used for fitting the weights **w** of neurons. These examples are reserved by dividing the dataset **D** into two non-intersecting subsets $\mathbf{D}_A = (\mathbf{X}_A, \mathbf{y}_A^o)$ and $\mathbf{D}_B = (\mathbf{X}_B, \mathbf{y}_B^o)$, the training and validating data sets, respectively. The sizes n_A and n_B of these subsets is usually recommended to be defined with $n_A \approx n_B$, and $n_A + n_B = n$.

Let now find a weight vector **w*** that minimizes the sum square error e of the neuron calculated on the subset $\mathbf{D}_A$:

$$e = \Sigma_k (g(\mathbf{v}^{(k)}, \mathbf{w}) - y_k^o)^2,\ k = 1, \ldots, n_A.$$

To obtain the desirable vector **w***, the conventional GMDH fits the neuron weights to the subset $\mathbf{D}_A$ by using a Least Square Method (LSM) [Bishop, 1995; Farlow, 1984; Madala & Ivakhnenko, 1994], which can produce effective evaluations of weights with Gaussian distributed noise in the data. As noise in real-world data is often non-Gaussian [Duda & Hart, 2000; Tempo, Calafiore & Dabbene, 2003], we will use the learning algorithm described in Section 3, which does not require a hypothesis about the noise structure.

Having found a desirable weight vector **w*** on the subset $\mathbf{D}_A$, we can calculate the value CR_i of the exterior criterion on the validation subset $\mathbf{D}_B$:

$$CR_i = \Sigma_k (g_i(\mathbf{v}^{(k)}, \mathbf{w}^*) - y_k^{(0)})^2,\ k = 1, \ldots, n_B,\ i = 1, \ldots, L_r. \quad (6)$$

We can see that the calculated value of CR_i depends on the behavior of the ith neuron on the unseen examples of the subset $\mathbf{D}_B$. Therefore, we may expect that the value of CR calculated on the data **D** would be high for the neurons with poor generalization ability.

The values CR_i calculated for all the candidate-neurons at the rth layer are arranged in an ascending order:

$$CR_{i1} \le CR_{i2} \le \ldots \le CR_F \le \ldots \le CR_I,$$

so that the first F neurons provide the best classification accuracy.

For each layer r, it is found out a minimal value CR_m^r corresponding to the best neuron, i.e., $CR_m^r = CR_{i1}$. The first F best neurons are then used at the next, $r + 1$, layer, and the training and selection of the neurons are repeated.

The value of CR_m^r decreases step-by-step while the number of layers increases and the network is built up. Once the value of CR reaches to a minimal point and then starts to increase and we can conclude that the network has been over-fitted. Here because the minimum of CR was reached at the previous layer, we stop the training algorithm and take the desirable network, which was grown at the third layer.

4.2 A GMDH-Type Algorithm

The conventional GMDH-type algorithms perform an exhaustive search for candidate-neurons in each layer. The number of candidate neurons increases very fast with increasing the number m of inputs as well as with the number F of selected neurons. For both the first and following layers these numbers are $L_1 = 0(m^2)$ and $L_2 = 0(F^2)$. Below we describe the GMDH-type algorithm we developed and applied to derive the polynomial networks from data represented by $m > 70$ input features.

The idea behind this algorithm is to select the neurons one-by-one and add them to the network along with calculated probabilities. For selecting the neurons we use the exterior criterion described above.

In contrast to the conventional exhaustive search, the algorithm randomly selects a pair of the neurons by using a "*roulette-wheel*" in which the wheel area is divided into F sectors. The area of these sectors is proportional to the classification accuracy of the selected neurons on the training data. The neurons selected in the pair are then mated with a probability, which is proportional to their classification accuracy on the validating examples. When adding the new layer to the network, the algorithm attempts to improve the accuracy of the network for a specified number of times.

```
X = [x1, ..., xm]; % a pool of m input variables
k = 0; % the number of neurons in the network NN
% Train neurons with p inputs and calculate accuracy
p = 1;      % the number of inputs
```

```
for i = 1:m
  N1 = create-neuron(p, X(i));
  N1 = fit-weight(N1);
  A(i) = calc-accuracy(N1);
end
% Create new two-input neurons for gno attempts
p = 2;
for i = 1:gno
  pair = turn-roulette(p, A);
  N1 = create-neuron(p, X(pair));
  N1 = fit-weight(N1);
  ac = calc-accuracy(N1);
  % Selection and Addition
  if ac > max(A(pair))
    k := k + 1;
    NN(k) = add-new-neuron(N1);
    A(m + k) = ac;
  end
end
```

As a result, the variable NN contains description of the neural network. This network provides the best classification accuracy on the validating examples.

4.3 Classification of EEGs of Alzheimer and Healthy Patients

In our experiments, we used EEGs recorded from an Alzheimer patient and EEGs recorded from a healthy patient via the standard 19-channels C1, ..., C19 during 8 second intervals [Duke & Nayak, 2002]. Muscle artifacts were deleted from these data by an expert. We used the standard Fast Fourier Transform technique to calculate the spectral powers into four standard frequency bands: delta (0-4 Hz), theta (4-8 Hz), alpha (8-14 Hz) and beta (14-20 Hz).

As the spectral powers were calculated into half second segments with a quarter second overlapping, each EEG record consisted of 31 segments represented by 76 spectral features. The first 15 segments were used for training and the remaining 16 for testing, so the training and testing data consisted of 30 and 32 EEG segments respectively.

Exploiting the non-linear polynomial from equation (5) above and with $F = 1$, our algorithm derived a polynomial network consisting of four input nodes and three neurons. In Figure 8, we depict this network. As you can see, the derived classification rule is described by a set of three polynomials:

$$y_1^{(1)} = 0.6965 + 0.3916x_{11} + 0.2484x_{69} - 0.2312x_{11}x_{69},$$
$$y_1^{(2)} = 0.3863 + 0.5648y_1^{(1)} + 0.5418x_{73} - 0.4847y_1^{(1)}x_{73},$$
$$y_1^{(3)} = 0.1914 + 0.7763y_1^{(2)} + 0.2378x_{76} - 0.2042y_1^{(2)}x_{76}$$

where x_{11} is delta in C11, x_{69}, x_{73}, and x_{76} are beta in C12, C16, C19, respectively.

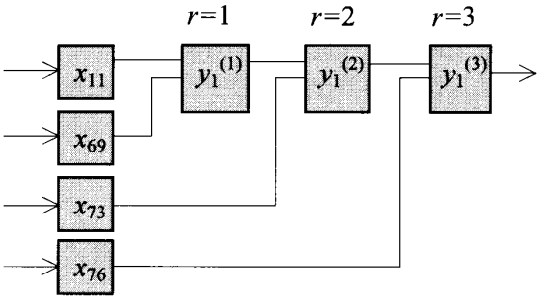


Figure 8. A polynomial network for classifying EEG of a Alzheimer and a healthy patient

Note that medical experts can interpret these polynomials as a weighted sum of two features. For example, polynomial $y_1^{(1)}$ is interpreted as a weighted sum of features x_{11} and x_{69}. The first two weights show the significance of these features for the polynomial output and the third weight shows the power of interaction between these two features.

Having applied the standard neural-network technique to these data, we found that a FNN, which consists of 8 input nodes and 2 hidden neurons, provides the best classification accuracy. We also applied a conventional GMDH-type technique to these data. All three neural networks misclassified one testing segment a testing error rate of 3.12%.

4.4 Recognition of EEG Artifacts

The EEGs used in our next experiments were recorded from two sleeping newborns. These EEGs were represented by 72 spectral and statistical features as described in [Breidbach et al., 1998] calculated in 10-second segments. For training, we used the EEG recorded from one newborn and for testing the EEG recorded from the other newborn. These EEGs consisted of 1347 and 808 examples in which an expert labeled respectively 88 and 71 segments as artifacts.

For comparison, we used the standard neural network and the conventional GMDH techniques. We found out that the best FNN consisted of 10 hidden neurons and misclassified 3.84% of the testing examples. The GMDH-type network was grown with an activation function, equation (5) above, with m = 72 inputs and F = 40. We ran our algorithm with the same

parameters and derived a polynomial network consisting of seven input nodes and 11 neurons as depicted in Figure 9.

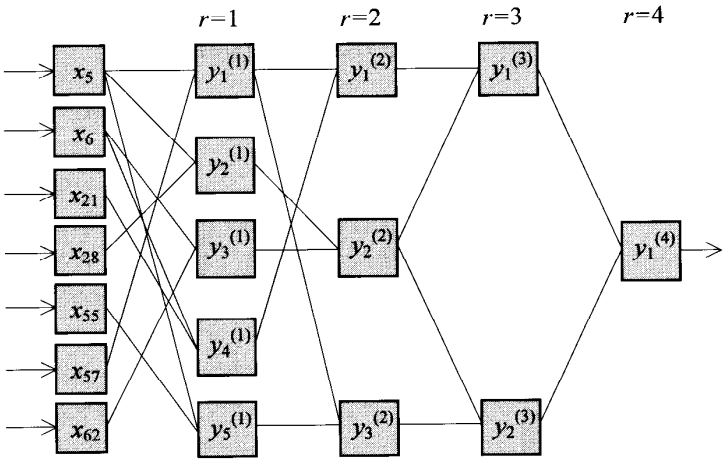


Figure 9. A polynomial network derived for recognizing EEG artifacts

This polynomial network misclassified 3.47% out of testing examples. This network is described by the following set of 11 short-term polynomials:

$$y_1^{(1)} = 0.9049 - 0.1707x_5 - 0.1616x_{57} + 0.0339x_5x_{57},$$
$$y_2^{(1)} = 0.9023 - 0.2128x_5 - 0.1389x_{28} + 0.0438x_5x_{28},$$
$$y_3^{(1)} = 0.9268 - 0.1828x_6 - 0.1195x_{62} + 0.0233x_6x_{62},$$
$$y_4^{(1)} = 0.9323 - 0.2057x_6 - 0.0461x_{21} + 0.0246x_6x_{21},$$
$$y_5^{(1)} = 0.9247 - 0.1822x_5 - 0.0951x_{55} + 0.0196x_5x_{55},$$
$$y_1^{(2)} = 0.0590 + 0.2810y_1^{(1)} + 0.3055y_4^{(1)} + 0.3670y_1^{(1)}y_4^{(1)},$$
$$y_2^{(2)} = 0.0225 + 0.4144y_2^{(1)} + 0.3812y_3^{(1)} + 0.1878y_2^{(1)}y_3^{(1)},$$
$$y_3^{(2)} = 0.0609 + 0.2917y_1^{(1)} + 0.2738y_5^{(1)} + 0.3880y_1^{(1)}y_5^{(1)},$$
$$y_1^{(3)} = 0.0551 + 0.3033y_1^{(2)} + 0.3896y_2^{(2)} + 0.2540y_1^{(2)}y_2^{(2)},$$
$$y_2^{(3)} = 0.0579 + 0.4058y_2^{(2)} + 0.2834y_3^{(2)} + 0.2549y_2^{(2)}y_3^{(2)},$$
$$y_1^{(4)} = -0.0400 + 0.6196y_1^{(3)} + 0.5702y_2^{(3)} - 0.1504y_1^{(3)}y_2^{(3)},$$

where x_5 is the absolute power of subdelta in C4, x_6 is the absolute power of subdelta, x_{21} is the real power of alpha, x_{28} is the absolute power of beta1 in C3, x_{55} is the absolute variance of theta in C4, x_{57} the is absolute variance of subdelta and x_{62} is the absolute variance of subdelta in C3.

Table 1 depicts the errors of the FNN, GMDH-type and polynomial neural networks (PNN) on the testing data. Note that both the FNN and the PNN were trained 100 times because their weights are initialized randomly. The conventional GMDH algorithm ran one time because it exploits the standard LSM technique of evaluating the synaptic weights.

Table 1. The classification errors of neural networks

	Error rate, %		
Data	***FNN***	***GMDH***	***PNN***
Train (patient 1)	2.00	2.06	2.23
Test (patient 2)	3.84	4.08	3.47

Observing the results listed in Table 1, we can conclude that the PNN trained by our method recognizes EEG artifacts slightly better than the FNN and GMDH-type network.

5. NEURAL-NETWORK DECISION TREES

In this section, we describe neural-network, decision-tree techniques, which exploit multivariate linear tests and algorithms searching for relevant features. The results of linear tests are easily visualized for medical experts. We also describe a new decision tree structure and an associated algorithm that is able to select the relevant features. This technique is shown to perform well on the large-scale clinical EEGs

5.1 Decision Trees

Decision tree (DT) methods have been successfully used for deriving multi-class concepts from real-world data represented by noisy features [Brodley & Utgoff, 1995; Duda & Hart, 2000; Quinlan, 1993; Salzberg, Delcher, Fasman & Henderson, 1998]. Experts find that results from a DT are easy to observe by tracing the route from its entry point to its outcome. This route may consist of the subsequence of questions which are useful for the classification and understandable for medical experts.

Conventional DTs consist of the *nodes* of two types. One is a *splitting* node containing a *test*, and other is a *leaf* node assigned to an appropriate class. A *branch* of the DT represents each possible outcome of the test. An example is presented to the *root* of the DT and follows the branches until the leaf node is reached. The name of the class at the leaf is the resulting classification.

A node can test one or more of the input variables. A DT is a *multivariate* or *oblique*, if its nodes test more than one of the features. Multivariate DTs are in general much shorter than those which test a single variable. These DTs can test Threshold Logical Units (TLU) or *perceptrons* that perform a weighted sum of the input variables. Medical experts can interpret such tests as a weighted sum of questions for example: Is 0.4 * *BloodPressure* + 0.2 *

HeartRate > 46? Weights here usually represent the significance of the feature for the test outcome.

To learn concepts presented by the numerical features [Duda & Hart ,2000], and [Salzberg et al. 1998] have suggested multivariate DTs which allow classifying *linearly separable patterns*. By definition such patterns are divided by linear tests. However using by these algorithms [Brodley & Utgoff, 1995; Frean, 1992; Parekh, et al., 2000; Salzberg et al., 1998], DTs can also learn to classify non-linearly separable examples.

In general, DT algorithms require computational time that grows proportionally to the number of training examples, input features, and classes. Nevertheless, the computational time, which is required to derive multi-class concepts from large-scale data sets, becomes overwhelming, especially, if the number of training examples is in the tens of thousands.

5.2 A Linear Machine

A Linear Machine (LM) is a set of r *linear discriminant* functions calculated to assign a training example to one of the $r \geq 2$ classes [Duda & Hart, 2000]. Each node of the LM tests a linear combination of m input variables $x_1, x_2, \ldots, x_m$ and $x_0 \equiv 1$.

Let us introduce a m-input vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_m)$ and a discriminant function $g(\boldsymbol{x})$. Then the linear test at the jth node has the following form:

$$g_j(\boldsymbol{x}) = \Sigma_i w_i^j x_i = \boldsymbol{w}^{jT}\boldsymbol{x} > 0, \quad i = 0, \ldots, m, \quad j = 1, \ldots, r, \tag{7}$$

where $w_0{}^j, \ldots, w_m{}^j$ are the real valued coefficients also known as a weight vector $\boldsymbol{w}^j$ of the jth TLU.

The LM assigns an example $\boldsymbol{x}$ to the j class if and only if the output of the jth node is larger than the outputs of the other nodes:

$$g_j(\boldsymbol{x}) > g_k(\boldsymbol{x}), \quad k \neq j = 1, \ldots, r. \tag{8}$$

This strategy of making a decision is known as Winner Take All (WTA).

While the LM is learning, the weight vectors $\boldsymbol{w}^j$ and $\boldsymbol{w}^k$ of the discriminant functions g_j and g_k are updated for each example $\boldsymbol{x}$ that the LM misclassifies. A learning rule increases the weights $\boldsymbol{w}^j$, where j is the class to which the example $\boldsymbol{x}$ actually belongs, and decreases the weights $\boldsymbol{w}^k$, where k is the class to which the LM has erroneously assigned the example $\boldsymbol{x}$. This is done using the following error correction rule:

$$\boldsymbol{w}^j := \boldsymbol{w}^j + c\boldsymbol{x}, \quad \boldsymbol{w}^k := \boldsymbol{w}^k - c\boldsymbol{x}, \tag{9}$$

where $c > 0$ is a given amount of correction.

If the training examples are linearly separable, the preceding procedure can yield a desirable LM giving maximal classification accuracy in a finite number of steps [Duda & Hart, 2000]. If the examples are non-linearly

separable, this training procedure may not provide predictable classification accuracy. For this case other training procedures have been suggested, we will discuss some of them below.

5.3 A Pocket Algorithm

To train the DT from data that are non-linearly separable, [Gallant, 1993] suggested a Pocket Algorithm. This algorithm seeks weights of multivariate tests that minimize the classification error. The Pocket Algorithm uses the error correction rule (9) above to update the weights $\boldsymbol{w}^j$ and $\boldsymbol{w}^k$ of the corresponding discriminant functions g_j and g_k. The algorithm saves in the Pocket the best weight vectors $\boldsymbol{W}^P$ that are seen during training.

In addition, Gallant has suggested the "ratchet" modification of the Pocket Algorithm. The idea behind this algorithm is to replace of the weight $\boldsymbol{W}^P$ by the current $\boldsymbol{W}$ only if the current LM has correctly classified more training examples than was achieved by $\boldsymbol{W}^P$. The modified algorithm finds the optimal weights if sufficient training time is allowed.

To implement this idea, the algorithm cycles training the LM for a given number of epochs, n_e. For each epoch, the algorithm counts the current number of input series of correctly classified examples, L, and evaluates the accuracy A of the LM on the training set.

In correspondence to inequality (8), the LM assigns a training example $(\boldsymbol{x}, q)$ to the jth class, where q is a class where the example $\boldsymbol{x}$ actually belongs. The LM training algorithm consists of the following steps:

```
W = init-weight();
[Wp, Lp, Ap] = set-pocket(W);
for i = 1:n % n is the number of training examples
  [x, q] = get-random(X);
  j = classify(x);
  if j ~= q
    Lp = 0;
    W(j) := W(j) + c*x;
    W(q) := W(q) - c*x;
  else
    if L > Lp
      A = calc-accuracy();
      if A > Ap
        % Update the pocket
        Wp = W;
        Lp = L;
        Ap = A;
      end
```

```
    end
  end
end
```

As the searching time that the algorithm requires grows proportional to the number of the training examples as well as of the input variables and classes, the number of epochs must be large enough to achieve acceptable classification accuracy. For example, in our case, the number of the epochs is set to the number of the training examples. The best classification accuracy of the LM is achieved if c is equal to 1.

When the training examples are not linearly separable, the classification accuracy of LMs may be unpredictable large. There are two cases when the behavior of the LM is destabilized during training. First, a misclassified example is far from the *hyperplane* dividing the classes. In such a case, the dividing hyperplane has to be substantially readjusted. Such relatively large adjustments destabilize the training procedure. Second, the misclassified example lies very close to the dividing hyperplane, and the weights do not converge.

To improve the convergence of the training algorithm, [Grean, 1992] has suggested a thermal procedure. This procedure decreases attention to the large errors by using the following correction

$$c = \beta/(\beta + k^2),\ k = (\boldsymbol{w}^{j} - \boldsymbol{w}^{(j)})^{\mathrm{T}}\boldsymbol{x}/(2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}) + \varepsilon,$$

where β is a parameter initialized to *2*, and $\varepsilon > 0.1$ is a given constant.

The parameter β is adjustable during training as follows. First, the magnitudes of the weight vectors are added. If the sum decreases for the current weight adjustment, but increased during the previous adjustment, the parameter β is reduced: $\beta = a\beta - b$, where a and b are given constants.

This reduction of β enables the algorithm to spend more time training the LM with small values of β that are needed to refine the location of the dividing hyperplane. However, experiments on the real-world classification problems show that the training time for the thermal procedure and the LM is comparable [Parekh et al., 2000].

5.4 Feature Selection Algorithms

In order to derive accurate and understandable DT models, we must eliminate the features that do not contribute to the classification accuracy of DT nodes. To eliminate irrelevant features, we use the Sequential Feature Selection (SFS) algorithms [Duda & Hart, 2000; Galant, 1993] based on a *greedy* heuristic, also called the *hill-climbing* strategy. The selection is performed while the DT nodes are trained from data. This avoids over-fitting more effectively than the standard methods of feature pre-processing.

The SFS algorithm exploits a *bottom up search* method and starts to learn using one feature. Then it iteratively adds the new feature providing the largest improvement in the classification accuracy of the linear test. The algorithm continues to add the features until a specified stopping criterion is met. During this process, the best linear test T_b with the minimum number of the features is stored. In general, the SFS algorithm consists of the following steps.

```
p = 1; % the number of features in the test
% Test the unit-variant tests T
for i = 1:m
  T(i) = test(p, X);
end
Tb = find-best-test(T);
while stop-rule(Tb, p)
  p := p + 1;
  T1 = find-best-test(p, T);
  % Compare the accuracies of T1 and Tb
  if T1.A > Tb.A
    Tb = T1;
  end
end
```

The stopping rule is satisfied when all the features have been involved in the test. In this case $m + (m - 1) + \ldots + (m - k)$ linear tests have been made, where k is the number of the steps. Clearly if the number of the features, m, as well as the number of the examples, n, is large, the computational time needed to terminate may be unacceptable.

To stop the search early and reduce the computational time, the following heuristic stopping criterion was suggested by [Parekhet et al., 2000]. They found that if at any step, the accuracy of the best test is decreased by more than 10%, then the chance of subsequently finding a better test with more features is slight.

However, the classification accuracy of the resulting linear test depends on the order in which the features have been included in the test. For the SFS algorithm, the order in which the features are added is determined by their contribution to the classification accuracy. As we know, the accuracy depends on the initial weights as well as on the sequence of the training examples selected randomly. For this reason the linear test can be non-optimal, i.e., the test can include more or fewer features than needed for the best classification accuracy. The chance of selecting the non-optimal linear test is high, because the algorithm compares the tests that differ by one feature only.

5.5 Derivation of Neural-Network Decision Trees

The idea behind our DT derivation algorithm is to individually train the test nodes and then group them in order to linearly approximate dividing hyperplanes. The DT test nodes, which are realized by TLUs, are individually trained to classify examples of two classes. For r classes, therefore, it is necessary to classify the $O(r^2)$ variants of the training subsets and train the same number of TLUs.

We can consider the trained TLUs, which deal with one class, as the hidden neurons of a neural network. The number of such networks is equal to the number of the classes, r. The contributions of these hidden neurons are summarized by the output TLU. Therefore each neural network makes a linear approximation to the dividing hyperplane between classes.

Let us introduce a TLU, $f_{i/j}$, performing the linear test (7) above, which learns to divide the examples of a pair of classes Ω_i and Ω_j. If the training examples of these classes are linearly separable, then the output y of the TLU is described as follows

$$\begin{aligned} y = f_{i/j}(\boldsymbol{x}) &= 1, \forall \boldsymbol{x} \in \Omega_i, \\ y = f_{i/j}(\boldsymbol{x}) &= -1, \forall \boldsymbol{x} \in \Omega_j. \end{aligned} \tag{10}$$

Indeed, medical experts can find that the features dividing two classes are simpler to observe than those dividing the multiple classes for $r > 2$. Fortunately, when the number of classes does not exceed several tens, such a *pairwise* approach can be efficiently applied to a multi-class problem by transforming it into a set of simple binary classifiers.

Having introduced the linear tests, now we can illustrate the idea of our derivation algorithm with a simple case of $r = 3$ classes. In Figure 10, we depict three classes Ω_1, Ω_2, and Ω_3, which hardly overlap each other. For this simple case, we need to train the $r\ (r - 1) / 2 = 3$ TLUs. The lines in Figure 10 depict the hyperplanes $f_{1/2}$, $f_{1/3}$, and $f_{2/3}$ of the TLUs trained to divide the classes Ω_1 and Ω_2, Ω_1 and Ω_3, as well as Ω_2 and Ω_3.

Also in Figure 10, we depict three new dividing hyperplanes g_1, g_2 and g_3. The first hyperplane, g_1, is a superposition of the linear tests $f_{1/2}$ and $f_{1/3}$, i.e., $g_1 = f_{1/2} + f_{1/3}$. The linear tests $f_{1/2}$ and $f_{1/3}$ here are summed with weights equal to 1, because both give us the positive outputs on the examples belonging to the class Ω_1. Correspondingly, the second and third dividing hyperplanes are $g_2 = f_{2/3} - f_{1/2}$ and $g_3 = -f_{1/3} - f_{2/3}$.

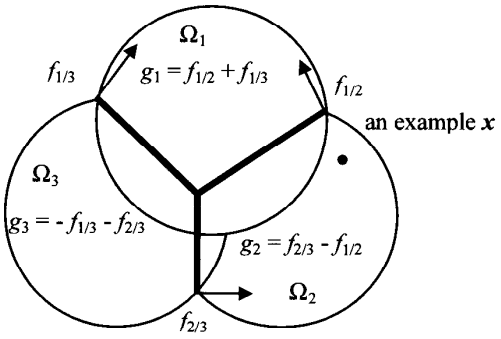


Figure 10. The approximation given by the dividing hyperplanes g_1, g_2 and g_3

We can see that an example $\boldsymbol{x}$ belonging to class Ω_2 causes the outputs of g_1, g_2 and g_3 to be equal to 0, 2, and –2, respectively:

$$g_1(\boldsymbol{x}) = f_{1/2}(\boldsymbol{x}) + f_{1/3}(\boldsymbol{x}) = 1 - 1 = 0,$$
$$g_2(\boldsymbol{x}) = f_{2/3}(\boldsymbol{x}) - f_{1/2}(\boldsymbol{x}) = 1 + 1 = 2,$$
$$g_3(\boldsymbol{x}) = -f_{1/3}(\boldsymbol{x}) - f_{2/3}(\boldsymbol{x}) = -1 - 1 = -2.$$

We can see that among g_1, g_2, and g_3, the second output is largest, $g_2 = 2$. Finally, the DT, using the WTA strategy, correctly assigns the example $\boldsymbol{x}$ to the class Ω_2.

For this case, the dividing hyperplanes g_1, g_2, and g_3 were approximated by $r = 3$ feed-forward neural networks consisting of the $(r - 1) = 2$ hidden TLUs. In Figure 11, we depict these networks in which hidden neurons perform the linear tests $f_{1/2}$, $f_{1/3}$, and $f_{2/3}$, respectively. The hidden neurons are connected to the output neurons g_1, g_2 and g_3 with the weights equal to (+1, +1), (–1, +1) and (–1, –1), respectively.

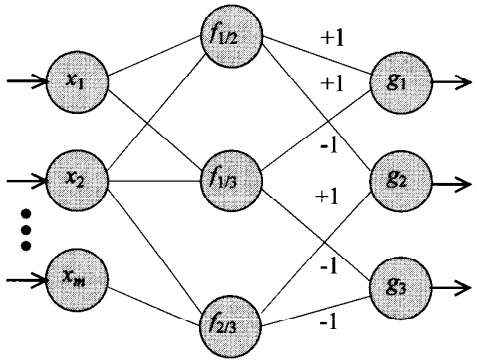


Figure 11. An example of the neural-network decision tree for r = 3 classes

In general for $r > 2$ classes, the neural network consists of $r(r - 1)/2$ hidden neurons $f_{1/2}, \ldots, f_{i/j}, \ldots, f_{r-1/r}$ and r output neurons $g_1, \ldots, g_r$, where $i < j, j = 2, \ldots, r$. The output neuron g_i is connected to $(r - 1)$ hidden neurons which are partitioned into two groups: the first group consists of the hidden neurons $f_{i/k}$ for which $k > i$, and the second group consists of the hidden

neurons $f_{k/i}$ for which $k < i$. The final step is to set up the weights of output neurons: each output neuron g_i is connected to the hidden neurons $f_{i/k}$ and $f_{k/i}$ with weights equal to +1 or -1.

As we see, each hidden neuron in the network learns to distinguish one class from another. The neurons learn independently of each other. However, the performance of the hidden neurons depends on the contribution of the input variables to the classification accuracy. For this reason, we next discuss a DT derivation algorithm which is able to select relevant features.

5.6 A Decision Tree Derivation Algorithm

The feature selection algorithm that we discussed in Section 5.4 searches for new features, which cause the largest increases of the classification accuracy of the linear tests. Recall that first this algorithm compares the tests that differ by one feature and that the algorithm then uses the greedy heuristic to select the new feature that provides the largest increase in the accuracy of the current test.

In our experiments, we have found that the comparison between the linear tests, which differ by more than one feature, increases the chance of accepting those tests, which improves the classification accuracy of the DT. We have also found that in real-world classification problems represented by noisy data, the greedy heuristic often finds a local minimum of the classification error. To increase the chance of escaping from local minima, we can evaluate the cross-validation classification error of linear tests. Using these heuristics, we developed the DT derivation algorithm shown below:

```
X = [x1, x2,..., xm];
% Test the unit-variant tests U
for i = 1:m
  U(i) = train-test(X(i));
  C(i) = calc-accuracy(U(i));
end
% Create the pools P and F
P = C/max(C);
[P, F] = sort-descend(P);
Tb = [];  % initialize
Ab = 0;
for k = 1:attempt-no
  T = [];
  i = 0;
  feature-no = 0,
  % Search for a candidate-test T
  while stop-rule(T, i)
```

```
    i := i + 1;
    if P(i) > rand(1)  % wheel of roulette
     T1 = [T X(F(i))]; % the features of test
     T1 = train-test(T1);
     A1 = calc-accuracy(T1);
     if A1 > A
      T := T1;
      A := A1;
      feature-no := feature-no + 1;
     end
    end
   end
   % Replace the best test Tb
   if A > Ab
    Ab := A;
    Tb := T;
   end
  end
```

To search for a best multivariate test, this algorithm exploits a strategy of evolution: it starts to train the single-variable tests including one feature x_i, $i = 1, \ldots, m$. Then, it calculates a probability p_i that is proportional to the accuracy of the ith test.

The calculated values of the probabilities are arranged in decreasing order, $p_{i1} \geq p_{i2} \geq \ldots \geq p_{im}$, and then they are placed in a pool P. Likewise the features $x_{i1}, x_{i2}, \ldots, x_{im}$ are placed in a pool F.

Next, the algorithm sets an empty array to the test, T, and 1 to the feature index, i. Then it attempts to add the feature x_i to T. If this occurs with calculated probability p_i, then a candidate-test T_1 is formed. The weights of this test are fitted to the training data, and then the classification accuracy A_1 of the test on the validation test is calculated.

If the accuracy A_1 becomes higher than the accuracy A of the current test T, then T is replaced by the candidate-test T_1. The number of features used in the new linear test T_1 increases by one.

The algorithm is repeated until a stopping criterion is met. This criterion is met in two cases: first, if the linear test T includes the given number N_f of the input variables, m, or second, if all the features have been tested.

Note that the algorithm compares the linear tests T and T_b which may differ by several features. This increases the chance of searching out a best linear test.

To increase the chance of locating the best solution, the linear tests are trained by the given number N_a attempts, each time with a different sequence of features. As a result, a unique set of the features is formed in the test T_b.

Using these features, the linear test classifies the training examples with the best classification accuracy A_b.

For fitting the DT linear tests, we used 2/3 of the training examples and evaluated the classification accuracy on all the training data. We varied the number of attempts N_a from 5 to 25.

5.7 Learning a Multi-Class Concept from the EEGs

Next we describe an application of the above DT algorithm for learning a multi-class concept from clinical EEG recordings. The EEGs were recorded from 65 sleeping patients via the standard EEG electrodes C3 and C4. These patients were healthy newborns with ages ranging between 35 and 51 weeks. The desired concept must distinguish the EEG recordings between these $r = 16$ age groups (classes).

Following [Breidbach, et al., 1998], the raw EEGs were segmented and transformed into 72 spectral and statistical features. Some of these features were redundant or irrelevant to the classification problem.

For training and testing the DT, we used 39399 and 19670 EEG segments respectively. For a given $r = 16$ classes, the DT included the $r\,(r-1)/2 = 120$ simple binary classifiers. The training errors of these classifiers varied between 0 and 15%, see Figure 12(a).

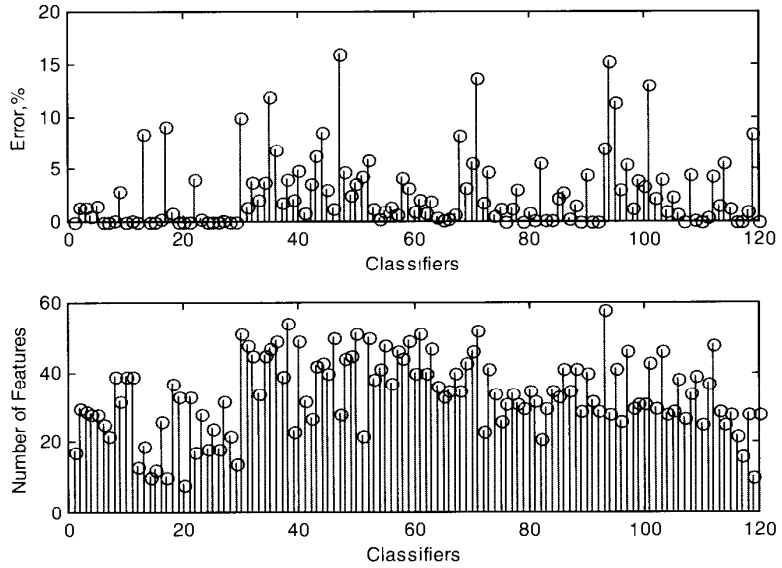


Figure 12. The training errors (a) and the number of features (b) for 120 binary classifiers

Note that the trained classifiers use different sets of the features (input variables). The number of these features varies from 7 to 58, see Figure 12(b).

The trained neural network DT correctly classified the 80.8% of the training and 80.1% of the testing examples. Summing all the segments belonging to one EEG recording, the trained DT correctly classified 89.2% and 87.7% of the 65 EEG recordings on the training and testing examples, respectively.

In Figure 13, we depict the distributions of the classified testing segments over all 16 classes for two patients belonging to the second and third age groups, respectively. Observing these distributions, we can give a probabilistic interpretation of the decisions. For example, we can decide that the patients belong to the second and third age groups with probabilities 0.92 and 0.58, respectively.

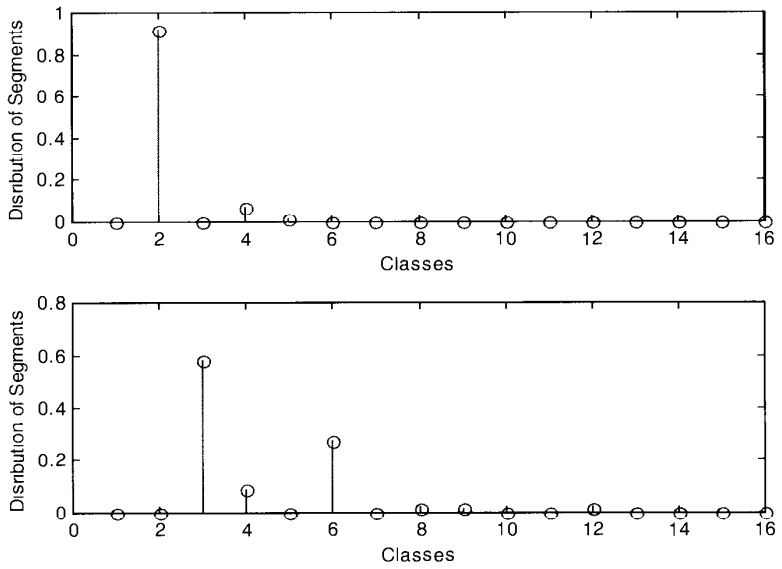


Figure 13. The distribution of the classified testing segments for two patients

We compared this DT technique with some data mining techniques on the same EEG data. First, we derived the LM described earlier. Second, we trained the feed-forward neural networks by using the standard back-propagation algorithm. The structures of the neural networks included from 8 to 20 input nodes and up to 20 hidden neurons. Third, we independently trained $r = 16$ binary classifiers to distinguish one class from the others. Fourth, we trained a binary decision tree consisting of $r - 1 = 15$ linear classifiers. However, in our experiments none of these standard techniques could achieve a desirable classification accuracy.

6. A RULE EXTRACTION TECHNIQUE

In some cases, the classification models trained by data from a neural network can be represented as decision tree rules [Avilo Garcez et al., 2001; Sethi & Yoo, 1997]. However in general this technique cannot guarantee that the resulting decision tree was not trapped in a local solution [Kovalerchuk & Vityaev, 2000]. In order to take this into account, we will next describe our technique developed to derive decision tree rules in *ad hoc* manner.

The idea behind our method is to project an original classification problem into an input space in which most of the training examples become separable. The dimensionality of such an input space can be significantly less than that of the original space. The neural-network techniques described in sections 3 and 4 are well suited for this role because they outperform standard neural networks.

Indeed, by removing the misclassified examples from the training data and eliminating noise and irrelevant features from the original feature set, we can significantly simplify the class boundaries and the solution of the classification problem. A decision tree derived from such data can be well suited for the classification of new observations.

To describe our technique, let us assume that the polynomial neural network performs well enough on the testing data. Next define the training subsets, X0 and X1, to consisting of n_0 and n_1 examples which have been correctly assigned by this network to the classes 0 and 1. These examples are represented in the new space of features, V, whose dimensionality is now equal to m. Then the DT derivation algorithm can be described as follows.

```
T = [];   % a decision tree T = ∅
V = 1:m;  % a pool V of features
find-node(X0, X1, V);
```

The procedure `find-node` is invoked with parameters X0, X1, and V. This procedure adds a new node to the decision tree T and then recursively calls itself as follows:

```
m = number-of-features(V);
% Search a threshold q_1 and an outcome p_1
for i = 1 to m do
  [q_1, p_1]= search-threshold-and-outcome();
end
[v_1, e_1]= find-feature-dividing X0 and X1;
f_1 = create-new-test();
T = [T, f_1]; % add new test to T;
% Calculate the outputs Y0 and Y1
Y0 = f_1(X0);
Y1 = f_1(X1);
```

```
V = remove-feature(v_i);
if V not empty
 % Find the examples A0, A10, A1 and A01:
  A0  = find(Y0 == 0);
  A01 = find(Y0 == 1);   % the errors of 0
  A1  = find(Y1 == 1);
  A10 = find(Y1 == 0);   % the errors of 1
  if A10 not empty
    find-node(X0(A0, V), X1(A10, V), V);
  end
  if A01 not empty
    find-node(X0(A01, V), X1(A1, V), V);
  end
end
```

We have used this algorithm to derive a decision tree for recognizing the artifacts in the clinical EEGs. First we trained the polynomial network described in section 4.4 from the training data which originally was represented by 72 features. Then we removed from these data all 30 misclassified examples and used the 7 discovered features to present the data in a new input space.

To derive a DT from the new data, the preceding algorithm was applied. This algorithm has derived a simple DT which exploits only one variable x_6, the absolute power of subdelta summed over channels C3 and C4, as depicted in Figure 14.

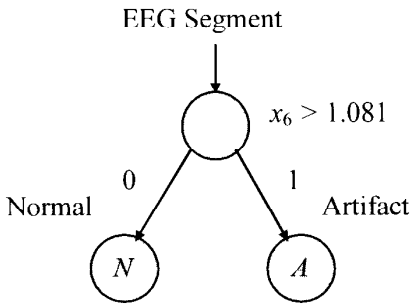


Figure 14. A decision tree rule for classifying the normal EEG segments and artifacts

Surprisingly, this decision tree has misclassified 24 testing examples while an original polynomial network misclassified 28. More experimental results can be found in the following paper devoted to the artifact recognition in the clinical EEGs [Schetinin & Schult, 2004].

Also we note that EEG experts can easily understand and interpret this decision tree as follows: an EEG segment is an artifact, if the value of absolute power of subdelta, x_6, is more than 1.081, otherwise, it is normal segment.

7. CONCLUSION

Standard neural networks can learn classification rules from real-world data well, however such classification models may not be comprehensible for experts. Classification models can become to be more understandable if they are represented in a visual form. To achieve such a representation, data mining techniques, based on a strategy of searching for a trade-off between complexity and accuracy of classification rules, are commonly used. In contrast to this strategy, the methods described in this chapter allow experts to present classification models in visual form and keep their classification error down.

We have presented examples of the application of standard techniques and our neural-network techniques to clinical EEG data for the extraction classification models which EEG-experts could easily represent visually. On testing data the new models performed slightly better than the standard feed-forward and GMDH-type networks. Thus, we conclude that our neural-network techniques can be successfully used for the visual data mining of clinical EEGs.

8. ACKNOWLEDGMENTS

The work has been supported by the University of Jena (Germany) and particularly by the University of Exeter (UK) under EPSRC Grant GR/R24357/01. The authors are personally grateful to Frank Pasemann for fruitful discussions, Joachim Frenzel and Burkhart Scheidt for the clinical EEG recordings we used in our experiments, to Richard Everson and Jonathan Fieldsend for useful comments.

9. EXERCISES AND PROBLEMS

1. Assume a fully connected neural network consists of 5 input nodes, 3 hidden and 2 output neurons. What is the minimum number of examples required to train this network by back-propagation? Why does user need to preset the structure of the neural network? What is changed in the neural network if the user applied PCA and determined two principle components?

2. Suppose a continuous exclusive OR (XOR) problem is described as follows

$$y = 1, \text{ if } x_1 x_2 > 0, \text{ and } y = 0, \text{ if } x_1 x_2 \leq 0,$$

where y are a target output, and $x_1 \in$ [-1, 1], $x_2 \in$ [-1, 1] are the input variables.

If the user uses a fully connected neural network, what structure has to be preset for this problem?

3. Assume an evolved cascade neural network which consists of 3 hidden neurons and 1 output neuron. How many examples are required to train this network? How many neurons are required to solve XOR problem?

4. Suppose a GMDH-type neural network uses a transfer polynomial

$$y = g(v_1, v_2) = w_0 + w_1 v_1 + w_2 v_2 + w_3 v_1 v_2 + w_4 v_1^2 + w_5 v_2^2,$$

where v_1 and v_2 are the input variables and $w_0, \ldots, w_5$ are the coefficients.

What minimal number of examples is required to train this network? When will GMDH-type neural networks out-perform fully connected neural networks and *vice versa*?

5. When and why will multivariate decision trees outperform decision trees which test single variables? Regarding the XOR problem (2.) above, which of these techniques is better?

6. Assume a 4-class problem. How many neurons are required to train linear machine? What should be the preset structure of a neural network based on pairwise classification for this case?

7. When and why will multi-class systems based on pairwise classification outperform the standard neural networks and decision trees?

10. REFERENCES

Avilo Garcez, Broda K., Gabbay D., Symbolic knowledge extraction from trained neural networks. Artificial Intelligence 2001; 125(1): 153-205.

Bishop C. M., Neural Network for Pattern Recognition. Oxford University Press, 1995.

Breidbach O., Holthausen K., Scheidt B., Frenzel J., Analysis of EEG data room in sud-den infant death risk patients. Theory Bioscience 1998; 117: 377-392.

Brodley C., Utgoff P., Multivariate decision trees. Machine Learning 1995; 19(11):45-77.

Brunner D., Vasko R., Detka C., Monahan J., Reynolds C., and Kupfer D., Muscle arti-facts in the sleep EEG: Automated detection and effect on all-night EEG power spectra. Journal of Sleep Research 1996; 5:155–164.

Duke D., Nayak K. The EEG data, Florida State University. Retrieved June 2002 from http://www.scri.fsu.edu/~nayak/chaos/data.html

Duda R.O., Hart P. E., Pattern Classification. Wiley Interscience, 2000.

Fahlman S.E., Lebiere C., The cascade-correlation learning architecture. In Advances in Neural Information Processing Systems II, ed. David Touretzky: Morgan Kaufmann Publishers Inc., 1990.

Farlow S., Self-organizing Methods in Modeling: GMDH-Type Algorithms. Marcel Dekker Inc., 1984.

Frean M.A., Thermal perceptron learning rule. Neural Computational 1992; 4: 946-957.

Galant S., Neural Network Learning and Expert Systems. MIT Press, 1993.

Iba H., deGaris, H., Sato T., Genetic programming with local hill-climbing. Proceedings of the Third International Conference on Parallel Problem Solving from Nature; 1994 October 9-14; Jerusalem; Berlin: Springer-Verlag, 1994.

Ishikawa M., Rule extraction by successive regularization. Neural Networks 2000; 13:1171-1183.

Kovalerchuk B., Vityaev E., Data Mining in Finance: Advances in Relational and Hybrid methods. Kluwer Academic Publishers, Boston, London, Dordrecht, 2000.

Madala H., Ivakhnenko A., Inductive Learning Algorithms for Complex Systems Modeling. CRC Press Inc., 1994.

Müller J.A., Lemke F., Self-Organizing Data Mining: Extracting Knowledge From Data. Canada: Trafford Publishing, 2003.

Morik K., Imhoff M., Brockhausen P., Joachims T., Gather U., Knowledge discovery and knowledge validation in intensive care. Artificial Intelligence in Medicine 2001; 19(3):225-49.

Parekh R., Yang J., Honavar V., Constructive Neural Network Learning Algorithms for Pattern Classification. IEEE Transactions on Neural Networks 2000; 11(2): 436-51.

Quinlan J., C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

Salzberg S., Delcher A., Fasman K., Henderson J., A decision tree system for finding genes in DNA. Computational Biology 1998; 5: 667-680.

Schetinin V, Schult J., A Combine Technique for Recognizing Artifacts in the Electroencephalograms of Sleeping Newborns. IEEE Information Technology in Biomedicine, to be published 2004.

Setiono R., Generating concise and accurate classification rules for breast cancer diagnosis. Artificial Intelligence in Medicine 2000; 18:205-219.

Sethi I., Yoo J., Structure-Driven Induction of Decision Tree Classifiers Through Neural Learning. Pattern Recognition 1997: 30(11):1893-1904.

Tempo R., Calafiore G., Dabbene F., Randomized Algorithms for Analysis and Control of Uncertain Systems. Springer Verlag 2003.

Towell G., Shavlik J., The extraction of refine rules from knowledge based neural networks. Machine Learning 1993; 13:71-101.

Chapter 15

VISUAL DATA MINING WITH SIMULTANEOUS RESCALING

Evgenii Vityaev[1] and Boris Kovalerchuk[2]
[1] *Institute of Mathematics, Russian Academy of Sciences, Russia*
[2] *Central Washington University, USA*

Abstract: Visualization is used in data mining for the visual presentation of already discovered patterns and for discovering new patterns visually. Success in both tasks depends on the ability of presenting abstract patterns as simple visual patterns. Getting simple visualizations for complex abstract patterns is an especially challenging problem. A new approach called inverse visualization (IV) is suggested for addressing the problem of visualizing complex patterns. The approach is based on specially designed data preprocessing. Preprocessing based on a transformation theorem is proved in this chapter. A mathematical formalism is derived from the Representative Measurement Theory. The possibility of solving inverse visualization tasks is illustrated on functional non-linear additive dependencies. The approach is called inverse visualization because it does not use data "as is" and does not follow the traditional sequence: discover pattern → visualize pattern. The new sequence is: convert data to a visualizable form → discover patterns with predefined visualization.

Key words: Visual data mining, simultaneous scaling, non-linear dependency, data pre processing, reverse visualization.

1. INTRODUCTION

Visual data mining is a growing area of research and applications [Keim, 2001; Fayyad, Grinstein & Wierse, 2001; Spence, 2001; Ware, 2000; Mille, 2001; Soukup & Davidson, 2002]. It includes two visually related tasks: the visual discovery of patterns and the visualization of discovered patterns in a specific form. This visual form should be *perceivable*, *understandable* and

interpretable in a domain. It is often formulated that the visualization should be simple - that is judged afterwards informally.

The most promising visual discovery approach is a *heterogeneous approach* that combines (a) analytical manipulation (AM) with data to transform them and (b) pure visual discovery (VD) by interactively observing transformed data.

In this chapter we attempt to formalize the concept of a *simple visualization* for data mining using ideas from classical physics, specifically from the theory of physical structures [Kulakov, 1971]. The next goal is to develop an AM technique that can provide a simple visualization. The chapter concludes with a simulation example showing the application of the AM technique on data.

Traditional *visualization* generally follows the sequence:

<patterns> → <visualization>.

In contrast, *visual discovery* reverses the sequence:

<visualization> → <patterns>.

That is, in visualization, we start from patterns and produce a visualization while in visual discovery, we start from visualization and produce patterns.

Thus, we call **visual discovery** an **Inverse Visualization Task (IVT)** with the goal to *find data transformations that permit the generation of a simple, clear visualization of data and patterns.* Success in this endeavor depends on the properties of the data transformations and the data mining methods involved. Many data mining practitioners share the opinion that practically any data mining method will discover meaningful patterns for "good" data while few if any will produce meaningful patterns for "poor" data.

One of the goals of IVT is to transform "poor" data into "good" data thus permitting a wide variety of data mining methods to be used for the successful discovery of hidden patterns.

For now, we will not attempt to define formally "poor" and "good" data. Rather, we will show that in classical physics such transformations have been used successfully for a long time to discover patterns, which are now classical (fundamental) physical laws, without formal definitions of "good" data. We note that the laws of classical physics are simple so the problem of their visualization is not difficult.

However, the lessons learned from classical physics can help in other domains where patterns do not appear to be simple, but first we need to understand the reasons for the simplicity of laws in classical physics. Are these reasons specific to physics or can they be exploited for domains such

reasons specific to physics or can they be exploited for domains such as finance, medicine, remote sensing, and image analysis?

An explanation of simplicity in physics follows from two theories: the Representative Measurement Theory [Krantz, Luce, Suppes & Tversky 1990] and the Physical Structures Theory [Kulakov, 1971; Mikhailichenko, 1985]. Measurement theory [Krantz et al., 1990, v.1] demonstrates that a system of physical quantities and fundamental laws will have a simple representation because they are obtained through a procedure that *simultaneously scales* the variables involved in the laws.

Traditionally data mining *does not involve simultaneous scaling.* Note that simultaneous scaling is different from the data *normalization* procedures used in neural networks to speed up search, see for example [Rao & Rao, 1995]. The typical normalization in neural networks transforms the scale of each variable *independently* and non-linearly to some interval, such as [-1,1]. On the other hand, simultaneous scaling of variables x, y and z might transform these variables into new scales x', y' and z' so that the law has the simple linear form, perhaps $y' = x'+z'$. In general, *laws of classical physics show that if all variables included in a law are scaled simultaneously then the law can assume a relatively simple form.*

The problem of finding efficient, simultaneous scaling transformations was not posed and solved by Representative Measurement Theory. This theory explains the simplicity effect but lacks a constructive way to achieve it. On the other hand, Representative Measurement Theory has wider area of application than physics only. For instance, psychology has benefited significantly from it [Krantz et al., 1971]. This observation raises a hope that simultaneous scaling will be beneficial in other areas too. This, of course, requires designing simultaneous scaling transformations.

Fortunately, the theory of Physical Structures provides an answer for this problem via the *constructive classification* of all functional expressions of all possible fundamental physical laws [Mikhailichenko, 1985]. Classes defined by this classification have an important property -- any other functional expressions of a physical law can be transformed to one of the given classes by a monotone transformation of all involved variables.

The procedure for deriving such transformation is the *simultaneous scaling* of these variables. This result shows, that every physical law can be described as class of expressions that can be converted to each other by monotone transformations of the variables contained in the law. This means that all laws can be enumerated in the classification from of all functional expressions of all possible fundamental physical laws [Mikhailichenko, 1985]. All laws of this classification have simple form and by extension, the problem of their visualization is simple too. All **complexity of visualization** of

these laws is thus converted into the design of a **monotone transformation** of the n-tuples of variables involved.

2. DEFINITIONS

Let us define a class of functions F, which can be transformed to the linear function $y = x + z$ by monotone transformations. There are many possible ways to define the class F. It is convenient to assume that F is given through a set of axioms. Suppose that a dataset D from a specific domain (e.g., finance) represents a set of triples x, y, and z where $y = f(x, z)$ and the function f is not known analytically. Suppose that the function f is known only through tabulated values from D and possibly some meaningful (for the domain) properties in the form of axioms. We assume that:

- real-world variables x, y and z are mapped to real numbers $\boldsymbol{R}$ by some measurement procedures,
- the order relation on $\boldsymbol{R}$ is not just a numeric relation but it has interpretation as a real-world relation for the variable y, and
- in the same way the equality relation on $\boldsymbol{R}$ is interpreted for variables x and z.

We define the class F of functions $f \in F$ on $X_f \times Z_f$, where $X_f \subset \boldsymbol{R}$, $X_f \neq \varnothing$, and $Z_f \subset \boldsymbol{R}$, $Z_f \neq \varnothing$. Functions from F satisfy the five properties of *additive conjoint structure* [Krantz et al., 1971, p.256]:

(1). $\forall z_1, z_2, \exists x \ (f(x, z_1) \geq f(x, z_2) \Rightarrow \forall x' (f(x', z_1) \geq f(x', z_2)))$

(2). $\forall x_1, x_2, x_3, z_1, z_2, z_3$
$((f(x_1, z_2) = f(x_2, z_1)) \ \& \ (f(x_1, z_3) = f(x_3, z_1)) \Rightarrow (f(x_2, z_3) = f(x_3, z_2))$

(3). For any three of x_1, x_2, z_1, z_2 the fourth of them exists such that
$f(x_1, z_2) = f(x_2, z_1)$

(4). $\exists x_1, x_2, z \ (f(x_1, z) \neq f(x_2, z))$

(5). For any $z_1, z_2 : z_1 \neq z_2$, if a sequence $x_1, x_2, \ldots, x_i, \ldots$ of elements of X_f is determined and satisfies the following properties: $\forall i, x_i < x_{max}$
$f(x_1, z_1) = f(x_2, z_2), f(x_2, z_1) = f(x_3, z_2), f(x_3, z_1) = f(x_4, z_2), \ldots,$
$f(x_i, z_1) = f(x_{(i+1)}, z_2), \ldots$
then this sequence is finite.

In addition properties (2) and (3) should also take place with x replaced by z and vice versa.

3. THEOREM ON SIMULTANEOUS SCALING

The theorem below is based on axioms (1)-(5) and is used for design of a simultaneous scaling procedure.

Theorem [Krantz et al., 1971, p.257]:

1. For any function $f \in F$ there are one-to-one functions φ_x, φ_z and a monotone function φ such that

$$\varphi f(x, z) = \varphi_x(x) + \varphi_z(z), \quad \langle x, z \rangle \in X_f \times Z_f.$$

2. If $\varphi`(x)$, $\varphi`(z)$ are two other functions with the same property, then there exist constants $\alpha > 0$, β_1, and β_2, such that

$$\varphi`_x(x) = \alpha\varphi_x(x) + \beta_1, \quad \varphi_z(z) = \alpha\varphi`_z(z) + \beta_2$$

$$f'(x', z') = \varphi f(\varphi_x(x'), \varphi_z(z'))$$

where φ is a strictly monotone function, and φ_x, φ_z are one-to-one functions from F.

Proof [Krantz et al., 1971, p.264-266]. The proof follows from the fact, that the set axioms (1) – (5) represents an additive conjoint structure.

Let function $f \in F$ on $X_f \times Z_f$ satisfy the axioms (1)-(5). By virtue of the axiom (4) there are points on the plane $\langle x_0, z_0 \rangle$, and $\langle x_1, z_0 \rangle$ such that

$$f(x_0, z_0) \neq f(x_1, z_0)$$

(see Figure 1).

Rescaling algorithm. Let's simultaneously scale X, Z, and Y ($y = f(x, z)$) as follows:

- assign value 0 to x_0 of the scale X; record it as $x_0 = 0$;
- assign value 1 to x_1;
- assign value 0 to z_0 of scale Z;
- assign values $f(x_0, z_0) = 0$ and $f(x_1, z_0) = 1$ for function f.

By virtue of the axiom (3) for three elements x_0, z_0, x_1 there exists fourth z_1, such that

$$f(x_0, z_1) = f(x_1, z_0).$$

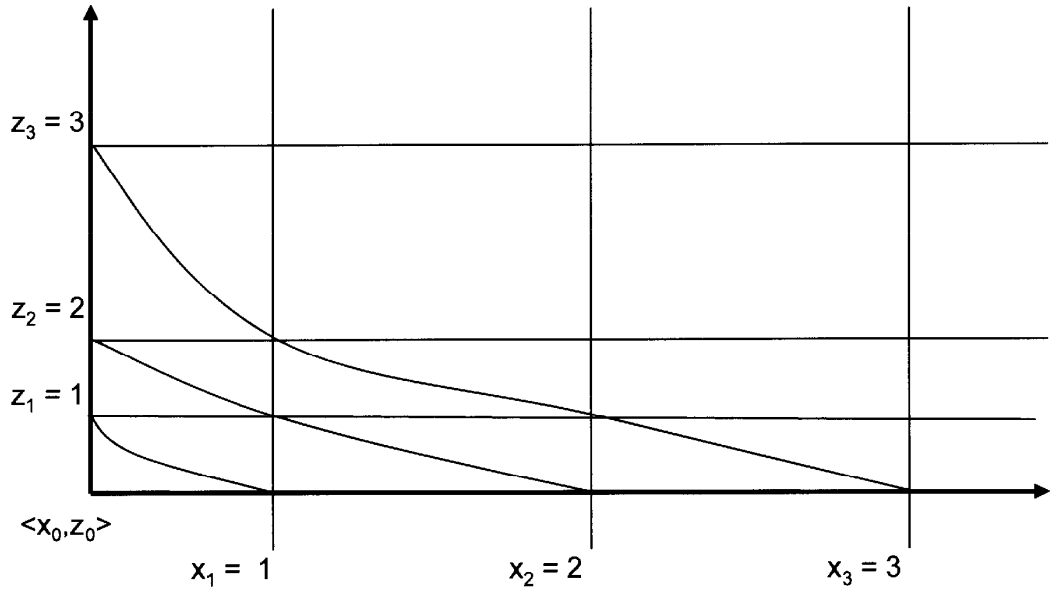


Figure 1. Simultaneous rescaling process

Let us link the points $\langle x_0, z_1\rangle$, $\langle x_1, z_0\rangle$ as shown in Figure 1. Along this line the function has identical values. These values are the values of Y scale (which is not shown on the picture). It is easy to see, that these values of x, z, and y satisfy the function $x + z = y$. We take a point $\langle x_1, z_1\rangle$ and assign value $y = f(x_1, z_1) = 2$ for this point.

Next we again apply the axiom (3). At first we apply it to values x_1, x_0, z_1 and receive x_2 such that $f(x_1, z_1) = f(x_2, z_0)$ and then we apply it to values x_1, x_0, z_1 and receive z_2, such that $f(x_0, z_2) = f(x_1, z_1)$. After that we assign value $y = f(x_0, z_2) = f(x_1, z_1) = f(x_2, z_0) = 2$. Now we consider new points $\langle x_2, z_1\rangle$ and $\langle x_1, z_2\rangle$.

To make the given construction possible for all new points $\langle x_0, z_3\rangle$, $\langle x_3, z_0\rangle$ it is necessary, that the values of the function would be identical $f(x_2, z_1) = f(x_1, z_2)$ for points $\langle x_2, z_1\rangle$ and $\langle x_1, z_2\rangle$. The equality $f(x_2, z_1) = f(x_1, z_2)$ follows from the axiom 2.

Figures 2 and 3 present such transformation. The surface in Figure 2 is transformed to the surface in Figure 3 by the simultaneous rescaling of variables x, z, and y. It follows from the theorem, that if properties (1)-(5) take place for some variables x, y, z, then the function $f \in F$ can be converted to function $y = x + z$ by rescaling variables. After this the visualization of rescaled data and function $y = x + z$ is obvious (see Figure 3).

The rescaling algorithm requires that values of a function f on specific pairs of values $\langle x, z\rangle$ satisfy properties (1)-(5) of the theorem. These properties are true for preference relations used in Decision Theory [Keeney & Raiffa, 1976], but this is not a universally true condition for other tasks.

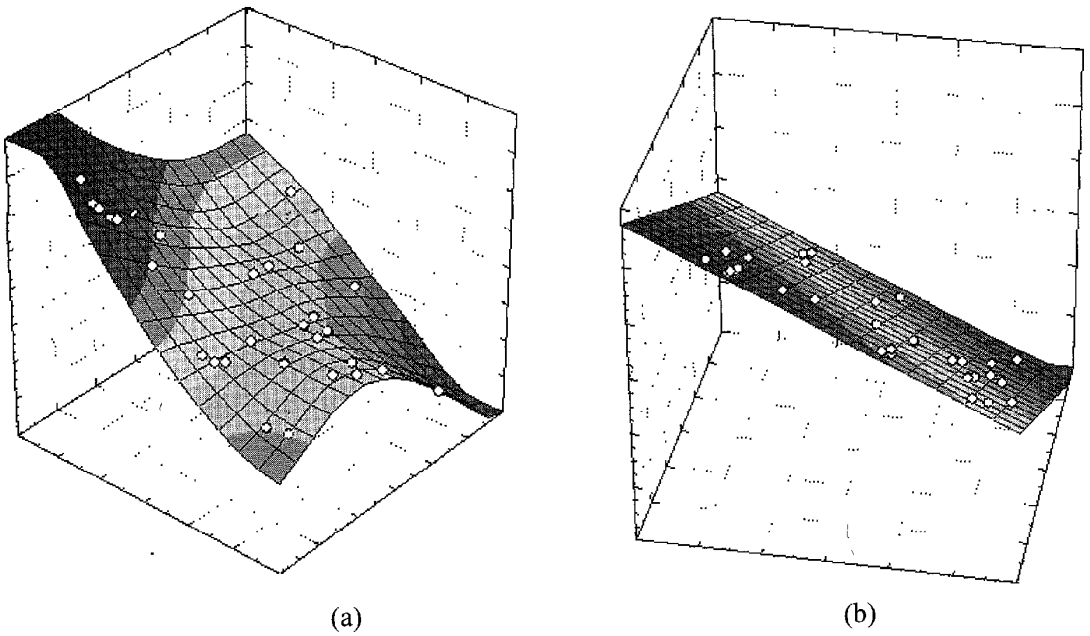


Figure 2. Data visualization: (a) original data, (b) simultaneously rescaled data. See also color plates.

4. A TEST EXAMPLE

A test example must satisfy several requirements to be really convincing:

1. It should contain regularities (patterns) known in advance;
2. These regularities should have at least a hypothetically meaningful interpretation in the domain;
3. These regularities should not be obvious when data is prescreened and visualized prior to rescaling as in Figure 2 (a).

Table 1, shown in section 5 below, contains data to meet these requirements. It is produced in the way described below:

- Attributes a_1, a_2, a_3 and a_5, a_6, a_7, a_8, a_9 are created by using a random number generator. For instance attributes a_1, a_2, a_3 could model some basic independent indicators of product manufacturing.
- Attribute a_4 is a sum of the first two attributes, $a_4 = a_1 + a_2$.
- Attribute a_{10} is a target attribute, it is equal to some random monotone transformation F of the difference $a_4 - a_3$, i.e., $a_{10} = F(a_4 - a_3) = F(a_1 + a_2 - a_3)$.

These attributes may have different *interpretations*. In one of them attributes a_5, a_6, a_7, a_8 and a_9 represent noise. They are random and unrelated to the target attribute a_{10}. A hypothetical interpretation of the regularity $F(a_1 + a_2 - a_3)$ could be productivity or production efficiency or revenue. Attribute a_1 may indicate initial capital (scaled from 0 to 10), attribute a_2 may indicate

quality of management (also scaled from 0 to 10) and attribute a_3 could be a tax level (scaled from 0 to 10). Attributes a_1 and a_2 contribute positively to revenue while attribute a_3 contributes negatively.

A relatively complex monotone transformation is motivated by an intention:

- to solve a **realistic task**. In real-world tasks, if there are any hidden patterns (regularities), they are usually disguised and significantly corrupted. Experience shows that monotone regularities are common for many data mining tasks.
- to show **unique capabilities** of the simultaneous scaling method. Traditional methods that do not use simultaneous scaling can not discover a regularity corrupted by a random monotone transformation. The only way to do this is to analyze all interpretable order relations $\leq_1, \leq_2, \leq_3, \ldots, \leq_{10}$ for all attributes. These regularities are revealed by the simultaneous scaling method.
- to find **meaningful patterns**, regularities, and functions. For instance, typically regression analysis produces functions that just interpolate data without meaningful interpretation in the domain. In contrast the simultaneous scaling method produces meaningful regularities such as

$$\forall\, a, b\ (a \leq_1 b\ \&\ a \leq_2 b \Rightarrow a \leq_{10} b)$$

The data in Table 1 encodes the following regularities by design:

$$\begin{aligned}
&\forall\, a, b\ (a >_3 b\ \&\ a \leq_4 b \Rightarrow a \leq_{10} b)\\
&\forall\, a, b\ (a \leq_3 b\ \&\ a >_4 b \Rightarrow a >_{10} b)\\
&\forall\, a, b\ (a \leq_1 b\ \&\ a \leq_2 b\ \&\ a >_3 b \Rightarrow a \leq_{10} b)\\
&\forall\, a, b\ (a >_1 b\ \&\ a >_2 b\ \&\ a \leq_3 b \Rightarrow a >_{10} b)
\end{aligned} \qquad (1)$$

5. DISCOVERING SIMULTANEOUS SCALING

The Discovery System [Kovalerchuk & Vityaev, 2000] can discover all monotone regularities including those shown in (1) above and are actually encoded in Table 1 along with random noise. When regularities (1) are discovered, a simultaneous monotone rescaling of the data can be arranged and the straightforward and simple visualization presented in Figure 3 below will be generated.

Thus the major challenge is discovering the monotone regularities. The Discovery System searches sequentially for monotone regularities starting from simplest ones:

$$\forall\, a, b\ (a \leq_i b \Rightarrow a >_{10} b),\ i = 1, \ldots, 9. \qquad (2)$$

Table 1. Test data

#	1	2	3	4	5	6	7	8	9	10	#	1	2	3	4	5	6	7	8	9	10
1	8	2	8	10	5	9	0	4	1	53	*41*	1	4	3	5	7	9	6	6	7	53
2	6	4	0	10	1	1	8	1	5	79	*42*	3	8	9	11	5	6	2	4	2	53
3	4	1	3	5	0	3	8	7	1	53	*43*	9	1	5	10	1	8	0	6	3	61
4	8	0	9	8	9	1	5	0	0	30	*44*	8	6	6	14	6	2	6	7	2	72
5	7	8	4	15	5	0	2	8	6	83	*45*	0	8	9	8	9	7	2	1	1	30
6	9	1	8	10	9	7	0	3	4	53	*46*	4	4	3	8	9	2	8	2	5	61
7	9	5	5	14	3	3	9	7	7	75	*47*	9	7	8	16	4	4	5	0	9	72
8	5	8	2	13	7	1	6	6	8	83	*48*	6	4	7	10	4	7	4	9	6	56
9	1	9	4	10	0	1	4	7	1	64	*49*	5	2	4	7	7	7	7	7	3	56
10	1	2	3	3	4	0	3	2	7	31	*50*	2	2	9	4	3	6	2	5	2	12
11	7	5	5	12	6	8	0	2	4	66	*51*	1	7	8	8	5	0	6	5	6	31
12	2	1	6	3	3	8	1	7	2	17	*52*	9	1	8	10	3	6	5	5	4	53
13	4	6	5	10	6	6	7	5	4	61	*53*	8	1	5	9	5	2	7	3	1	58
14	1	5	8	6	4	9	0	9	5	24	*54*	2	7	7	9	8	0	4	8	9	53
15	7	3	9	10	1	4	5	8	2	51	*55*	3	5	6	8	5	6	2	4	2	53
16	2	6	9	8	9	5	3	5	2	30	*56*	5	5	1	10	8	7	9	1	3	75
17	1	8	9	9	6	6	3	8	2	31	*57*	3	5	7	8	8	3	3	6	7	51
18	9	3	7	12	3	9	1	2	1	61	*58*	4	5	4	9	8	6	4	8	1	61
19	0	4	4	4	4	2	0	3	4	31	*59*	2	9	4	11	4	7	4	0	1	66
20	9	1	5	10	5	4	2	6	1	61	*60*	4	1	9	5	9	0	2	8	5	14
21	8	7	6	15	1	2	7	1	1	75	*61*	9	1	5	10	4	0	7	9	2	61
22	7	9	6	16	9	6	7	3	1	79	*62*	5	8	4	13	8	6	2	5	8	75
23	8	9	2	17	1	5	6	3	9	96	*63*	5	8	6	13	6	0	5	9	9	66
24	8	2	7	10	9	4	1	1	6	56	*64*	9	2	5	11	2	1	9	7	2	64
25	0	1	7	1	6	9	0	6	6	9	*65*	5	8	8	13	3	6	2	1	7	61
26	7	7	1	14	5	0	5	2	5	91	*66*	3	7	9	10	2	1	6	4	7	51
27	2	9	7	11	2	1	1	5	3	58	*67*	8	4	8	12	8	2	4	9	0	58
28	5	5	1	10	2	4	5	8	8	75	*68*	7	0	8	7	6	0	8	1	5	30
29	1	7	0	8	6	3	6	9	7	72	*69*	4	1	6	5	9	2	8	9	7	30
30	4	6	0	10	0	9	8	2	7	79	*70*	8	6	8	16	4	8	8	2	8	72
31	0	6	1	6	3	7	8	6	1	61	*71*	1	5	5	6	2	3	7	6	8	51
32	8	4	4	12	2	6	6	4	6	72	*72*	1	3	9	4	2	4	5	6	4	12
33	3	4	6	7	7	2	1	7	6	51	*73*	7	3	0	10	6	3	7	6	6	79
34	2	7	8	9	3	3	8	4	4	51	*74*	9	0	4	9	3	8	0	0	5	61
35	6	5	9	11	7	2	4	5	5	53	*75*	5	7	4	12	1	6	8	6	8	72
36	9	3	9	12	8	5	8	6	0	56	*76*	5	1	9	6	1	4	8	8	4	17
37	1	6	2	7	0	7	6	0	8	61	*77*	3	0	5	3	7	6	5	4	1	24
38	6	0	0	6	0	2	3	7	5	64	*78*	7	7	7	14	7	9	7	5	7	66
39	2	7	9	9	0	1	6	9	8	31	*79*	0	2	9	2	5	8	5	1	8	5
40	4	8	2	12	0	9	5	1	5	79	*80*	3	4	4	7	0	9	9	7	8	56

After testing them we discover regularity

$$\forall\, a, b\ (a \leq_4 b \Rightarrow a \leq_{10} b).$$

with a statistical confidence level equal to 0.0001. This regularity is not in the list of regularities (1) above, although it is true for data from Table 1. Next the systems tests more complex regularities:

$$\forall\, a, b\ (a >_i b\ \&\ a \leq_j b \Rightarrow a \leq_{10} b)\ \ i, j = 1, ..., 9$$

and finds a regularity

$$\forall\, a, b\ (a >_3 b\ \&\ a \leq_4 b \Rightarrow a \leq_{10} b)$$

with a statistical confidence level equal to 0.025.

Similarly, another parametric set of hypothetical regularities is generated and tested to discover the second regularity in (1). Next we can discover a regularity with all three variables in the antecedent if we substitute given attributes with parameters *i, j* and *k* that are the indexes of attributes. For instance, for discovering

$$\forall\, a, b\ (a \leq_1 b\ \&\ a \leq_2 b\ \&\ a >_3 b \Rightarrow a \leq_{10} b),$$

we generate a parametric set

$$\forall\, a, b\ (a \leq_i b\ \&\ a \leq_j b\ \&\ a >_k b \Rightarrow a \leq_{10} b)\ i, j, k = 1, ..., 9$$

and test it. The test reveals the needed regularity with a confidence level equal to 0.1.

6. ADDITIVE STRUCTURES IN DECISION MAKING

In section 2, we assumed an *additive conjoint structure* that permitted us to build a simple linear visualization. In this section, we discuss the motivation of using an additive structure from a decision-making viewpoint invoking an approach presented in [Keeney & Raiffa, 1976].

A decision-making problem is considered as a tradeoff between contradictory goals such as maximizing profit and minimizing risk. The tradeoff means that we try to substitute a chunk Δ_1 of goal G_1 that we cannot satisfy with a chunk Δ_2 of another goal G_2 that we can satisfy. The tradeoff assump-

tion must be such that substitution makes sense. This is, in essence, leading us to an additive conjoint structure assumption. Under this widely accepted assumption, the practical issue is finding the chunks Δ_1 and Δ_2.

One of the options that can be used to solve this problem is the *explicit way* where a SME (subject matter expert) declares, say, that $\Delta_1=3$ and $\Delta_2=5$ are equivalent for substitution purposes, that is SME formalizes preferences as a model. This is typically a very difficult task. Another approach is the *implicit* approach. In this approach, we just ask a SME to define preferences for, say, about 100 pairs of multi-criteria decisions.

A SME can say that a decision with attributes $(a_1, a_2, \ldots, a_n) = (1, 5, \ldots, 7)$ is better than a decision with attributes $(a_1, a_2, \ldots, a_n) = (3, 2, \ldots, 5)$. Alternatively, we may ask a SME to assign a priority to each $(a_1, a_2, \ldots, a_n)$ alterative using a 0 to 100 percentage scale. Table 1 can be interpreted in this way, where a_{10} can be viewed as a priority.

Both implicit alternatives provide us with a partially defined a scalar priority function, v:

$$v(x_1, \ldots, x_n) \geq v(y_1, \ldots, y_n) \Leftrightarrow (x_1, \ldots, x_n) \geq_{SME} (y_1, \ldots, y_n).$$

Function $v(x_1, \ldots, x_n)$ is *additive* if $v(x_1, \ldots, x_n) = v_1(x_1) + v_2(x_2) + \ldots + v_n(x_n)$. There are several sets of axioms known for the relation $\geq_{SME}$. If a set of such axioms is assumed to be true, then theorems can be proved that a numeric additive function v exists. One of these sets of axioms is presented below.

To be able to benefit from such mathematical results, we need to be able to test that the axioms are satisfied for an individual task and a dataset.

Two options are available: (1) to test axioms on a tabulated function v, such as that presented in Table 1, where $v(a_1, a_2, \ldots, a_9) = a_{10}$, and (2) to test axioms on tabulated on SME preferences $\{p_{ij}\}$, that record relations between different pairs of alternatives:

$$p_{ij} = 1 \Leftrightarrow x_i \geq_{SME} x_j,$$

where $x_i = (x_{i1}, \ldots, x_{in})$ and $x_j = (x_{j1}, \ldots, x_{jn})$.

However, testing axioms provides only a conclusion that an additive function v exists, but functions $v_1(x_1), v_2(x_2), \ldots, v_n(x_n)$ would still need to be built. To build v, we need to apply a simultaneous scaling procedure to $x_1, x_2, \ldots, x_n$. Then when the functions $v_1(x_1), v_2(x_2), \ldots, v_n(x_n)$ are found we need to analyze what they mean. To do this, we can view a dataset using these functions with new axis (scales) as shown in figure 2(b). It is a linear surface. The SME can analyze how well this sum corresponds to the estimates of $v(x_1, \ldots, x_n)$. Thus visualization provides a simple representation of SME knowledge that can be communicated to another SME and tested independ-

ently. If a regularity v is multiplicative, $v(x_1, \ldots, x_n) = v_1(x_1)v_2(x_2)\ldots v_n(x_n)$ then

$$\ln v(x_1, \ldots, x_n) = \ln v_1(x_1) + \ln v_2(x_2) + \ldots + \ln v_n(x_n),$$

that is the logarithm of v is an additive function. Thus, we can use the same technique for multiplicative regularities.

7. PHYSICAL STRUCTURES

The simultaneous scaling linearization and visualization procedure described in section 5 is fully applicable to physical structures. It is based on the theorem that is a described below. In addition to this theorem, another theorem was proved by [Vityaev, 1985], which states that every physical structure of the rank (2, 2) satisfies all axioms (1)-(5) of the additive conjoint structure described in section 2.

The most general characteristic of all physical laws is that they are equally applicable to all objects. This fundamental property permits the derivation of the structure of physical laws by formulating fuctional equations of a special type and solving them.

Let us consider two arbitrary sets of objects: set M with elements $i, k, \ldots$ and set N with elements $\alpha, \beta, \ldots$. Let us further suppose that for each pair $i \in M$, $\alpha \in N$ is mapped to a real number $a_{i\alpha} \in \mathbf{R}$ by some experiment; that is, the set M×N is mapped to the matrix $A = \| a_{i\alpha} \|$ of such numbers.

If sets M and N are two sets of physical objects of different type, then the matrix $\| a_{i\alpha} \|$ is the result of experiments that describe the relationship between objects $i \in M$ and $\alpha \in N$.

We will say that the **physical structure of the order (r, s)** is defined on sets M and N if a functional equation:

$$\Phi\begin{pmatrix} a_{i\alpha}, a_{i\beta}, \ldots, a_{i\gamma} \\ a_{k\alpha}, a_{k\beta}, \ldots, a_{k\gamma} \\ \ldots\ldots\ldots\ldots\ldots \\ a_{q\alpha}, a_{q\beta}, \ldots, a_{q\gamma} \end{pmatrix} = 0 \qquad (3)$$

is satisfied for any $r \cdot s$ real numbers from matrix $A = \| a_{i\alpha} \|$,

$$\begin{matrix} a_{i\alpha}, a_{i\beta}, \ldots, a_{i\gamma} \\ a_{k\alpha}, a_{k\beta}, \ldots, a_{k\gamma} \\ \ldots\ldots\ldots\ldots\ldots \\ a_{q\alpha}, a_{q\beta}, \ldots, a_{q\gamma} \end{matrix}$$

that are located on the intersection of any r rows i, k,..., q and any s columns $\alpha, \beta, \ldots, \gamma$.

The function Φ must not depend on:

1. the selection of r objects from the set M_r, where $M_r = \{ i, k, \ldots, q \} \subset M$, and
2. the selection of the s objects from the set N, $N_s = \{ \alpha, \beta, \ldots, \gamma \} \subset N$

We assume that the function Φ is analytical and cannot be presented as superposition of other analytical functions with a smaller number of variables.

We will say that the functional equation (3) gives us a **physical law** of order (r, s) that is invariant relative to selection of finite sets M_r and N_s from sets M and N. The equation (3) is a symbolically written, infinite system of functional equations relative to an unknown function $\Phi(x_{11}, x_{12}, \ldots, x_{rs})$ of $r \cdot s$ variables and one unknown infinite matrix $A = \| a_{i\alpha} \|$, which represent one real valued function $a_{i\alpha}$ of two nonnumeric arguments *i* and α from M and N.

Mikhailichenko [Mikhailichenko, 1972] solved this system of equations and derived the analytical expressions for all possible physical laws that satisfy the equation (3). He proved a theorem stating that functions Φ and $a_{i\alpha}$ may have only one of the following forms:

1. for $r = s = 2$ $\quad a_{i\alpha} = \Psi^{-1}(x_i + \xi_\alpha)$,

$$\Psi(a_{i\alpha}) - \Psi(a_{i\beta}) - \Psi(a_{j\alpha}) + \Psi(a_{j\beta}) = 0;$$

2. for $r = 4, s = 2$

$$a_{i\alpha} = \Psi^{-1}[(x_i \xi_\alpha^{1} + \xi_\alpha^{2}) / (x_i + \xi_\alpha^{3})],$$

$$\begin{vmatrix} \Psi[a_{i\alpha}] & \Psi[a_{i\beta}] & \Psi[a_{i\alpha}] & \Psi[a_{i\beta}] & 1 \\ \Psi[a_{j\alpha}] & \Psi[a_{j\beta}] & \Psi[a_{j\alpha}] & \Psi[a_{j\beta}] & 1 \\ \Psi[a_{k\alpha}] & \Psi[a_{k\beta}] & \Psi[a_{k\alpha}] & \Psi[a_{k\beta}] & 1 \\ \Psi[a_{l\alpha}] & \Psi[a_{l\beta}] & \Psi[a_{l\alpha}] & \Psi[a_{l\beta}] & 1 \end{vmatrix} = 0;$$

3. for $r = s \geq 3$

$$a_{i\alpha} = \Psi^{-1}(x_i^{1} \xi_\alpha^{1} + \ldots + x_i^{m-2} \xi_\alpha^{m-2} + x_i^{m-1} \xi_\alpha^{m-1}),$$

$$\begin{vmatrix} \Psi[a_{i\alpha}] & \Psi[a_{i\beta}] & \dots & \Psi[a_{i\tau}] \\ \Psi[a_{i\alpha}] & \Psi[a_{i\beta}] & \dots & \Psi[a_{i\tau}] \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ \Psi[a_{v\alpha}] & \Psi[a_{v\beta}] & \dots & \Psi[a_{v\tau}] \end{vmatrix} = 0;$$

and also

$$a_{i\alpha} = \Psi^{-1}(x_i^{1}\xi_\alpha^{1} + \dots + x_i^{m-2}\xi_\alpha^{m-2} + x_i^{m-1} + \xi_\alpha^{m-1}),$$

$$\begin{vmatrix} 0 & 1 & 1 & & 1 \\ 1 & \Psi[a_{i\alpha}] & \Psi[a_{i\beta}] & \dots & \Psi[a_{i\tau}] \\ 1 & \Psi[a_{i\alpha}] & \Psi[a_{i\beta}] & \dots & \Psi[a_{i\tau}] \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ 1 & \Psi[a_{v\alpha}] & \Psi[a_{v\beta}] & \dots & \Psi[a_{v\tau}] \end{vmatrix} = 0;$$

4. for $r = s + 1 \geq 3$

$$a_{i\alpha} = \Psi^{-1}(x_i^{1}\xi_\alpha^{1} + \dots + x_i^{m-2}\xi_\alpha^{m-2} + \xi_\alpha^{m-1}),$$

$$\begin{vmatrix} 1 & \Psi[a_{i\alpha}] & \Psi[a_{i\beta}] & \dots & \Psi[a_{i\tau}] \\ 1 & \Psi[a_{i\alpha}] & \Psi[a_{i\beta}] & \dots & \Psi[a_{i\tau}] \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ 1 & \Psi[a_{v\alpha}] & \Psi[a_{v\beta}] & \dots & \Psi[a_{v\tau}] \end{vmatrix} = 0;$$

5. for $r - s \geq 2$, except the case $r = 4$, $s = 2$ physical structures are not exist.
   Ψ- is a strictly monotone function of one variable in some vicinity;
   Ψ^{-1} - is an inverse function; x_i, ξ_α - are independent parametrs.

8. CONCLUSION

Evidence that an additive conjoint structure plays a fundamental role in two very different fields such as multi-criteria decision making and fundamental physical laws provides the basis for our belief that it also can play the same fundamental role in other fields. Thus, sumultaneous rescaling has a great potential as a major tool in visual data mining for the simplification of patterns to be visualized in a variety of fields. Further study will be directred

at providing computationally efficient simultaneous rescaling algorithms for multi-dimensional data.

9. EXERCISES AND PROBLEMS

1. Define an additive conjoint structure for $f \in F$ on $X_f \times W_f \times Z_f$ as a generalization of such structure for $f \in F$ on $X_f \times Z_f$ presented in section 2.

2. Formulate an analogue of the theorem presented in section 3 for the 3-D function $f \in F$ on $X_f \times W_f \times Z_f$.

3. Develop a rescaling algorithm for 3-D similar to that presented in section 3 for 2-D.

Advanced

4. Solve the problems in exercises 1-3 for the n-dimensional case.

5. Formulate axiom (1) presented in section 2 for n dimensions ($n \geq 3$) and prove that axiom (2) presented in section 2 for $n = 2$ follows from axiom (1) for $n \geq 3$ (Lemma 14 [Krantz et al., 1971, p.306]).

10. REFERENCES

Fayyad U., Grinstein G., and Wierse A., Eds., Information Visualization in Data Mining and Knowledge Discovery, Morgan-Kaufman, 2001.

Keim D., Visual Exploration of Large Data Sets, Communication of ACM, vol. 44, N.8, 2001, pp. 39-44.

Keeney R.L., Raiffa H. Decisions with Multiple Objectives: preferences and value Tradeoffs. John Wiley& Sons, 1976

Krantz DH, Luce RD, Suppes P, Tversky A: Foundations of Measurement v.1-3, Acad. Press, NY, London. 1971, 1989, 1990.

Kuakov Yu.I. The One Principal Underlying Classical Physics, Soviet Physics – Doclagy, V.15, #7, Jan., 1971, 666-668.

Kovalerchuk, B., Vityaev, E. Data Mining in Finance: Advances in Relational and Hybrid Methods, Kluwer Acad. Publ., Boston, 2000.

Mikhailichenko G.G. Phenomenological and Group Symmetry in the Geometry of two Sets (Theory of Physical Structures), Soviet Math. Docl. 32(2), 1985, 371-374.

Mikhailichenko G.G. Solution of functional equations in the theory of physical structures, Doklady, Soviet Academy of Sciences, 1972, v 206, N.5 1056-1058.

Mille, H. (Ed) Geographic Data Mining & Knowledge Discovery, Taylor and Francis, 2001

Rao II., Rao V., C++ Neural Networks and Fuzzy Logic, Hungry Minds, Inc, 1995.

Soukup T., Davidson I., Visual Data Mining: Techniques and Tools for Data Visualization and Mining, Wiley, 2002

Spence R., Information Visualization, Addison-Wesley, 2001.
Vityaev E.E. Numerical, algebraic, and constructive representation of one physical structure. In: Logical foundations MOZ (Methods for discovery of regularities), Computer systems, #107, Novosibirsk, 1985, pp.40-51 (in Russian).
Ware C. Information Visualization: perception for Design, Acad. Press, 2000.

Chapter 16

VISUAL DATA MINING USING MONOTONE BOOLEAN FUNCTIONS

Boris Kovalerchuk[1] and Florian Delizy[1,2]
[1]*Central Washington University, USA*
[2]*University of Technology of Belfort-Montbéliard, France*

Abstract: This chapter describes a new technique for extracting patterns and relations visually from multidimensional binary data using monotone Boolean functions. Visual Data Mining has shown benefits in many areas when used with numerical data, but that technique is less beneficial for binary data. This problem is especially challenging in medical applications tracked with binary symptoms. The proposed method relies on monotone structural relations between Boolean vectors in the n-dimensional binary cube, E^n, and visualizes them in 2-D as chains of Boolean vectors. Actual Boolean vectors are laid out on this chain structure. Currently the system supports two visual forms: the multiple disk form (MDF) and the "Yin/Yang" form (YYF). In the MDF, every vector has a fixed horizontal and vertical position. In the YYF, only the vertical position is fixed.

Key words: Visual Data Mining, explicit data structure, Boolean data, Monotone Boolean Function, Hansel Chains, Binary Hypercube.

1. INTRODUCTION

The goal of *visual data mining* (VDM) is to help a user to get a feeling for the data, to detect interesting knowledge, and to gain a deep visual understanding of the data set [Beilken & Spenke, 1999]. One of especially important aspects of visual data mining is *visualizing the border* between patterns. A visual result in which the border is simple and patterns are far away from each other matches our intuitive concept of the pattern and serves as important support that the data mining result is robust and not accidental.

VDM methods have shown benefits in many areas when used with **numerical data**, but these methods do not address the specifics of **binary data**, where there is little or no variability in the visual representation of objects for each individual Boolean attribute. VDM is especially challenging task when data richness should preserved without the excessive aggregation that often happens with simple and intuitive presentation graphics such as bar charts [Keim, Hao, Dayal, & Hsu, 2002]. Another challenge is that often such data lack natural 3-D space and time dimensions [Groth, 1998] and instead require the visualization of an abstract feature.

The purpose of this chapter is to develop a technique for visualizing and discovering patterns and relations from *multidimensional binary data* using the technique of monotone of Boolean functions, which are also reviewed at the end of the chapter. We begin with an analysis of the currently available methods of data visualization.

Glyphs. A *glyph* is a 2-D or 3-D object (icon, cube, or more complex *"Lego-type"* object). Glyph or **iconic visualization** is an attempt to encode multidimensional data within the parameters of the icons, such as the shape, color, transparency, orientation [Ebert, Shaw, Zwa, Miller & Roberts, 1996; Post, van Walsum, Post & Silver, 1995; Ribarsky, Ayers, Eble & Mukherja, 1994].

Typically, glyphs can visualize up to *nine attributes* (three positions x, y, and z; three size dimensions; color; opacity; and shape). Texture can add more dimensions. Shapes of the glyphs are studied in [Shaw, Hall, Blahut, Ebert & Roberts, 1999], where it was concluded that with large superellipses, about 22 separate shapes can be distinguished on the average. An overview of multivariate glyphs is presented in [Ward, 2002]. This overview includes a taxonomy of glyph placement strategies and guidelines for developing such a visualization. Some glyph methods use data dimensions as positional attributes to place glyphs; other methods place glyphs using *implicit or explicit structure* within the data set.

From our viewpoint, the placement based on the use of data structure is a promising approach. We believe that the placement of glyphs on a data structure is a way to increase the data dimensions that can be visualized. We call this the **GPDS** approach (**Glyph Placement on a Data Structure**). It is important to notice that in this approach, some attributes are implicitly encoded in the data structure while others are explicitly encoded in the glyph. Thus, if the structure carries ten attributes and a glyph carries nine attributes, we can encode a total of nineteen attributes. The number of glyphs that can be visualized is relatively limited because of possible glyph overlap and occlusion.

Spiral Bar and others techniques. Alternative techniques such as Generalized Spiral and Pixel Bar Chart are developed in [Keim, Hao, Dayal &

Hsu, 2002]. These techniques work with large data sets without overlapping, but only with a few attributes (these range from a *single attribute* to *perhaps four to six attributes*). Another set of visualization methods, known as Scatter, Splat, Map, Tree, and Evidence Visualizer, are implemented in MineSet (Silicon Graphics), which permits up to *eight dimensions* to be shown on the same plot by using color, size, and animation of different objects [Last & Kandel, 1999].

Parallel coordinate techniques. This visualization [Inselberg & Dimsdale, 1990] can work with ten or more attributes, but suffers from record overlap and thus is limited to tasks with well-distinguished cluster records. In parallel coordinates, each vertical axis corresponds to a data attribute (x_i) and a line connecting points on each parallel coordinate corresponds to a record. Figure 1 depicts vectors

$$01010;11010;01110;01011;01111;11011;11111;10101;11101;10111 \quad (1)$$

Can we discover a regularity that governs the dataset in Figure 1? This figure represent these data in parallel coordinates. It is difficult, but the regularity is a simple monotone Boolean function $(x_2 \,\&\, x_4) \vee (x_1 \,\&\, x_3 \,\&\, x_5)$. This function is true for vectors from (1).

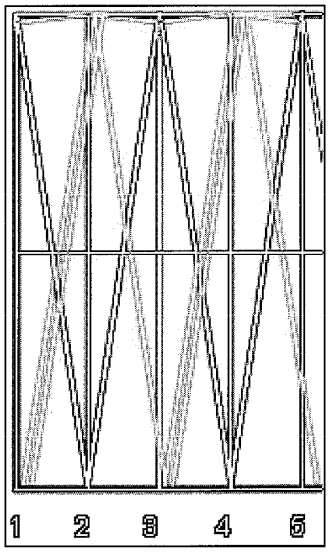


Figure 1. Ten Boolean records in parallel coordinates

Limitations. Technically, the number of dimensions is limited only by the screen resolution. However, this is typically too overwhelming to permit any real understanding of the data [Goel, 1999]. Serious limitations in traditional visualization techniques are summarized in [Last & Kandel, 1999]:

- The *subjectivity of the visual representation* can cause different conclusions looking to the same data.

- The *poor scalability* of visual data analysis can fail when representing hundreds of attributes.
- Humans *unable* to perceive more than six to eight dimensions on the same graph.
- The *slow speed* of manual interactive examination of the multi-dimensional, multi-color charts is a drawback.

We are interested developing a technique that can work with ten or more Boolean attributes. Many data mining problems can be encoded using Boolean vectors, where each record is a set of binary values {0; 1} and each record belongs to one of two classes (categories) that are also encoded as 0 and 1. For instance, a patient can be represented as a Boolean vector of symptoms along with an indication of the diagnostic class (e.g., benign or malignant tumor) [Kovalerchuk, Vityaev & Ruiz, 2000, 2001].

For n-dimensional Boolean attributes, traditional glyph-based visualizations are useful but somewhat limited. Attributes of a Boolean vector can be encoded in glyph lengths, widths, heights, and other parameters. There are only two values for the length, width, and other parameters for each Boolean vector. Thus, there is not much variability in visual representation of objects. When plotted as nodes in a 3-D binary cube, many objects will not be visually separated.

The approach and methods described below do not follow the traditional glyph approaches that would put n-dimensional Boolean vectors ($n > 3$) into 3-D space, making them barely distinguishable. The methods rely on **monotone structural relations** between Boolean vectors in the n-dimensional binary cube, E^n. Data are visualized in 2-D as chains of Boolean vectors. Currently, the system supports two visual forms: the **Multiple Disk** Form (**MDF**) and the "**Yin Yang**" Form (**YYF**).

2. A METHOD FOR VISUALIZING DATA

The numeric order of data layout. Consider a set of n-dimensional Boolean vectors V be given similar to those shown in (1) above. Every n-dimensional Boolean data set V can be encoded as a Boolean function in a disjunctive normal form (DNF) or conjunctive normal form (CNF). Thus, *visualization of a Boolean data set is equivalent to visualization of a Boolean function.* Next, every Boolean function can be decomposed into a set of monotone Boolean functions [Kovalerchuk et al., 1996].

The monotone structure is important for the data mining tasks, because most of data mining methods are based on the *hypothesis of local compactness*: if two objects have similar features, then they belong to the same class.

In this chapter, we assume that the data satisfy the **property of monotonicity**, that is, if vector ***a*** *belongs* to class 1, then a vector ***b*** that is greater than or equal to ***a*** also belongs to class 1. The formal definition of monotonicity is given in section 6. Informally, monotonicity means that if a patient with symptoms ***a*** has a malignant tumor then another person with symptoms ***b*** that include all ***a*** symptoms and some additional symptoms most likely also has a malignant tumor.

Below we describe the structure used to allocate these Boolean vectors in 2-D rendering. Boolean vectors are ordered vertically by their Boolean norm (sum of "1"s) with the largest vectors are rendered on the top, starting from (11111).

Our intent is to visualize the border between the two classes in a clear way. Each vector will be first placed in the view and then drawn as colored bar: white for the 0 class, black for the 1 class.

Figure 2 depicts the hierarchy of levels of Boolean vectors with $n = 10$, that is 11 levels from 0 to 10, where level 0 and 10 contain vectors (00000 00000) and (11111 11111), respectively. Several vectors are shown in Figure 2 as small bars. The bar on the top row represents vector (11111 11111). The vector on the third row can not be identified from this figure without additional assumptions, but we call tell for sure that it has eight "1"s, because of its location on the third line. To be able to identify in a more specific way the location of the "0"s in this vector, we need to assume that the vertical positions (columns) are associated with specific numbers.

For instance, we can assume that all vectors with the same norm are ordered, where the leftmost column can be reserved for the vector (11111 11100) and the rightmost position can be reserved for vector (00111 11111) in the third row.

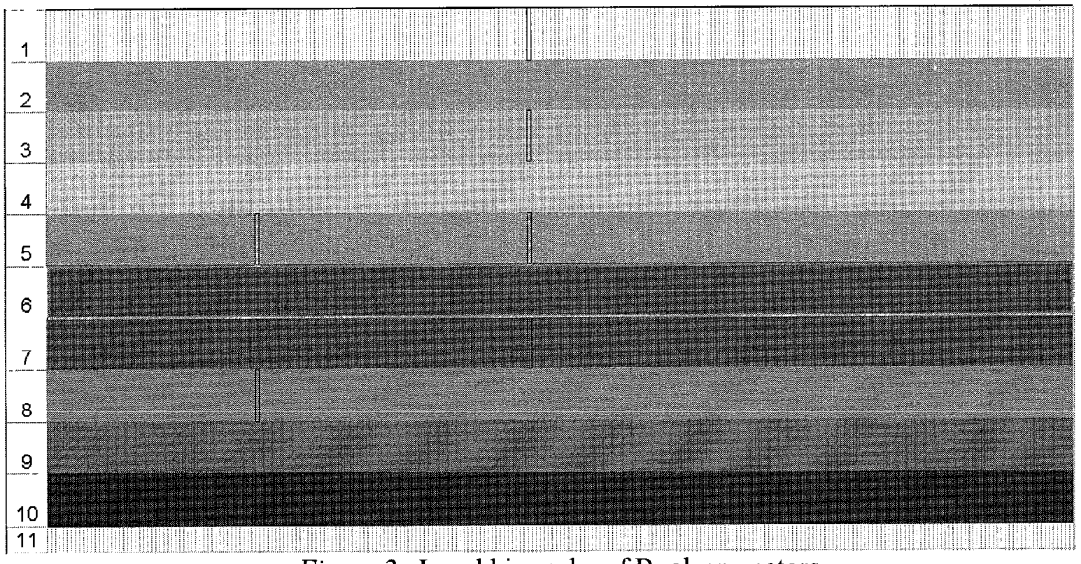


Figure 2. Level hierarchy of Boolean vectors

Other vectors can occupy fixed positions in between based on their numeric value (binary/decimal) where Boolean vectors are interpreted as numbers with the low-order bits located on the right. In this case, since the vertical position is centered, the vector in row 3 is an average of binary numbers (not vectors) $00111\ 11111_2$ and $11111\ 11100_2$ and that can be computed. Here the subscript 2 indicates a binary number. We call this visualization a **Table Form Visualization (TFV)**.

Note that for $n = 10$ the maximum number of vectors on each row is 252 that is the number of combinations for five "1"s out of ten. Consider a display having a horizontal screen resolution of $1024 = 2^{10}$ pixels, we can use 4 pixels per bar and can easily visualize 10-dimensional space in 2-D as it is shown in Figure 2.

Using only one pixel per bar we can accommodate 12 binary dimensions since we would have 924 combinations for choosing six "1"s out of 12 and that is still less then 1024. With a higher resolution screen and/or multiple monitors, we could increase the dimensionality, but this has obvious limits of about $n = 14$ where 3432 pixels are needed on a row.

Several options are available to deal with this exponentially growing dimensionality. One of them is grouping, that is showing high dimensional data by their projections to, say, 12-D, which is much larger than traditional conversion to 2-D. Then methods such as principal components can be used with visualization of first 12 principal components instead of first two principal components.

Beyond this, we need to notice that for $n = 20$, the number of elements in the space is $2^{20} = 1{,}048{,}576$. If a dataset contains $8192 = 2^{13}$ vectors, then they would occupy no more than $2^{13}/2^{20} = 2^{-7} = 1/128$ fraction of the total space, that is less than 1%. This means that a visualization like that shown in Figure 2 may not use 99% of the screen. Thus, the visualization columns that are not used can be reduced in order to enable the visualization of vectors with $n = 20$ or more.

The visualization shown in Figure 3 is a modification of the visualization shown in Figure 2, where all repeating vectors are deleted. In Figure 2, the top and bottom vectors are repeated 252 times. Each level in Figures 3 is called a disk and the entire visualization is called the **multiple disk form (MDF)**. In the MDF form, every vector has a fixed horizontal and vertical position as shown in Figure 3.

3. METHODS FOR VISUAL DATA COMPARISON

In sections 3 and 4, we detail procedures for MDF that successively provide an increased ability for making visual comparisons of Boolean vectors

chosen from a set of n-dimensional Boolean vectors V. Examples of applications of these procedures are provided in section 5.

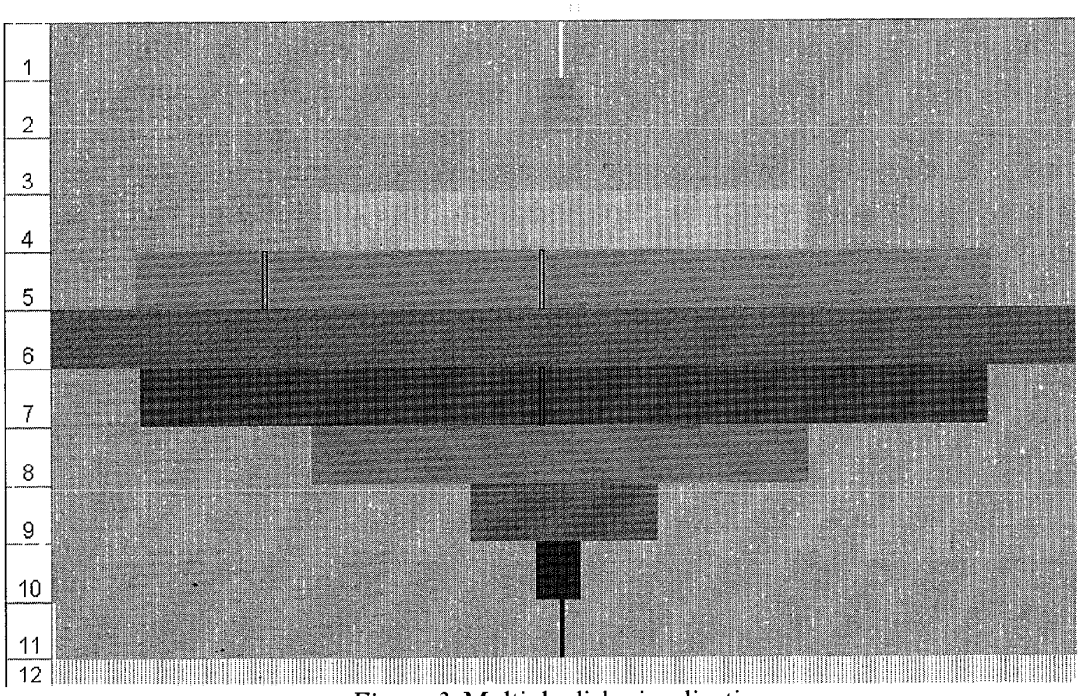


Figure 3. Multiple disk visualization

Location-based procedure. The Boolean vectors are placed to the disk by their level as explained in the previous section. Specifically, vectors are placed horizontally inside the appropriate disk using a placement procedure based on natural numerical order. Each binary vector is converted to its decimal equivalent. For instance, the decimal equivalent of the vector 0000000010 would be 2_{10}. Each vector is then placed horizontally in its disk based its value with value 0 being on the right side. Call this procedure P_1.

The advantage of this procedure is to allow the user to compare more than one binary dataset or Boolean function at a time. This is possible because each vector always is assigned the same fixed location of the disks based on its value. If two data sets or Boolean functions are equal then they have exactly the same layout.

Procedure P_1 produces the same border scheme for different Boolean functions and datasets, a condition that is necessary for direct **comparison**. In general, direct comparisons are not always easy. Moreover, often the goal is finding the differences between functions rather than similarities. If two datasets or functions are different then their difference will be revealed by their layout on the on disks.

Chain-based procedure. P_1 can only be used for comparing two data sets or functions, but does not really visualize any border or structure of the Boolean function; therefore, we have created the second procedure P_2. This

procedure for MDF is based on the decomposition of the binary cube, E^n, into chains.

For instance, the vectors given in (1) above form several natural chains as shown in Table 1 where the gray elements appear in several chains. Chain 1 contains four elements starting from (01010) and ending up with (11111). Each following element is greater than previous one, that is, some 0 positions are exchanged for 1's as in (01010) and (11010) where the first position is changed to 1. We also note that each vector in the second row has two 1's. Similarly each vector in third and forth rows has three or four 1's. As previously noted such numbers indicate the **level of the Boolean vector** and are also referred to as its norm.

Table 1. Boolean data chains

Chain 1	Chain 2	Chain 3	Chain 4	Chain 5
01010	01010			
11010	01011	01110	10101	
11011	01111	01111	11101	10111
11111	11111	11111	11111	11111

While vectors on the same chain are ordered, vectors on the different chains may not be ordered. There is only a partial order on Boolean vectors.

The **partial order** is defined as follows: vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ is greater or equal to vector $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ if for every $i = 1, 2, \ldots, n$; $a_i \geq b_i$. This partial order means that chains may overlap as shown in Table 1. We will use the notation $\boldsymbol{a} \geq \boldsymbol{b}$ if Boolean vector $\boldsymbol{a}$ is greater than or equal to Boolean vector $\boldsymbol{b}$. A set of vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ is called a **chain** if

$$\boldsymbol{v}_1 \geq \boldsymbol{v}_2 \geq \ldots \geq \boldsymbol{v}_n .$$

We focus special chains of vectors called **Hansel chains**, which are described in section 6. The Hansel chains are computed and then aligned vertically. Procedure P_2 applied to MDF moves vectors with regard to Hansel chains. First Hansel chains for vectors of size (dimension) n are computed, and then every vector belonging to the chain will be moved to align the Hansel Chain vertically. Hansel chains have different lengths with possible values from 1 to n elements. To keep the integrity of the MDF structure, we have to place these chains so that no elements fall out of the disks. Hence, the longest chain will be placed on the center of the disk and the others chains will be placed alternatively to the right and left of the first chain. Moreover, procedure P_2 will always assign a fixed position to each vector. This position does not change from one dataset to another. This again allows direct comparison to be done between different Boolean functions. P_2 visualizes a certain extent the structure of the Boolean function, but it does not really visualize the border between classes.

4. A METHOD FOR VISUALIZING PATTERN BORDERS

The goal of this VDM method is to show patterns in a simple visual form. If cases of two classes (say, benign and malignant) are separated in the visual space and the border between classes is simple, then the goal of VDM is reached. We want to reveal such border visually using the technique of monotone Boolean functions.

Procedures P_1 and P_2 produce a border that can be very complex and, thus, not easy to interpret as Figures 4 and 5 indicate. These figures illustrate the methods for visualizing a data set that can be described by a simple Boolean function $f(x_1,x_2,...,x_{10}) = x_1$. Figure 4 shows the system placing elements at a fixed place using MDF and P_1. Because elements have a fixed place, this procedure permits the comparison between multiple functions. For example, we could superimpose another function in the same disks with a different color for the bars or we could place it side-by-side in the second panel.

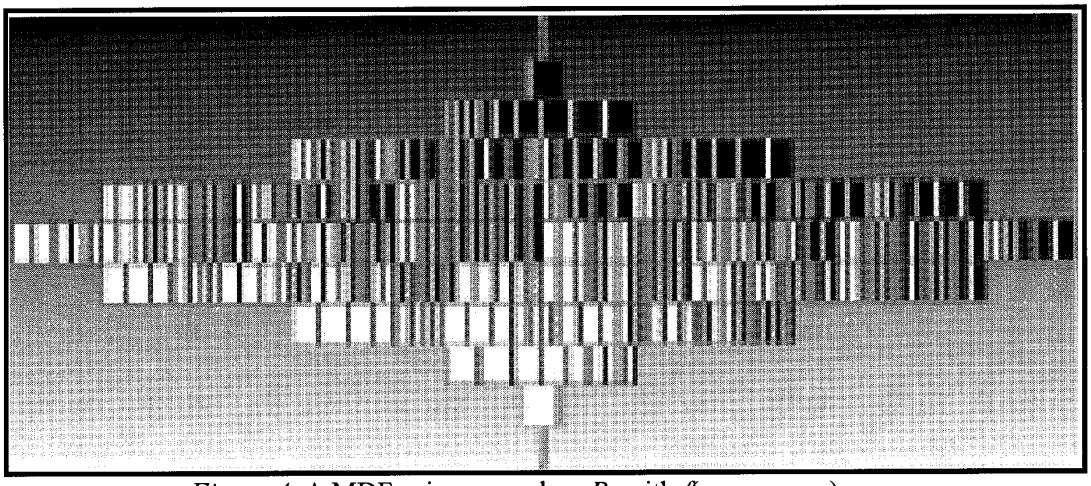


Figure 4. A MDF using procedure P_1 with $f(x_1,x_2,..., x_{10}) = x_1$

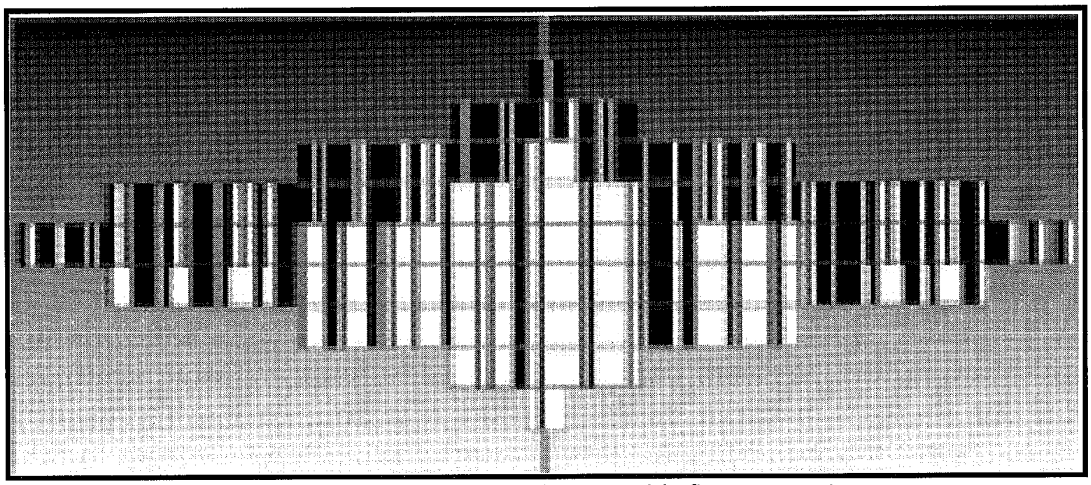


Figure 5. A MDF using procedure P_2 with $f(x_1,x_2,..., x_{10}) = x_1$

Next, the procedure P_2 unveils parts of the structure of the Boolean function, see Figure 5. Recall, P_2 still has a fixed place for each Boolean vector. Hence, it still permits the comparison of multiple functions. However as Figure 5 shows, the border visualized can be very complex.

Procedure P_3. Hence, we introduce a third procedure, P_3, for MDF. This procedure tries to move all Hansel chains to the center of the disk. It is based on: the level of the first "1" value in each chain for a given Boolean function, and the requirement that the disk architecture should be preserved. In this way, two different functions will produce distinct visualizations. Figure 6 demonstrates that the results of P_3 more easily visualize the border between the two classes.

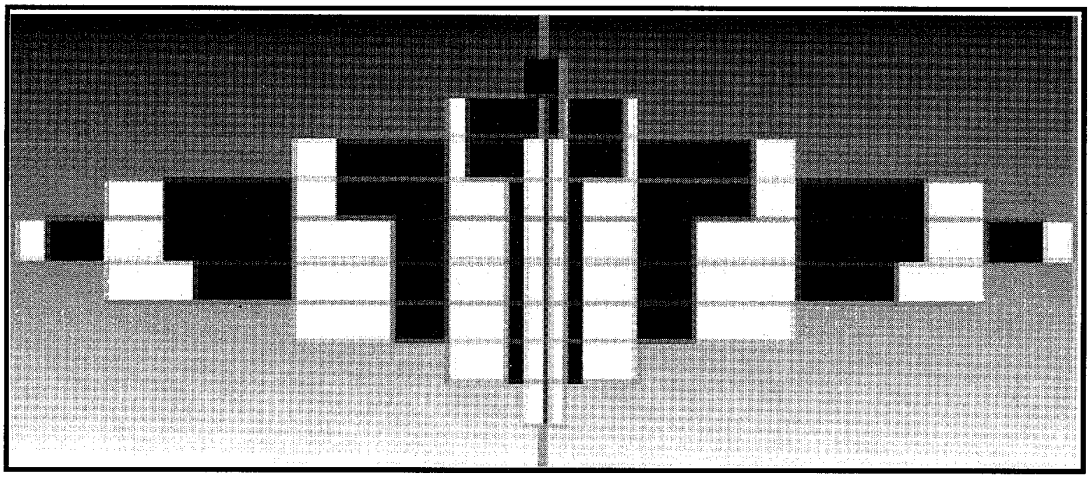


Figure 6. A MDF using procedure P_3 with $f(x_1,x_2,\ldots,x_{10}) = x_1$

Here, the concept of the first "1" on the chain means that chain may contain elements from both classes, say benign (class "0"), and malignant (class "1"). Procedure P_3 is a derivative of P_2. After computing and placing the vectors using P_2, every Hansel chain will be given a value l equal to the level of the first 1 value present within the chain. Next, every Hansel chain will be moved so that the chain with the highest l value is located in the center of the disk so that the MDF structure is kept. Using this procedure, we are able to group classes within the MDF. Nevertheless to keep the MDF structure, the chains have to be placed in a position related to their length. Unfortunately, this potentially introduces a complex border between classes because of a possible *gap* between groups of vectors within the same class.

Procedure P_4. Using P_3 makes the border obvious, but the border is still divided into several pieces because of the structure of the MDF itself. We can see the gaps between the black parts. However, each white element placed on the top (belonging to class 0) actually expands to an element of the class. Therefore, the border should be visualized as continuous rather than

interrupted. Since the borders produced by P_3 can still be complex and thus difficult to interpret visually, we introduce the **YYF structure** along with procedure P_4.

In the YYF, the movement of all chains is based on only the level of the first 1 in each chain. Additionally, this structure allows filling the gaps between the groups. The set of chains is sorted from left to right according to the indicated level and providing a clear, simple border between the two classes of Boolean vectors. Procedure P_4 extends every Hansel chains created before placing them according to the same method used in P_3. The first step consists in *extending the Hansel chain* with elements extended in relation to the edge elements both up and down. The YYF structure shows this continuous border build using Procedure P_4, see Figure 7.

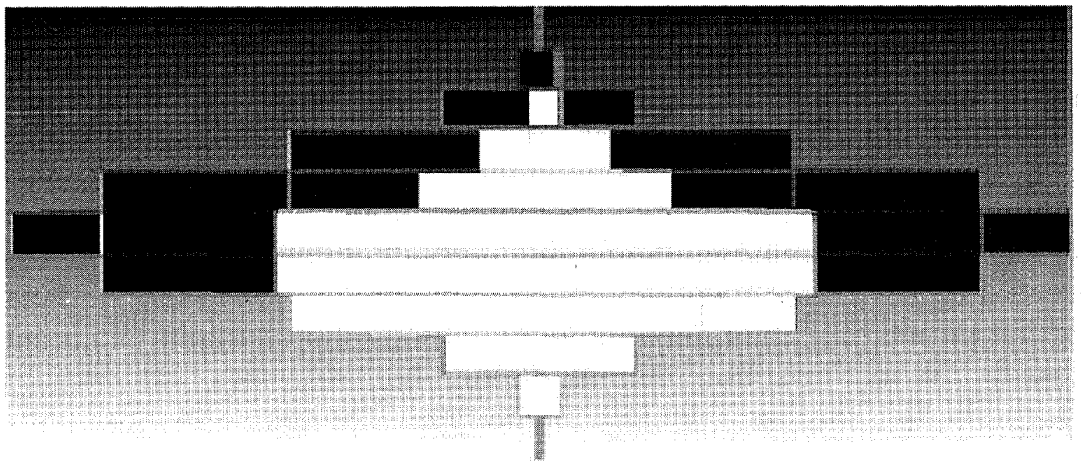


Figure 7. A YYF using procedure P_4 with $f(x_1,x_2,\ldots, x_{10}) = x_1$

Edge elements are Boolean vectors that form the border between two classes on the chain. To extend a chain up, we try to find the first vector belonging to class 1 above the edge element. That is, given the edge vector $\boldsymbol{x}$ we look for a vector $\boldsymbol{y}$ where $\boldsymbol{y} \geq \boldsymbol{x}$, if no such $\boldsymbol{y}$ is found on the level just above the $\boldsymbol{x}$ level, we then add a vector $\boldsymbol{z}$ from class 0 so that $\boldsymbol{z} \geq \boldsymbol{x}$ and so that the path to the first vector $\boldsymbol{y} \geq \boldsymbol{z}$ of class 1 is minimized.

We repeat these steps until we find a vector $\boldsymbol{y}$ from the 1 class and add it to the chain. To expand a chain down, we apply the same steps reversing the relation $\boldsymbol{y} \geq \boldsymbol{x}$ and swapping the classes 0 and 1. Using this procedure, we duplicate some of the vectors which would display them more than once. This is justified because we keep a consistent relative relationship between the vectors.

Once the chains are expanded, a value l will be assigned to each chain, just as in the procedure P_3. Then, the chains will be sorted with regard to this value. This approach visualizes a simple border between classes 0 and 1. In our attempt to visualize the borders between the two classes of elements, we moved the data out of the MDF structure. In the YYF vectors are ordered vertically in the same way as in the MDF but they are not centered anymore.

All vectors are moved with regard to the data in order to visualize to the border between the two classes. In this way, a clear border will appear, class 1 being above class 0 thus giving the YYF the *"Yin Yang"-like* shape ☯ ,that responsible for its name.

5. EXPERIMENT WITH A BOOLEAN DATA SET

This experiment was conducted with breast cancer data that included about 100 cases with an almost equal number of benign and malignant results. Each case was described by 10 binary characteristics (referred to as "symptoms" for short) retrieved from mammographic X-ray images [Kovalerchuk, Vityaev & Ruiz, 2001]. The goal of experiment was to check the monotonicity of this data set which is important from both radiological and visualization viewpoints. Figure 8 shows initial visualization where cases are aligned as they were in Figure 4; that is, by allocating cases as bars at fixed places using MDF and P_1. As we mentioned in section 5, this procedure permits the comparison of multiple functions and data sets. Comparison of Figures 4 and 8 immediately reveals their difference. Figure 8 presents the layered distribution of malignant and benign cases visualized as bars, where white bars represent the benign cases and black bars represent the malignant cases.

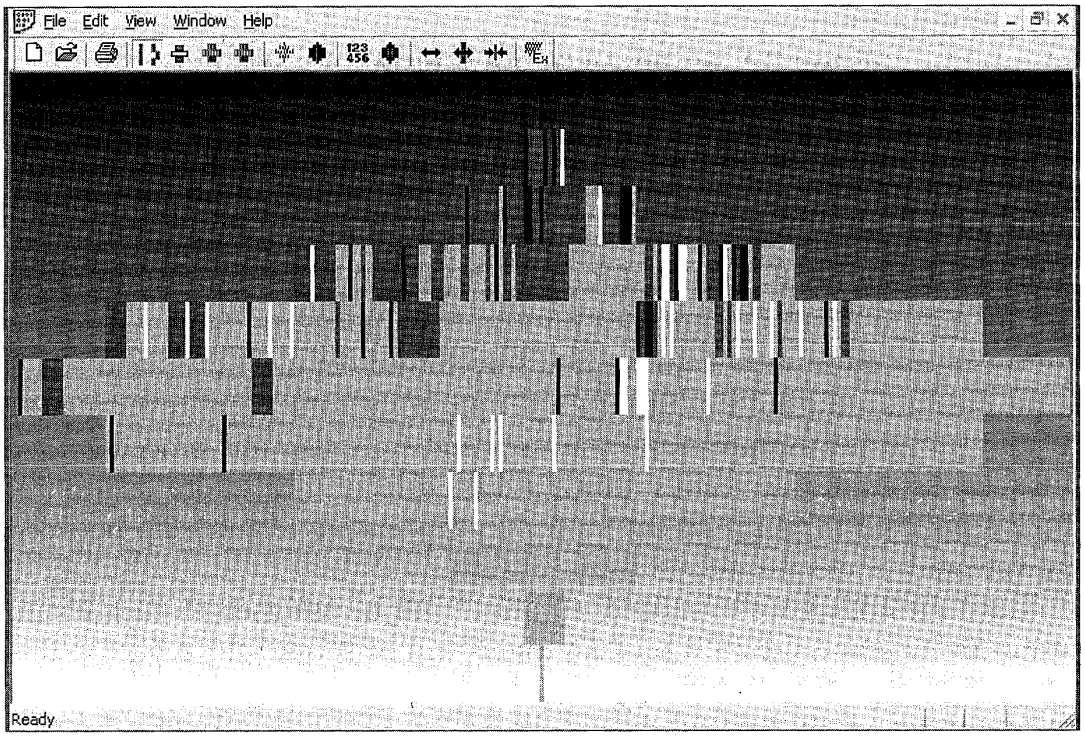


Figure 8. Breast cancer cases based on characteristics of X-ray images visualized using fixed location procedure P_1

All cases in the same layer have the same number of cancer positive symptoms, but the symptoms themselves can be different. Light grey areas indicate monotonic expansion of benign cases to lower layers for each benign case and dark grey areas indicate monotonic expansion of malignant cases to upper layers for each malignant case. Figures 9, 10 and 11 are modifications of Figure 8 based on procedures P_2 and P_3.

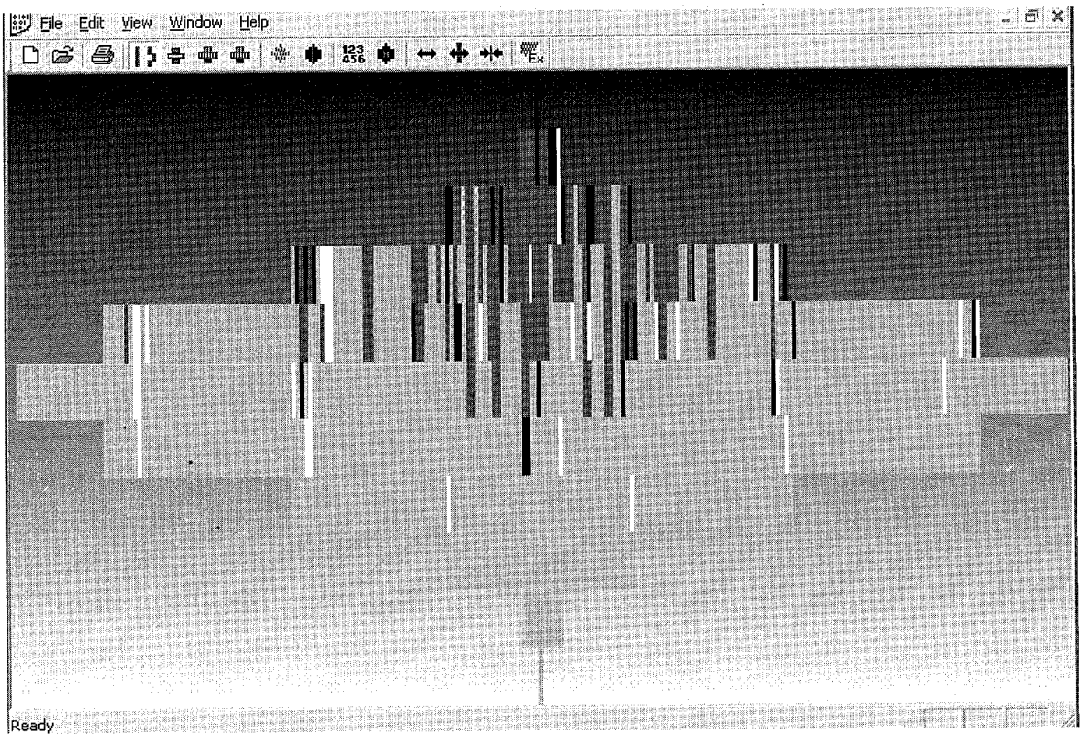


Figure 9. Breast cancer cases visualized using procedure P_2

Benign cases are lined up monotonically. That is, each benign case below a given benign case contains only a part of its positive cancer characteristics. Similarly malignant cases (bars) are also lined up monotonically. Thus, a malignant case above a given malignant case contains more positive cancer characteristics than the given malignant case. The vertical lines (chains) that contain both benign and malignant cases are most interesting for further analysis as we shall see in Figure 11. Figure 11 is a simple modification of Figure 10 where the cases or areas around the bars are separated by frames. Figure 12 shows a fragment of the chain from Figure 11 that is rotated 90°. This chain demonstrates a violation of monotonicity, where after a benign (white) case on the layer 7 we have a "malignant" (grey) case on the layer 6 that was obtained via monotonic expansion. It also shows how narrow the border is between benign and by two malignant (black) cases on layers 8 and 9 with out any gap. An actual benign case on layer 7 is immediately followed by two actual malignant (black) cases on layers 8 and 9 with out any gap.

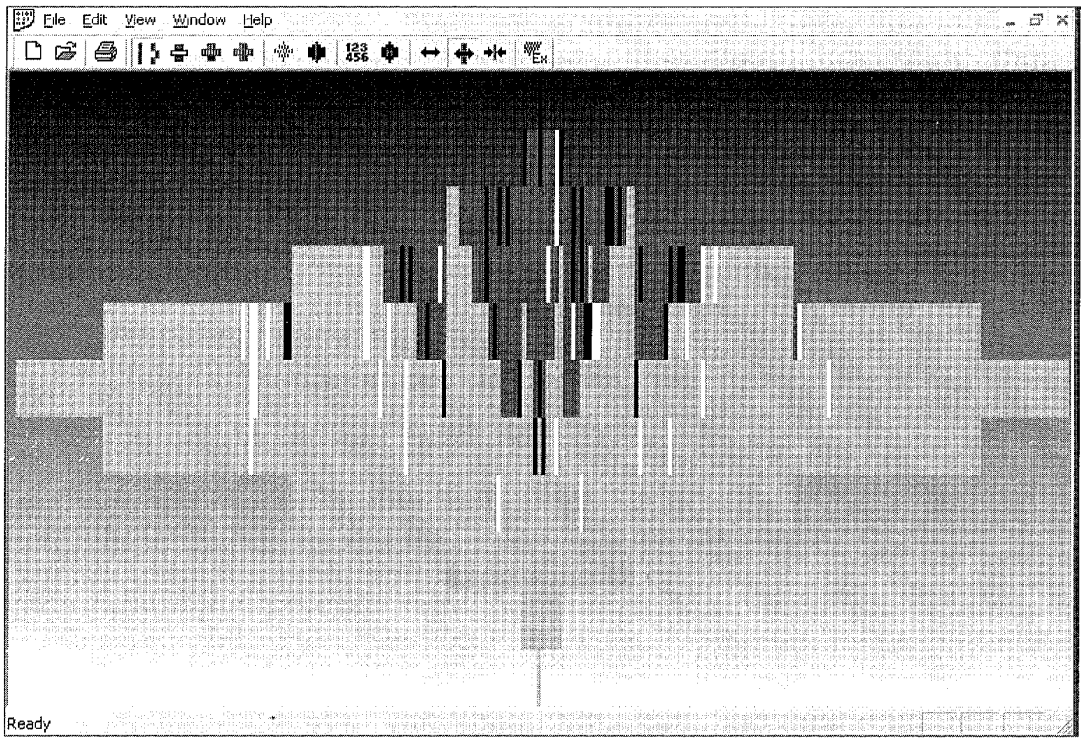


Figure 10. Breast cancer cases visualized using procedure P_3

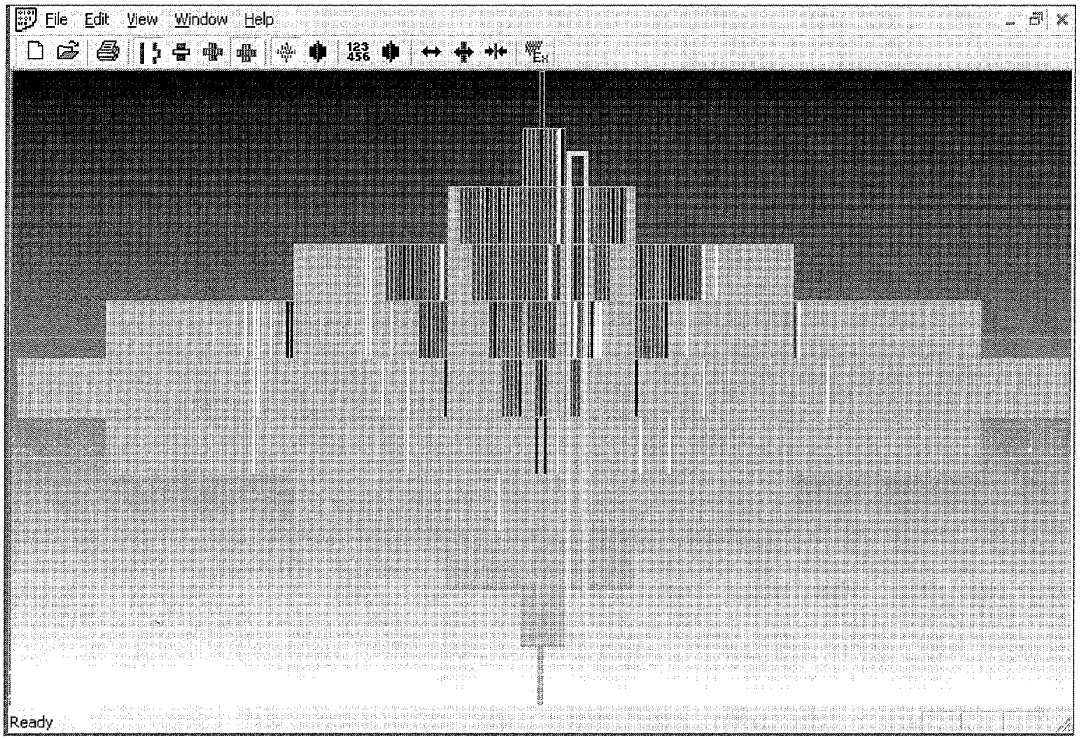


Figure 11. Breast cancer cases visualized using procedure P_3 with cases shown as bars with frames. See also color plates.

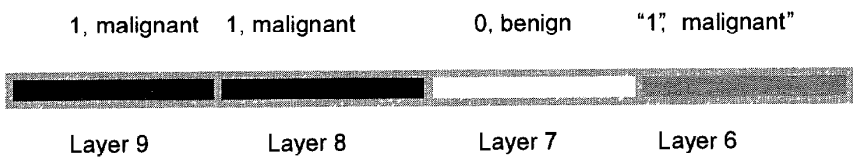


Figure 12. Fragment of individual chain with violated monotonicity

Figures 8-12 reveal that there are *inconsistencies* with monotonicity for several of the cases. Note, here cases are shown without frames and several bars may form continuous areas filled with the same color.

The "white" case is an *inconsistent case* if there are black and dark grey bars (areas) above and below it. Similarly the "black" case is inconsistent if there are white and light grey cases above and below it. Both types of inconsistent cases are presented in Figure 9. We will call them *white and black inconsistencies*, respectively.

This visualization permits us building different monotone Boolean functions interactively and visually for situations with inconsistencies. The first way to do this is to find all white inconsistencies and convert all elements below them to white bars. This process is called a *white precedence monotonization.* Similarly, we can use a *black precedence monotonization* that converts all white and light grey elements above inconsistent black cases to black.

If we use the black precedence, then such monotone expansion will cover 100% of the malignant cases. This means that we will have some false positive cases (benign cases diagnosed as malignant), which of course is better than having a false negative (cancer cases diagnosed as benign). The latter occurs when we give precedence to monotonic expansion of benign cases (white precedence monotonization). If the black case monotonization produces too many false positives we may check the sufficiency of the parameters used. The violation of monotonicity may indicate that more parameters (beyond 10 parameters used) are needed. The advantage of described approach is that we build a visual diagnostic function that is very intuitive. Inconsistencies can be analyzed globally followed by pulling up inconsistent cases for further analysis as shown in Figure 12.

Two 3-D versions of the **Monotone Boolean Visual Discovery (MBVD)** method (programmed by Logan Riggs) are illustrated in Figures 13 and 14. The first version uses only vertical surfaces and is quite similar to the 2-D versions. The second version (Figure 14) uses both vertical and horizontal surfaces. 3-D versions have several advantages over 2-D versions. The first one is the ability to increase the dimensionality n that can be visualized. It is done by using front, back and horizontal surfaces of disks and by grouping similar Hansel chains and by showing only "representative" chains in the global disk views (see Figure 15 for grouping illustration). More detail can

be provided by changing camera location, which permits one to see the back side of the disks combined with the semantic zoom that permits one to see all chains not only the "representative" ones when the camera closes up on the disk.

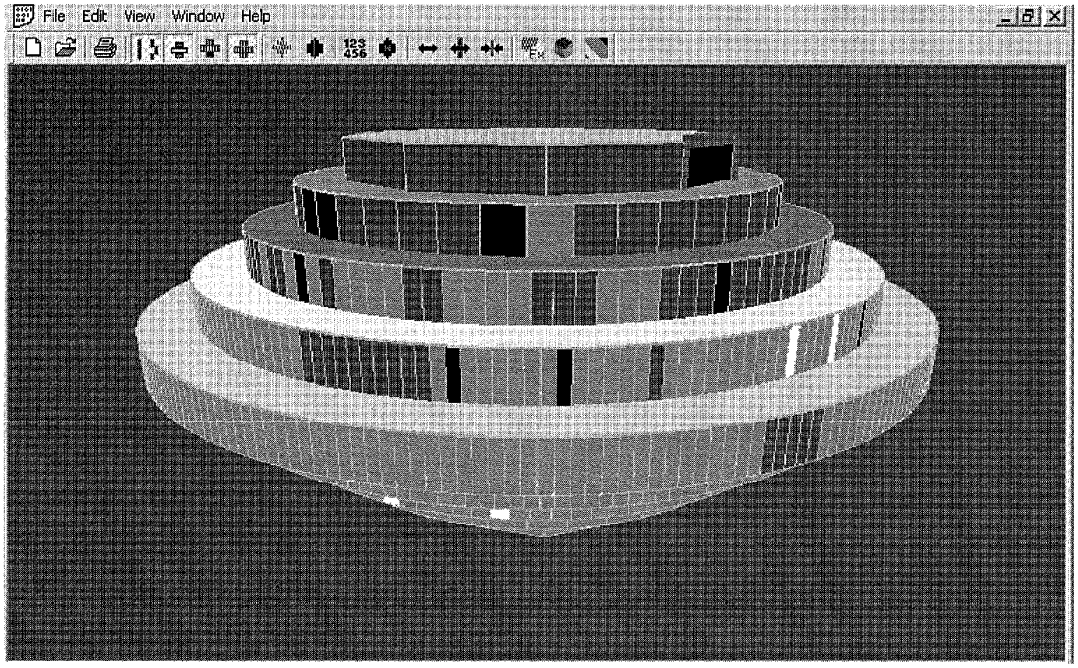


Figure 13. A 3-D version of Monotone Boolean Visual Discovery with only vertical surface used

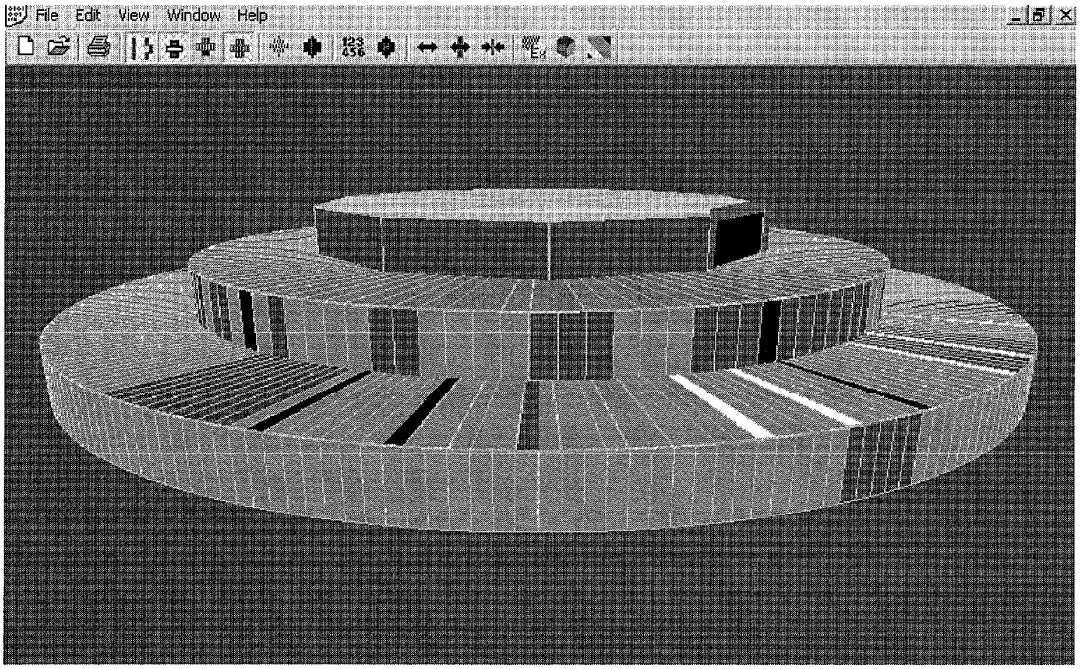


Figure 14. A 3-D version of Monotone Boolean Visual Discovery with vertical and horizontal surfaces used. See also color plates.

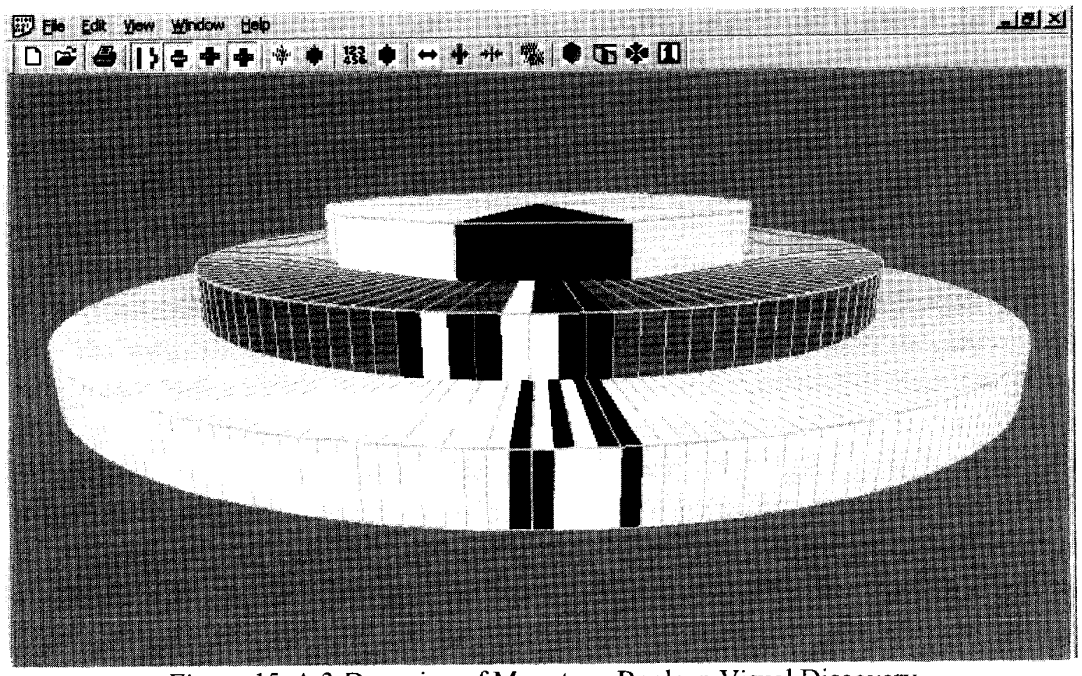


Figure 15. A 3-D version of Monotone Boolean Visual Discovery with grouping Hansel chains. See also color plates.

6. DATA STRUCTURES AND FORMAL DEFINITIONS

A **Boolean vector** is an ordered set of n Boolean values 0 and 1. For instance, a sequence of 10 bits such as 1011100100, which would be a 10-dimensional Boolean vector. We can assign a set of properties to a Boolean vector such as its size (dimension n) and its norm. The **level of a Boolean vector**, also referred to as the **Boolean norm**, is the sum of the components of the Boolean vector. We use these norms for splitting the set of vectors into $n + 1$ levels. For instance, the level of the vector 0000000000 would be 0, whereas the level of the vector 1111111111 would be 10.

Boolean vectors can be represented as vertices of a cube (or a **hypercube** if $n > 3$). For instance, if $n = 1$, the cube is formed by the two elements {0; 1}. For $n = 2$, the cube is a square formed by the elements {00; 01; 10; 11}. **Hansel chains** [Hansel, 1966; Kovalerchuk, Triantaphyllou, Despande & Vityaev, 1996] provide a way to browse a hypercube without overlapping.

We build Hansel chains recursively. The Hansel chain for n=1 is the trivial segment (0; 1). To obtain the Hansel chains for the level 2, we first duplicate the Hansel chains of the level 1 by generating two identical sets G=(0;1), G=(0; 1) and then by adding the prefix of 0 to the first set and then by adding the prefix 1 to the second set. This results in two sets

$E_{min} = (00; 01)$, $E_{max} = (10; 11)$.

We then cut the maximum element of E_{max} and add it to E_{min}. Thus, the Hansel chains for the size 2 (dimension n=2) are

$\{(00; 01; 11), (10)\}$.

By repeating those operations of *duplicating* and *cutting for* n=3, n=4 and so on, we will be able to build the Hansel chains for any size vectors.

A Boolean function f is **monotone** if for any two Boolean vectors $x = (x_1, x_2, \ldots x_5)$ and $y = (y_1, y_2, \ldots, y_n)$ such that x precedes y, that is $\forall\, i \in \{1, \ldots, n\}$ $x_i \geq y_i$, then $f(y) \geq f(x)$. Such functions divide the set of Boolean vectors into two classes: vectors assigned to the value 0 and vectors assigned to the value 1, thus forming a border between the two classes.

The number of vectors on level l can be obtained by the formula $C_n^l = \frac{n!}{l!(n-l)!}$. For instance if n = 10 there is 1 element on the level 0, and 10 elements on the level 1. Note the maximum number of elements is reached on the level $n/2$ with 252 elements for n = 10 ($C_{10}^5 = 252$). The number of elements increases from the level 0 to the level $n/2$ and decreases to the level n.

7. CONCLUSION

Visual data mining had shown benefits in many areas when used with numerical data. However, classical VDM methods do not address the specific needs of binary data. In this chapter we had shown how to visualize real patterns contained in Boolean data. The first attribute to consider while ordering Boolean vectors is its norm. Using the Boolean norm of the vectors, we are able to split the data into n + 1 groups (n being the number of elements of vectors). Each group is assigned to a vertical position. Then, multiple methods can be used to assign the horizontal position.

We created two different structures to handle data: MDF and YYF. In each structure, horizontal position of vectors is then handled by a specific procedure. Three procedures are specific to the MDF structure and one is specific to the YYF structure.

The first procedure P_1 is specific to MDF and orders vectors with regard to the natural order, converting a Boolean value into a numerical decimal

value. This procedure does not visualize the real structure of data, but, permits direct comparison between multiple functions.

The second procedure P_2 is specific to MDF and orders vectors in order to visualize Hansel chains. This procedure visualizes the structure of the data itself. However, it does not really visualize relations among data but it still permits a direct comparison between several Boolean functions.

The third procedure P_3 is specific to MDF and orders the Hansel chains. This procedure unveils the border between the two classes of elements (0 and 1). In order to keep the MDF structure intact, the border is visualized as being interrupted and thus differences can be visualized between multiple Boolean functions. However, in monotone Boolean functions, vectors belonging to class 0 actually all expand to a vector of class 1. Hence, the border should be continuous.

The last procedure, P_4, is specific to YYF. This procedure visualizes the real border that exists between the two classes of elements. This is done by expanding Hansel chains up and down duplicating some elements in the process. This new approach has proved to be appropriate to handle discovery of patterns in binary data. By further developing these procedures for non-monotone Boolean functions and data and k-valued data structures, we believe that the new approach can be used in variety of applications.

8. EXERCISES AND PROBLEMS

1. Define Hansel chains for $n = 3$ using the chains for $n = 2$ and a recursive procedure described in Section 6.

2. Define Hansel chains for $n = 4$ using the chains for $n = 3$ built in exercise 1 and a recursive procedure described in Section 6.

3. Build a parallel coordinate visualization of the Boolean function $f(x_1,x_2,x_3) = x_1x_2 \vee x_3$. Discuss clarity of this visualization as a way to separate sets of Boolean vectors $\{x\}$ with $f(x) = 0$ from vectors with $f(x) = 1$.

4. Draw Hansel chains for n=3 and visualize the Boolean function $f(x_1,x_2,x_3) = x_1x_2 \vee x_3 x_4$ 3 by marking each element of each chain with its value "1" or "0".

9. REFERENCES

Beilken, C., Spenke, M., Visual interactive data mining with InfoZoom -the Medical Data Set. The 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD '99, Sept. 15-18, 1999, Prague, Czech Republic. http://citeseer.nj.nec.com/473660.html

Ebert, D., Shaw, C., Zwa, A., Miller, E., Roberts, D. Two-handed interactive stereoscopic visualization. IEEE Visualization '96 Conference.1996, pp. 205-210.

Goel, A., Visualization in Problem Solving Environments. Virginia Polytechnic Institute and State University, 1999.MS. Thesis, 64 p. www.amitgoel.com/vizcraft/docs/ masters_thesis.pdf

Groth, D., Robertson, E., Architectural support for database visualization, Workshop on New Paradigms in Information Visualization and Manipulation, 53-55,1998, .pp. 53-55.

Hansel, G., Sur le nombre des functions Bool'eenes monotones de n variables. C.R. Acad. Sci., Paris (in French), 262(20), pp.1088–1090, 1966

Inselberg, A., Dimsdale, B., Parallel coordinates: A tool for visualizing multidimensional Geometry. Proceedings of IEEE Visualization '90, Los Alamitos, CA, IEEE Computer Society Press, 1990, pp. 360–375.

Keim,D., Ming C. Hao, Dayal, U., Meichun Hsu. Pixel bar charts: a visualization technique for very large multiattributes data sets. Information Visualization, March 2002, Vol. 1, N. 1, pp. 20–34.

Kovalerchuk, B., Triantaphyllou, E., Despande, A.,Vityaev, E., Interactive Learning of Monotone Boolean Functions. Information Sciences, Vol. 94, issue 1-4, pp. 87–118, 1996.

Kovalerchuk, B., Vityaev, E., Ruiz, J., Consistent knowledge discovery in medical diagnosis. IEEE Engineering in Medicine and Biology, (Special issue on Data Mining and Knowledge Discovery), v. 19, n. 426–37, 2000.

Kovalerchuk, B., Vityaev, E., Ruiz, J., Consistent and complete data and "expert" mining in medicine. Medical Data Mining and Knowledge Discovery, Springer, 2001:238–280.

Last, M., Kandel, A., Automated perceptions in data mining, invited paper. 1999, IEEE International Fuzzy Systems Conference Proceedings, Part I, Seoul, Korea, Aug 1999, pp.190–197.

Post, F., T. van Walsum, Post,F., Silver, D., Iconic techniques for feature visualization. In Proceedings Visualization '95, pp. 288–295, 1995.

Ribarsky, W., Ayers, E., Eble, J., Mukherja, S., Glyphmaker: creating customized visualizations of complex data. IEEE Computer, 27(7), pp. 57–64, 994.

Shaw, C., Hall, J., Blahut, C., Ebert, D., Roberts. A., Using shape to visualize multivariate data. CIKM'99 Workshop on New Paradigms in Information Visualization and Manipulation, ACM Press, 1999, pp. 17-20.

Ward, M., A taxonomy of glyph placement strategies for multidimensional data visualization. Information Visualization 1, 2002, pp. 194–210.

PART 5

VISUAL AND SPATIAL PROBLEM SOLVING IN GEOSPATIAL DOMAIN

Chapter 17

IMAGERY INTERGRATION AS CONFLICT RESOLUTION DECISION PROCESS

Methods and appraoches

Boris Kovalerchuk, James Schwing, and William Sumner
Central Washington University, USA

Abstract: With the growing use of geospatial data arising from multiple sources combined with a variety of techniques for data generation and a variety of requested data formats, the problem of data integration poses the challenging task of creating a general framework for both carrying out this integration and decision making. This chapter features a general framework for combining geospatial datasets. The framework is task-driven and includes the development of task-specific measures, the use of a task-driven conflation agent, and the identification of task-related default parameters. The chapter also describes measures of decision correctness and the visualization of decisions and conflict resolution by using analytical and visual conflation agents.

Key words: Visual decision-making, geospatial data integration, conflation, task-driven approach, measure of correctness.

1. INTRODUCTION

In a computer environment, almost any two datasets can be combined with the hope that the result is "better" than the sum of the initial datasets. This is not always the case, as the appropriateness of the combination depends upon two primary factors: the quality of the input data and the task-specific goal of the combination process itself.

The goal of *imagery registration* is **integrating** a given image to geospatial coordinates. The goal of *imagery conflation* is the **correlation** and integration of two or more images or geospatial databases. "The process of transferring information (including more accurate coordinates) from one geospatial database to another is known as 'conflation'" [FGDC, 2000]. Typically,

the result of the conflation is a combined image produced from two or more images with: (1) matched features from different images and (2) transformations that are needed to produce a single consistent image. Registration of a new image can be done by conflating it with a registered image.

Conflation has been viewed as a *matching technique* that fuses imagery data and preserves inconsistencies (e.g., inconsistencies between high and low resolution maps, "best map" concept, [Edwards & Simpson, 2002]). This approach tries to preserve the pluralism of multi-source data. The traditional approach [DEMS, 1998] uses an "artistic" match of elevation edges. If the road has a break on the borderline of two maps then a "corrected" road section starts at some distance from the border on both sides and connects two disparate lines. This new line is artistically perfect, but no real road may exist on the ground in that location (see Figure 1).

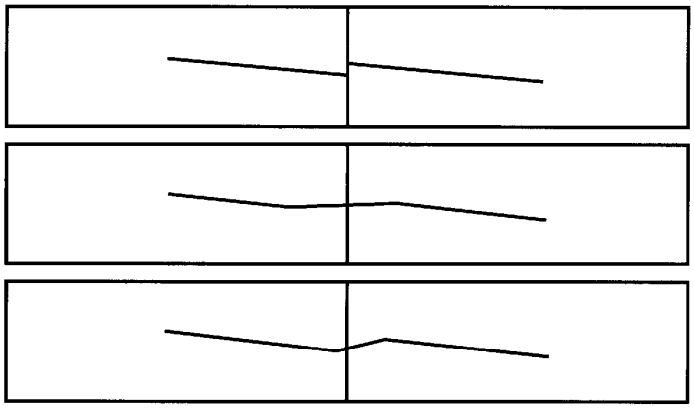


Figure1. Initial mismatch and "artistic" conflations

Thus, *conflation* and *registration* are typical and important parts of *geospatial decision making* process that is highly visual by its nature. The demand for visual geospatial decision making is coming from ecology, geography, geology, archeology, urban planning, agriculture, military, intelligence, homeland security, disaster relief operations, rescue missions, and construction tasks. The range of examples is abundant and includes tasks such as:

- assessment of *mobility/trafficability* of an area for heavy vehicles,
- dynamic assessing *flood damage,* and
- assessing *mobility in the flood area.*

Traditional cartography provided paper maps for analysis and decision-making in these domains. Recently there has been a massive proliferation of spatial data, and the traditional paper map is no longer the final product of cartography. In fact, the focus of cartography has shifted from map production to the management, combination, and presentation of spatial data. Maps can be (and often are) produced on-demand for any number of specialized purposes. Unfortunately, data are not always consistent. As such, data com-

bination (or conflation [Jensen, Saalfeld, Broome, Price, Ramsey & Lapine, 2000; Cobb, Chung, Foley, Petry, Shaw & Miller, 1998; Rahimi, Cobb, Ali, Paprzycki & Petry, 2002]) has become a significant issue in cartography, where mathematical methods and advanced visual decision making must have a profound impact upon cartography.

This task has a dual interest in the process of visual decision making (the focus of this book). First, conflation creates a *base* for making decisions such as selection of transportation routes and construction sites. Second, many decisions are made visually in the process of conflation itself. Conflict resolution in matching features from different sources often requires visual inspection of maps and imagery. Feature f_1 in a geospatial dataset may have two features f_2 and f_3 in another dataset that appear to match f_1. An analyst using **contextual visual information** may resolve the ambiguity by deciding that f_1 should be matched to f_2 only. Such visual problem solving process may involve computing *mathematical similarity measures* between features, analysis of their names and other *non-spatial attributes* as well as using the analyst's *tacit knowledge* about the context of the task and data.

The conflation problem is discussed in several chapters of this book. This chapter provides a wide overview and a conceptual framework including visual aspects. A task-driven approach is elaborated in Chapter 18. Chapter 19 describes algebraic mathematical approach to conflation and a combined algebraic, rule rule-based approach is presented in Chapter 21.

2. COMBINING AND RESOLVING CONFLICTS WITH GEOSPATIAL DATASETS

In essence, the conflation problem is a **conflict resolution decision problem** between disparate data. Inconsistencies in multisource data can be due to scale, resolution, compilation standards, operator license, source accuracy, registration, sensor characteristics, currency, temporality, or errors [Doytsher, Filin & Ezra, 2001]. Conflict resolution strategies are highly *context and task dependent.* Jensen et al. [Jensen et al., 2000] discuss the dependency of conflation on the task under consideration.

In solving a conflation problem, experts are unique in extracting and using non-formalized context and in linking it with the task at hand (e.g., finding the best route). Unfortunately, few if any contexts are explicitly formalized and generalized for use in conflating other images. It is common that the context of each image is unique and not recorded. For example, an expert conflating two specific images may match feature F1 with feature F3, although the distance between features F1 and F2 is smaller than the distance between features F1 and F3. The reasoning (that is typically not recorded)

behind this decision could be as follows. The expert analyzed the whole image as a context for the decision. The expert noticed that both features F1 and F3 are small road segments and are parts of much larger road systems A and B that are structurally similar, but features F1 and F2 have no such link. This conclusion is very specific for a given pair of images and roads. The expert did not have any formal definition of structural similarity in this reasoning. Thus, this expert's reasoning may not be not sufficient for implementing an automatic conflation system. Moreover, the informal similarity the expert used for one pair of images can differ from the similarities the same expert might use for two other images.

There are two known approaches for incorporating context: (1) *formalize the context* for each individual image and task directly and (2) *generalize the context* in the form of expert rules. Here for the first approach, the challenge is that there are too many images and tasks and there currently is no unified technique to for context formalization. The second approach is more general and more feasible, but in some cases may not match a particular context and task, thus a human expert still needs to "take a look." **Visual decision making is necessary.**

Consider a simple example. Suppose that a **task-specific goal** may be to locate individual buildings at a spatial accuracy of +20 meters. Suppose further that there are two spatial datasets available – one set with roads at ±5 meters and the other set containing both roads and buildings at ±50 meters. Obviously, neither dataset can properly answer the question. Now suppose the two datasets are conflated. Is the spatial accuracy of the new image ±20 meters or better? If so, the process is a success. If not, the users will have to either find new data or just accept the inaccuracies in one dataset. As a side note, it is important that the conflation process should not be used to simplify datasets (i.e., combine two datasets into one and delete the original data), but rather to answer specific questions.

The conflation task includes:

- combining geospatial data (typically represented by geospatial features),
- measuring conflict in the combined data,
- deconflicting the combined data, and
- testing the appropriateness of the conflation relative to the stated problem/task definition.

A single common flexible framework is needed that will integrate diverse types of spatial data with the following capabilities [Jensen et al., 2000]: (1) horizontal data integration (merging adjacent data sets), (2) vertical data integration (operations involving the overlaying of maps), (3) temporal data integration, (4) handling differences in data content, scales, methods, standards, definitions, practices, (5) managing uncertainty and represen-

tation differences, (6) detecting and deal with redundancy and ambiguity of representation, (7) keeping some items unmatched, and (8) keeping some items to be matched with limited confidence.

Several challenges have been identified in [Mark, 1999].

- The **representational challenge** – finding a way of merging spatial data from variety of sources without contradiction. Often this challenge cannot be fully met.
- The **uncertainty challenge** – finding a way of measuring, modeling, and summarizing inconsistencies in merged data. Often inconsistencies are inevitable in merging spatial data.
- The **visualization challenge** – finding a way to visualize differences between different digital representations and real phenomena.

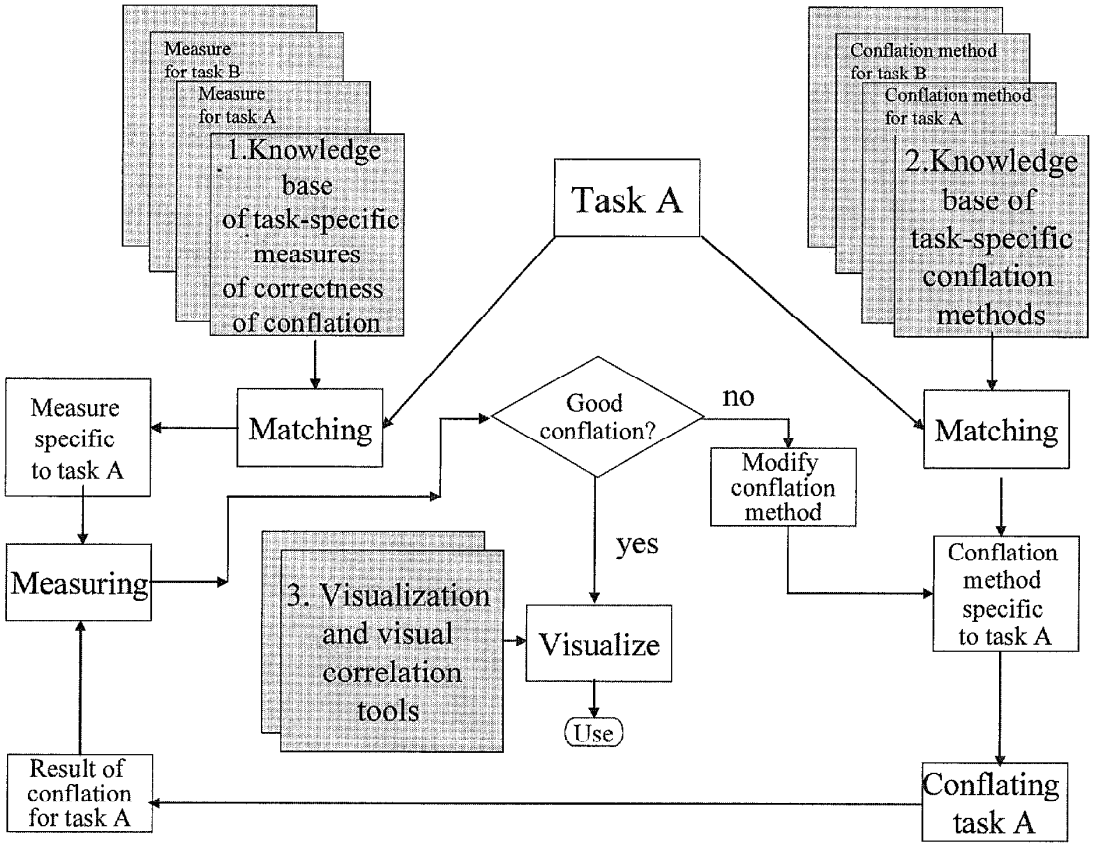


Figure 2. Framework for task driven conflation process

Figure 2 illustrates the framework for the overall process. It includes three systems:

(1) System of task-specific measures of correctness of conflation,
(2) System of task-specific conflation methods, and
(3) System of visualization and visual correlation tools.

This design of the system integrates analytical and visual problem solving processes. We note that different tasks may surely require different accuracies and different measures of correctness of the conflation.

The system of task-specific measures of correctness of the conflation serves a repository of such measures as a part of the **conflation knowledge base**. Different goals along with measures may also require different conflation methods.

The system of task-specific conflation methods is another component of the **conflation knowledge base**. The system of **visualization and visual correlation tools** opens the way for visualizing conflicts in the data being conflated, for portraying relations, and for helping to discover relations.

In Figure 2, a specific task, *Task A,* is matched to both knowledge bases (1) and (2) and measures and methods specific for *Task A* are retrieved. Then the specific conflation method is applied to *Task A* and the result is tested using the measure of correctness specific to *Task A*. If the result is appropriate for *Task A*, then it is visualized and delivered to the end user. Otherwise, the parameters of the conflation method are modified and the procedure is repeated until an acceptable level is achieved.

The conflation process can use a variety of data along with metadata. The geometry and topology data classes that provide information on points, vectors, and structure are most critical data classes.

Quality metadata include: (1) *statistical information* such as random error, bias characteristics of digital terrain elevation data, and location error for a feature, and (2) *expert-based information* such as "topologically clean," "well matched," "highly contradictory," "bias," "consistent around polygons," and "noticeable" (e.g., noticeable can mean edge breaks of approximately 1 to 3 vertical units of resolution).

Typically, data quality is assessed using measures such as accuracy, precision, completeness, and consistency of spatial data over space, time, and theme. It is identified relative to the database specifications (i.e., if the database specification states that objects must be located within ±100 meters, and all objects are located to that accuracy, the data is 100% accurate). As such, appropriate use (or combination) of data is always relative to both the *desired output accuracy* stated in the goal and the *quality of the input data.*

2.1 Rule-based and task-driven approach

Typically the conflation process is guided by using If-Then rules. Below we combine a *rule-based approach* and a *task-driven approach* for the conflation problem by designing a knowledge base. This *knowledge base* (KB) contains two types of rules. The first set of rules, $\mathbf{R_A} = \{\mathbf{R_{Ai}}\}$ derived from user's task. These rules match features $\mathbf{F_1}$ and $\mathbf{F_2}$ of two images I_1 and I_2,

$$R_{Gi}: \mathbf{F}_1 \Rightarrow \mathbf{F}_2 .$$

Such R_{Gi} rules are called **task-driven rules**. Another set of rules in KB are the **task-free rules** that are used when the user is uncertain about the goals of data conflation and cannot formulate a definitive goal for conflation. A subset of these rules, $\mathbf{T}_G$ can be tested for randomly selected potential goals and matched by the time user-identified task rules are ready to be fired. The rule below is an example of a *task-free conflation rule*:

> IF two candidate features f_1 and f_2 from image I_1 are matched to the feature of interest f_3 in the second image I_2
> THEN select that feature from f_1 and f_2 that is closer to f_3 according to Euclidian distance between features:
> $$min\ (D(f_1, f_3), D(f_2, f_3)).$$

In order to contrast the two types of rules, let us generate an example of a **task-driven rule** by modifying the last rule. For instance, we can add a requirement to further test that the smaller distance derived above is less than a threshold $H(A)$ acceptable for user's task A; specifically:

$$min\ (D(f_1, f_3), D(f_2, f_3)) < H(A).$$

2.2 Conflation Methods

Zitova and Flusser's [Zitova & Flusser, 2003] survey of conflation methods and classify them according to their nature (*area-based* and *feature-based*) and according to the four basic steps of image registration: *feature detection*, *feature matching*, *mapping function design*, and *image transformation and re-sampling*. This comprehensive review concludes with the following statement: "Although a lot of work has been done, automatic image registration still remains an *open problem*. Registration of images with complex nonlinear and local distortions, multimodal registration, and registration of N-D images (where N > 2) belong to the most challenging tasks at this moment. . .The future development on this field could pay *more attention to the feature-based methods*, where appropriate invariant and modality-insensitive features can provide good platform for the registration. In the future, the idea of an ultimate registration method, able to *recognize the type of given task* and to decide by *itself about the most appropriate solution*, can motivate the development of *expert systems*. They will be based on the combination of various approaches, looking for consensus of particular results." Zitova & Flusser's vision of the future reflects the process depicted in Figure 2.

The intent here is not to summarize the more than 1000 papers on registration and conflation published in the last 10 years nor repeat the excellent surveys [Zitova & Flusser, 2003; Brown, 1992], but to review some of the techniques developed by the image processing community for integration of raster and vector images relevant to this book. It is also important to note the significant activities of medical imaging community in image integration/registration such as the Second International Workshop on Biomedical Image Registration (WBIR'03) in Philadelphia. Below we list some representative methods from recent research on registration and spatial correspondence presented at this workshop for rigid and non-rigid image registration based on: vector field regularization, curvature regularization, spatio-temporal alignment, entropy, mutual information, variational curve matching, normalized mutual information, K-means clustering, shading correction, piecewise affine transformation, elastic transformation, multiple channels, modalities and dimensions, similarity measures, the Kullback-Leibler distance, block-matching features, voxel class probabilities, intensity-based 2D-3D spines, orthogonal 2D projections. Despite similarities there are significant differences between medical and geospatial imagery. Medical imagery is produced in a more controlled environment with less dynamics and variability in resolution but often more metadata.

Hirose et al. [Hirose, Furuhashi, Kitamura & Araki, 2001] do not assume that fiducial points are known. Rather they automatically extract four corresponding points from images. These points are used to derive an affine transformation matrix, which defines a mutual position relationship between two consecutive range images. Such images can then be concatenated using the derived affine transformation.

Wang et al. [Wang, Chun & Park, 2001] use GIS-assisted background registration that discerns different clutter regions in the initial image frame by using a feature vector composed of vertical and horizontal autocorrelation. The authors also build filters tuned to each class. In the successive frames, they classify each region of different clutter from a contour image obtained by projecting the GIS data and by registering it to the previous image.

Finally, the reader is directed to [Bartl & Schneider, 1995] who demonstrate that knowledge of the geometrical relationship between images is a prerequisite for registration. Assuming a conformal affine transformation, four transformation parameters are determined on the basis of the geometrical arrangement of characteristic objects extracted from images. An algorithm is introduced that establishes a correspondence between (centers of gravity of) objects by building and matching so-called angle chains, a linear structure for representing a geometric (2D) arrangement.

Several other related areas that face similar challenges are *image retrieval* from multimedia databases that use image content [Shih, 2002], *multimedia data mining* [Perner, 2002], and information extraction from heterogeneous sources [Ursino, 2002]. Progress in each of these areas along with progress in image integration should prove to be mutually beneficial.

Pope and Theiler [Pope & Theiler, 2003] and Lofy [Lofy, 2000] apply image *photogrammetric georeferencing* using metadata about sensors combined with the *edge extraction and matching* at three levels of resolution. The last of these systems has been used to automatically register synthetic aperture radar (SAR), infrared (IR), and electro-optical (EO) images within a reported 2-pixel accuracy.

Growe and Tonjes [Growe & Tonjes, 1997] present a *rule-based approach* to imagery registration for automatic control point matching when flight parameters are inaccurate. Prior knowledge is used to select an appropriate structure for matching, i.e. control points from a GIS database, and to extract their corresponding features from the sensor data. The knowledge is represented explicitly using semantic nets and rules. The automatic control point matching is demonstrated for crossroads in aerial and SAR imagery.

A recent special issue of Computer Vision and Image Understanding Journal [Terzopoulos, Studholme, Staib & Goshtasby, 2003] is devoted to non-rigid image registration based on point matching, distance functions, free boundary constraints, iconic features and others. The next step in geospatial data registration is video registration [Shah & Kumar, 2003], which faces many of the same challenges as static image registration discussed above.

The new use of algebraic invariants described in Chapter 19 is a very general image conflation method. For example, it can be used with digitized maps (e.g. in USGS/NIMA vector format), aerial photos and SRTM **data that have no common reference** points established in advance. The images may have **different, (and often) unknown scales, rotations and accuracy**.

The method assumes that map, aerial photo or SRTM data images each have as least 5 well-defined linear features that can be presented as polylines (continuous chains of linear intervals). A feature on one image might be only a portion of the same feature on another image. Also features might overlap or have no match at all. It is further assumed that these well-defined linear features can be relatively easy extracted. The major steps of the method are:

Step 1. Extract linear *features* as sets of points (pixels), S.

Step 2. Interpolate these sets of points S as a specially designed *polyline*.

Step 3. Construct a matrix P of the relation between all *lengths of intervals* on the polyline (see below for more details). These matrixes are computed for all available polylines.

Step 4. Construct a matrix Q of the relation between all *angles* on the polyline. These matrixes are computed for all available polylines.
Step 5. Search *common submatrixes* in the set of matrices P and compute a measure of closeness.
Step 6. Search *common submatrixes* in the set of matrices Q and compute a measure of closeness.
Step 7. *Match* features using the closest submatrix.

At first glance, it is not clear how useful a visual approach can be for algebraic, rule-based and task-driven problem solving. The algebraic approach described in Chapter 19 benefits from visuals by having visual spatial relations as a *source of intuitive problem understanding.* It provides insight for discovering an algebraic formalization of human conflation process. For instance, the matrix for relations between angles of the line segments captures a human way of analyzing similarities between two lines by noticing that relations between angles are similar in both lines. This insight is used in the algebraic approach described in Chapter 19.

2.2.1 Are topological invariants invariant for the conflation task?

Recently at the NSF-funded Workshop on Computational Topology, Bern et al. [Bern, Eppstein, Agarwal, Amenta, Chew, et al., 1999] identified several major shape identification and recognition problems. Two of these topics are most relevant to our main problem of interest – matching/correlating spatial objects, namely, Qualitative Geometry and Multiscale Topology and Topological Invariants. The reason is that often it is impossible to conflate maps and data while preserving all local geometrical properties. The hope is that some more global shape properties can be preserved, discovered, and used for matching spatial objects. Obviously topological properties present some global invariants. They may also provide a more meaningful description than geometric measures [Bern et al., 1999]. However, "Topological invariants such as Betti numbers are insensitive to scale, and do not distinguish between tiny holes and large ones. Moreover, features such as pockets, valleys, and ridges -- which are sometimes crucial in applications -- are not usually treated as topological features at all" [Bern et al., 1999].

Two ideas have been generated to deal with this important problem [Bern et al., 1999]:

- Using topological spaces naturally associated with a given surface to capture scale-dependent and qualitative geometric features. For example, the lengths of the shortest linking curves [Dey & Guha, 1998] or closed curves through or around a hole, which can be used to distinguish small holes from large ones.

- Using the topology of offset or "neighborhood" surfaces for classifying depressions in a surface. For example, a sinkhole with a small opening will seal off as the neighborhood grows, whereas a shallow puddle will not.

The first idea uses highly scale-dependent, geometric characteristics such as length. This idea has an obvious limitation for the conflation problem. Different and unknown scaling of maps, aerial photographs, and sensor data can distort sizes -- a large hole could appear to be small, while a small hole could appear to be large. The second idea just shifts the same limitation to "neighborhood" surfaces.

A fundamentally new mathematical approach is needed for solving the conflation problem. The algebraic approach described below presents such new approach. Algebraic and differential geometry ideas are central to this approach and include establishing homomorphisms and homeomorphisms between algebraic systems. Further, this approach solves the conflation problem for a wide range of realistic settings. The algebraic approach introduces a new class of invariants which are much more robust than common geometric characteristics such as coordinates and lengths. In addition, they detect specific geospatial features better than common topological invariants. Obviously common topological invariants were not designed specifically for solving the conflation problem.

It was noticed in [Bern et al., 1999] that "the most useful topological invariants involve homology, which defines a sequence of groups describing the 'connectedness' of a topological space. For example, the Betti numbers of an object embedded in R^3 are respectively the number of connected components separated by gaps, the number of circles surrounding tunnels, and the number of shells surrounding voids. Technically, the Betti numbers are the ranks of the free parts of the homology groups. For 2-manifolds without boundary, the homology can be computed quite easily by computing Euler characteristics and orientability."

The number of connected components (in the same drainage system) can vary significantly between different maps, aerial photographs, and sensor data for the same area. It depends on characteristics such as human error, map resolution, sensor capabilities and parameters, obstacles (e.g. clouds), data processing methods. Similar problems exist for other topological numbers. Thus, in general, they are not invariant for the conflation problem.

Bern et al. [Bern et al., 1999] suggested that the estimation of topological invariants may be appropriate if it is not possible to determine the topology of an object completely. Unfortunately, in the case of a drainage system this estimate can be far from useful. A given drainage system might have 10 or even 100 connected elements on some maps, but only one or two connected elements on another. In general, objects such as drainage systems, road systems and/or lakes can be presented without some of their components such as

individual canals, roads, and islands, which can change the calculation of their topological invariants. Study needs to be done to determine the extent that Betti numbers and Euler characteristics would be useful for matching incompletely defined spatial objects. Similarly, the extent to which Betti numbers and Euler characteristics would be useful for matching spatial objects that are defined with errors such as roads and canals with incorrect connections needs further consideration.

2.2.2 Rubber sheeting problem: definitions

Dey et al. [Dey, Edelsbrunner & Guha, 1999] address two important computational topology problems in cartography: (1) rubber sheets and (2) cartograms. These problems involve two purposeful deformations of a geographic map:

(1) bringing two maps into correspondence -- (e.g. two geographic maps may need to be brought into correspondence so that mineral and agricultural land distributions can be shown together) and
(2) deforming a map to reflect quantities other than geographic distance and area (e.g., population).

Both tasks (1) and (2) belong to the group of problems that consist of matching similar features from different databases. However, there is an important difference between them and our spatial object correlation task.

Dey and Guha [Dey & Guha, 1998] assume that n reference matched points are given between the two maps for rubber sheeting: "To model this problem let $P \subseteq M$ and $Q \subseteq N$ be two sets of n points each together with a bijection $b: P \rightarrow Q$. The construction of a homeomorphism $h: M \rightarrow N$ that agrees with b at all points of P is popularly known as rubber sheeting [Gillman, 1985]. Suppose K and L are simplicial complexes whose simplices cover M and N: $M = |K|$ and $N = |L|$. Suppose also that the points in P and Q are vertices of K and L: $P \subseteq$ Vert K and $Q \subseteq$ Vert L, and that there is a vertex map v: Vert K $\rightarrow$ Vert L that agrees with b at all points of P . The extension of v to a simplicial map $f : M \rightarrow N$ is a simplicial homeomorphism effectively solving the rubber sheet problem."

Dey and Guha [Dey & Guha, 1998] also review variations of the construction of such complexes K and L. They note that [Aronov, Seidel & Souvaine, 1993] consider simply connected polygons M and N with n vertices each where they show there are always isomorphic complexes $|K| = M$ and $|L| = N$ with at most $O(n^2)$ vertices each. They also prove that sometimes $\Omega(n^2)$ vertices are necessary and they show how to construct the complexes in $O(n^2)$ time. In addition, Dey notes that [Gupta & Wenger, 1997] solve the same problem with at most $O(n + m \log n)$ vertices, where m is the minimum number of extra points required in any particular problem instance.

2.2.3 Related computational topology problems

Below we discuss relationship between our framework and the tasks discussed by [Bern et al., 1999].

Shape Reconstruction from Scattered Points is an important part of our algebraic approach. Methods for reconstructing a linear feature shape using a criterion of **maximum of local linearity** can be developed using this approach.

Shape Acquisition. Matching/correlating spatial objects requires shape **acquisition** from an existing physical object. We believe the solution of this problem depends on developing a formal mathematical definition for features such as roads and drainage system as complex objects in terms that coincided with the USGS/NIMA Topological Vector Profile (TVP) concept.

Shape Representation. Bern et al. [Bern et al., 1999] list several representation methods: unstructured collections of polygons ("polygon soup"), polyhedral models, subdivision surfaces, spline surfaces, implicit surfaces, skin surfaces, and alpha shapes.

In light of this, we believe that an **algebraic system representation** such as that described below, which includes scale-dependent but **robust** invariants, axioms and permissible transformations should be developed.

Topology Preserving Simplification. It is critical for many applications to be able to replace a polygonal surface with a simpler one. However, such a process is "notorious for introducing topological errors, which can be fatal for later operations" [Bern et al., 1999].

A generic approach developed in [Cohen, Varshney, Manocha, Turk, Weber, Agarwal, et al., 1996] can be used where a simplified 2-manifold is fitted into a shell around the original. In addition, we are developing a specific method for the simplification of linear features critical for the algebraic approach using the similar ideas.

Our goal is to enforce algebraic invariants using simplification. The main idea is that the simplification criteria should not be completely defined in advance but be adjusted **using extensive simulation experiments and machine learning tools** to identify the practical limits of robustness of the algebraic invariants.

This includes the classification of linear features to identify simplification options. For example, let a be **a** linear feature, **b** its simplification, λ a measure of the closeness between **a** and **b** with $\lambda(\mathbf{a},\mathbf{b}) < \delta$, where δ is a limit of acceptable simplification. This limit can be developed as an adjustable parameterized function of a feature **a**, $\delta = \delta(\mathbf{a})$.

3. MEASURES OF DECISION CORRECTNESS

A wide range of measures of correctness of the results have been developed, they need to be analyzed for building a common ground for geospatial decision making. Systems can be classified in a way that varies in their level of exact definition and presentation of measures of closeness and their confidence as follows:

- *High level.* A system makes exact and clear measure of closeness on the design stage.
- *Medium level.* A system does not measure closeness of spatial objects in advance, but provides a user with interactive tools such as the curve-matching cursor.
- *Low level.* A system mostly relies on an informal human perceptual measuring mechanism, providing similar graphical presentation of entities and some pointing mechanisms between them.

A major concern with the first approach (referred to as the high level above) is in the nature of a measure of closeness as a *single numeric indicator*. If we try to catch the closeness of data sets with 1000 points in each of them, this measure may capture:

- *average closeness* – the averaged distance between entities using some formal definition of the measure of closeness between individual points,
- *optimistic closeness* – a measure with higher weights on smaller discrepancies in distances between entities using some formal definition of the measure of closeness between individual points,
- *pessimistic closeness* – a measure with higher weights on larger discrepancies in distances between entities using some formal definition of the measure of closeness between individual points.

Probability theory, mathematical statistics, functional analysis, fuzzy logic, machine learning, and pattern recognition provide plenty of examples of measures for all three alternatives and many intermediate measures between them. The problem is that each of them hides/ignores some of the *distortions*, which may be critical for a specific task. A custom design of measures of closeness for a specific task and spatial entities falls into another trap -- *loss of the universality* of the approach and tools. In this case, there is the need for highly task-specialized (unique) measure designs. Successful introduction of measures of the closeness helps to evaluate conflation process quality and to resolve ambiguities in matching features.

Multidimensional measures of conflation correctness and their visual presentation are a way to meet these challenges. Figure 3 illustrates a 3-dimensional measure with three components: (1) optimistic (short arrow), (2) average (medium arrow) and (3) pessimistic (large arrow). We also utilize features such as: location, rotation, blinking, and lighting to present contra-

dictory characteristics of conflated maps. Figure 3 visualizes a conflation procedure, showing directions of movement and matching lines for moving the upper map to conflate the two maps. This visualization also can serve as a guide in manual visual conflating, when, for instance, an automatic conflation fails.

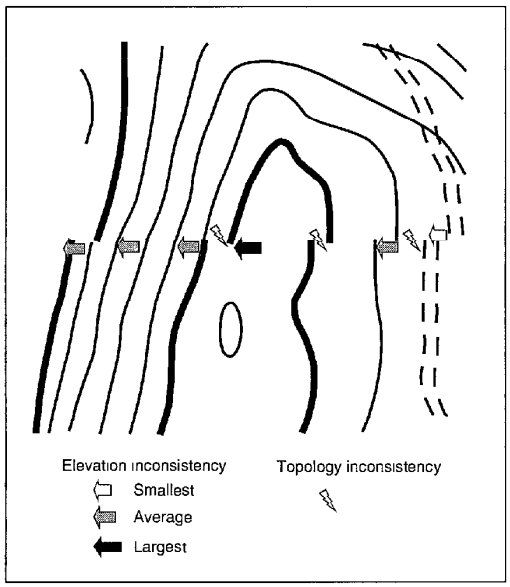


Figure 3. Multidimensional measures of correctness

Figure 3 uses the symbol ↯ to portray contradictory breaks in elevation lines. This is an example of our matching attribute approach: if two objects *a* and *b* contain contradictory data then *a* is associated with the attribute "Contradicts *b*" and *b* is associated with the attribute "Contradicts *a*", both attributes are portrayed by the same symbol ↯ with the same color and this symbol is attached to the both objects. Similarly, if two other objects contradict each other, then the same symbol is attached to both objects.

Contour lines match on the left side of Figure 3, but are shifted by one. An examination of this limited area could lead to the mistaken conclusion that the conflation was better than it really is. This visualization makes the conflation decision-making process apparent.

To distinguish two pairs of contradictory objects, the symbol of contradiction is used with an attribute of a different color for these pairs. Thus, matching attributes support *visual correlation between contradictory objects.* For portraying the *magnitude of contradiction* between spatial objects, we use measures of conflation correctness (including fuzzy logic measures). The symbol of contradiction is rotated in accordance with this measure value: the larger the rotation angle (up to 180°), the larger the contradiction.

A conflation procedure will be called a **consistent** conflation procedure if a resulting combined map: preserves relative distances between elevation

lines and objects on each map preserves absolute elevations and locations on the border connecting maps, and avoids discontinuity of the objects on the border.

In the situation when consistent conflation is impossible, some non-linear distorting conflation methods are used as part of USGS standard. These methods differ in the number of neighboring elevation profiles involved in interpolation. The number of profiles depends on the number of vertical resolution units where the edge breaks. For such situations, measures of correctness of conflation are compositions of:

- measures of the distortion of relative distances on both maps,
- measures of the distortion of absolute elevations and locations of objects on the edge and on the interpolated profiles,
- measures of the discontinuity on the border, and
- measures of the distortion of topology of objects due to composing two maps.

We represent each of these measures as one of the three measures forms described in the previous section: *pessimistic, optimistic, and average measure forms.*

Traditionally, all measures use a Euclidian *separation distance* between parts of the spatial object. For instance, the standard *measure of vertical distortion* used in digital elevation models (DEM) is a *root-mean-square (RMSE) error statistic, E* of an average Euclidian closeness between elevation data. Next two thresholds, T_1 and T_2 are set up for these error statistics:

If $E < T_1$ then it is desirable to use conflated data;
If $T_1 < E < T_2$ then conflated data can be retained temporarily before better conflated data will be available;
If $T_2 > E$ then conflated data are rejected.

For instance, the US Geological Survey provides thresholds $T_1 = 7$ m and $T_2 = 15$ m for the 7.5-minute Digital Elevation Model. These thresholds are not flexible and are task-independent. If $E = 7.4$ m is desirable, why then would $E = 7.6$ m not be desirable. In essence, the answer should depend on the task, for one task, it might be desirable, for another task, it might not. Fuzzy logic membership functions provide more flexibility in setting thresholds and accommodating task specifics. Figures 4 and 5 translate thresholds T_1, and T_2 into a fuzzy logic format. The first membership function might represent the *desired* RMSE. Similarly, the second and third membership functions data might represent *temporarily retained* and data *rejected* values of RSME respectively. Figure 5 may better fit a practice when there is not much difference between 6.9m and 7.0m RMSE in contrast with a sharp border of less than 7.0 m as in Figure 4.

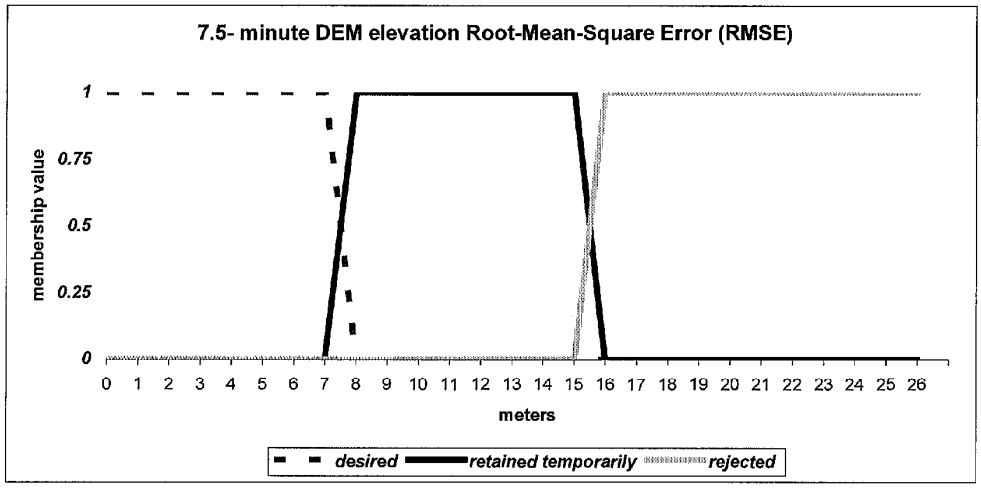


Figure 4. Use of fuzzy logic membership functions - sharp distinctions

Figure 5 shows a more flexible version of these functions with wider sets of uncertain values between desired, retained temporarily, and rejected values of RMSE, which can be more realistic measures in some tasks.

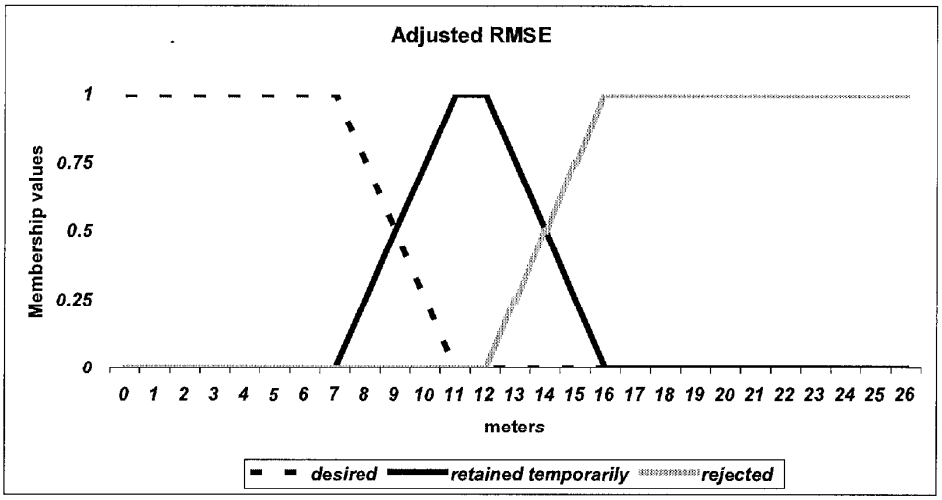


Figure 5. Use of fuzzy logic membership functions - more relaxed distinctions

These membership functions are constructed and stored for each specific task in the conflation knowledgebase providing task-specific formalizations of concepts "desired", "retain temporarily", and "reject." Different tasks may have different levels of desired distinction and their definitions, as we have illustrated in Figures 4 and 5. The value of RMSE is a property of the data and exists *independently of the task*, but its *evaluation is a task-specific*. A fuzzy logic approach also permits introducing a hierarchy of fuzzy logic membership functions to capture a variety of task-specific evaluations. Next, we consider a more general approach based on context spaces [Kovalerchuk

& Vityaev, 2000] that can capture a richer collection of task specifics and context.

4. VISUALIZATION

There are several stages in the problem solving process: *discovering*, *implementing* (in hardware and software), *using*, and *presenting results*. To associate a *visual process* with this, all major steps should be *visual*. A *complete visual step* is performed visually, represented visually (visualized) and animated when considering dynamic steps.

Most visualization activities concentrate on two stages: *using* a well-defined process for solving tasks and *representing results* [Mille, 2001]. Visualization of these stages permits speeding up the process and assures *quality control*. Examples are abundant, e.g., finding North by using stars such Polaris and the Big Dipper and Little Dipper stars. Animation of Pythagorean Theorem proof shown in Figure 4 in Chapter 1 provides another example of a visual process. The first example has been very useful for solving the navigation problem for centuries and the second example has been used in education for two millennia. AutoCAD provides examples of *visual implementation stage*. Much less has been done for visualizing *process discovery*. Typically, on this stage, visuals serve the role of an informal *insight for process algorithm development*. There are historical facts that such insights played a critical role in the whole problem solving process. Consider again Albert Einstein's evidence quoted in Chapter 3. This is the most challenging and creative stage of problem solving. *Visual and spatial data mining* have recently emerged as major tools for this stage.

Modern geospatial studies form a natural domain for advanced visual decision-making techniques where typical the spatial relations between objects such as larger, smaller, above, below are visual by their nature. Cognitive science research reviewed in Chapter 3 indicates that human reasoning with such **spatio-visual relations** is more efficient than with relations that are only **visual relations** (e.g., cleaner – dirtier).

As an illustration, consider two vector image data sets consisting of 1497 line segments and 407 line segments respectively. Are these two images of the same scene? Figures 6 and 7 display both the data sets and the parts of the sets that are held in common and those that are unique.

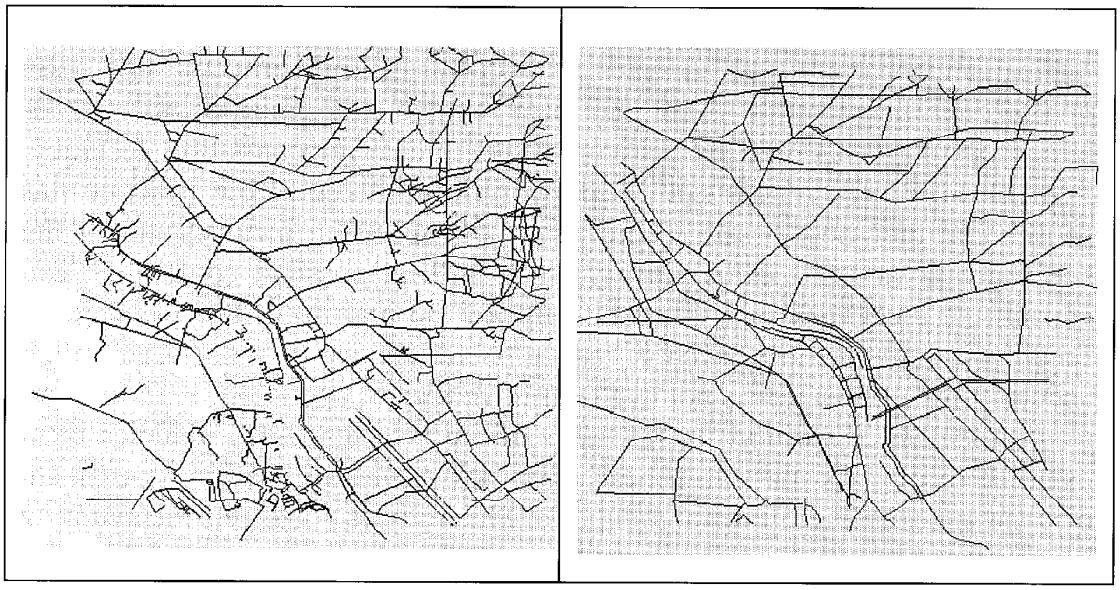


Figure 6. Vector image data sets of 1497 and 407 line segments

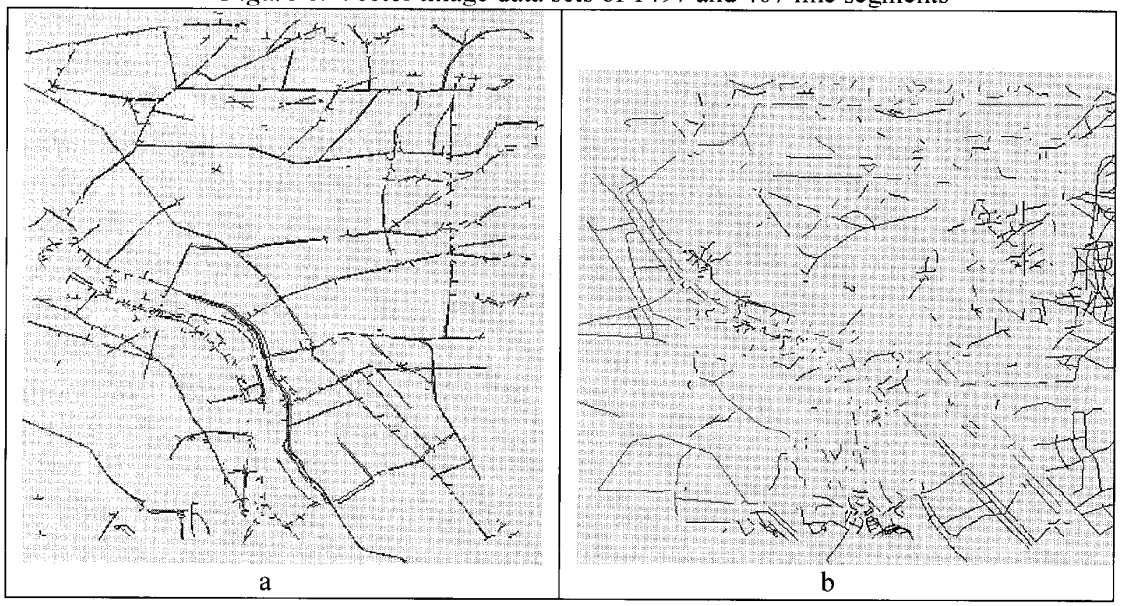


*Figure 7.*Common (a) and unique (b) segments for the vector data sets in Figure 6. These segments were found using methods described in Chapter 19.

The number of common polylines supports the conclusion that these are two images of the same scene. The number of polylines not in common illustrates the need for additional information to know whether these are the result of incomplete or faulty feature spaces, different image resolutions, or changes in the scene between the times of acquisition. With additional information from the metadata associated with this vector data or perhaps from other sources, it may be possible to quantify the differences that this visualization makes clear.

A tight link between visual decision-making and conflation is clarified when we notice that the decision-making task provides a specific goal for conflation. We call such a conflation approach a **task-driven (or mission**

specific) conflation approach. This approach shows the synergy of mathematical and spatio-visual analyses in decision-making process. The *task-driven approach* intends to conflate data relative to a given decision problem. Different tasks may set different requirements for conflation accuracy. Sometimes a user requests a "best possible conflation" but practice has shown that the "best possible conflation" might be too inconsistent for the specific task at hand. Alternatively, such a "best possible conflation" can consume critical time and effort far beyond the real need. For instance, in time critical applications, such as rescue operations we may not need a high accuracy for an area that is not in a restrictive area of operation.

Assessment of mobility/trafficability of the area for heavy vehicles is an example of the task that can benefit from task-driven approach. Assessment of differences in elevations and depth of water zones (both represented by z coordinate distribution in the area) are among critical components of this task. This task may be less sensitive to errors in assessing horizontal distances in x and y coordinates than in z coordinates in each of the maps being conflated. For instance, a two-meter error for x and y might be acceptable while such a deviation might be too large for z coordinate.

Another task could be the dynamic *assessment of flood damage* to an area by combining a map of the area before flood and aerial photos taken every two hours since the flood began. The quantitative goal of this task is computing the area of the flood zone under water. Thus, in contrast with the mobility task, this task may need higher accuracy in the values of x and y coordinates while the z coordinate might need to be less accurate. Requirements for a third task of assessing *mobility in the flood zone* may differ from both of the first two tasks. It may require high accuracy for all three x, y, and z coordinates.

In order to concentrate efforts for the task-driven approach, we need to identify **specific tasks** more formally. Specifically, it can be represented as a triple $A=<G, K, D>$, where G is a *goal*, K is a *domain specific knowledge* that includes the *ontology* and D is the available *data*. The goal G for a **well-posed task** provides also a *criterion*, C_G, for identification that the goal of the task, A, has been reached, i.e., $C_G(A) = 1$ (true).

5. CONFLICT RESOLUTION BY ANALYTICAL AND VISUAL CONFLATION AGENTS

In this section, we demonstrate how a set of task-driven intelligent **conflation agents** can accomplish the task-driven conflation approach. An agent can carry out a single task while a community of agents can carry a wide ar-

ray of tasks. Such agents operate with multiple feature representations and resolve conflation conflicts using rules or other conflict resolution strategies according to the task at hand. For instance, if the task is a global strategic overview of country's conditions, then a **strategic conflation agent** is activated. If the goal is to support a local reconnaissance unit, then the system monitor should activate a specialized **local reconnaissance conflation agent**.

In [Doytsher et al., 2001; Rahimi et al., 2002] hierarchies of spatial **conflation agents (CA)** are suggested. The top-level agents are defined according to the type of matching geometric elements they use: points, lines, or areas. Agents based on matching points are called **point agents**. Next point agents are classified in subcategories: For instance, some agents can match images using *corner points* on buildings (*building agents*).

Other agents can match images using *distinguishing points* on roads such as intersections and *turning points*. Agents based on lines (**line agents**) can be classified further similarly as *building, road, or railroad agents*. For instance, a line building agent matches images using lines on buildings such as roof lines and agents based on area (area agents, e.g., lake agent). The reasoning behind the introduction of the dynamic selection of conflation agents versus a static approach with a single conflation agent is the flexibility of the *dynamic approach*. The dynamic approach is **task-specific**; a specialized agent can be selected depending on the task at hand. A dynamic system can monitor and map discrepancies related to a specific user's task and select an appropriate conflation agent to resolve the problem.

The prototype described in [Rahimi et al., 2002] provides a set of menus for a user: (i) to declare the conflation tools and their parameters to be used, (ii) to select method for determining matched features, and (iii) to select method for evaluating links. Below we briefly describe some conflation agents [Doytsher et al., 2001].

The **point agent** (PA1)

- selects points (nodes) as counterpart features and
- builds a local rubber-sheeting transformations based on selected counterpart points, and
- converts one map to another map by using the found transformations.

This agent is adequate only for cases where rubber-shitting between known control points does not create significant errors in matching intermediate points that differ from control points.

The **line agent (LA1)** conflates maps by running code that implements the following sequence of algorithmic steps:

- detecting counterpart linear features,

- partitioning the whole map into sub-regions according to the network of counterpart lines,
- transforming counterpart elements to their new positions, and
- transforming the remaining elements within each subregion according to the boundary transformation.

Chapter 19 provides another **line agent (LA2)** based on the algorithm called Algebraic BSD (algebraic sequential binary division).

Conflation agents [Rahimi et al., 2002] fulfill several functions:

- eliminating non-matches using simple measures that can be quickly computed,
- determining potential matches using more subtle checks such as attribute set and value similarity, and
- determining whether the overall matching score computed as a function of attribute score, geometry score, topology score and filter score, exceeds the threshold level.

Topological matching approaches are based on graph theory [Lynch & Saalfeld, 1985] and artificial intelligence methods using if-then rules [McKeown, 1987; Cobb et al., 1998].

The overall score approach taken in [Rahimi et al., 2002], where individual scores are combined in a single overall matching score has well-known drawbacks -- it is hard to justify a specific form of the converting function. Different converting functions can provide different feature matches. This is a common problem of converting a multi-criteria task to a single-criteria task. Functions can follow different heuristic strategies (e.g., "optimistic", "pessimistic", or "average") as we have discussed above. Often these choices are highly subjective. In the truly multi-criteria situations where different converting functions provide different matching results, we suggest using a **visual conflation agent**. This agent provides tools for a human expert to analyze the context of the situation in depth using unique human visual abilities. An attribute matching function is introduced in [Rahimi et al., 2002]. In this approach, each feature object is considered as a set of attribute-value pairs:

$$((a_{i1}, v_{i1}), (a_{i2}, v_{i2}), \ldots, (a_{ik}, v_{ik}), \ldots, (a_{in}, v_{in}))$$
$$((a_{j1}, v_{j1}), (a_{j2}, v_{j2}), \ldots, (a_{jk}, v_{jk}), \ldots, (a_{jn}, v_{jn})),$$

where a_{ik} identifies attribute a_k for feature f_i and v_{ik} identifies the value of attribute a_k for feature f_i and terms for feature f_j on the second line are defined similarly. For matching numeric attributes a_k, membership matching functions are used and for linguistic attributes similarity tables are used. The overall Matching Score (MS) for attributes is given by:

$$MS_{i,j} = (\Sigma_{k=1..N} [S_k(f_i, f_j) \times W_k])/N,$$

where $S_k(f_i, f_j)$ is the similarity function between features f_i and f_j for their attribute a_k, N is the number of attributes that are common to both features f_i and f_j, and W_k is the weight computed by the rule-based expert system for the attribute a_k. For instance, a rule for computing weights W_k and W_m for attributes a_k and a_m could be:

$$If\ v_{1k} = 1\ \&\ v_{2m} = 3\ \&\ v_{1m} = v_{2m} = 10,\ then\ W_k = 0.8\ \&\ W_m = 0.4.$$

Lakin [Lakin, 1994; Lakin 1987] described **Visual Agents (VAs)** as software entities, which assist people in performing graphical tasks, such as making a text-and-graphic record of a group meeting, live and on-the-fly, and showing it to participants during the meeting to enhance collaboration. The group members can see the record as it is being made, offering suggestions and corrections. For instance, a visualization agent can act as a whiteboard assistant helping to graphically record the conversation and concepts of a working group on a large display. The visualization agent can help display on-the-fly global objectives, immediate goals, tools, factual data and R&D options that are been discussed by the group that is making decision on a business strategy.

Similarly, a conflation visualization agent may have a complete access to imagery analyst's actions doing the conflation in collaboration with other analysts and software conflation agents. Thus, it can be true visual human-computer collaboration in problem solving. The visual agent includes computational engines for processing text-graphic activity, both static images resulting from the activity as well as actual moment-to-moment dynamics of the activity itself.

6. CONCLUSION

Imagery integration is a conflict-resolution decision problem among disparate data with inconsistencies due to scale, resolution, compilation standards, source accuracy, registration, and many other factors. There are representational challenges and others from uncertainty and how to visualize it. This chapter reviews several imagery integration techniques and concludes that no single method is suitable for all data sets and for every task. The most effective approach is to create a general framework for carrying out image integration and develop task-specific measures of decision correctness for each process.

It was shown that a set of task-driven intelligent conflation agents can accomplish the task-driven conflation approach. An agent can carry out a sin-

gle task while a community of agents can carry a wide array of tasks. Such agents operate with multiple feature representations and resolve conflation conflicts using rules or other conflict resolution strategies according to the task at hand.

7. ACKNOWLEDGEMENTS

This research has been supported by a University Research Initiative grant from the US National Geospatial-Intelligence Agency (NGA) that is gratefully acknowledged. Thanks to S. Cento and P. Brennan for providing data.

8. EXERCISES AND PROBLEMS

1. Suggest a classification of task-driven conflation rules that are conceptually described in the section devoted to rule based- and task driven approach.

2. Design a framework to conflate two Digital Elevation Models (DEM) with different resolutions and areas of coverage.

3. Design a way to visualize the areas of agreement and disagreement in this conflation and use this information to improve the conflation.

4. Design a framework to conflate a raster satellite image and vector stream data with different resolutions and areas of coverage.

5. Design a way to visualize the areas of agreement and disagreement in this conflation and use this information to improve the conflation.

9. REFERENCES

Aronov, B., R. Seidel and D. Souvaine. On compatible triangulations of simple polygons. Comput. Geom. Theory Appl. 3, 27—35, 1993

Bartl, R., and W. Schneider. Satellite image registration based on the geometrical arrangement of objects. Proc. SPIE, v. 2579, p. 32-40, 11/1995

Bern M. et al. Emerging challenges in computational topology. cite-seer.nj.nec.com /bern99 emerging.html, 1999.

Bern, M., Eppstein, D., Agarwal, P., Amenta, N, Chew, P. at al. Emerging challenges in computational topology, 1999, citeseer.nj.nec.com/bern99emerging.html

Brown, L. A Survey of Image Registration Techniques, ACM Computing Sur-veys,vol.24 (4),pp. 325--376, 1992, citeseer.nj.nec.com/brown92survey.html

Cobb, M. Chung, M., Foley, D., Petry. F., Shaw, K., and Miller, H., A rule-based approach for the conflation of attributed vector data, GeoInformatica, 2/1, 1998, 7-36.

Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks Jr., F., and Wright, W. Simplification envelopes. In Rushmeier, H., Ed., Proc. SIG-GRAPH '96, 119–128. Addison Wesley, 1996

Dey T., Guha, S. Computing homology groups of simplicial complexes. JACM, 45(2):266–287, March 1998.

Dey, T., H. Edelsbrunner, and S. Guha. Computational Topology. In Advances in Discrete and Computational Geometry (Contemporary mathematics 223), ed. B. Chazelle, J. E. Goodman, and R. Pollack, American Mathematical Society, 109—143, 1999. http://citeseer.nj.nec.com/dey99computational.html

Digital Elevation Model Standards, 1998, USGS http: //rockyweb.cr.usgs.gov/nmpstds/ demstds.html

Doytsher, Y. Filin, S., Ezra, E. Transformation of Datasets in a Linear-based Map Conflation Framework, Surveying and Land Information Systems, Vol. 61, No. 3, 2001, pp.159-169.

Edwards D., Simpson J. Integration and access of multi-source vector data, In: Geospatial Theory, Processing and Applications, ISPRS Commission IV, Symposium 2002, Ottawa, Canada, July 9-12, 2002, http://www.isprs.org/ commission4/proceedings/pdfpapers/ 269.pdf

Federal Geographic Data Committee FGDC-STD-999.1-2000, 2000, http://www.bts.gov/gis/ fgdc/introduction.pdf

Gillman D., Triangulation for rubber-sheeting, Auto Cartography 7, 191-197, 1985

Growe S., R. Tonjes, A Knowledge Based Approach to Automatic Image Registra-tion, International Conference on Image Processing 97 (ICIP 97), 26-29 October 1997, Santa Barbara, USA, ftp://ftp.tnt.uni-hannover.de/pub/papers/1997/ICIP97-SGRT.pdf

Gupta, H., and Wenger, R. Constructing pairwise disjoint paths with few links. Technical Report OSU-CISRC-2/97-TR16, The Ohio State University, Columbus, Ohio, 1997. http://citeseer.nj.nec.com/gupta97constructing.html

Hirose, M., Furuhashi, H., Kitamura, N., and Araki, K. Method for automatic regis-tration using real-time multiview range images. Proc. SPIE Vol. 4572, p. 174-182, 10/2001

Jensen J., Saalfeld, A., Broome, F., Price, K., Ramsey, D., and Lapine, L. Spatial Data Acquisition and Integration, 2000, NSF Workshop GIS and Geospatial Activities. Accessed June 2001 http://www.ucgis.org/research_white/data.html

Kovalerchuk B., E. Vityaev, Data Mining in Finance: Advances in relational and hybrid methods, Kluwer Acad. Publ., 2000.

Lakin, F. Visual Grammar for Visual Languages, Proceedings of AAAI-87, confer-ence of the American Association for Artificial Intellugence, Seattle, Washington, July 1987, pp 683-688.http://www.pgc.com/pgc/home-stuff/papers-archive/vizgram-abstract.html http://www.pgc.com/pgc/home-stuff/papers-archive/dave.ps

Lakin, F., A Visual Agent For Performance Graphics, The 1994 AAAI Spring Sym-posium on Believable Agents, Stanford University, March 1994, http://www.pgc.com/pgc/home-stuff/papers-archive/dave-abstract.html

Lofy B., High Accuracy Registration and Targeting, 29th IEEE Applied Imagery Pattern Recognition Workshop (AIPR'00), October 16 - 18, 2000, Washington, D.C., pp. 235-244

Lynch, M. and Saalfeld, A., Conflation: Automated map compilation -- A video game approach, Proceedings, AutoCarto 7, 1985, Washington, 343-352.

Mark, D., Geographic information science: critical issues in an emerging cross-disciplinary research domain, NSF workshop to assess the needs for basic research in Geographic Information Science and Technology, January 14-15, 1999.

Mille, H. (Ed) Geographic Data Mining & Knowledge Discovery, Taylor and Francis, 2001

McKeown, Jr., D.R., 1987, The role of artificial intelligence in the integration of remotely sensed data with geographic information systems, Geoscience And Remote Sensing, Ge-25/3,330-348.

Perner, P., Data Mining on Multimedia Data, Springer, 2002.

Pope P., Theiler, J.,Photogrammetric Image Registration (PIR) of MTI Imagery, Space and Remote Sensing Sciences Group, Los Alamos National Laboratory, Los Alamos, NM,2003, http://nis-www.lanl.gov/~jt/Papers/pir-spie-03.pdf

Rahimi, S., Cobb, M., Ali, D., Paprzycki, M., Petry, F. A Knowledge-Based Multi-Agent System for Geospatial Data Conflation, Journal of Geographic Information and Decision Analysis, 2002, Vol. 6, No. 2, pp. 67-81

Shah, M., Kumar, R., (Eds.) Video Registration, Kluwer, 2003

Shih, T., Distributed Multimedia Databases: techniques and applications, Idea Group Publ., 2002.

Terzopoulos D., C. Studholme, L. Staib, A. Goshtasby (Eds) Nonrigid Image Registration, Special issue of Computer Vision and Image Understanding Journal, vol. 89, Issues 2-3, pp. 109-319, 2003

Ursino, D. Extraction and exploitation of intensional knowledge from heterogeneous inforlation sources, Springer, 2002

Wang, J., Chun, J., and Park, Y.W. GIS-assisted image registration for an onboard IRST of a land vehicle. Proc. SPIE Vol. 4370, p. 42-49, 2001

Zitová B., Flusser J., Image registration methods: a survey, Image and Vision Computing. 21 (11), 2003, pp. 977-1000

Chapter 18

MULTILEVEL ANALYTICAL AND VISUAL DECISION FRAMEWORK FOR IMAGERY CONFLATION AND REGISTRATION

George G. He[1], Boris Kovalerchuk[2], and Thomas Mroz[3]
[1]*Pacific Northwest National Laboaratory, USA*
[2]*Central Washington University, USA*
[3]*National Energy and Technology Laboratory, USA*

Abstract: This chapter addresses imagery conflation and registration problems by providing an Analytical and Visual Decision Framework (AVDF). This framework recognizes that pure analytical methods are not sufficient for integrating images. Conflation refers to a process similar but more complex than what is traditionally called registration, in the sense that there is, at least, some conflicting information, which predates it and post conflation evaluation that postdates it. The conflation process studies the cases of two or more data sources where each has inaccuracy and none of them is perfect. The chapter covers complexity space, conflation levels, error structure analysis, and a rules-based conflation scenario. Without AVDF, the mapping between two input data sources is more opportunistic then definitive. A partial differential equation approach is used to illustrate the modeling of disparities between data sources for a given mapping function. A specific case study of AVDF for pixel-level conflation is presented based on Shannon's concept of mutual entropy.

Key words: Imagery conflation, registration, analytical and visual decision framework, complexity space, conflation level, rule base, entropy, mutual information.

1. INTRODUCTION

It is self-evident that major modern scientific and technological endeavors; e.g., precision farming, space and resources (oil and gas) exploration,

and security and monitoring, use imagery for diagnostics, measurement, feature extraction and decision making.

Scientists, engineers and managers base their decisions on increasingly complex and higher dimensional images captured by new instruments and/or generated using models and algorithms though vastly different scales. These images may be composed from different viewing angles with different physical characteristics and environment constraints. Common to these scientific and application efforts is the challenge of performing information assimilation from multiple modality imagery sources to provide sufficient evidences so that a decision can be made with incomplete information and under operational constraints; e.g. real time practices.

Analytical and Visual Decision Making (AVDM) framework refers to a process using visual environments by and/or for decision makers to acquire quality information to support spatial decision making. This framework advocates the method of *mission specific approach* (MSA). Mission specific is defined as a generic scope augmented with a specific task; e.g., map making is a generic job whereas making a road map in area X is a mission specific task. The integration of information from different maps is in general a generic conflation job, whereas the conflation of roads for trafficability evaluation using different data sources in a well-defined area will be a mission specific task. In other words, a conflation process will have a set of data sources with a well-defined time, spatial, and attribute framework within which conflicts can be modeled and managed. In general, the state space attributes outside the conflict extents serve as the reference framework.

In general, visual decision making is a nonlinear process either purely visual or combined with analytic means. For example, conflation can be either a very complex decision making task for trafficability assessment under a combat situation (visual) or a simple translation function for a well mapped local street from high quality resolution imagery (analytic). Prospecting in oil and gas exploration is a typical conflation type decision making process using combined visual and analytic means.

The purpose of conflation, according to the National Technical Alliance (NTA) is to create a third dataset that is better than either of the original sources by combining information from the two. The report by Swiftsure Spatial Systems Inc. [2002] concluded that no one has yet achieved fully automated conflation; and vector-to-imagery conflation is required for future development of the method.

Conflation is a process of identification, correction, and synthesis of disparate information including individual features from multiple imagery (literal and non-literal) and/or vector sources. Conflation consists of three types: vector-to-vector, imagery-to-imagery and imagery-to-vector or vice versa.

For **vector images**, with identified features (e.g., ESRI shape file format) conflation means finding features or segments *both matched and unmatched.* For **raster imagery**, theoretically, two images are conflated if values of *all corresponding pixels are equivalent*, that is the matching ratio R=1.0 in the object space, given fully registered and calibrated spatial and spectra response. For conflation between **vector and raster** images, **conflation** means the establishment of *correspondence for features* from all given objects.

AVDM framework for mission specific (task specific) conflation is a process of using visuals to reduce the degrees of freedom and/or increase the efficiency for identifying the relationships between data sets. Here the task is a triple A=<G, K, D>, where G is a goal, K is domain specific knowledge that includes the domain ontology and D is available data. The goal G for a well-posed task provides also a criterion for identification that the goal of the task A is reached. In formal terms, the goal criterion can be expressed as some predicate, C_G, such that if $C_G(A)=1$ (true) for the task A then the goal G has been reached.

To a certain extent when a well-defined mapping function exists between two data sets, registration can be treated as a simplified case of conflation. Extensive review of registration methods can be found in [Brown, 1992; Zitová & Flusser, 2003]. A NSF-funded research-planning workshop on approaches to combat terrorism [Moniz & Baldeschwieler, 2002] also listed image registration as an open problem:

> ...an important area of research is the registration of images from different times and modalities. Registration of such images onto a single coordinate system is vital for automated analysis but can be extremely challenging. The lack of robust image registration algorithms remains a limiting factor in many fields.

Registration is closely related to, but differs from conflation. The goal of registration is to provide geo-reference for a pair of images *without matching individual features*. Traditionally, registration is a process of seeking the mapping function among all the data points using its subset of so-called control points. Conflation, on the other hand, includes the matching of features. Thus, registration can provide a less specific match than conflation and, in essence, conflation is a tougher challenge.

Source conflict and information change diagnosis become the characteristics of conflation process, while seeking the mapping function between two images is the key to registration. Typical cases for conflation include the matching of a subset of roads among the road networks or across a mapping boundary, the mapping of extracted features across different scales or resolution, and the combining of information from sources with large spectral

variation/change due to viewing geometry, environment variation, or technology evolution. Conflation could also come from the need of information integration from multiple disparate sources; e.g., reservoir characterization using stratigraphy based on well log and seismic cross section. Conflation provides a unique application for applying AVDM techniques.

2. IMAGE INCONSISTENCIES

2.1 Local and global inconsistencies

Typically, geometric disparities or inconsistencies (misalignment) may be global, local or subpixel in scope. Global disparity, scene/image wide, is analogous to similarity and direct linear transformation models in the domain of photogrammetry [Ghosh, 1998]. Local disparity, on the order of 10s to 100s of pixels, parallels the use of Affine and Rational Polynomial Coefficient transformation for registration. While all these models have analytical solutions, the modeling of subpixel disparities remains an elusive research objective.

Global Inconsistency: Figure 1 illustrates a complex case of global inconsistency where multiple information sources exist. Images 1 through n are the information sources for the digitized vector products 1 through m. There might exist one to one mapping among each pair of the imagery and/or product to meet some preset quality measure for registration. However, there may not exist a single global mapping function for all the imagery sources and their products. In addition, an overall solution for a mapping function may not satisfy mission specific criteria though a least square error measure that fits the requirement. In an AVDM framework, the modeling of this overall global function only serves the function of conflation diagnostics. The purpose of conflation is to increase the precision and accuracy further than what the registration type function can provide for some given mission specific requirements.

Local Inconsistency: Figure 2 illustrates the existence of multiple mapping functions for different objects of interests given a pair of imagery and its vector product. The correspondence mapping for Road 172 between the map and imagery is relative effortless, either visually or mathematically. The same cannot be said about all of the Camp Lejeune and Marines roads. For example the road marked straight (Marines) in the vector map appears to be curved in the imagery. Because of this obvious difference in visual signature, the mapping function for the Marines road will be very much different from that of Road 172. The solution will be inadequate based on registration be-

cause it is an average among all the potential conjugate points when the solution for specific set of features requires the partition of feature sets. The limited extent of the feature mismatch makes the matching problem "local" in nature. In this case, neither the global nor the local solution may be used for a mission specific solution, but together they may. Further more, the local solution may differ from one road (Marines) to the next (Camp Lejeune), thus, the mission specific nature of conflation.

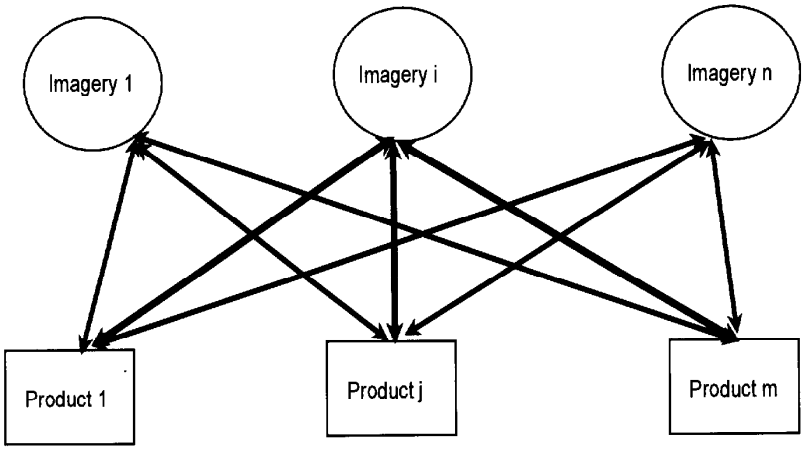


Figure 1. Inconsistency among imagery sources and their products

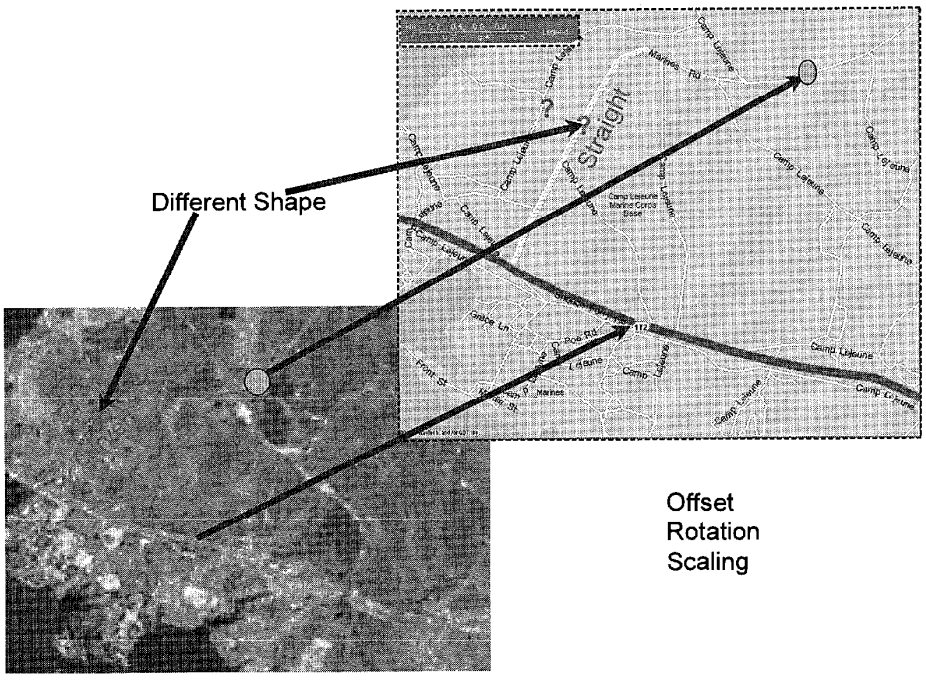


Figure 2. Local inconsistency between imagery source and vector product. See also color plates.

Human activities and natural occurrences are the two most common causes of the inconsistencies that occur in the imagery and its derived products. Among all the conflicts, vector-to-vector is the most studied case. Its disparities originate frequently from multi-source data over a common observation area.

The disparities come from two sources; i.e., raw data and vector generation, due to scale, resolution, compilation standards, operator license, source accuracy, registration, sensor characteristics, currency, temporality, or errors [Edwards & Simpson, 2002].

The disparities are two types of inconsistencies; i.e., spatial and attributes. Spatial disparities tend to be analytic while attributes disparities are due to decision-making from operators based on intensities. Other characteristics of inconsistency include discrete in space and time, variable in magnitude and direction, and often times incomplete. Below is a case study to illustrate disparity structure given a pair of information source and a mapping function.

2.2 Disparity structure analysis and inconsistencies decomposition

The use of geospatial information is becoming more and more automated. The trend towards automation demands better data quality and confidence measures. In the case of conflation, both data sources carry inaccuracies and render the *standard error analysis*, the dependent-and-independent variable analysis methodology, less useful.

For conflation, a mapping function, continuous or discrete, between the input data sources, is necessary and required, but often incomplete. The function used often is arbitrarily chosen and only approximate. Thus, an understanding of error sources in their fundamental form becomes an important factor in conflation decision-making. In this section, partial derivatives are applied to a basic mapping function, a similarity transformation, to reveal the **disparity structure** in conflation resolution. **Disparity structure** is used here to refer the inconsistency between the data sources in contrast to **error** that is reserved for the difference between a measurement and the truth.

The compact form of a typical **photogrammetric similarity transformation** is described in equation (1). Equation (1) has three components: translation, rotation and a *single* scaling factor. (X_T, Y_T) are the translations along the X and Y-axes, S is the scale factor, and θ is the rotation angle between the two sources. Equation (2) is its expanded form.

$$\begin{bmatrix} X_T \\ Y_T \end{bmatrix} = \begin{bmatrix} X_s \\ Y_s \end{bmatrix} - S \begin{bmatrix} Cos\theta & -Sin\theta \\ Sin\theta & Cos\theta \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \end{bmatrix} \quad (1)$$

where: X_T, Y_T are the translations,
X_s, Y_s are the coordinates from the standard,
X_c, Y_c are the coordinates from the conflation source,
S is the single scaling factor, and θ is the rotation.

$$X_T = X_s - S * Cos\theta * X_c + S * Sin\theta * Y_c$$
$$Y_T = Y_s - S * Sin\theta * X_c - S * Cos\theta * Y_c \quad (2)$$

For both equations (1) and (2), the assumption is that the similarity transformation is the known functional relationship between the two data sources, with the translation, rotation and scaling as the unknown parameters. Note that equations (1) and (2) represent a relatively special case of a general affine transformation, mathematically speaking. Thus, a more general theory would work with an arbitrary affine transform, but make the physical meaning of the derived parameter set less clear, and defeat the purpose of pursuing disparity structure understanding. Equation (3) is a set of partial derivatives for translation, assuming a constant scaling factor.

$$\begin{aligned} \partial X_T / \partial X_s &= 1 \\ \partial Y_T / \partial Y_s &= 1 \\ \partial X_T / \partial X_c &= -S * Cos\theta \\ \partial Y_T / \partial X_c &= -S * Sin\theta \\ \partial X_T / \partial Y_c &= S * Sin\theta \\ \partial Y_T / \partial Y_c &= -S * Cos\theta \\ \partial X_T / \partial \theta &= S * Sin\theta * X_c + S * Cos\theta * Y_c \\ \partial Y_T / \partial \theta &= -S * Cos\theta * X_c + S * Sin\theta * Y_c \end{aligned} \quad (3)$$

Equation (4) calculates the total differential along the X direction using the derived partial derivatives. This total differential includes nonlinear components for angle measurements.

$$\begin{aligned} dX_T = {} & \partial X_T / \partial X_s * dX_s \\ & + \partial X_T / \partial X_c * dX_c \\ & + \partial X_T / \partial Y_c * dY_c \\ & + \partial X_T / \partial \theta * d\theta \end{aligned}$$

$$
\begin{aligned}
&= 1 * dX_s \\
&- S * Cos\theta * dX_c \\
&+ S * Sin\theta * dY_c \\
&+ (S * Sin\theta * X_c + S * Cos\theta * Y_c) * d\theta
\end{aligned}
\tag{4}
$$

Equation (5), similarly, calculates the total differential along the Y direction using the derived partial derivatives. This total differential is also a combination of linear and nonlinear components. The Y components are generally symmetrical to the total differential along the X direction.

$$
\begin{aligned}
dY_T &= \partial Y_T / \partial Y_s * dY_s \\
&+ \partial Y_T / \partial X_c * dX_c + \partial Y_T / \partial Y_c * dY_c + \partial Y_T / \partial \theta * d\theta \\
&= 1 * dY_s - S * Sin\theta * dX_c - S * Cos\theta * dY_c \\
&+ (-S * Cos\theta * X_c + S * Sin\theta * Y_c) * d\theta
\end{aligned}
\tag{5}
$$

In an attempt to *relate the total disparity to the original information sources*, equation (6) calculates the *square of the total differential* along both X and Y directions as the function of measurements.

Equations (1) through (5) represent the **disparity structure** that carries an orthogonal component assumption with a description of various types of disparity terms, their relationship and relative importance, while equation (6) describes the combined total under the orthogonal assumption. Some of the terms are data source related, and some of them only reveal themselves when information sources are combined.

$$
\begin{aligned}
&d^2X_T + d^2Y_T = dX_s^2 + S^2Cos^2\theta d^2X_c + S^2Sin^2\theta d^2Y_c + \\
&S^2(Sin^2\theta X_c^2 + Cos^2\theta Y_c^2)d^2\theta - 2Cos\theta S dX_c dX_s \\
&+2Sin\theta S dY_c dX_s + 2Sin\theta S X_c d\theta dX_s + 2Cos\theta S Y_c d\theta dX_s \\
&-2Cos\theta Sin\theta S^2 dY_c dX_c - 2Cos\theta Sin\theta S^2 X_c d\theta dX_c \\
&-2Cos^2\theta S^2 Y_c d\theta dX_c + 2Sin^2\theta S^2 X_c d\theta dY_c \\
&+2Cos\theta Sin\theta S^2 Y_c d\theta dY_c + 2Cos\theta Sin\theta S^2 Y_c X_c d^2\theta \\
&+dY_s^2 + S^2Sin^2\theta d^2X_c + S^2Cos^2\theta d^2Y_c + \\
&S^2(Cos^2\theta X_c^2 + Sin^2\theta Y_c^2)d^2\theta - 2Sin\theta S dX_c dY_s \\
&-2Cos\theta S dY_c dY_s - 2Cos\theta S X_c d\theta dY_s + 2Sin\theta S Y_c d\theta dY_s \\
&+2Cos\theta Sin\theta S^2 dY_c dX_c + 2Cos\theta Sin\theta S^2 X_c d\theta dX_c \\
&-2Sin^2\theta S^2 Y_c d\theta dX_c + 2Cos^2\theta S^2 X_c d\theta dY_c \\
&-2Cos\theta Sin\theta S^2 Y_c d\theta dY_c - 2Cos\theta Sin\theta S^2 Y_c X_c d^2\theta
\end{aligned}
\tag{6}
$$

The simplified version of equation (6) is present in equation (7) for the square of the total disparity, where one of the two information **sources** is assigned as the **standard** (S) information source and the other is the **conflicting** (C) information source. This assignment is arbitrary because both of them have their original source **errors** that lead to the conflicting information when they are put together into the same framework viewed under a selected mapping function. Before this step of disparity analysis, conflicting information is simply a concept.

$$
\begin{aligned}
&d^2X_T + d^2Y_T \Rightarrow Total\ \ Disparity\ \ Budget \\
&\text{Disparity due to Standard source} \\
&dX_s^2 + dY_s^2 \Rightarrow Trans.\ \ disparity \\
&\text{Disparity due to Coflicting source} \\
&S^2 d^2X_c + S^2 d^2Y_c \Rightarrow Trans.\ \ disparity \\
&S^2(X_c^2 + Y_c^2)d^2\theta \Rightarrow Rotation\ \ disparity \\
&2S^2 d\theta(X_c dY_c - Y_c dX_c) \Rightarrow Dependent\ \ terms \\
&\text{Disparity due to cross terms of Standard and Coflicting sources} \\
&2S*(Sin\theta dY_c dX_s - Cos\theta dY_c dY_s \\
&\quad - Sin\theta dX_c dY_s - Cos\theta dX_c dX_s \\
&\quad + Sin\theta X_c d\theta dX_s - Cos\theta X_c d\theta dY_s \\
&\quad + Sin\theta Y_c d\theta dY_s + Cos\theta Y_c d\theta dX_s)
\end{aligned}
\tag{7}
$$

The total disparity describing the conflicting information in detail using sums of squares, the left side of Equation (6), has 30 different terms, the right side of Equation (6). Equation (7) presents a choice of grouping of the 30 terms by **disparity types**; i.e., terms from **sources** designated the standard and conflicting, and **cross terms** between the standard and conflicting sources. After using partial derivative analysis, and placing the disparities into a standard-conflation framework, the right side of the equation (7) includes several types of disparities that form a **total disparity structure**:

- the translation disparity along the X and Y directions,
- their combinations with the scaling factor, that is labeled Errors, due to a conflicting source in equation (7),
- the position dependent rotation disparity, and
- position canceling but rotation dependent disparity.

In equation (7), the cross term between S and C are all mixed terms either position independent or position dependent. The designation for "due to standard source" is for terms related to the standard information sources, and "due to conflicting source" for conflicting information sources. The cross

terms are related to both standard and conflicting information sources. The standard and conflicting terms are coded by their corresponding subscripts.

In essence, the integration of data has some prospect of providing better information; but it also presents opportunities to introduce new errors after conflation resolution, and potentially larger ones if the mapping function or its parameter derivation is deemed incompatible with the data sources. This makes AVDF an important component of conflation resolution. Among the conflict and cross categories, the assessment of disparity terms that are *location dependent* is key to visualization and decision-making.

The coupling and magnification of location dependent and angle-measurement error terms arguably will be the most challenging disparity to quantify. This necessitates a *visual decision making* contribution in addition to analytic analysis. For example, if the translation error terms from the designated standard information source are ignored (as in the standard dependent-and-independent variable analysis methods), the corresponding disparities have to be compensated by the other terms in the equation. This creates additional inconsistencies in the subsequent analysis. The illustrated similarity transformation disparity analysis example shows the fundamental difference between conflation and standard registration where one information source is considered **error free**. For more complicated transformation cases, direct linear transformations, or more relaxed mathematical models are required to map the spatial relationship between information sources. They will result in much more complicated disparity terms than using the similarity transformation equation as the mapping function. The state-of-the-art methodology for conflation resolution gets around the complicated and necessary disparity model by using vector attributes as a linking mechanism [Edwards & Simpson, 2002], thus completely bypassing the spatial modeling aspects of the process in the first step of the process. AVDM recommends a perspective of reducing the dependency of new source disparities on location by reducing analytic scopes, using visual cues to aid the conflation process.

3. AVDM FRAMEWORK AND COMPLEXITIES SPACE

For AVDM, the term **registration** is defined as the finding one and only one *mapping function* between two geospatial information sources, and **conflation** resolution as matching *features (pixel group)* in geospatial information sources with spatial consistency at a segment (subpixle) level and with both matched and un-matched features (pixel group) identified; i.e., the spatial and feature (pixel group) matching are location and attribute dependent. Thus, the purpose of registration and conflation is to combine information from data sources for the potential of creating a new dataset that is better

than the originals. To achieve better-fused data quality, different algorithms and methods are permissible under AVDM and its different stages of the data and information processing. The facts that no one has yet achieved fully automated conflation and that very few researchers are working on the imagery to vector registration, point to the reality of the challenges in the conflation process.

The AVDM framework provides a multi-level and/or stage solution for registration and conflation using multiple observations with requirements across multiple scales of geographical extents, and different attributes. One of the advantages for this divide-and-conquer strategy, utilizing the potential of consistency at one scale, location, extent for a set of attributes to diagnose the inconsistency at the other, is to manage the potential complexity at its appropriate time or process stage for the benefit of reducing uncertainties. The result of AVDF is the improvement of information quality by simplifying and compartmentalizing the complexities. Figure 3 shows a graphic presentation, to the first order, of the potential **complexities in registration and conflation** in the AVDF. Typically, accuracy requirements come from a decision-making process for a particular purpose; e.g., a specific change detection function requiring registration of 1/5 pixel accuracy. The "registration requirement" states the minimum number of control pairs necessary for a particular algorithm or method to form a solution in order to achieve the goal set by the accuracy decision-making process. The "searching requirement" refers to the iterations an algorithm has to go through to be sure the accuracy requirements have been met. For example, to locate a 60% common overlapping area for a footprint of 1K×1K imagery with 1 pixel accuracy, a algorithm is projected to go through a 400×400 = 160,000 points of a searching grid without rotation. Thus, Figure 3 can also be viewed as a potential solution space.

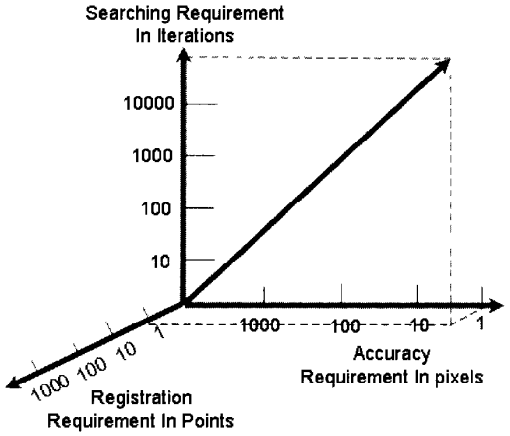


Figure 3. Complexities space covers the accuracy, registration, and algorithm searching requirements for registration and conflation

The complexity variation can be quite dramatic from the origin to the solution point in Figure 3, because of the complexity dependency on the under-

lying model and quality of the data sources. Given specific data sets and conflation objectives, the complexity space provides a first level framework to guide the conflation process. For example, when a particular objective calls for a very complex model, huge amounts of data, and very high accuracy requirement, the first level AVDF analysis in the complexity space may indicates the impossibility of solution attainment. Subsequently, the complexity space may provide visual decision making alternatives for a less demanding solution.

The complexity space provides a framework to view, identify, establish, and partition the conflation objectives. The conflation levels in the next section are proposed to provide a process and mechanism to navigate through the compartmentalized complexity subspace when a conflation objective is partitioned into actionable conflation levels. Disparity analysis provides a mathematical framework and quantities for constructing a practical process and its associated algorithms to estimate and/or solve the conflation problem at a specific level. The goal of AVDM framework is to put the complexity space, conflation levels, and disparity structure analysis into proper geospatial region of interests or scale and boundary. For example, in the case of emergency operation, a high decision-making level official needs only to know an approximate location for a large scale environment disaster, such as a large oil tanker spill, to be able to mange resource allocation, whereas a firefighter needs to know the exact location to be able to rescue potential victims successfully. In addition, this AVDF provides a mechanism to categorize and compare different **conflation scenarios**; e.g., planning an air strip for air trafficability vs. defining a route for tank columns in a hostile environment. Different conflation scenarios have different needs in **visual and analytical information content integration** from multiple information sources for decision making. Synergistic integration of analytic and visual decision-making provides a mechanism to elucidate conflation scenarios batter than using either one individually.

4. CONFLATION LEVELS

Conflation level partition subdivides the complex solution space so that conflation tasks can be handled recursively through a hierarchical structure where each individual level will have well-defined processes, methods, algorithms and tools.

First order of division of **conflation levels** is twofold: *Upper and Lower Level* divisions. The goal of *Lower Level* conflation is the diagnoses of mutual information and disparities in the data; mathematically speaking, to provide the appropriate variables and sub-datasets for integration. The goal of

Upper Level conflation is to provide the guidelines or objectives; mathematically speaking, the objective functions. The grouped conflation levels described above are depicted in Figure 4 where upper level conflation levels (decision levels) provide a guideline for designing a formalized objective function. Lower levels (measurement/signal levels) can examine data disparities such as absence of exact match of coordinate values and missing features.

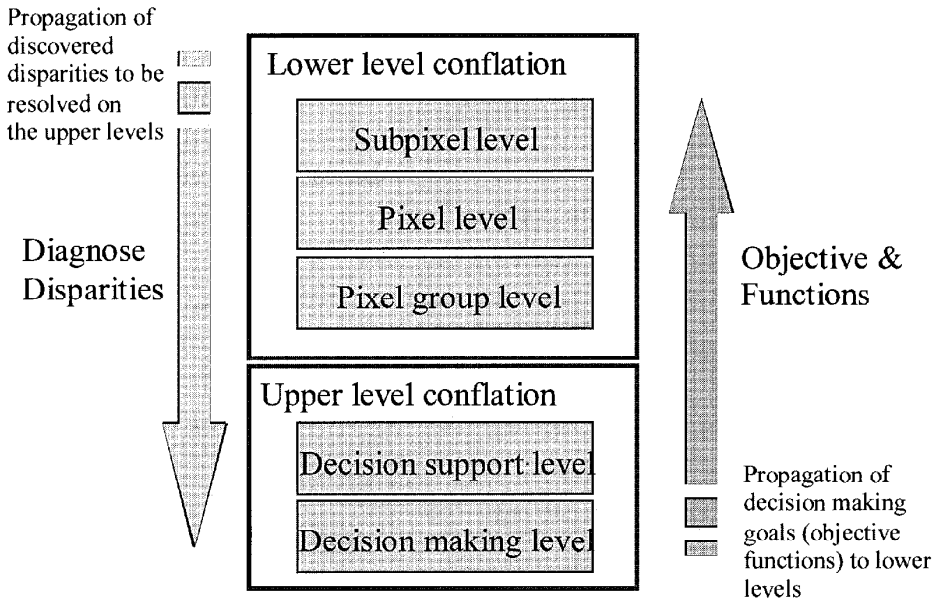


Figure 4. Conflation levels: propagation of decisions and disparities

The **Upper Level** conflation is further subdivided into *Decision-Making* and *Decision-Support Levels.*

The goal for **Decision-Making Level** is to provide the context type match based on input from a decision maker about parameters for the task; e.g., errors in the road elevation should not be larger than 5m for supporting trafficability.

The tasks for **Decision-Support Level** typically include change detection, feature discovery and target location. On this level data sources are matched/conflated to identify objects needed for decision making.

The **Lower Level** tasks partition the analytic aspect of conflation into *Pixel group, Pixel and Subpixel Level* matching.

The **Subpixel Level** match is at the signal level; the match of signals at this level may come from the same type of physical sources but may be separated in time and space.

The **Pixel Level** conflation studies the relationship with pixels as a whole; e.g., building corners are treated as pixel points disregarding the fact

that the intensity of a pixel could be a mixture of signals from different physical sources.

The **Pixel Group Level** conflation investigates the geometrically relationship with topological information. At *Pixel Group Level*, feature properties; e.g., roads, and their abstract mathematical quantities become integral part of the conflation resolution.

At times, correspondences exist between the ancillary information for data sets that are well calibrated internally, e.g., GPS information on a sensor location, and precise viewing geometry for the sensor. For this type of scenario, the conflation can be performed at the level in-between called **Metadata Level** match. Below we show examples of different levels. Typical cases of the decision level conflation are (1) rescue operation when life is at stake and (2) the trafficability from location A to B through the area C via ground, air, or water. The trafficability problem has several sub-problems that have different requirements for conflation:

- Military: Advance heavy armor column;
- Geology: Relocate oil & gas drilling platform;
- Rescue: Transport rescue teams to earthquake and hurricane locations; population evacuation;
- Engineering: Transport construction equipment to the high voltage power line corridor, and
- Engineering: Construct a highway.

Examples of decision support level conflation tasks are Change Detection, Feature Discovery, and Target Location (see Table 1 for more detail).

Table 1. Decision support level conflation tasks

Tasks	***Subtasks***	
Change Detection	Military:	Damage assessment
	Geology:	Plate movement
	Rescue:	Disaster assessment
	Agriculture:	Crop yield assessment
Target Location	Military:	Identify/illuminate target using GPS/laser, conflated map and imagery.
	Geology:	Locate a site for new drilling
	Rescue:	Locate the Epicenter and most severe damage location.
	Agriculture:	Locate and assess diseased crops.
Feature Discovery	Military:	Discovery of a new WMD facility
	Geology:	Discovery of a volcanic belt
	Rescue:	Discovery of a volcano using aerial photo and orbital imagery
	Geography:	Discovery of land use categories (crop, forest etc)

The example shown in Figure 5 illustrates the use of conflation levels. The *decision level task* here is to move troops from the blue point (see small circle) to the red point (see large circle) using the fastest route (using or not using roads). To solve this task, imagery and a map should be integrated and

their features conflated. This is a *decision level conflation problem.* In this example:

- the *decision support level task* is to identify feature types (e.g., orchards, rivers, buildings) that help to define the off road route (obstacles or camouflage);
- the *pixel group level* is to locate the features (e.g., roads, bridges, orientation of the orchards). The *pixel and subpixel level task* is to locate geometric attributes (e.g., line segments and points) and usability of a specific feature.

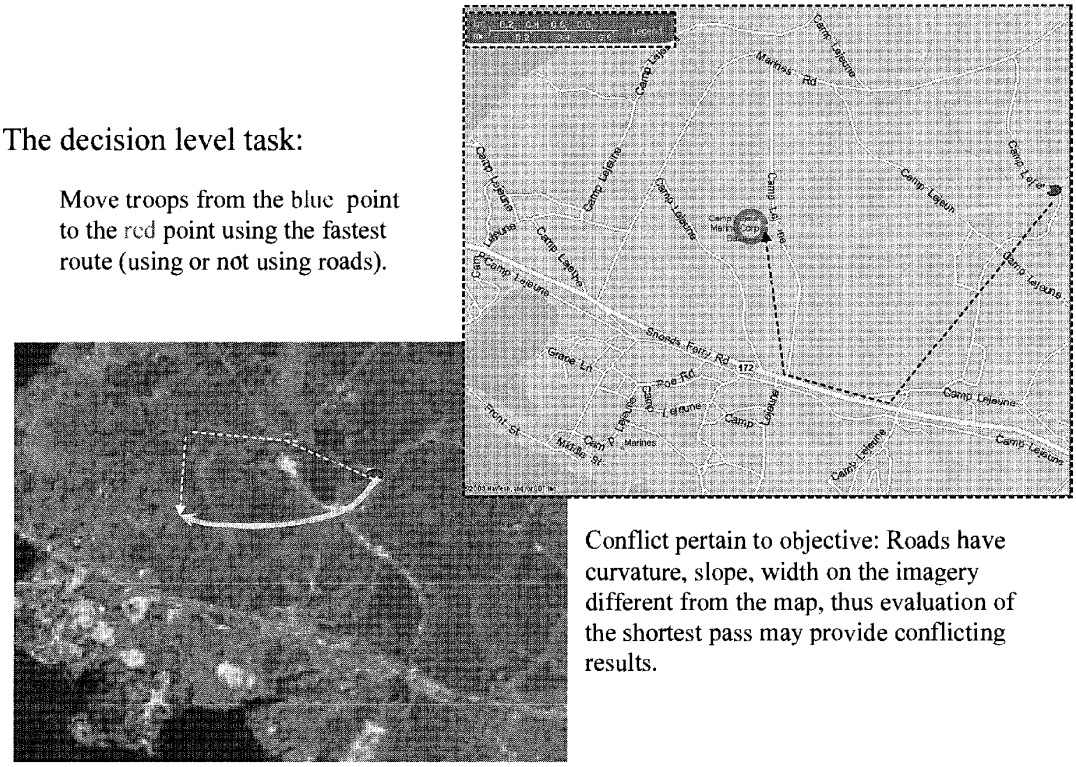


Figure 5. Decision level conflation problem. See also color plates.

Each of these levels L_i has its own conflict between map and imagery that needs to be resolved. Criteria for resolving conflicts in level L_i is coming from an upper level L_{i+1}. Below in Table 2 we illustrate types of conflict for each level.

5. SCENARIO OF CONFLATION

Below we discuss a **conflation scenario** that is based on the concept of the Gold Standard. A **Gold Standard (GS)** is a *description of the conflation*

situation for which there exists a suitable conflation method. Thus, this is a description of the situation and the conflation method. It is not necessary that the method be very precise as provided by GPS. For instance, for two satellite images with metadata on sensor model and location, the gold standard could be an orthorectification model. We consider both a conceptual orthorectification model and a model populated with actual parameter values using the description for the data source (sensor model and location).

The *Knowledge Base* (KB) contains *descriptions of data sources* (DDS) to be conflated and *prior knowledge* (PK). Prior knowledge is information that can be used to populate a GS for data source type. It differs from data that is directly coming from image metadata.

Table 2. Conflation levels and associated conflict types

Level	***Conflict type***
Level 6, L_6 Decision level	Conflict pertained to objective: Roads have curvature, slope, width on the imagery different from the map, thus evaluation of the shortest path may provide conflicting results.
Level 5, L_5 Decision support level	Conflict pertained to feature set: The map may or may not show the terrain condition but the imagery will have the information. On the other hand satellite imagery does not have feature names for navigation but map does.
Level 4, L_4 Metadata level	Conflict pertained to inaccuracies of data on sensor location and sensor model.
Level 3, L_3 Pixel group level	Conflict pertained to individual feature: the map has approximation information but the imagery has detailed and up-to-date information; e.g., a bridge and roads are under construction.
Levels 1-2, L_1, L_2 Pixel and Sub-pixel level	Conflict pertained to geometric attributes of a given feature: the width of the roads and bridges and the spacing of an orchard.

DDS can include metadata if available. Some pairs (DDS, PK) are matched with an individual GS by a **matching function**, *M*,

$$M(\text{DDS, PK}) = \text{GS}.$$

Function *M* is not fully defined, for some DDS and PK there may be no Gold Standard. One of the reasons could be that DDS and PK are not complete. Function *M* should be computed for every specific DDS and PK to produce GS. The **conflation scenario** based on the Gold Standard consists of four steps:

(1) Identification of **Conflation Situation** (**CS**) that includes identification of DDS and PK,
(2) Identification of **Gold Standard** (**GS**) by computing M(DDS, PK) = GS.
(3) **Conflation using Gold Standard** (This step is abbreviated as **CG**),
(4) **Assess disparities** for feature of interest.

A huge variety of possible problems that require conflation motivates the introduction of the Gold Standard concept. Several categories of such problems from different domains have been described in the previous sections.

Creating of task-specific conflation methods that we argue for (see chapter 17, section 2) has many advantages for properly solving conflation problems, but it is not easy to implement having in mind a huge number of different tasks that need conflation. Covering just a few "important" specific tasks can be only a temporary solution. The situation is not static, new tasks can become "important".

The Gold Standard approach tries to solve this problem by creating gold standard tasks, where all information of the given type is known and a method based on this information is known. For instance, if the meta-information about the sensor models and sensor locations is known for two panchromatic satellite images, classical photogrammetric methods of orthorectification can be applied using standard GIS software such as ArcMap. Thus, the Gold Standard approach is a new advance in implementing the task specific conflation. Creating a set of conflation gold standards is more realistic than creating specific conflation methods for every possible conflation task.

In the gold standard approach, the first step is the identification of a *conflation situation* that is sufficient to be able to identify a gold standard for the *task at hand* at the second step. More specifically **conflation steps** are:

- Step 1: Establish the gold standard for conflating specific input sources for the task at hand;
- Step 2: Populate a gold standard with data from input source features;
- Step 3: Conflate input sources using a gold standard;
- Step 4: Assess disparities for feature of interest in input sources.

These steps are show in more detail in Table 3.

Table 3. Conflation scenario steps

Step 1	Identification of lower level Conflation Situation (matching with predefined categories of conflation situations.
Step 2	Identification of gold standard Step 2.1: Matching Conflation Situation with one of the predefined Gold Standard categories. Step 2.2: Populate the gold standard with feature sets from data sources
Step 3	Conflation using Gold Standard Step 3.1: Conflate input sources using the gold standard Step 3.2: Assess disparities for feature of interests in data sources Step 3.3: Modify conflation for decreasing disparities
Step 4	Accumulation of conflation knowledge base

Steps are looped repeatedly in the process of refining conflation and resolving conflicts and disparities. Figure 6 shows these steps looped.

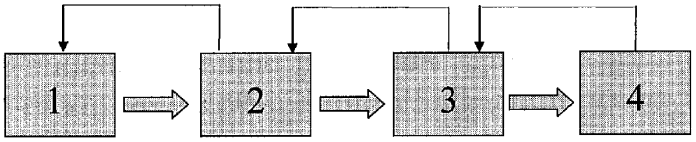


Figure 6. Conflation loop cycle

These conflation scenario steps can be implemented in a **rule-based framework** with several categories of rules: (1) rules for conflation situations and knowledge updates, (2) rules for references configuration using GS, and (3) rules for conflation using GS. The purpose of the first step in conflation resolution is to identify the type of conflation that matches the current mission specific task at hand.

A rule can be relatively simple propositional statement or a complex model M with clearly defined model use conditions C:

If <condition C is true> then <use model M>.

A set of rules forms a **virtual expert rule base** that is part of the **conflation knowledge base**. A specific conflation scenario for the task at hand is identified through applying rules that serve as guidelines.

If the identified scenario matches closely with one in the database of scenarios, this particular conflation task becomes a case of applying an existing methodology by following the stored set of rules in succession. Figure 7 illustrates a rule-based approach employing CS and GS. In more detail, rules listed in this figure are described in Section 6. When a closely related rule is not in the Virtual Expert knowledge base, it is necessary to work with an expert in the domain to build a new scenario; i.e., the Rule of CS-GS2 and associated Virtual Expert System updates along with its quality evaluations.

Building a new conflation scenario is a two-step process. The first of which is to establish the GS to be followed by conflation resolution. This GS represents a first order solution because a perfect solution is unobtainable under the assumption of the existence of conflicts. This first order solution can also be viewed as a framework, foundation, boundary condition, etc...

Photogrammetry models are good candidates for the GS because of their "rule of thumb" nature and because they require only a few points to satisfying the necessary modeling conditions.

Without the GS, the subsequent conflation resolution runs a risk of solution without context. With the establishment of GS, each data point from the conflation source pairs has well-defined coordinates that will become the starting point for conflation in the subsequent process.

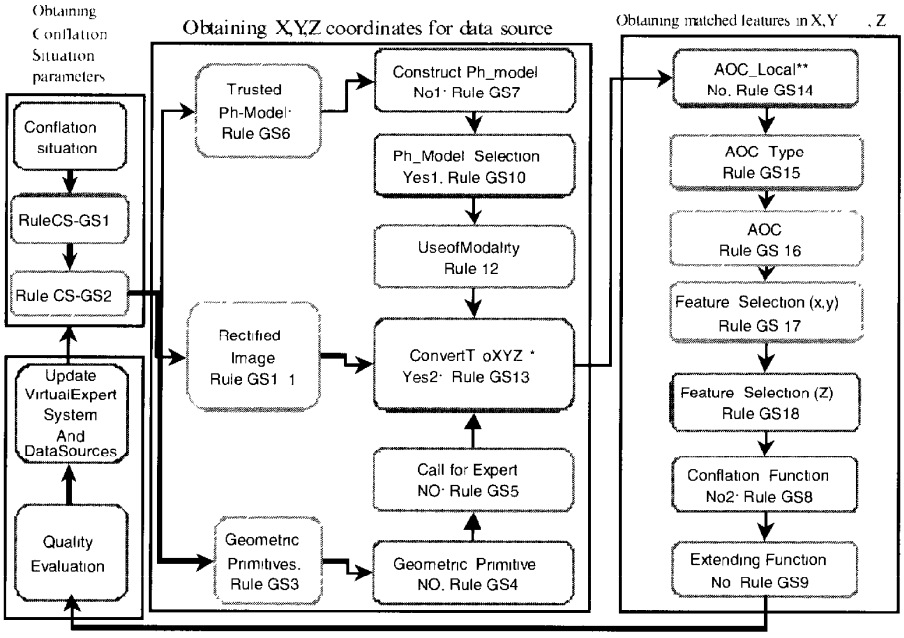


Figure 7. Top-level view of Gold Standard based conflation elements and their links, see section 6.2 for details for specific rules

There are three possibilities in establishing the GS with increasing difficulties: 1) data sources that are rectified, 2) a trusted photogrammetry model exists for the data source, and 3) there is very little existing information for the data sources. Consulting experts from photogrammetry field can counteract the increase of degree of difficulties. Upon successful conflation, the solution to the new problem naturally becomes part of the system for future references. In accordance with definitions introduced above, one of the **conflation rules** is formulated as follows:

> If the description of data source (DDS) to be conflated and prior knowledge (PK) about them are in the Knowledge Base (KB) then compute a matching function *M* that provides a Gold Standard (GS) otherwise apply another rule. The second rule tests if both <DDS, PK> and additional information about <features> in images are not available and suggests to call an expert or to enhance available DDS and PK.

More exactly two rules are presented below.

Rule GS1 "Computing GS":

> If pair <DDS,PK> is in KB then compute M(DDS,PK)=GS
> else apply *rule CS-GS2.*

Rule GS2 "Expert matching GS":

> If pair <DDS,PK> and <features> are not in KB
> then call <expert> to match <DDS,PK> with some GS or enhance <DDS,PK> to pairs that are in KB.

6. RULES FOR VIRTUAL IMAGERY EXPERT

A virtual imagery expert system is a system that intends to assist a real imagery expert in solving imagery analysis problems including conflation and registration problems. Below we illustrate such a system by presenting a conflation rule base.

6.1 Rules for identification of conflation situation

The major source for identifying the conflation situation is the data source description available. An upper level identification criteria of conflation situation is given by a simple predicate Available (<data source>)=Y/N.

On the next level of detail the same predicate is applied for Metadata and Prior Knowledge:

Available(<Metadata>)=Y/N, Available(<Prior Knowledge>)=Y/N.

Here Metadata describe a specific data set and Prior Knowledge may be applicable to a variety of datasets.

Next it is assumed that datasets to be conflated are available, that we do not need to test predicates Available(<datasets>), but we need to identify types of data available: hardcopy, data that describe geospatial features (feature-based data), physical sensor parameters, Earth parameters, or data about a specific object.

These data types can be represented by predicates, for instance, it can be written as Earth_parameter(<data source>)=Y/N and Hardcopy(<data source>)=Y/N. Table 4 and Figure 8 show upper-level rules to enrich data sources and conflation situation to be able to conflate data.

Table 4. Upper level Data Source (DS) rules

Rule	***IF-part (condition)***	***Then-Part (action)***
CS1	Hardcopy(<data source>)=Y	Digitize
CS2	Features-based(<data source>)=Y	Obtain features
CS3	Physical_sensor_parameters(<data source>)=Y	Apply sensor model
CS4	Earth_Parameters (<data source>)=Y	Apply environment model
CS5	Specific_object(<data source>)=Y	Obtain object model

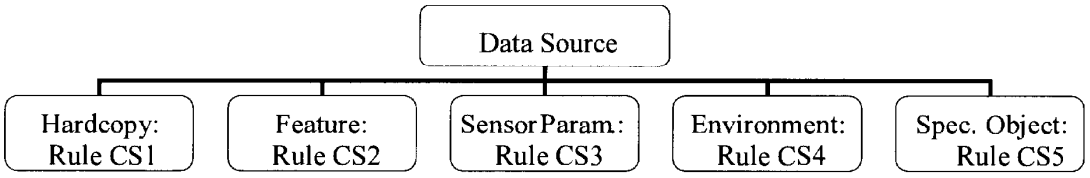


Figure 8. Links between conflation situation rules

Detailed CS rules are more specific and include a variety of parameters, such as parameters of physical sensors, parameters of a feature extraction mechanism, and feature relations. For instance, some rules can be applicable only if edges do not overlap.

Other rules may require that spatial (lenses or. focal planes) and spectral calibration models be known. Further rules may require different derivatives (partial, mixed, second and higher orders) for edge detection (Sobel, Canny, Laplace methods). Some rules assume that the noise is symmetrical and normally distributed.

The following rule is representative for lower level rules based on pixel-group level match (e.g., road pixels). The intent is to diagnose disparity in features in lower levels and to resolve the disparity on the decision support level by applying criteria known only on this upper level. For instance, for trafficability task we may have two roads, R1 and R2. Both roads have disparity 1 m in two sources. This disparity is discovered in the lower pixel group level. At the lower level, it is not known whether any of these roads can be accepted for trafficability. On the decision support level, any conflation resolution may be accepted for trafficability task if the disparity is accessed to be less than 1m.

If additional information indicates that road R1 goes to point B and road R2 goes to point D; and if on the decision level it is known that the goal is to reach point B, then the conflation resolution should be in favor of road R1. Thus, the inconsistencies diagnosis of the conflation situation on the lower level is resolved by applying criteria from upper levels.

The advantages of using AVDM in the stated real world scenario derive from the guidance of conflation levels. Conflation level based approach aids conflation process by having a mechanism to group types of disparities to be matched and resolved with appropriate decision support requirements. Thus, AVDM provides a framework for computing and allocating resources for only relevant disparities.

6.2 Use of gold standard rules

The focus of this section will be on the links between rules in addition to what have provided in Figure 7. In essence, virtual imagery expert can be thought of as a closed operating system completed with a set of linked hierarchical rules. Assuming that rules are built first, mission specific approach (MSP) will make use of the rules to evaluate disparity between two imagery data sources. The logic of dealing with disparity is presented in Table 5 with two rules: GS1 and GS2.

Rule GS1 can be further specified in its then-part. "Evaluate <disparity>" can include two steps also encoded as rules

- Derive <disparity(data source 1,data source2)>,
- Model <disparity>, and
- Evaluate <disparity>.

Table 5. Evaluate disparity using Gold Standard

Rule ID	***Name***	***If***	***Then***
GS1	Evaluate disparity	<data_source1> is converted to <GS> and <Features are in X,Y,Z coordinates> and <data_source2> is converted to <GS> and <Features are in X,Y,Z coordinates>	evaluate <disparity> and set flag=1 if disparity is low and { set flag=0 if disparity is high and use Rule CG2 }
GS2	Modify	<disparity> is high	obtain more information from <expert> or <other sources> for adjusting <GS>

Figure 9 shows links between rules for building conflation rules based on a gold standard and the analysis of prior knowledge. The logic of rule CG1 is depicted in Figure 10.

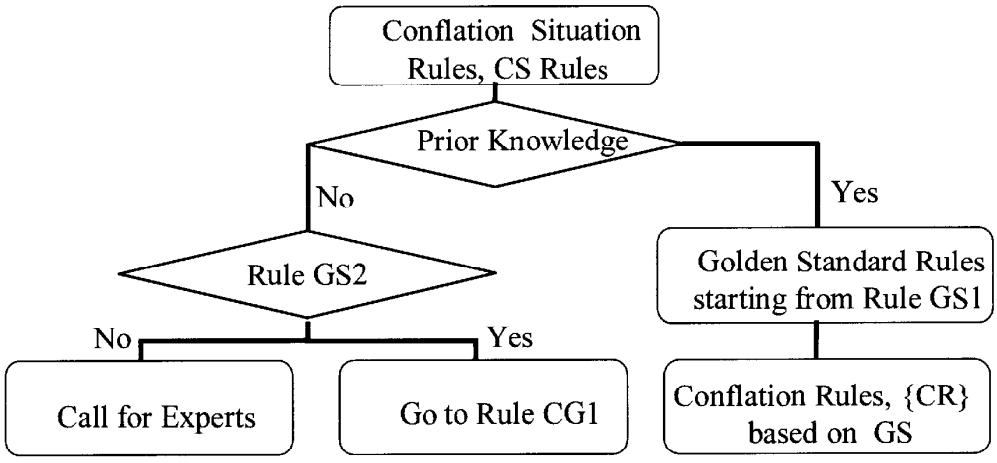


Figure 9. Links between rules for building gold standard

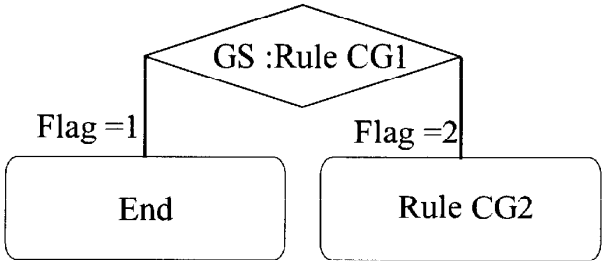


Figure 10. Links between conflation rules based on gold standard

6.3 Rules for GS identification

It is natural to require all information sources be addressable by the <x,y,z coordinates> -spatial part of the GS, albeit these can be relative. Thus, a key attribute to the rules for the virtual expert system is to built <function F> that will provide geo-reference coordinates <x,y,z>.

The *theoretic concept* of Gold Standard at rules level is to identify specific and appropriate means that provide geo-referencing such as GPS to facilitate <function F> building for computing geo-references. Informally the concept of the gold standard is coming from the idea that GPS provides a <gold_standard> for geo-location, e.g., for targeting.

Tables 6 and 7 contain rules that are designed to work in this environment. Table 6 deals with prior knowledge that can lead to an appropriate <Function F>. For instance, <prior knowledge> in rule GS5 could be that image 1 has <a crater> then a crater (bomb, or gas explosion) expert could be called, whose knowledge can help to match crater in one image with original unaltered information from the other to provide corresponding location between the two.

Table 6. Rules for getting rules and models

Rule	***If***	***then***	***else***
GS3	There is <GS> for <data source> with defined <x,y,z> coordinates> and <geometric primitives>	convert <data source> to <GS> and apply Rule CG1	apply Rule GS4
GS4	There are <x,y,z coordinates> and <geometric primitives> suitable for <data source>	set flag=1 and define <x,y,z coordinates> and <geometric primitives> for <data source>	use Rule GC6 set flag=0 and call <experts> for <rules> on <prior knowledge> or <conclusion>
GS5	Rule GS4 fails	set flag=0 and call <experts> for <rules> on <prior knowledge> or <conclusion>	

In Table 7, <Ph_model> notation stands for a Photogrammetry Model that includes orbit and sensor information. In this table, Rule GS 12 is described by using the concept of modality that is denoted as <QQQ>. Modality <QQQ> could stand for <LIDAR>, <Vector Image> and others.

Table 7. Rules for coordinates <x.y,z> and photogrammetric models

Rule	***If***	***then***	***else***
GS6	<prior knowledge> is a trusted <Ph_model>	use <Ph_model> as <prior knowledge> in Rule GS2.	construct <Ph_Model> by using Rule GS7.
GS7	<data source> is suitable to use <Ph_model>	select <Ph_model> and calculate <Ph_parameter> and convert <Ph_model_info> to <Ph_model> for using by Rule GS2.	
GS8	<1:1 function F> from <data source> to a <x,y,z> reference coordinate exists but may not be known	attempt to identify function F as <GS>	attempt to create a new <function F> as <GS>
GS9	<function F> has <new argument>	Attempt to identify <function F> with this new argument	request a <function F> from <expert> or build a local function using Rule GS13
GS10	<Ph_Model> is <plane_to_plane> or <3D_to_2D>	select <projective_model>	If <Ph_Model> is <3D_to_3D> then select <conformal_model> else use standard coordinates
GS11	<image rectified>	identify <scale> and use Rule CG1	convert image to <GS> using <aspect ratio> or <measuring units> or <variation param> and apply Rule CG1
GS12	<modality_type> is <QQQ> and <QQQ parameter set> is matched completely with <GS parameter set>	Setup flag 1	Setup flag 0 and extract <GS parameters> set from <data_source>, or use <analytic_model> or make <assumption> on <GS> parameters
GS13	<Ph_model_info> of <data_source> of modality <QQQ> is complete and applied for the whole image	convert <data_source> to <x,y,z>	If <z values> are <missing> then make <assumptions>

Table 8 presents rules when geo-referencing <function F> does not apply to the whole image. It deals with areas of interest (AOI) and with areas of conflation (AOC). Rule GS18 deals with their specific type --<flood type>. Other types could be such as <war_type> or <fire_type>.

Table 8. Area of interest rules

Rule	*If*	*then*
GS14	<function F> does not apply for the whole area	define <Area_of_Conflation>
GS14a	Too many <control points> or <lack_of_good_strategy> for defining <AOC>	Attempt to select <sample_strategy> if fail call an expert to create <strategy> for defining <AOC>
GS17	<Change occurred> from <image 1 to image 2>	mask_out <changed area> mask_in <useful features> from <mask_out> and if <expert_can_help> encode <expert's input> to <AOC>
GS18	<Flood type>	get <DEM> mask out <lower elevation> mask in <tall feature>

7. CASE STUDY: PIXEL-LEVEL CONFLATION BASED ON MUTUAL INFORMATION

In this section we describe a pixel-level case study of the multilevel conflation approach. Feature-based decision support level cases are presented in other chapters. Signal quantization to pixels is one of the error sources that conflation of images should deal with. The quality of a digitally recorded image signal depends on quantization characteristics such the number and size of discrete units (pixels) per image and the number of bits per pixel.

A *digital number (DN)* is the number identified in the process of an analog signal quantization. The digital number is characterized by the number of bits n allocated to encode the signal value. A resolution of analog/discrete (A/D) conversation is identified by n.

If n bits are used per pixel, then 2^n intensity values can be encoded in the image, thus an 8 bit image is limited to 2^8=256 intensity values. For larger n such as 16 or 32 intensity values can be 2^{16} or 2^{32}, respectively.

Each individual wave band can be displayed by one intensity parameter using n-bit digital number. Any three bands can be combined together using a composite of three "false" colors by assigning red color to the first band, green to the second band and green to the third band. This requires $3n$-bit DN to encode contribution of all colors. A standard false color composite of red, green, and blue is used to encode near-infrared, red, and green colors, respectively, in vegetation analysis.

7.1 Mutual Information: theory and applications to conflation

7.1.1 Entropy

The concepts of Mutual Information stem from information theory. Shannon's Information Theory deals with the fact that the "actual message is one selected from a set of possible messages". The fundamental concept in Shannon's information theory is that information is conveyed by *randomness*. Information from Shannon's theory is defined as a measure for the statistical dependency of messages selected from a set of possibilities. The focus of mutual information, however, is on "two messages". While information theory has been successful in serving the needs of communication, its application in imagery processing and pattern recognition has been limited because of the difficult challenge in formulating the required probability density function.

Conflation provides a unique imaging application for mutual information (MI) when it is considered as a subject of studying some finite combinations of parameters in a given model from observations separated in space, time, and wavelength. The challenge of applying MI to conflation types of applications in imagery and its derived information sources lies in the area of quantifying and evaluating information beyond mean square error criterion. This case study exploits MI for image conflation.

The basis of information theory is entropy, H, that characterizes information or its rate production [Shannon, 1948; Shannon & Weaver, 1963].

Suppose a random variable $X=\{x_i\}$ with values x_i $(i=1,\ldots,n)$ has probabilities of x_i occurrences $p_1=p(x_1)$, $p_2=p(x_2),\ldots,p_n=p(x_n)$. The **entropy** measure H is defined as

$$H(X)=-K\,\Sigma_{i=1:n}\,p_i\,\lg p_i \tag{8}$$

where K is a positive constant amounts to a unit measure. H is continuous in the p_i and $H(X) \leq \lg n$. If all p_i are equal $(p_i=1/n)$ then H should be a monotonic increasing function of n, $H(X) = K \lg n$.

7.1.2 Mutual Information (MI)

Suppose there are two random variables, X and Y, in question, with n possibilities x_i for the first and m possibilities y_i for the second. Let $p_{ij}=p(x_i,y_j)$

be the probability of the joint occurrence of x_i for the first and y_j for the second. The joint entropy of two variables X and Y is

$$H(X,Y) = -\sum_{i,j} p_{ij} \lg p_{ij}$$
$$H(X,Y) \le H(X)+H(Y)$$

with equality only if variables are independent, i.e., $p_{ii} = p_i p_j$.

Mutual Information $MI(X,Y)$ is the relative entropy between the joint distribution and the product distribution:

$$MI(X,Y)=\sum_{i,j} p_{ij} \lg \frac{p_{ij}}{p_i p_j}. \quad (9)$$

Note that $MI(X,X)=0$ when X and Y are independent variables, because in this case $p_{ij}/p_i p_j = 1$ and $\log 1 = 0$. Mutual Information $MI(X,Y)$ is related to the rate of transmission R from Shannon, i.e., the amount of addition information that must be supplied to correct the error

$$R = I_{send} - I_{corr} = H(X)+H(Y)-H(X,Y)=MI(X,Y). \quad (10)$$

The state-of-the-art registration and conflation processes, methods and algorithms have synergistic needs in: defining overlapping areas, mitigating multiple solutions in parameter space, refining local solutions, and reducing computation complexities.

This chapter presents two basic aspects of MI using case study approach. First, we illustrate the potential use of MI for defining the overlapping area by reducing computation complexities compared with cross correlation methods. Second, we illustrate the concept of using MI to find conjugate registration points when cross correlation fails, thus demonstrating its potential for mitigating multiple solutions in parameter space and refining local solutions.

7.1.3 Histogram and Entropy

The first step in MI application is the calculation of pixel radiance intensity histogram and entropy. A window of data is selected from the input imagery to calculate the histogram before it is converted to entropy. Figure 11 is a Landsat imagery over Idaho of the size of 1K x 1K pixels; it represents a typical agriculture area in a Northwest semi-arid region. The circular features nested in a regular grid in the imagery indicate a nominal irrigation pattern.

Generally for the Landsat class of imagery, 256 probability distribution bins exist because of its 8 bits digitizing precision. Figure 12 shows a set of histograms formed by dividing the entire imagery into equal quadrangles. The histogram in the center of Figure 12 represents the whole image.

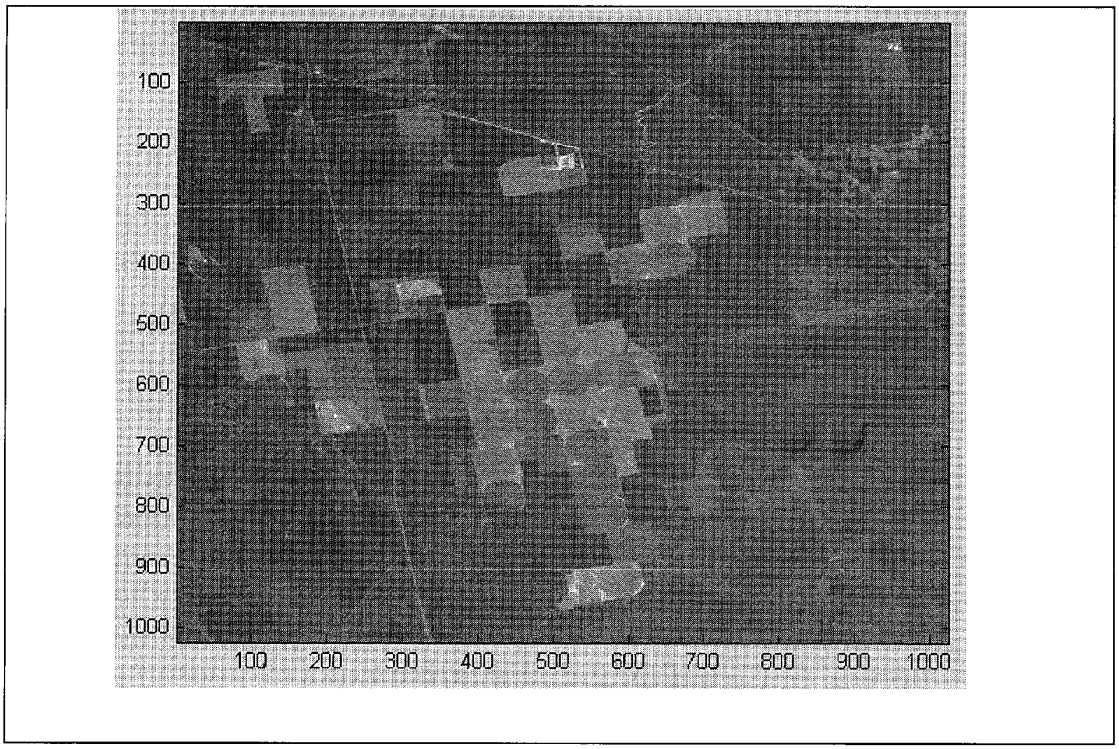


Figure 11: A 1K×1K frame of Landsat imagery from an agriculture area in Idaho. Circular features nested in a regular grid in the imagery indicate a nominal irrigation pattern.

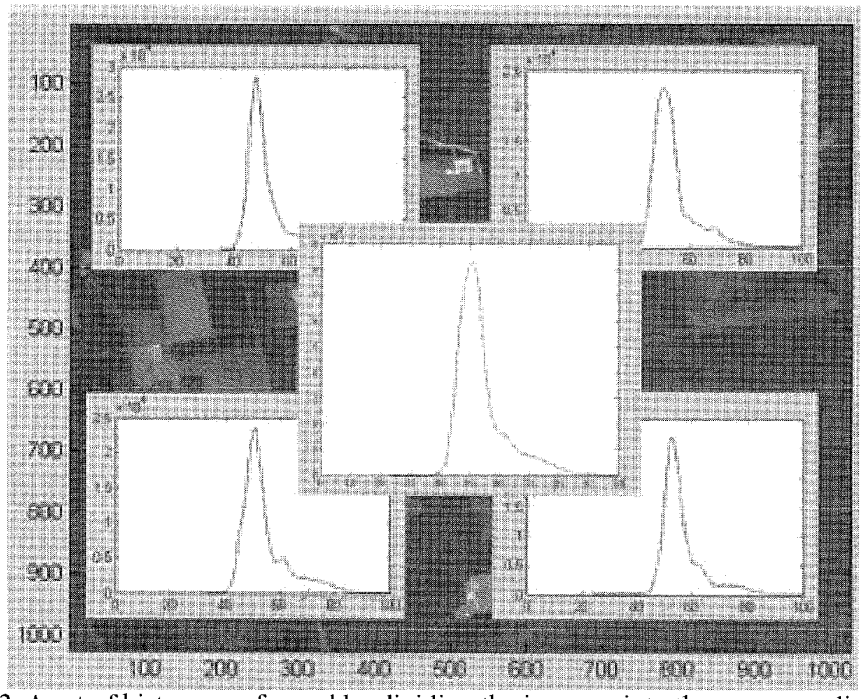


Figure 12. A set of histograms formed by dividing the imagery into the corresponding equal quadrangles. The subplot in the center of the figure shows the histogram of the whole image.

The comparison of these histograms indicates that they are **location and size dependent** (see also Figure 14). The *location dependency* forms the basis for spatial *information correlation.* The *size dependency* becomes a factor in the efficacy of applying *specific methods.*

To reduce the dependency of the histogram of accumulative counts on window size, the histogram values are divided by the total of number of points in the selected window, providing a *probability density function p* and enabling the calculation of entropy. The transformation from probability density distribution p to entropy E reduces all the information from the selected window of data to only a single number, i.e., the number of bits over the area of interests.

This data compression results in quantifying the information from the selected window as an average number of bits needed to convey radiance intensity information for all the image pixels. This averaging nature is inherited from the normalization process of converting the frequency counts to probability in addition to the application of unit measure provided by the base value of logarithmic operation in the entropy formula. Figures 12 and 13 illustrate the conversion of histogram to a probability density function. The magnitude of vertical axis in Figure 12 is on the order of 10^4, while the magnitude of the vertical axis in Figure 12 is normalized to around 1/10 of the total number of points in the selected window. The horizontal axis represents pixel intensity values ranges from 0 to 100 out of 256 bins. P_i in Figure 13 indicates the probability for a given intensity digital number bin.

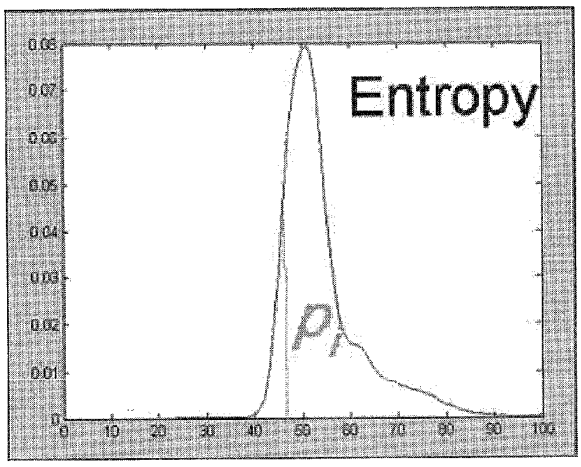


Figure 13. Illustration of the conversion of histogram to probability. The entropy value calculated using Eq. 8 is of 4.7576 bits. It is within the 8 bits of Landsat accuracy as expected.

7.1.4 Use of mutual information in image registration and conflation

Formulation of mutual information has been evolving since Shannon formulated the information theory in 1946. Mae et al. [Maes, Colignon, Vandermeulen, Marchal & Suetens, 1997] and Wells et al. [Wells, Viola, Atsumi, Nakajima, & Kikinis, 1996] have successfully introduced it to the medical fields. Studholme [Studholme, 1999] introduced normalization into MI to further normalizing the information content through a ratio of summation of the marginal probability over the joint probability. These basic research developments signify the continued refinement of using MI for image registration. Such evolution comes from the fact that MI is a result of a set of nonlinear transformations that reduce the first order dependencies of intensity on location and window size.

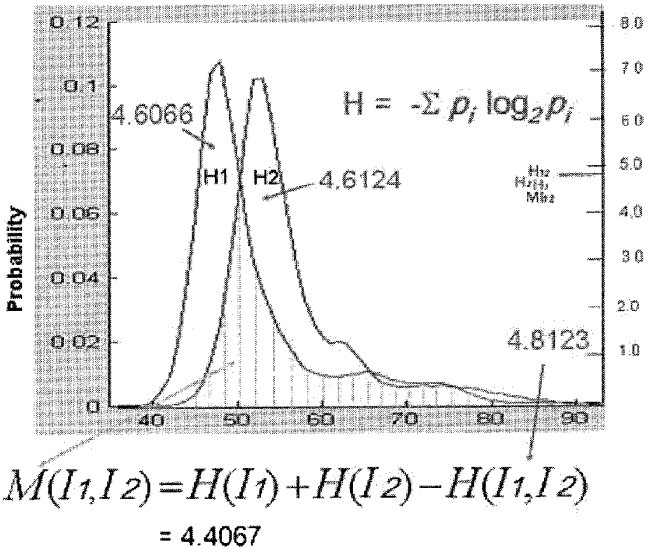


Figure 14. Entropy (H1, H2) and mutual information (M (I_1,I_2)) calculated based on the quadrangles 2 and 4 from Figure 13. Vertical axis on the left marks the measure for the individual DN bin while the vertical axis on the right marks the mutual information in terms of number of bits.

Registration and conflation involves the relationships between conjugate pairs of the information sources. Figure 14 illustrates this principle by using two windows, H1 and H2, of data selected from Figure 12; i.e., top left and bottom right quadrangles.

As the word "mutual" implies, the focus is on depicting the information common to both of the input entities. Vertical cyan (will color be used?) lines in Figure 14 mark the intersection of H1 and H2. The entropies for the selected quadrangles and their joint histogram are 4.6066, 4.6124 and 4.8123, respectively.

7.2 Computation reduction for overlapping area

First order registration and/or conflation are of great interest to authors of any alignment type algorithms for the reasons of computation reduction. The interests and advantages start from the reduction in the necessary searching space. Additionally, when large numbers of pixels are involved, the convergence is not guaranteed for many registration algorithms due to the potential for increased *error* and *large intensity variation*, both of which contribute to the reduced correlation. For MI to succeed in registration and conflation, it is necessary to demonstrate its power in computation reduction and stability across large radiance variability. The first example in this section demonstrates the use of MI for computation reduction while the stability will be demonstrated in the next example. Figure 15, using the same image as in Figure 12, demonstrates a situation of finding one registration tie point for global translation in both horizontal and vertical directions.

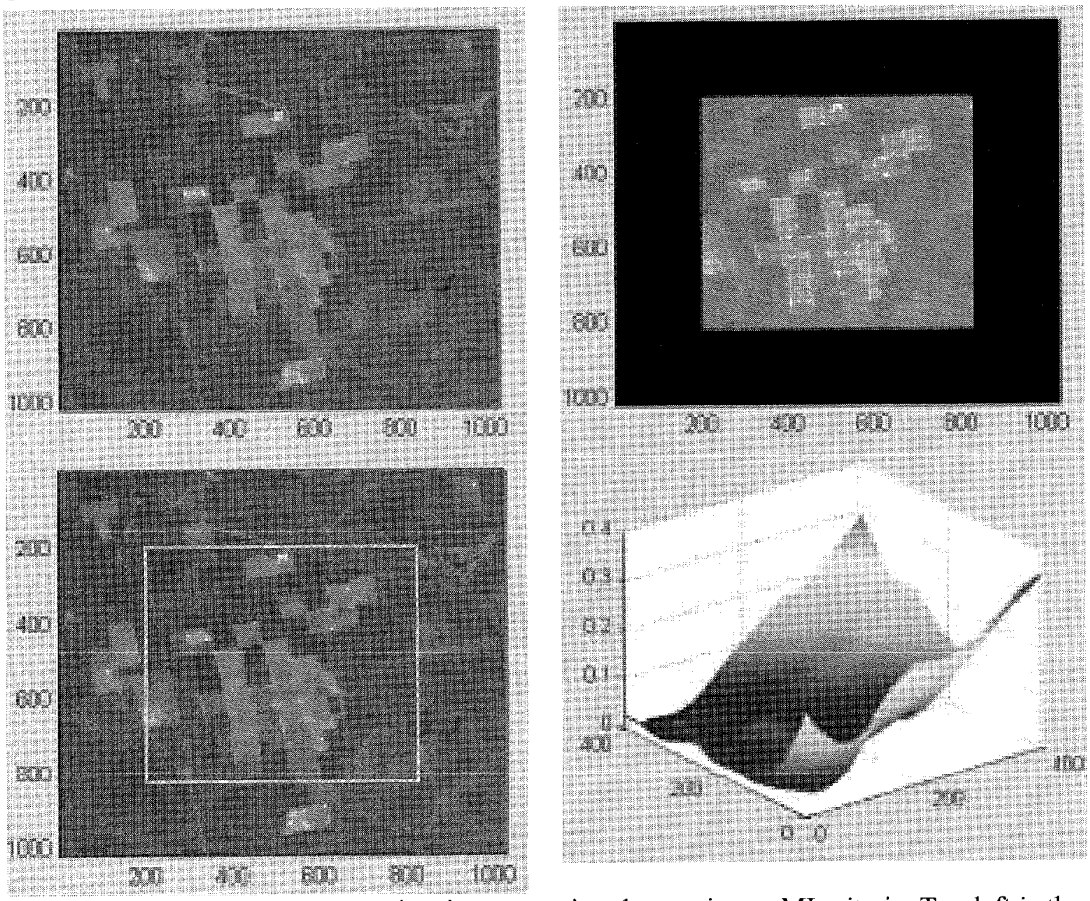


Figure 15. Defining imagery overlapping area using the maximum MI criteria. Top left is the input imagery. Top right is the image chip for the center of the original imagery. Lower left is the results from MI maximization. Lower right pictures the absolute entropy difference between a given window from the original imagery and the imagery chip.

To achieve a desired accuracy 1 pixel for a predefined 60% coverage of an image chip without rotation, it is necessary to have the image chip overlaying the original 1K by 1K imagery 400x400 = 160,000 times before a satisfaction solution is derived. If rotation is needed, it will take much more computation. The computation time used in finding the solution of a (200,200) offset with the window size of 624 by 624 is about 320 units of CPU time measured by Matlab tic and toc utility on a desktop machine.

For comparison, the calculation of a single iteration of a correlation of the center image chip with itself used up more than 0.3 cpu time units on the same computation setup as in MI calculation. This 0.3 CPU time duration translates to more than 42,000 units of total cpu time for searching the entire 160,000 points on the iteration grid. Thus MI shows about 100 times improvement in computation efficiency when compared with correlation based method.

7.3 MI for Control Point Selections

Registration requires control (conjugate) points. At the top level, there are three different ways of coming up with control points: manual, autocorrelation based, and mathematic fitting using equations. Autocorrelation is acknowledged to be the *current standard* used by many state-of-the-art software packages. Normally, in registration, cross correlation of radiance intensities from two or more images uses window sizes on the order of a dozen pixels. When compared with the size of normal Landsat or Spot imagery on the order of 6K, this local registration tie point from a dozen pixels is of different order of magnitude from imaging warping. The need to develop and test algorithms on the order of 100s pixels will be obviously beneficial to warping type registration/conflation tasks at the scale of 100s of pixel misalignment.

At the scale of 100s of pixels, inevitably, misalignment will contain a component of rotation. *Correlation is not known to be stable when rotation and scaling exist in the pair of data set matrix,* particularly when there is intensity variation. The example illustrated in Figure 16 demonstrates the utility of using entropy instead of radiance intensity for finding registration tie points using correlation, thus demonstrating the advantages of using entropy and mutual information for registration and conflation.

The image pair in Figure 16 has obvious translation, rotation, and scaling disparities. There are many readily mapable features in this imagery pair. Among them, a river bend and a lake are mapped using a dark blue vector. When the dark blue vector from the river bend is translated to the location of

the white vector mapping the lake with the sharing a common original, a difference vector results marking the misalignment pointed by a curved arrow.

The curved arrow in Figure 16 points to the vector representation of misalignment with components of offset, scaling and rotation. To show the benefit of MI, this pair of radiance intensity images is first transformed to entropy using a window size on the order of 60 pixels. The reduction in high frequency signal from the entropy transformation demonstrates MI's capability of reducing local variability. Figure 16 also reveals that the transformation of the radiance intensity to entropy is a non-literal one; e.g., both the river (dark pixels) and agriculture (bright pixel) can have medium to high entropy. The comparison will be demonstrated using correlation of data windows with size on the order of 600 pixels.

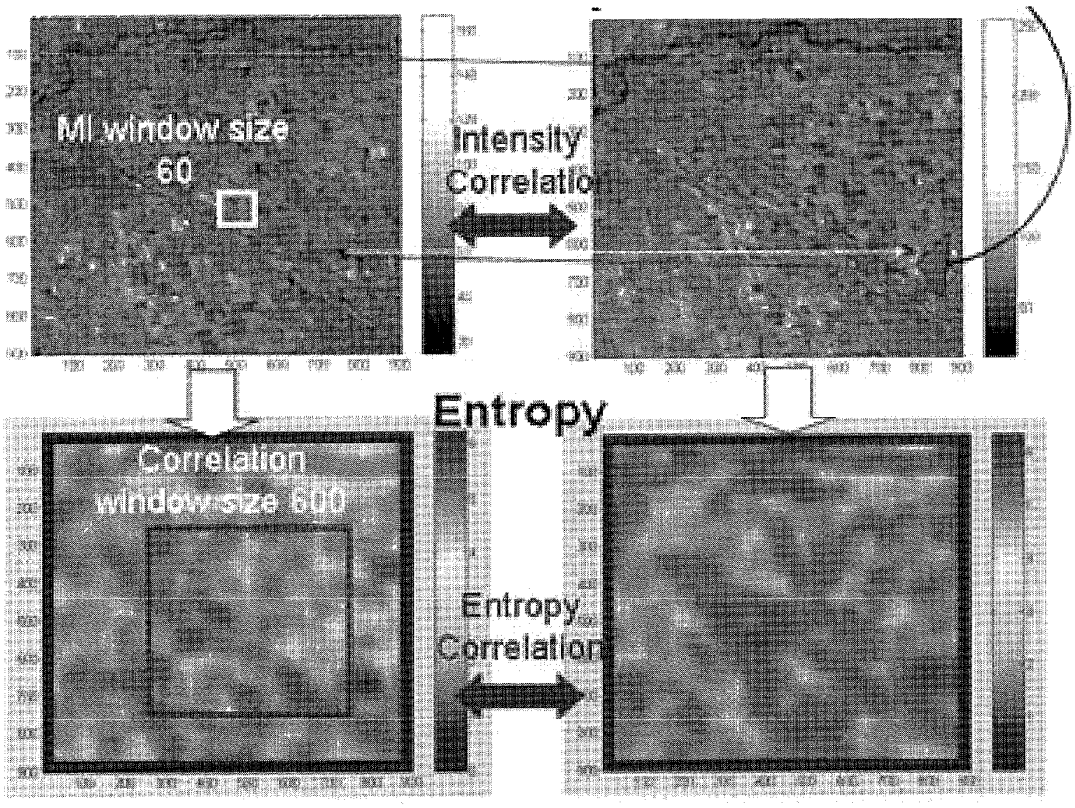
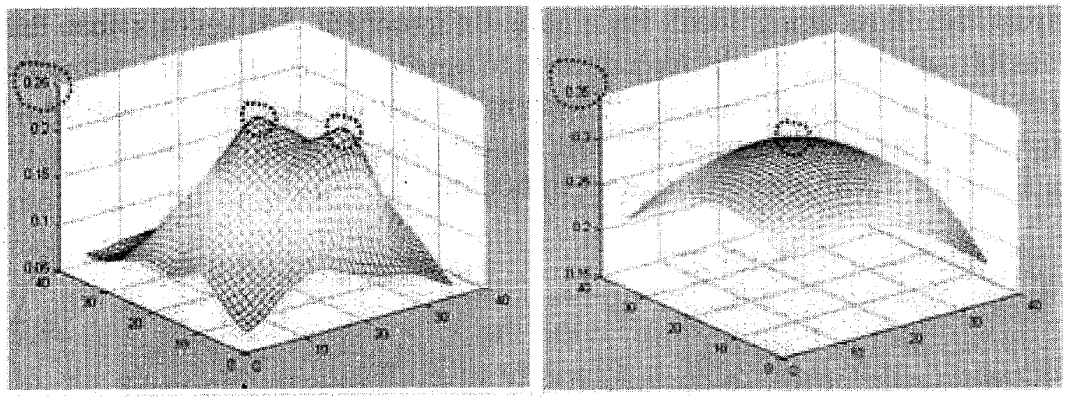


Image (a) Image (b)

Figure 16. Illustration of offset, scaling, and rotation phenomenon at the scale of warping; i.e., 100s pixels. Image (a) and (b) or the time lapsed image pair. Blue vector maps corresponding river bend features. White vector maps the lake. The dark vector maps the registration vector for the selected features. The lower two figures are the displays of entropy transformed using equation 8 on the corresponding intensity imagery pair. See also color plates.

Figure 17 illustrates the correlation of both radiance intensity and its entropy images with 35-pixel lag in both horizontal and vertical directions. Radiance intensity-based correlation is to the left side of the figure and entropy

to the right. The correlation window is centered near the locus of the image. The comparison of these two correlations indicates the entropy-based data has number about 0.1 higher than that of intensity based. Further more, entropy shows only one peak while the correlation appears to have a sharper peak but with multiple potential solutions. Notice the improvement of correlation coefficient of entropy over intensity, and the existence of a unique peak for entropy



(a) Intensity based cross correlation (b) Entropy based cross correlation

Figure 17. Comparison of the cross correlation between intensity and entropy with the rest of parameters staying constant

To further demonstrate the benefit of using entropy and MI, this process is repeated over a 7 by 7 regular grid centered across the imagery pair. The results are plotted in Figures 18 and 19 with radiance intensity-based correlation displayed in Figure 18 and entropy-based results in Figure 19. There are total 49 sub plots corresponding to their locations within the image in each of the correlation figures. The peaks, or the maximum correlation, supposedly are the solutions for control registration points.

The advantages are at least twofold: 1) the improvement on computational complexity, and 2) the consistent and unique solution for mapping control pairs across 100s of pixels. These advantages could potentially provide solutions to map overlapping areas and provide the necessary inputs for higher order registration beyond standard physical photogrammetry models.

One of the potential key contributions from the MI based approach is its ability to match the required solution for a pre-selected model or method at the similar spatial resolution and extent requirements. The ability of MI to transform imagery and its derived product to the quantity of information domain may provide the necessary mathematic foundation for conflation.

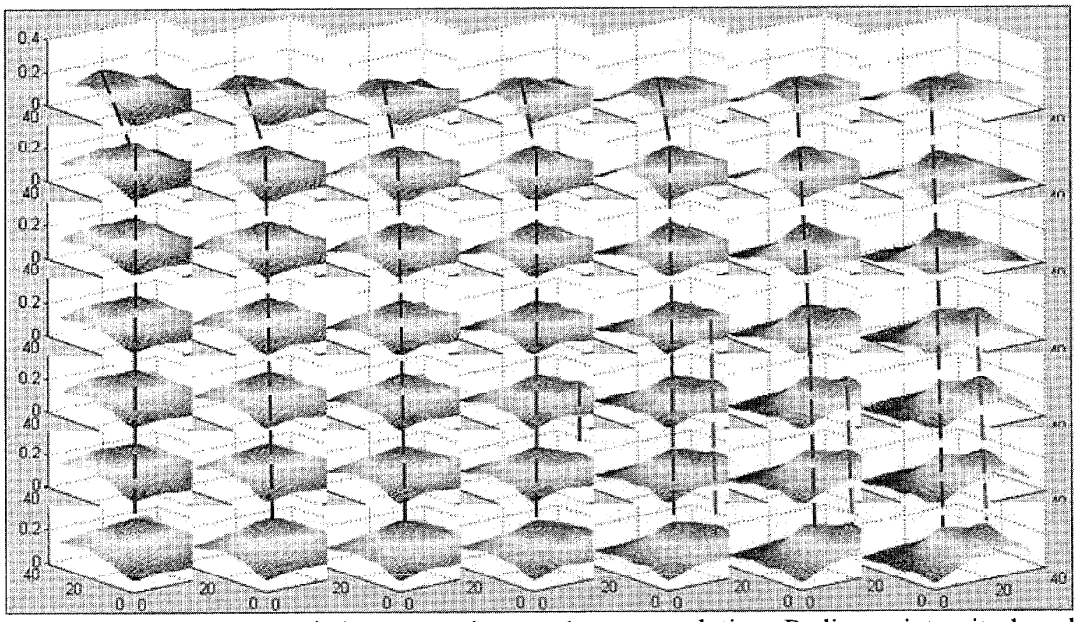


Figure 18. Intensity correlation non-unique and wrong solution. Radiance intensity-based correlation from total of 49 locations centered on a 7x 7 grid over the image pairs in Figure 16. The connection of the correlation peaks does not have the same trend as the registration vector displayed in Figure 16. See also color plates.

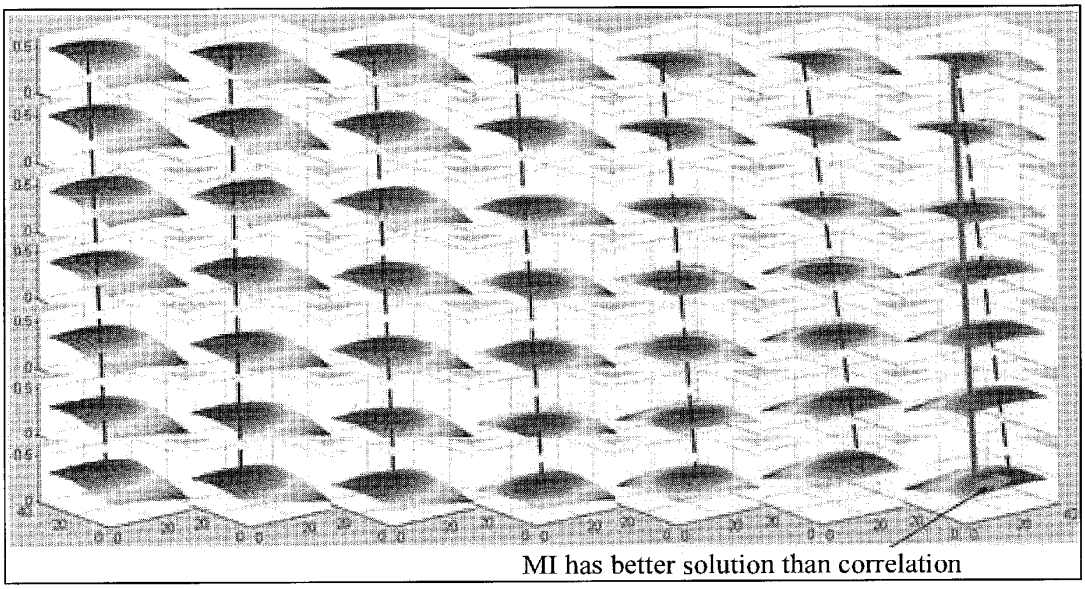


Figure 19. Entropy correlation. The trend matches with data. Entropy-based correlation with the same lay out in 18. The connection of the of the correlation peaks does have the same trend as the registration vector from Figure 16. See also color plates.

8. CONCLUSION

Scientific, economic, and national security interests gain benefits from decision makings utilizing multiple information sources. These information sources are increasingly becoming higher in dimension. Imagery and the information extracted from them become increasingly more complex to integrate under time and space operation constraints. The state-of-the-art methods and algorithms turn out to be less adequate to meet the needs of information analysis and integration. This chapter is intended to be the beginning of a required systematic approach to address the complexities in the integration of information derived from imagery. The three important aspects in addressing the information integration include (1) decision making framework and data, (2) information quality and/or error analysis, and (3) methods, rules and algorithms. This chapter defined imagery and geospatial data conflation and its levels accompanied by the examples.

Figure 3 illustrates a framework incorporating potential contributions from the integration of visual and analytic methods. This framework is adaptable to specific scenarios of similar cases as well as individual tasks of a given application case. It provides a starting point for defining conflation scenarios, cases, tasks, and their refinements to meet mission specific requirements. It is equally applicable to algorithmic selection and computational analysis.

Despite the potential complexities of inconsistencies in any conflation scenario, the geometric discrepancies are mathematically quantifiable for a given task and application case. The error analysis indicates that the key to increasing the quality of reducing spatial disparities lies in the decoupling of the dependencies of the rotation component from its location dependency. Correlation has been the standard in spatial information integration, though it has been known to fail in the cases where either information is overwhelming or inconsistent. Mutual information has the potential to overcome some of the shortcomings of the use of correlation in conflation. Further work in MI is vital to understand the advantages and disadvantages in the area of information integration using MI algorithm at multiple scales and across different extents. It is anticipated that large amounts of knowledge will be accumulated in each of the three areas discussed here as research activities proceed, and an expert system with the ability to capture and analyze the performance of a framework and its associated methods and algorithms will be a great catalyst for information integration using imagery.

The scenario of conflation was described along with linked concepts of conflation situation and a gold standard for conflation. The conflation scenario is based on sets of rules: (1) rules for linking conflation situation to a gold standard, (2) rules for identification of conflation situation; (3) rules for

identification of the gold standard; (4) conflation rules based on the gold standard. Future work includes developing and implementing more rules and detailed quality assurance tools.

9. ACKNOWLEDGEMENTS

This research has been supported by the US National Imagery and Mapping Agency (currently NGA) and the US Department of Energy that are gratefully acknowledged.

10. EXERCISES AND PROBLEMS

1. Build 3-D error equations by generalizing the 2-D model presented in equations (1)-(7).

2. Decompose your 3-D errors from exercise 1 in a way similar to what was done in equation (7) for 2-D.

3. Build a 2-D error equation for an affine transform:
   $x' = m_{11}x + m_{12}y + m_{13}$
   $y' = m_{21}x + m_{22}y + m_{23}$
   Tip: use differential approach taken in equations (1)-(6).

4. Decompose your 3-D errors from exercise 3 in a way similar to what was done in equation (7) for 2-D.

5. Build new conflation rules similar to those presented in section 6.

11. REFERENCES

Brown, L. A Survey of Image Registration Techniques, ACM Computing Sur-veys,vol.24 (4),pp. 325--376, 1992, citeseer.nj.nec.com/brown92survey.html

Edwards D., Simpson J. Integration and access of multi-source vector data, In: *Geospatial Theory, Processing and Applications*, ISPRS Commission IV, Symposium 2002, Ottawa, Canada, July 9-12, 2002.

Edwards, D., Simpson, J. Integrating, Maintaining, and Augmenting – Multi-source Data through Feature Linking , In: *OEEPE/ISPRS Workshop: From 2D to 3D; Establishment and Maintenance of National Core Geospatial Databases,* 8-10 October 2001, Hannover; OEEPE, Frankfurt am Main, Germany, 2002,

Federal Geographic Data Committee FGDC-STD-999.1-2000, 2000, http://www.bts.gov/gis/

fgdc/introduction.pdf

Ghosh, S.K. Analytical Photogrammetry, Pergamon Press, New York, 1998.

Maes F, Colignon A, Vandermeulen D, Marchal G and Suetens P, Multimodality image registration by maximization of mutual information, *IEEE Trans Medical Imaging*, 1997, vol. 16, pp 187-198.

Moniz, E., Baldeschwieler J., Approaches to Combat Terrorism (ACT): Opportunities for Basic Research, Chantilly, VA, November 19-21, NSF, 2002, http://www.mitre.org/public/act/10_22_final.pdf

Scott, D., Conflation for Feature Level Database (FLDB), NIMA National Technology Alliance, 2003, http://www.nta.org/gi.htm.

Shannon, C. E. A Mathematical Theory of Communication. The Bell System Technical J. 27, 379-423 and 623-656, July and Oct. 1948. http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf.

Shannon, C. E. and Weaver, W. Mathematical Theory of Communication. Urbana, IL: University of Illinois Press, 1963.

Studholme C., Hawkes D.J., and Hill D.L.G., An overlap invariant entropy measure of 3D medical image alignment, Pattern Recognition, vol. 32, pp. 71-86, 1999.

Swiftwure Spatial Systems Inc., Near-term conflation requirements at NIMA, and existing conflation capabilities, Report, 2936 Phyllis Street, Victoria, BC V8N 1Z1, Nov. 30, 2002.

Wells, W. M., Viola, P., Atsumi, H., Nakajima, S., and Kikinis, R., Multi-modal volume registration by maximization of mutual information, Medical Image Analysis, 1(1):35--51, 1996

Zitová B., Flusser J., Image registration methods: a survey, Image and Vision Computing. 21 (11), 2003, pp. 977-1000

Chapter 19

CONFLATION OF IMAGES WITH ALGEBRAIC STRUCTURES

Boris Kovalerchuk, James Schwing, William Sumner, and Richard Chase
Central Washington University, USA

Abstract: Spatial decision making and analysis heavily depend on quality of image registration and conflation. An approach to conflation/registration of images that does not depend on identifying common points is being developed. It uses the method of algebraic invariants to provide a common set of coordinates to images using chains of line segments formally described as polylines. It is shown the invariant algebraic properties of the polylines provide sufficient information to automate conflation. When there are discrepancies between the image data sets, robust measures of the possibility and quality of match (measures of correctness) are necessary. Such measures are offered based on image structural characteristics. These measures may also be used to mitigate the effects of sensor and observational artifacts. This new approach grew from a careful review of conflating processes based on computational topology and geometry. This chapter describes the theory of algebraic invariants, a conflation/registration method with measures of correctness of feature matching.

Key words: data fusion, imagery conflation, algebraic invariants, geospatial feature, polyline match, measure of correctness, structural similarity, structural interpolation

1. INTRODUCTION

Algebraic invariants form a new methodology that automates the combining and correlation/registration of images from many sources with various resolutions and reliability, giving them common scales and coordinates (for analysis of different approaches see Chapters 17 and 18). Algebraic invariants use techniques that are based on matching linear features described by mathematically constrained line segments (polylines). This method matches polylines using their robust structural characteristics instead of more traditional matches based on less robust geometric distances.

traditional matches based on less robust geometric distances. The new method permits high speeds and automation since matches are done using only a small fraction of the total image data. Structural characteristics are measured separately for selected features (this can be done off-line) instead of time intensive feature pair comparison as required by other methods. This approach can also be used to automate the identification of locations that change in time, and be used to automatically search for specific objects of interest by predefining their abstracted linear shapes.

The registration and conflation method based on algebraic invariants using polylines may be done in several ways. Consider the two satellite images of the Kyrgyz lake Sonkyl in Figure 1.

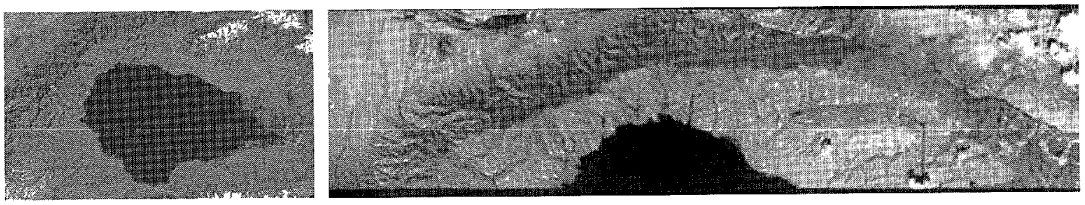


Figure 1. Two images of the lake Sonkyl in Kyrgyzstan

Feature extraction programs can be used to construct numerous polylines, with the most obvious one being the shoreline of the lake. The results are shown in Figures 2 and 3.

While the overall structure of the extracted shorelines is apparent, the polylines differ in detail for a variety of reasons. Robust ways of comparing these polylines are necessary to determine image transformations and to assess the quality of the result.

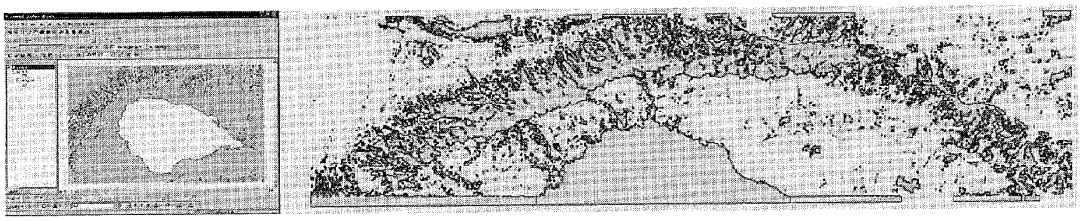


Figure 2. Extracted Sonkyl features

The angles between segments and individual segment lengths are two algebraic characteristics of polylines that can be used. For smooth features extracted from images with comparable scales and resolutions, either comparison works well. When there are marked differences in image scale and resolution, the choice of angles or lengths becomes more important. This chapter examines characteristics of extracted polylines and how they may be interpolated and compared and used to conflate images.

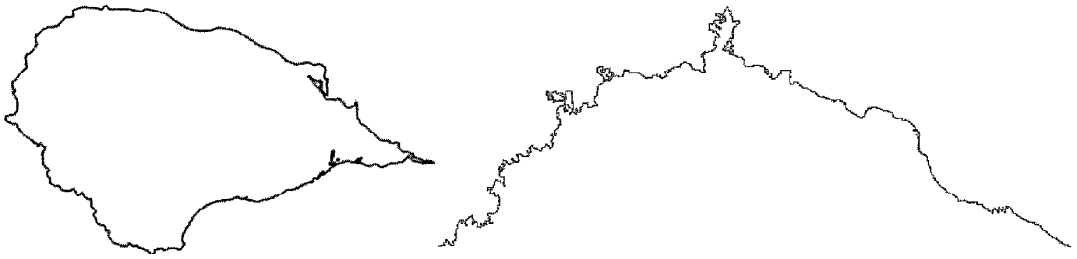


Figure 3. Lake shores extracted from the photographs of Sonkyl

2. ALGEBRAIC INVARIANTS

We considered both topological and geometrical models before deciding on an algebraic approach as a major tool for registering images. Features within different images have different properties such as the number of segments and different beginning and ending points. These variations in the same feature make the use of algebraic invariants very attractive, particularly when compared to the challenges that they present for other methods. (Our comparison of methods is summarized in Chapter 1.)

The algebraic invariant approach to conflating images makes the following **assumptions**:

(1) The images (satellite images, gravity maps, aerial photos, digital elevation maps, synthetic aperture radar (SAR), etc.) have **no common reference points** established in advance for matching them.

(2) The images have **different (and often) unknown scales, rotations and accuracy**.

(3) Each image has several well-defined "features" that can be represented as polylines (continuous chains of line segments). A feature is a wider concept than is commonly used in image sciences. **Anything with a reasonably well-defined shape will work as a feature.** A closed polygon is also a feature in this sense. The only requirement is that the feature can be fit with a polyline. It is not necessary to know what it is or if there are any correspondences with polylines in other images.

(4) These well-defined features can be relatively easily extracted.

The example considered above illustrates the method of the use of polylines as a base for applying algebraic invariants to image conflation/registration.

We now describe the major concepts related to algebraic invariants that are the basis of our techniques, beginning with definitions of terms.

2.1 Algebraic definitions

Definition. A pair **a** = ⟨A, Ω⟩ is called an **algebraic system** [Mal'cev, 1973] if A is a set of elements, Ω a is a set of predicates {P} and operators {F} on A and on its Cartesian products, where P: A×A×...×A → [0, 1] and F: A×A×...×A → A.

Definition. A triple **a** = ⟨A, R, Ω⟩ is called an **multisort algebraic system** if A and R are sets of elements, Ω_a is a set of predicates {P} and operators {F} on A and R and on their Cartesian products, where P: $B_1 \times B_2 \times ... \times B_n \rightarrow [0, 1]$ and F: $B_1 \times B_2 \times ... \times B_n \rightarrow B_{n+1}$, where each B_i is A or R.

A set of axioms can be associated with an algebraic system to generate a specific system such as a group, a field or a linear feature.

Definition. An algebraic system $\boldsymbol{a} = \langle A, R, \Omega_a \rangle$ is called a **linear feature** if R is a set of real numbers, Ω_a consists of two operators (functions) $D(a_i)$ and $L(a_i, a_j)$ and three predicates (linear order relations) $>_a$, $\geq_D$, $\geq_L$. Thus $\Omega_a = \langle D(), L(,); >_a, \geq_D, \geq_L \rangle$ where:

1. $\forall a_i, a_j \in A$: $a_i >_a a_j$ or $a_j >_a a_i$ (All elements of A are totally ordered.)
2. D: $A \rightarrow [0, \infty)$ (An element a is called a linear interval and $D(a)$ is the length of a.)
3. L: $A \times A \rightarrow [0, 360]$ ($L(a_i, a_j)$ is called an angle between a_i and a_j.)
4. $a_i \geq_D a_j \Leftrightarrow D(a_i) \geq D(a_j)$. (This links $\geq_D$ with $D()$). We call elements a_i, a_j linear intervals and say that element a_i is no shorter than element a_j if $a_i \geq_D a_j$, that is, $D(a_i) \geq D(a_j)$).
5. $\forall a_i, a_j, a_k, a_m \in A$: $(a_i, a_j) \geq_L (a_k, a_m) \Leftrightarrow L(a_i, a_j) \geq L(a_k, a_m)$ (This links $\geq_L$ with $L(a_i, a_j)$).

Properties:

- $\forall a_i, a_j \in A$: $a_i \geq_D a_j$ or $a_j \geq_D a_i$.
- $\forall a_i, a_j, a_k, a_m \in A$: $(a_i, a_j) \geq_L (a_k, a_m)$ or $(a_k, a_m) \geq_L (a_i, a_j)$.

2.2 Co-reference: definitions and a theorem

Definition. An algebraic system $\tilde{a}$ is called an **abstracted linear feature** of feature a if $\Omega_{\tilde{a}}$ consists of three predicates (linear order relations) $>_a$, $\geq_D$, $\geq_L$, with $\Omega_a = \langle >_a, \geq_D, \geq_L \rangle$ from the linear feature a.

Definition. An algebraic system $e = \langle E, \Omega_e \rangle$ is a **subsystem** of an algebraic system $a = \langle A, \Omega_a \rangle$ if $E \subseteq A$ and $\Omega_e = \Omega_a$. The subsystem relation is denoted by $e_a \subseteq a$.

Definition. An algebraic system $e = \langle E, \Omega_e \rangle$ is a **shared subsystem** of algebraic systems
$a = \langle A, \Omega_a \rangle$ and $b = \langle A, \Omega_a \rangle$ if $E \subseteq A$, $E \subseteq B$ and $\Omega_e = \Omega_a = \Omega_b$.

Definition. Algebraic systems $a = \langle A, \Omega_a \rangle$ and $b = \langle B, \Omega_b \rangle$ are **co-referenced** if they have a shared subsystem $e = \langle E, \Omega_e \rangle$.

This property is not easy to test because it requires matching equal elements of A and B in advance, which is a major goal of conflation.

Definition. Algebraic systems $e_a = \langle E_a, R, \Omega_e \rangle$ and $e_b = \langle E_b, R, \Omega_e \rangle$ are **isomorphic** if there is a one-to-one mapping $\mu: E_a \rightarrow E_b$ such that for every predicate P and operator F from Ω_e

$$\forall e_1, e_2, \ldots, e_n: P(e_1, e_2, \ldots, e_n) = P(\mu(e_1), \mu(e_2), \ldots, \mu(e_n)) \text{ and}$$
$$\mu(F(e_1, e_2, \ldots, e_n)) = F(\mu(e_1), \mu(e_2), \ldots, \mu(e_n)).$$

Definition. Linear features $a = \langle A, \Omega_a \rangle$ and $b = \langle B, \Omega_b \rangle$ are **co-reference candidates (CRC)** if they are homeomorphic and have isomorphic linear subfeatures $e_a = \langle E_a, R, \Omega_e \rangle$ and $e_b = \langle E_b, R, \Omega_e \rangle$, where $e_a \subseteq a$ and $e_b \subseteq b$.

Definition. The number of elements $n = |E_a| = |E_b|$ in isomorphic linear subfeatures $e_a = \langle E_a, \Omega_e \rangle$ and $e_b = \langle E_b, \Omega_e \rangle$ such that $e_a \subseteq a$ and $e_b \subseteq b$ is called an **index of co-reference**.

Definition. Subsytem e is called a **maximum co-reference** if e is the largest co-reference subsystem in a and b.

Theorem 1. If the number of elements in linear features a and b equals n, then their maximum co-reference subsystem e can be found in $O(n^3)$ comparisons of matrixes for the worst-case scenario.

Proof. To prove this theorem we note that the task is equivalent to finding the largest common submatrix such as shown in Tables 1 and 2 below. This submatrix should be centered on the diagonal of the two matrixes for **a** and **b** as shown in Table 2. The total number of such matrixes is $n+(n-1)+(n-2)+...+2+1=(n+1)n/2$. To find the largest common submatrix we need to compare submatrixes of the same size $i \times i$ in both matrixes.

Table 1. Illustrative matrix for feature **a**

	L1	L2	L3	L4	L5	L6
L1	1	0	1	1	0	1
L2		1	0	1	0	0
L3			1	0	1	0
L4				1	1	0
L5					1	1
L6						1

Table 2. Illustrative matrix feature **b**

	L 1	L2	L3	L4	L5	L6	L7	L8	L9	L10
L1	1	0	1	1	0	1	1	1	0	1
L2		1	0	1	0	0	0	0	0	1
L3			1	0	1	0	1	0	0	1
L4				1	1	0	1	1	1	1
L5					1	1	0	0	0	0
L6						1	1	1	0	0
L7							1	0	0	0
L8								1	1	1
L9									1	1
L10										1

There are n smallest 1×1 submatrixes that contain a single binary number. These submatrices are actually individual diagonal elements. Each submatrix (element) from A_a should be compared with n submatrixes from A_b, that is n^2 matrix comparisons. Because every 1×1 submatrix contains only one element, there are the same n^2 binary number comparisons.

There are n-1 matrixes of the next size 2×2, each such matrix contains 4 elements (only three of them really need to be compared because the matrix in antisymmetric). Each of $(n-1)$ matrixes in A_a needs to be compared with all $(n-1)$ submatrixes in A_b. This requires $(n-1)^2$ matrixes, and $3(n-1)^2$ binary number comparisons.

Similarly for every matrix of size $(n-k) \times (n-k)$, there are $(n-k)^2$ matrix comparisons.

The total number of matrix comparisons is $O(n^3)$ because:

$$n^2 + (n-1)^2 + (n-2)^2 + (n-3)^2 + \ldots + (n-k)^2 + \ldots + 2 + 1 = n(n+1)(2n+1)/6.$$

To compute the complexity of binary comparisons we need to consider the number of elements in the matrix. Each matrix of size $(n-k) \times (n-k)$ contains $(n-k)^2$ binary elements. Because the matrixes are antisymmetric we need to compare only $(n-k) + (n-k-1) + \ldots + 3 + 2 + 1 = (n-k+1)(n-k)/2$ binary elements. Combining the number of matrixes of each size, $(n-k)^2$ and the total number of elements in each matrix, $(k+1)^2$will give us the total number of binary comparisons:

$$n^2 * 1^2 + (n-1)^2 2^2 + (n-2)^2 3^2 + \ldots + (n-k)^2 (k+1)^2 + \ldots + 2^2 (n-2)^2 + \ldots + 1^2 (n-1)^2 .$$

To estimate this number of comparisons we note that each element is smaller than n^4 and there are n elements in the sum, so the sum is less then n^5. Thus, the number of binary comparisons is $O(n^5)$.

This polynomial complexity can be significantly reduced by converting each matrix A_a and A_b to the special linear structures with n elements as we discuss in section 3 below. Similarly to the previous consideration it is $O(n^3)$ for linear structure comparisons, but it requires less binary comparisons. For submatrices of size $s \times s$ it is s instead of s^2. Thus, the total complexity is $O(n^4)$ in contrast with $O(n^5)$ shown above. Generation of the special linear structure itself requires $nlgn$ binary comparisons for sorting, so that does not change the total complexity, that is $O(n^4)$.

Another important computational issue is the fact of monotonicity, that is if none of the common submatrices of size $s \times s$ are found then there is no reason to search for common submatrices of a larger size. These submatrices can not be shared by A_a and A_b. Thus, the worst-case complexity often is not the case.

2.3 The linear feature as an invariant

Theorem 1 sets up a framework for using *co-reference candidates* (CRC) as algebraic invariants for conflation. In this section, we describe a simplified simulation procedure to identify the scope for such invariants. The informal hypothesis is that if an image contains linear features with significant variation of angles then the CRC will work better than for images containing smaller variations. A similar hypothesis can be made about the lengths of linear intervals in a feature -- if lengths vary significantly (e.g. one interval is double the size of the next) for a given feature then the co-reference candidates extracted would provide a better reference between images. Testing these hypotheses means studying robustness of the algebraic invariants.

Analysis of robustness. Let us be given a sequence of connected linear intervals $a_1 = [v_0, v_1]$, $a_2 = [v_1, v_2]$, ... , $a_n = [v_{n-1}, v_n]$ in an image which forms a linear feature, Figure 4.

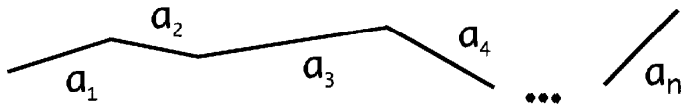


Figure 4. Linear feature

We define and compute a relation P on this feature as follows:

Let $P_i = P(a_i , a_{i+1})$ where

$P(a_i , a_{i+1}) = 0 \Leftrightarrow D(a_i) < D(a_{i+1})$ and

$P(a_i , a_{i+1}) = 1 \Leftrightarrow D(a_i) \geq D(a_{i+1})$.

This relation can be represented as a binary vector, $\mathbf{t} = (P_1, P_2, P_3, \ldots , P_k, \ldots , P_{n-1})$. For example, consider a specific collection of linear intervals that generate a vector $\mathbf{t}_1 = (1, 0, 1, \ldots)$. Note that this vector contains information about the relative lengths of successive intervals. For example, $\mathbf{t}_1$ states that a_1 is no shorter than a_2 while a_2 is shorter than a_3. Denote the i^{th} component of $\mathbf{t}_1$ as t_{1i} .

Next, for the purpose of simulation, suppose we apply non-linear transformations to the points that comprise each of the a_i intervals. Let a vertex $v = (x, y)$, then the transformed vertex v will be transformed component-wise as

$$v' = (f(x, y), g(x, y)).$$

The relation P would then be recomputed for the transformed intervals. Let $\mathbf{t}_2 = (P'_1, P'_2, P'_3, \ldots , P'_k, \ldots , P'_{n-1})$ collect the results of the transformed relation. The vector $\mathbf{t}_2$ can then be compared with the vector $\mathbf{t}_1$ by computing the Hamming distance H between these vectors:

$$H(\mathbf{t}_1 , \mathbf{t}_2) = \Sigma_{k=1,n-1}(t_{1k}-t_{2k})^2.$$

This distance measures the **number of distortions** produced by transformations f and g in relation P. If the distance is 0 then the feature is a P-**invariant for the (f, g) transformation**.

Consecutive angles can be treated in a similar way.

Let $Q_i = Q(a_i , a_{i+1}, a_{i+2})$ where

$Q(a_i , a_{i+1}, a_{i+2}) = 0 \Leftrightarrow L(a_i , a_{i+1}) < L(a_{i+1}, a_{i+2})$,

$Q(a_i , a_{i+1}, a_{i+2}) = 1 \Leftrightarrow L(a_i , a_{i+1}) \geq L(a_{i+1}, a_{i+2})$ and

$\mathbf{t} = (Q_1, Q_2, Q_3, \ldots , Q_k, \ldots , Q_{n-2})$.

As before a transformed vector $\mathbf{t}_2$ can then be compared with a vector $\mathbf{t}_1$ by computing the Hamming distance H:

$$H(\mathbf{t}_1,\mathbf{t}_2)=\Sigma_{k=1,n-2}(t_{1k}-t_{2k})^2.$$

Again, this measures the **number of distortions** in the angles produced by transformations f and g in relation Q and if H is 0 then the feature is called Q-**invariant for the (f, g) transformation**. Note, all angles are measured consistently.

2.4 Similarity measures

Common problems with polylines are discontinuities that result from image resolutions, differences in image acquisition, and artifacts of feature extraction algorithms. Extracted features can be modified in two ways to give a common resolution complexity to facilitate comparisons.

Consider the case of a curvilinear feature that is segmented as a result of something obscuring it, such as a road shaded by a tree. By connecting segments that are "close enough" and have "small" deviation angles, a composite feature can be formed. The maximum separation distance and the maximum deviation angle parameters permitted are clearly critical to feature creation and to one's confidence in the result.

Another type of feature modification is necessary to simplify curvilinear features with one or more relatively narrow lobes for comparison to one with no narrow lobes. For example, a state road map will typically depict a coastline with very little structure. A high-resolution aerial photograph, on the other hand, will show the same coastline with lots of structure, showing that it goes inland for miles along river channels and juts out around spits of land. The higher resolution feature can be simplified by removing these lobes if they are "sufficiently narrow." Critical parameters here are the unit size of feature sampling and maximum jumping distance permitted.

With images prepared with this preprocessing, it is possible to register images that appear at first to have few if any features in common and are of unknown scale and orientation.

Measures of spatial similarity of polylines also need to be developed. Here, the focus is on spatial similarity characteristics while the similarity of non-spatial feature attributes can be matched after a spatial match is confirmed. If two images are matched using only a few reference points, the similarity of other points also needs to be assessed.

The issue of variability of points that form a polyline also needs to be addressed. Different feature extraction algorithms and imagery analysts can assign points differently on the same physical feature. This can affect finding

co-reference candidate features. The technique described below addresses this problem in a computationally efficient way.

Consider the polyline in Figure 5 and the successive approximations defined by taking points defined by the mid point of lengths along the polyline as indicated. The method is called **Binary Sequential Division (BSD) method** computed by finding a curve middle point along the curve, then repeated for each half, halves of halves and so on.

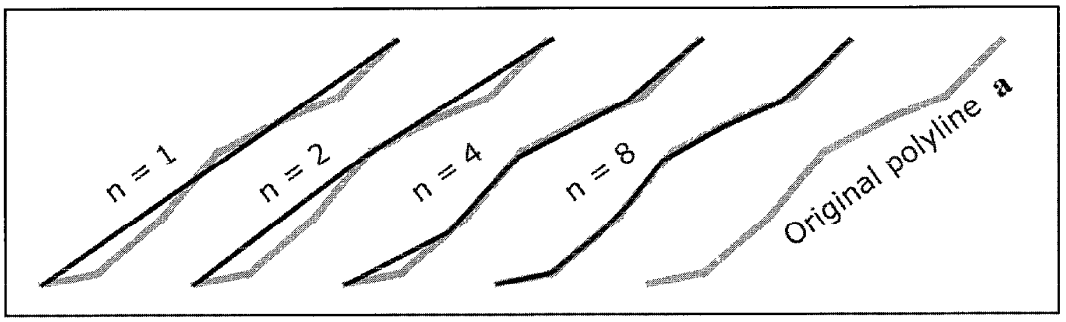


Figure 5. Structural interpolations of a polyline. Our experiments show that 8 binary sequential divisions with $256=2^8$ linear segments is typically sufficient for interpolation.

More formally a function G can be defined as follows for its first three values:

$G(2^0) = G(1) =$ [p1, p2];
$G(2^1) = G(2) =$[p1, middle(p1, p2), p2];
$G(2^2) = G(G(2^0)) =$ [p1, middle(p1, middle(p1, p2)), middle(p1, p2), middle(middle(p1, p2), p2), p2]

In general, notation $G(n)$ will be to denote n-th **interpolation of the polyline** and notation $S(n)$ will be used for the **structure** of polyline $G(n)$. Together the pair $<G(n),S(n)>$ is called **a structured polyline** G(n). Next, we can define the concept of the structure S(n) of the polyline using algebraic concepts for imagery conflation problems. For general polylines we use the pair of matrixes, A and L that represent relations between angles (matrix A) and lengths of intervals (L) in the polyline. These matrices are called **structural matrices**. For each $G(n)$ polyline, the relation between lengths are trivial – all of them are equal by the G(n) definition. Table 3 provides an example of the structural matrix for angles for the original polyline depicted in Figure 5. For instance, bold **0** as a relation between Angle 1 and Angle 2 indicates that Angle 1 < Angle 2.

Table 3. Structural matrix A for polyline **a**

	Angle 1	Angle 2	Angle 3	Angle 4	Angle 5
Angle 1	1	**0**	0	0	0
Angle 2	1	1	0	0	0
Angle 3	1	1	1	1	1
Angle 4	1	1	0	1	1
Angle 5	1	1	0	0	1

Similarly we can construct matrixes A(n) for each polyline $G(n)$. The matrix shown in Table 3 only reflects the upper level structure. For more detailed structure we may include in structure S more matrixes that reflect more specific structural properties, such as relation between differences between angles and relations between second, third and so on differences between angles, similar to second and third derivatives.

Two **structured polylines** $G_a(n)$ and $G_b(n)$ are **structurally equivalent** if their structures $S_a(n)$ and $S_b(n)$ are the same, that is there is a isomorphism between structures. If we restrict the structure by the matrix A shown in Table 3 then the equality of structures means the equality of such matrixes for two different polylines.

Now we can discuss how to define measures of structural similarity between two arbitrary polylines **a** and **b** and use these definitions for matching features and conflating images.

Definition. Two polylines a and b are *n*-**structurally equivalent** if

$$S_a(n) = S_b(n). \tag{1}$$

Definition. We say that there is a **monotone** *n*-**similarity** between polylines **a** and **b**, if for every $n' \leq n$ property (1) is true, i.e., $S_a(n') = S_b(n')$.

Definition. The number n is called the **measure of structural similarity** between polylines **a** and **b** if (i) similarity between **a** and **b** is monotone and (ii) n is the maximum of all n' for which (1) is true. Such n is denoted as n_{max}.

There is no structural equivalency for n' greater that n_{max}:

$$\forall\, n' > n_{max}\ \ S_a(n) \neq S_b(n).$$

The following definition describes the concept of **stable structure** of the polyline. Let $\mathbf{a}_e$ be a polyline obtained from polyline **a** by extending or cutting the end intervals of the polyline (see Figure 5). Index e at $\mathbf{a}_e$ indicates the added or deleted length of polyline **a**. Now we can produce structured

interpolations $<G_{ae}(n), S_{ae}((n)>$ for $\mathbf{a}_e$ and measure similarity between **a** and $\mathbf{a}_e$.

Definition. Polyline **a** has $(\mathbf{a}_e,n)$**-stable structure** if the n-similarity between **a** and $\mathbf{a}_e$ is monotonic.

For matching purposes we use sets of n-stable polylines, say {**a**} and {**b**}, in both images, We search among them for those features that have the same structure. If such feature do not exist then we search for subpolylines of polylines in {**a**} and {**b**} that have the same structure. If such features do not exist then we search for pairs of features with highest measures of structural similarity introduced above as the measure of **monotonic n-similarity**.

$G(n)$ is used to denote the n-th interpolation of a polyline. As mentioned above $n=2^k$, and k is called the **BSD level.** The first four steps of the conflation process are:

Step 1. For raster images *extract* several linear features as sets of points (pixels), S. For vector images skip this step.
Step 2. *Vectorize* extracted linear features. For vector images skip this step.
Step 3. For both raster and vector images analyze the complexity and connectivity of vectorized linear features. If features are too simple (contain few points and are small relative to the image size) combine several features in a *superfeature.* If features are too complex, simplify features by applying a gap analysis algorithm. In the ideal situation we also should be able to separate feature extraction algorithm artifacts from real features. In the example here, the algorithm introduced artifacts, by capturing vegetation as a part of the shoreline in several places.
Step 4. *Interpolate* each superfeature as a specially designed polyline using the BSD method.

Figure 6 depicts level 1 BSD interpolations with k=1 and n=2 for the vectorized features shown in Figure 3. The middle points of each feature are as shown, computed along each line. Significant fluctuations have been lost in the lower resolution image. Feature M as interpolated has angles A_1, A_2, and A_3. Feature L as interpolated has angles B_1, B_2, and B_3

Figure 7 depicts the next level of BSD interpolations for the same vectorized features from Figure 3.

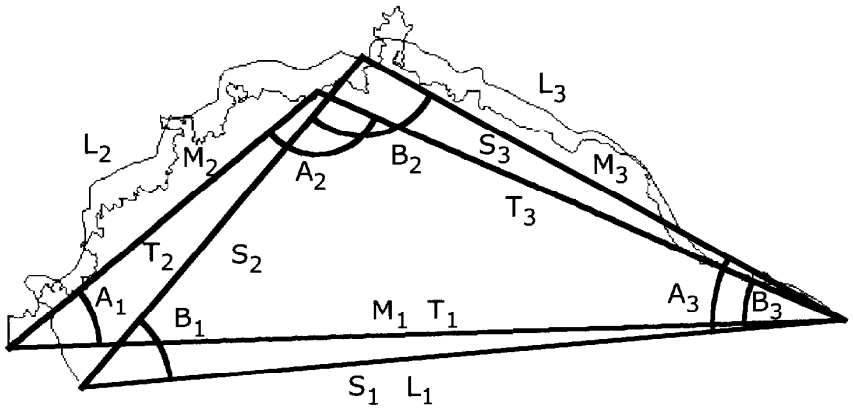


Figure 6. Sections of the extracted shorelines with the first level BSD interpolations with k=1 and n=2

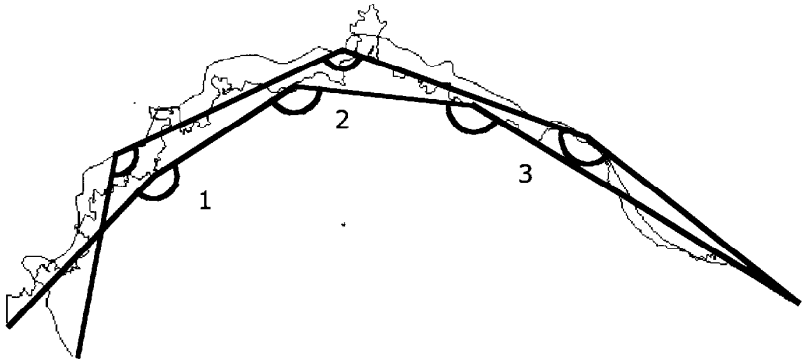


Figure 7. Fragment of BSD level 2 for the two polylines

We also developed a **modification** of the method that is based on **lengths** that permits to measure a *structural difference* in lines in the following way:

Step 1: Compute L, the length of the line from its start to the middle point M along the line.

Step 2: Compute the lengths of two "shoulders", S1 and S2, that is S1 the length of the straight line [T, M] between the start point T and the middle point M. Similarly compute S2, the length of the straight line [M,E] between the middle point M and the end point E.

Step 3. Compute ratios R1=S1/L and R2=S2/L, where L is the half of length of the feature computed along the feature. If a ratio R1 is close to 1 then the first part of the feature is close to the straight line, similar with R2. If both R1 and R2 are significantly larger than 1 and have similar values, say R1= 10.5 and R2=11.4 then on the first level of structural similarity features F1 and F2 are similar.

Step 4. Repeat steps 1-3 for subfeatures [T, M] and [M, E] recurrently for theirs subfeatures until all ratios will be equal to 1 (straight line).

For any polyline with n nodes it is required to repeat step 4 no more than $n\lg n$ times to get all ratios equal to 1.

This can be shown by considering an extreme case (EC) : where after the finding a middle point M only the single end points will be in the right "shoulder." This means that the n-1 nodes are in the part between the start node T and M including node p_{n-1}.

We know that the polyline between node p_{n-1} and the end node E= p_n is a straight line [p_{n-1}, E]. Above we also assumed that M is between them, thus [M, E] is a straight line and a part of the longer straight line [p_{n-1}, E]. That is ratio R_2 is equal to 1 for [M, E] and Step 4 took only one iteration for the subfeature between M and E.

Now we assume that the same extreme case (EC) is true for the left sub-feature from T to M and subsequently for all its subfeaures. That is we need to repeat step 4 only n times for this extreme case.

If our previous assumption is not true and we have more than one node in the subfeature between M and E, we can repeat such binary search process at most for $\lg n$ times to come to the situation with a single straight line. Having total n nodes it may take $n\lg n$ loops with Step 4 in the worst case.

We visualize the structural length type of the feature related to lengths in Figure 8.

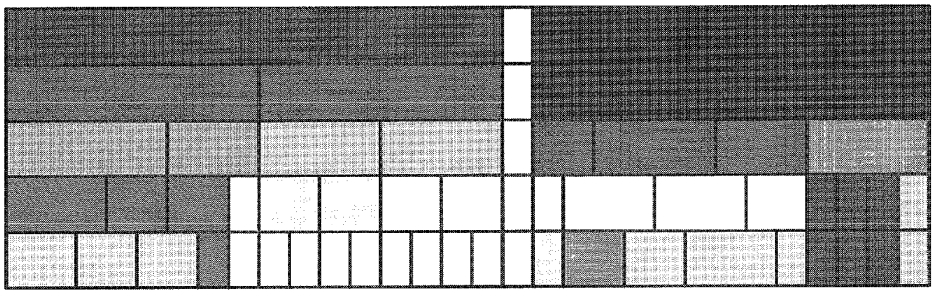


Figure 8. Illustration of structural lengths. See also color plates.

Figure 8 shows that ratios R_1 and R_2 on the first level were basically the same. On the second level it is the same, but on the third level the right shoulder is much larger than the left shoulder. In the fourth level the right part is symmetric, but left part is not. The last level 5 also provide a mix of symmetric and asymmetric cases.

2.5 Generation of matrices

Matrices are constructed from the angle relationships and from the length relationships of the polylines by using two algorithms, denoted as the Angle

Algorithm (AA) and the Shoulder Algorithm (SA). Matrix computation form steps 5 and 6 of the conflation algorithm:

Step 5. Compute a matrix Q of the relation between all angles on the polyline by using AA Algorithm.
Step 6. Compute a matrix P of the relation between all lengths of intervals on the polyline by using SA algorithm.

Values of 0 and 1 are used to indicate ≥ and < respectively. For this example, the angular relations for the two polylines are presented in Table 4 and the length (or shoulder) relations are in Table 5.

Table 4. Angular relations for A and B

	A1	A2	A3
A1	A1≥A1	A1<A2	A1≥A3
A2	A2≥A1	A2≥A2	A2≥ A3
A3	A3< A1	A3< A2	A3≥ A3

	A1	A2	A3
A1	1	0	1
A2	1	1	1
A3	0	0	1

	B1	B2	B3
B1	B1≥ B1	B1<B2	B1≥ B3
B2	B2≥B1	B2≥B2	B2≥ B3
B3	B3< B1	B3< B2	B3≥ B3

	B1	B2	B3
B1	1	0	1
B2	1	1	1
B3	0	0	1

Table 5. Length (shoulder) relations for S and T

	S1	S2	S3
S1	S1≥S1	S1≥S2	S1≥S3
S2	S2<S1	S2≥S2	S2<S3
S3	S3<S1	S3≥S2	S3≥S3

	S1	S2	S3
S1	1	1	1
S2	0	1	0
S3	0	1	1

	T1	T2	T3
T1	T1≥T1	T1≥T2	T1≥T3
T2	T2<T1	T2≥T2	T2<T3
T3	T3<T1	T3≥T2	T3≥T3

	T1	T2	T3
T1	1	1	1
T2	0	1	0
T3	0	1	1

Table 6. Matrices for features from images A and B for angles marked 1,2, and 3. Bold numbers indicate differences between angular relations in two features.

	1	2	3
1	1	**0**	**0**
2		1	0
3			1

Image A: angles
1: 2.206752
2: 2.389911
3: 2.797306

	1	2	3
1	1	**1**	**1**
2		1	0
3			1

Image B: angles
1: 2.906888
2: 2.467343
3: 2.702809

The tables above only show matrixes for BSD level 1. In the general case for BSD level k there are $2^k-1=n-1$ segments and each matrix generated has size $(n-1)\times(n-1)$. Table 6 shows part of the BSD levels 2 for the same features.

Another way to compare interpolated polylines is to construct a matrix of shoulder/length ratios S/L, where the length L is computed as a distance between the shoulder end points along the polyline. We use two relations: $S_i/L_i \geq S_j/L_j$ and $|S_i/L_i - S_j/L_j| < \varepsilon$. In the matrix for relation $S_i/L_i \geq S_j/L_j$ all diagonal cells ($i=j$) are equal to 1 because $S_i/L_i \geq S_i/L_i$ is always true. Also $S_1/L_1=1$, since by definition for the base line $S_1=L_1$.

Next, $L_2 = L_3$ by the middle point design, thus $S_3/L_3 \geq S_2/L_2$ is true if and only if $S_3 \geq S_2$. We also know by definition that $1 \geq S_i/L_i$, because S_i as a straight line is shorter or equal to the curve L_i between the same points. This analysis shows that relations for ratios S/L are the same as for S. But ratios can provide more relations, for instance we may discover that $2*S_2<S_3$, i.e., S_2 is twice smaller than S_1.

2.6 Testing data closeness

Once the matrices have been generated, measures of closeness may be calculated for each polyline in one image with each polyline in the other. The AA and SA algorithms make these comparisons. These comparisons form steps 7 and 8 of the Conflation Method:

Step 7. Testing closeness for angles.
Step 8. Testing closeness for lengths.

In the example, Tables 4 and 5 show that matrixes for two features are identical in both images. This means that we have the highest closeness of given features on BSD level 1. However, Table 6 reveals that matrixes for BSD level 2 are different and there is no match on this level. If there are successful matches, higher BSD levels are explored until the match fails. A **match level tree** that shows the deepest match level reached for each section on each BSD level can be used to make a final judgment about feature match.

Table 7. Angle comparisons using thresholds

	1	2	3
1	1	1	**1**
2		0	1
3			**0**

Level 2 BSD angle comparison matrix: image A (columns) and image B (rows) with threshold = 0.209440. Note, 1 indicates the difference in the column and row values is greater than the threshold, $| L_i - L_j | > \varepsilon$. This matrix indicates that Level 2 BSD match fails. There is only one value in the threshold limits.

The set of relations computed by the AA and SA algorithms in practice also include thresholds to make the methods more robust to noise and multi-resolution mismatch. This is illustrated in Table 7.

2.7 Match forecasting and evaluation

The goal of Step 9 is to identify the deepest level of structural match for two features. The brute force version of this step is to repeat steps 7 and 8 for all deeper BSD level k+1, k+2 and so on. The more efficient version of step 9 described below first forecasts feature match on the next BSD level and

computes the next BSD level only if the forecast is successful. Otherwise, the forecasting algorithm explores potential match for halves of the feature (shoulders). If it fails too then the features are cut down as described in Steps 10 and 11. Step 9 contains the following substeps:

Step 9.1. If Steps 7 and 8 are successful on BSD level k, forecast potential match in the next BSD level k+1 using forecasting algorithm FA (see algorithm described below).

Step 9.2. If match is forecasted by FA algorithm for BSD level k+1, repeat step 9 for BSD level k+1 until mismatch is forecasted. If mismatch is forecasted by FA algorithm for BSD level k+1, use FA algorithm for match forecasting for respective halves of the features (right and left shoulders).

Step 9.3. Construct a match level tree that shows the deepest match level reached for each section on each BSD level.

Step 9.4. Evaluate the tree to make a final judgment about feature match.

2.7.1 Match prediction using SA algorithm

Algorithm AA for BSD level k=0 provides us the lengths of three curves L_1, L_2 and L_3 (see Figure 5), where L_1 is the base line connecting two ends of the feature curve, and L_2=L_3 by the definition of the line middle point used to built them. Similarly SA algorithm for k=0 produces three straight lines (shoulders) S_1, S_2 and S_3 connecting two ends and the middle point of the feature curve (see Figure 6). Similarly, for every other k, AA and SA algorithms provide a set of L_i for each polyline segment along the curve and a set of shoulders S_i between those segments (see an example in Figure 7).

Next we compute all S_i /L_i for BSD level k=0 for image A and similar ratios for image B, denoted as T_i /M_i in Figure 6. Say these ratios are 1, 0.5 and 0.8 for image A and 1, 0.3 and 0.83 for image B.

After that we search for a pair of shoulders $<S_i/L_i, T_i/M_i>$ such that $|S_i/L_i - T_i/M_i| > \varepsilon$ in the two images. If shoulders are found, this is declared a mismatch forecast. If there are no such shoulders, the FA algorithm forecasts a potential match on the next level and the Conflation Method proceeds to compute BD level k+1.

In the example above, $|S_2/L_2 - T_2/M_2| = |0.5-0.3| > \varepsilon$, if ε=0.1. Thus, means that the CM method will not compute BSD level k+1=2 for the whole feature but will compute BSD for the third section of the curve with $|S_3/L_3 - T_3/M_3| = |0.8-0.83| < 0.1$.

The motivation for this algorithm is the following. The significant difference in ratio values between two features is indicative that these differences will show up on a deeper level of BSD and features would not match at

higher levels. However, we cannot say at what higher level it will actually happen. The probability to get match on the next BSD level is lower for shoulders with very different ratios than for shoulders with similar ratios. Our simulation experiments confirm this. It is also consistent with an expectation that the probability of a high level of match (k>4) for images with significant noise and different resolution is low. Thus, we cannot expect many of these very good cases and having the majority of low-level matches the FA algorithm will significantly shorten computation time.

2.7.2 Cutting units from features

The next step is to explore the situation when a feature match is not found. In this case, a search for matching sub-features is initiated by cutting a predefined unit from the first superfeature and repeating, keeping other superfeatures unchanged. Currently we use a unit size between 1/200 of the image largest measurement and half the size of a large feature. It is suggested to start from large units to save time.

If no match is found, the process is repeated by sequentially cutting units from all superfeatures until a match is found or the superfeatures are gone. This process forms steps 10 and 11 of the conflation algorithm:

Step 10. If Steps 7 and 8 fail cut a predefined unit from the first superfeature and repeat steps 4-9 for this modified superfeature and unchanged other superfeatures.

Step 11. If match is not found repeat step 10 by sequentially cutting units from all superfeatures and until match will be found or superfeatures are completely cut down.

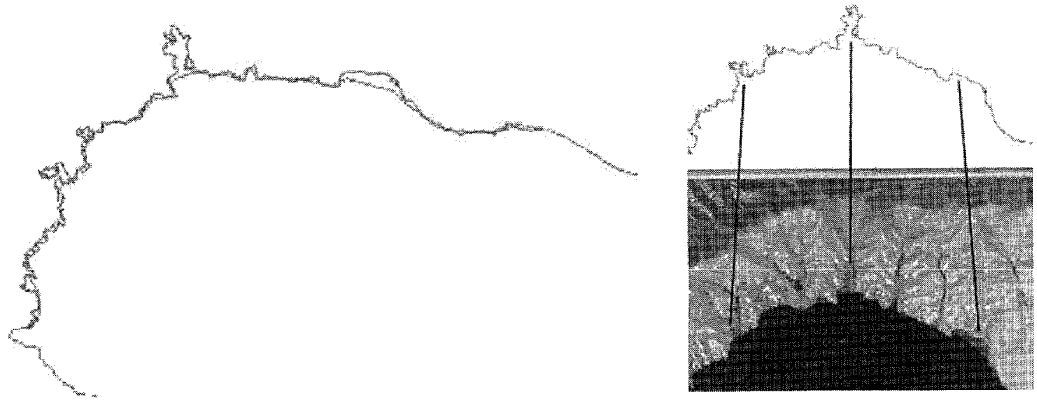


Figure 9. Step 9: Successful match of vectorized features and original image.

Figure 9 shows the final conflation result for the example discussed.

Optimizing unit size. The unit size u used in steps 10 and 11 is the most critical parameter of the whole CM method. If the unit size is equal to the length L of the largest superfeature then steps 10 and 11 are empty. If the unit is a half of the L, u=L/2 then at most step 11 will be repeated two times for each superfeature, that is total $n \times m$ times, where n is the number of superfeatures in image A and m is the number of superfeatures in image B. Our experiments show that three superfeatures per image were sufficient to find the match. Thus, $n \times m$ could be bounded by 9 times of running step 11. If each superfeature contains r units then step 11 will be run $(rn) \times (rm) = r^2 nm$ times, if there is no optimization of time. This is polynomial time complexity function. The combination of AA and SA algorithms provides such time optimization by cutting time of running step 11 by half.

3. FEATURE CORRELATING ALGORITHMS

We begin this section by presenting a simplified version of an algorithm that finds a maximum co-reference. In essence this algorithm finds the largest n-gram common for two Boolean vectors $\mathbf{t}_1$ and $\mathbf{t}_2$. This common n-gram cannot be longer than $\min(n_1,n_2)$ - $H(t_1,t_2)$. There are several known algorithms for finding n-grams with a finite alphabet. In our case, the alphabet is a smallest possible -- {0,1}. Thus, the search is can be relatively fast.

At times, it may be necessary to consider more than consecutive intervals. Using a notation similar to the previous section, let us denote the angle between a_i and a_{i+1} as $L_i = L(a_i, a_{i+1})$. Now record the comparisons between all pairs of angles in a given feature in a matrix. As above, record the i,j entry as 0 iff $L_i < L_j$ and 1 iff $L_i \geq L_j$. Note the resulting matrix is symmetric. Now when two such matrices are built for two features ***a*** and ***b***, a more complex approach in determining a maximum co-reference is based on searching for the largest common part these matrices. This is done by "sliding down" the diagonals of the matrices. Consider the example in Tables 1 and 2 where the largest common parts are highlighted. We will refer to this Matrix Matching Algorithm as MMA1. Later we will also examine an improved Matrix Matching Algorithm, MMA2.

Next we will define a concept used in representing new Linear Structure Algorithms which we will refer to as LS1 and LS2.

Definition. An algebraic system $LS=<I,R,\Omega_l>$ is called a **linear structure for feature** A if

(1) I is the set of integer indexes of elements a_i from feature A according to their order $>_a$ in A. That is $I=\{1,2,\ldots n\}$.

(2) The signature $\Omega_1 = \langle \geq_{cD}, >_{cl}, \varphi, \psi \rangle$ contains two relations $\geq_{cD}$, $>_{cl}$, and two sorting functions φ and ψ that are defined on I such that:

$\forall$ *i,j* from *I*: $\varphi(i) \geq_{cD} \varphi(j) \Leftrightarrow D(a_i) \geq D(a_j)$ and
$\forall$ *i, j, k, m* from *I*: $\psi(i, j) \geq_{cD} \psi(k, m) \Leftrightarrow L(a_i, a_j) \geq L(a_k, a_m)$.

The function φ shows the results of sorting intervals according their lengths and ψ shows the results of sorting angles between adjacent intervals according to their magnitude.

Example: The sequence of lengths of intervals a_1, a_2, a_3, a_4 is 0.1m, 4.3m, 7.6m and 0.5m. Their indexes can be written as follows {1, 2, 3, 4} then by sorting the lengths in descending order, the structure's sequence can be represented by the reordering of the original indices thus: {3, 2, 4, 1}.

In formal terms, this means that we have a substitution (permutation) φ for indexes of lengths such that

$$\varphi = \begin{pmatrix} 1234 \\ 3241 \end{pmatrix}.$$

A similar index substitution function is produced for angles (i.e. for pairs of linear segments with indexes [i, j] and [k, m]).

3.1 The Matrix Matching Algorithms

In this section, we begin by presenting a detailed analysis of MMA1 summarized above. We then analyze an improved Matrix Matching Algorithm MMA2.

3.1.1 An exact complexity formula for MMA1

It has been shown above that an upper bound on the complexity of MMA1 is $O(n^5)$ but the exact formula was not provided [Kovalerchuk & Sumner, 2003]. The general bound is:

$$n^2 1^2 + (n-1)^2 2^2 + (n-2)^2 3^2 + \ldots + 2^2 (n-1)^2 + 1^2 n^2 .$$

We will now demonstrate an exact representation for the complexity of MMA1.

Theorem 2: The complexity of MMA1 when applied to $n \times n$ matrices is given by

$$\frac{1}{120}[2n^5 + 5n^4 - 5n^2 - 2n].$$

Proof. Let S1 and S2 be $n \times n$ *comparison* matrices that share a common subsystem. As mentioned in [2, 3], the matrixes being considered are anti-symmetric so that only upper triangular comparisons are required. Thus,

$$(n\text{-}k) + (n\text{-}k\text{-}1) + \ldots + 3 + 2 + 1 = (n\text{-}k+1)(n\text{-}k)/2$$

binary comparisons need to be made for each subsystem. Thus, the first terms of each item in the sum can be replaced by $(n\text{-}k+1)(n\text{-}k)/2$, which leads to the following representation for the full system:

$$(n(n\text{-}1)/2)*1^2 + ((n\text{-}1)(n\text{-}2)/2)*2^2 + \ldots + 1*0*(n)^2.$$

We can represent and then manipulate the equation as follows.

$$\sum_{k=1}^{n} k^2(n-k+1)(n-k)/2 = \frac{1}{2}\sum_{k=1}^{n}(k^4 - (2n+1)k^3 + (n^2+n)k^2) =$$

$$\frac{1}{2}(\sum_{k=1}^{n} k^4 - (2n+1)\sum_{k=1}^{n} k^3 + (n^2+n)\sum_{k=1}^{n} k^2) =$$

$$\frac{1}{2}[(\frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n) -$$

$$(2n+1)(\frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2) +$$

$$(n^2+n)(\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n)] =$$

$$\frac{1}{120}(2n^5 + 5n^4 - 5n^2 - 2n).$$

Thus, we have shown this is indeed $O(n^5)$. For comparison purposes as we proceed, note that for $n = 10$ this is 2079.

It is interesting to note that for small n ($n \leq 60$) this equation behaves more like $O(n^4)$ due to the small leading coefficient. However, it is not generally unusual to have $n > 60$ segments in a feature.

3.1.2 The improved algorithm MMA2

The performance of MMA1 can be improved further using the principle of monotonicity. In this method, the largest potentially matching matrices are searched first. If the matrix match in progress should fail at a given location, we then continue from where the match was last successful. Thus, we are able to continue looking for a smaller match originating from the same starting positions without repeating work already performed.

In [Kovalerchuk & Sumner, 2003], the issue of *monotonicity* is presented by noting that the worst-case is often not what occurs. That is, if no common matrices of size s×s are found, then there is no reason to search for larger

common submatrices. Now we have chosen to take a reverse viewpoint, while still utilizing the same basic principle. We shall attempt to verify the largest subsystems first from a given set of starting positions in the matrices. If during comparisons, elements are found that do not match, the entire match is not abandoned, the match is just scaled down to include the last matching elements. In this way, the principle of monotonicity can be combined in a dynamic programming approach that reduces the complexity involved.

This can be represented by the equation

$$\sum_{k=1}^{n} k(n-k)(n-k+1)/2 =$$

$$\frac{1}{2}[(n^2+n)\sum_{k-1}^{n} k-(2n+1)\sum_{k=1}^{n} k^2+\sum_{k=1}^{n} k^3]=$$

$$\frac{1}{2}[(n^2+n)(\frac{1}{2}n^2+\frac{1}{2}n)$$

$$-(2n+1)(\frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n)$$

$$+(\frac{1}{4}n^4+\frac{1}{2}n^3+\frac{1}{4}n^2)=$$

$$(n^4+2n^3-n^2-2n)/24.$$

We have shown the following theorem.

Theorem 3: The complexity of MMA2 when applied to $n \times n$ matrices is $O(n^4)$, and is given by

$$(n^4+2n^3-n^2-2n)/24.$$

Again as an example, consider $n = 10$ which yields 495.

3.2 The Linear structure algorithms

In this section, we present two versions of the Linear Structure algorithm LS1 and LS2.

3.2.1 The Linear Structure algorithm, LS1

When setting up a comparison matrix, the values of the table reflect whether the value of element x is greater than or equal to or less than the value at element $x+1$, $x+2$, $x+3$, and so on. In deriving a program to model

all possible combinations of values that also satisfy these same relationships, we found the best method was to first sort the values, keeping track of the indices from which they came, set up a new group of values from lowest to highest, then distribute them back into the proper order as per the original indices. That is, the order of the original indices after sorting fully describes the comparison matrix. Given this order, it is a simple matter to reconstruct the matrix. In fact, this order itself can be used to compare two features in a way in which you do not need the comparison matrices at all.

This is what we call the *linear structure's* sequence. The following example illustrates this concept.

Let us use the length measurements for feature S1: {5, 2, 8, 9, 4}, which have the comparison matrix shown in Table 8. The values 0 and 1 are used to represent $x < y$ and $x \geq y$ respectively, where x is a row value and y is a column value.

Table 8. Comparison of lengths for feature S1

Length	5	2	8	9	4
5	-	1	0	0	1
2		-	0	0	0
8			-	0	1
9				-	1
4					-

Now, we also have S2: {8, 9, 4, 6, 7} (see Table 9). The matching segment is highlighted in both Table 8 and Table 9.

Table 9. Comparison of lengths for feature S2

Length	8	9	4	6	7
8	-	0	1	1	1
9		-	1	1	1
4			-	0	0
6				-	0
7					-

Let us build the index sequence for Table 8. This table depicts the set of lengths of intervals {5, 2, 8, 9, 4} and their relationships. The structural sequence for this table will begin with the value 2, which is the index of the smallest element (3). The next value 5 indicates the second smallest element is from index 5 whose value was 4. This process continues and the final sequence of the indices in S1 is {2, 5, 1, 3, 4}. The sequence of the values by index in S2 is {3, 4, 5, 1, 2}, and is derived from Table 8 in a manner similar to S1. At first it seems that, there is no similarity, but when the first two elements of S1 have been stripped off and the values ranked again, we find the S1 LS-sequence to be {3, 1, 2}. Further, when the last two elements of S2 have been stripped off, we find the S2 LS-sequence to be {3, 1, 2} also. We have found the matching segments. All that is necessary is to delineate all

sequential element segments, convert them to linear structures then compare these sequences with other LS-sequences starting with the longest segments first until we find an exact match. Each sequence will require sorting which can be done in $O(n \log n)$ using, say, a mergesort algorithm [Neapolitan, Naimipour, 1998], and the overall total will involve the addition of all combinations.

This can be represented by the formula:

$$\sum_{k=1}^{n}(n+1-k)k\log(k)$$

and since $\log k \leq \log n$ we have

$$\sum_{k=1}^{n}(n+1-k)k\log(n) = n\log(n)\sum_{k=1}^{n}k + \log(n)\sum_{k=1}^{n}k - \log(n)\sum_{k=1}^{n}k^2 =$$

$$n\log(n)(\frac{1}{2}n^2 + \frac{1}{2}n) + \log(n)(\frac{1}{2}n^2 + \frac{1}{2}n)$$

$$-\log(n)(\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n) =$$

$$\frac{2}{3}n^3\log n + n^2\log n + \frac{1}{3}n\log n.$$

We have shown the following result.

Lemma 1. The complexity of generating LS-sequences, which is $O(n^3 \log n)$, is given by

$$\frac{2}{3}n^3\log n + n^2\log n + \frac{1}{3}n\log n.$$

This process of generating sequences can be performed for each feature before any comparisons between features are made because it requires no information from any other feature. It needs to be noted that mergesort or another stable sorting method is preferred to a sort such as quicksort here because of the importance of preserving the correct order in the event that two equal values are found.

We can now analyze the complexity of the actual comparisons between features.

- For each sequence of length n-k+1 a comparison must be exact for the entire matching section. This requires n-k+1 comparisons. Thus, this is a simple $O(n)$ operation.
- The number of subsequences of length n-k+1 produced from a sequence of length n is k.

- Each of k subsequences of length n-k+1 generated in one feature must be cross referenced with each sequence of the same length in feature 2, which will be $O(k^2)$.

This can be represented by the equation:

$$\sum_{k=1}^{n} k^2(n-k+1) = (n+1)\sum_{k=1}^{n} k^2 - \sum_{k=1}^{n} k^3 =$$

$$(n+1)(\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n)$$

$$-(\frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2) =$$

$$(n^4 + 4n^3 + 5n^2 + 2n)/12 .$$

We have shown the following theorem.

Theorem 3. The complexity of LS1 when applied to $n \times n$ matrices is $O(n^4)$ and given by

$$(n^4 + 4n^3 + 5n^2 + 2n)/12 .$$

As noted, this is $O(n^4)$ which is similar in complexity to the MMA2 method. However, it is not quite as efficient as the MMA2 method. Consider for example $n = 10$. Here the value is 1210.

3.2.2 The Linear Structure algorithm LS2

We can improve the performance of the LS1 method by incorporating some additional storage and sorting of the linear structures created. If linear structures of same length sequences are stored in a sorted order (e.g. a binary tree) then binary searching of this information for the exact match would require only $O(\log n)$ time instead of $O(n)$. The ordering does take extra time, amounting to $O(n^4 \log(n))$, but it can be done off-line, i.e. Before comparing two features. This modifies the previous analysis for LS1 method. The LS1 method has a smaller "off-line" part. Below we estimate the complexity of the "on-line" part of the method LS2. We define the "*on-line*" part the method that is applied for comparison of two features from two images.

We have k sequences of length n-k+1 in each feature F1 and F2 to be compared. Let's take an arbitrary sequence, say (1,2,5,4,...) generated for feature F2 with n-k+1 elements. To find a match for this sequence in an ordered set of sequences for feature F1, we need $\log(k)$ comparisons of sequences, because we have k such sequences.

Next we have n-k+1 elements in each sequence, and we need to compare them with the same n-k+1 elements in another sequence element by element

for each position, that is n-k+1. Thus, for a single sequence, say (1,2,5,4,...) we need (n-k+1)log k individual comparisons of values (indices).

Now we notice that there are maximum k different sequences of length n-k+1, that is the total number of comparisons is: (n-k+1)*k*log k. Now, we derive the formula for all k:

$$\sum_{k=1}^{n}(n-k+1)k*\log(k)=\sum_{k=1}^{n}((n+1)k-k^2)*\log(k)=$$

$$(n+1)\sum_{k=1}^{n}k\log(k)-\sum_{k=1}^{n}k^2\log(k)=$$

Since always k ≤ n, we can substitute log(n) for log(k), which yields:

$$(n+1)\sum_{k=1}^{n}k\log(n)-\sum_{k=1}^{n}k^2\log(n)=$$

$$(n+1)\log(n)\sum_{k=1}^{n}k-\log(n)\sum_{k=1}^{n}k^2=$$

$$(n+1)\log(n)(\frac{1}{2}n^2+\frac{1}{2}n)$$

$$-\log(n)(\frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n)=$$

$$\log(n)*(n^3+3n^2+2n)/6$$

Thus, we have an upper bound for our analysis of

$$\frac{1}{6}n^3\log(n)+\frac{1}{2}n^2\log(n)+\frac{1}{3}n\log(n)$$

Now we have demonstrated the following.

Theorem 4. The complexity of the "on-line" part of LS2 method when applied to two linear features with $n\times n$ matrices is $O(n^3\log n)$.

Table 10. Comparison of complexities of methods

N	10	20	40	80
MMA1	2079	59983	1813266	56319732
MMA2	495	7315	111930	1749060
LS1	1210	16170	235340	3586680
LS2	731	6656	61096	559870

Table 10 shows comparison of complexities of four methods presented above for some typical n values used in linear feature encoding.

For cross-referencing more than two features, this LS2 matching process would need to be extended. We can build this extension using the same

method of storing all linear structures with equal lengths sequences in a sorted order.

For $n = 10$ we have $1*10 + 2*9 + 3*8 + 4*7 + 5*6 + 6*5 + 7*4 + 8*3 + 9*2 + 10*1 = 165$ segments. These segments can then be found in $(n^3 - n)/6$ or $O(n^3)$ time.

If we let m be the total number of features, and we assume for this case that all features are of length n, then to compare to all other features we will have $O(mn^3 \log n)$.

The complexity of the LS2 has shown to be the best for processing efficiency. We have not, however, taken into account the *storage* requirements for maintaining the list of sequences which is $O(n^2)$.

An image with 1000 polylines of 100 points each would require $1000*100*101/2$ storage locations. Since all the indexes in this case are < 256 we could use a single byte of storage apiece, resulting in a storage requirement of 5,050,000 or 5 Mb, which is quite reasonable.

Of the methods investigated, we have found the modified linear structure method LS2 provides the best efficiency with $O(n^3 log\ n)$ followed by the modified matrix matching algorithm MMA2 with complexity $O(n^4)$.

3.3 Robustness of algorithms for images with different resolution and noise level

One of the major conflation challenges is matching features from images with significantly different resolution. Feature extraction artifacts can also contribute to feature disparity. As we have seen in Figure 3, such features have different levels of smoothness and detail. In Figure 10 we show the behavior of the SA algorithm in the presence of random noise that simulates effect of difference in resolution. Binary sequential division (BSD) was applied two times and produced a polyline shown in Figure 10.

To build structural matrices we compute all S_i/L_i ratios for the low resolution line (straight line) and the high resolution line with spikes. Here L_i is the length of the i-th polyline segment along the curve and S_i is the length of the straight line between end points of the i-th polyline segment. These ratios are: 1, 1, 1, 1 for the straight line and, say, 0.4, 0.45, 0.48, and 0.51 for the curve. Next, we check that all these ratios are close enough in each feature separately, that is $|S_i - S_i| < \varepsilon$ for $\varepsilon = 0.15$. All pairs of ratios are in these limits in both features. Thus, the SA algorithm matches given features on this BSD level. Significant differences in ratio values between two features are indicative that on a deeper level of BSD these differences will show up and features will not match at higher levels.

Similarly, we can compute three sequential angles (170°, 165°, and 175°) for the high resolution feature. These angles are all 180° for the straight line.

All angle differences are less than the 15° limit for both images. Thus, they are matched by the AA algorithm on this BSD level. But the AA algorithm does not give any clue that this match may deteriorate in the next levels, in contrast with the SA algorithm that provides such clues.

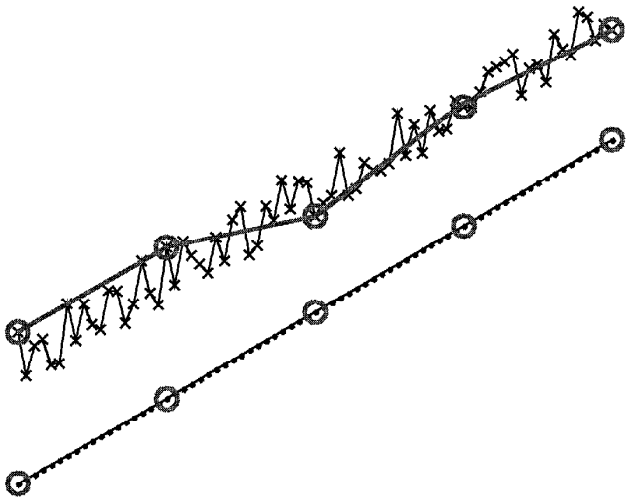


Figure 10. This example indicates that the SA algorithm based on lengths is more robust for similar cases than the AA algorithms based on angles

4. CONFLATION MEASURES

Characterizing and evaluating the quality of conflated data relative to a given problem is one of the most important open problems in geospatial data integration. A major part of this development is producing multidimensional measures of the correctness of conflation, since no single measure of quality of conflated data sets is appropriate for every conflation problem.

Traditionally registration, co-registration, and conflation have been done with images of a similar type [Brown, 1992]. Some processes have used every pixel or every vector in each image. For these processes, the *feature space* is essentially the *image* (refer to Figure 4) and the similarity metrics chosen to guide the process and the transformations allowed between the images have an important bearing on the outcome as does the presence of local minimums in the search space and the search strategy.

Other conflations use feature spaces that have been derived from the images using a variety of techniques including the *raw intensities, extractable edges, salient and statistical features, matches against models*, and so on [Zitova & Flusser, 2003]. There are numerous possible choices for similarity metrics used for both images and features including, *cross-correlation* with and without pre-filtering, *correlation coefficients, phase-correlations, sums of absolute differences of intensity, sums of absolute differences of contours,*

other *contour/surface differences*, *number of sign changes* in point-wise intensity differences, *sums of squares of differences* between nearest points, *minimum changes in entropy*, etc.

Conflation performance is defined by the *quality* and *time* of conflation. We are measuring the quality of conflation by a certainty that features are matched correctly. The speed of conflation is measured by the time required to conflate images. The quality of conflation is a multidimensional characteristic. In this paper, we focus on how close structurally features are in the two images. For each pair of features, it is measured by the BSD level parameter k. We assume that $k \geq 5$ indicates a high quality of conflation match. The final match is done by checking that there are no other matching features nearby with the same or higher level of match. In this is the case, we continue BSD process with both AA and SA algorithms and all highly matched features to find a pair with the highest match.

Error estimates. We need to know how good the conflation result is. A standard approach is to measure the difference between control points after a matching transformation is completed. In the base case, the distance is 0 for all control points after transformation. Does it tell us how good the conflation is or how good is the match for other points? Say, we have 10 control points in an image of 1000x1000 pixels, that is 10 points out of 10^6 points. This means that we evaluated the quality of conflation using 0.001% of the total number of points. The question is how we can evaluate the quality for the other 99.999% of the points without actually having 10^6 control points. The classical smooth interpolation approach does not answer this question explicitly. The only statement that can be made is that smooth continuous interpolations such as linear, polynomial and splines transfer close points to close points. However, how close is not clear. An expectation is that the error for other points should be similar to the errors computed for control points. If we have zero error for 0.001% of the total number of points, we can not claim that we will have zero error for the rest of the image too. Thus, an innovative approach is needed to measure quality of conflations.

Matching non-control points. A transformation T based on control points will also transform all other image points, $T(x)=y$. However, it is not done specifically for them. It is a byproduct of matching control points. Why should point x be matched to point y if x is not a control point? There is a huge number of transformations T' that match control points, but differ from T in other points. Transformation T' can convert point x to points y' that can be close to y but not identical to y. Our conflation method identifies matched non-control points in a more meaningful way using a combination of measures provided by AA and SA algorithms. This is important for enhancing a standard control point matching approach.

Virtual imagery expert. In addition, we are automatically capturing some human abilities to match images using context. In Figure 3 we may assume that an expert knows that the left image is a low-resolution image and the right image is a high-resolution image. Such context knowledge permits the expert to match these features despite their apparent differences. The combined AA and SA algorithm can help to discover this from the image itself by analyzing shoulder relations.

Three similar methods have been developed for processes similar to ours. The first is a technique of matching planar polylines described by Cohen and Guibas [Cohen & Guibas, 1997]. We analyzed their task, called the polyline shape search problem or PSSP for short, to clarify the applicability of their approach to the **algebraic structural conflation (ASC)** task formalized here and to compare the differences in task formalizations and interpretations. Our conclusions are as follows:

(1) Both tasks PSSP and ASC assume that matching may require rotation, scaling and translation of polylines
(2) The task of PSSP is to find a part of a polyline A that matches a polyline B. Polyline B can be a relatively small "template" polyline (e.g., corner-shape or step-shape). See Figure 11 for examples.
(3) The task of ASC is to find the largest matched part C of two larger polylines A and B.
(4) PSSP uses **two criteria for judging the quality of a match**:
    a. the error of the match computed as a distance between polyline A and the matched part of polyline B after B is transformed by shifting, rotating and scaling;
    b. the length of the match is computed as a number of matched breakpoints on the polylines multiplied by α (stretching parameter). This parameter is used to transform B to A.

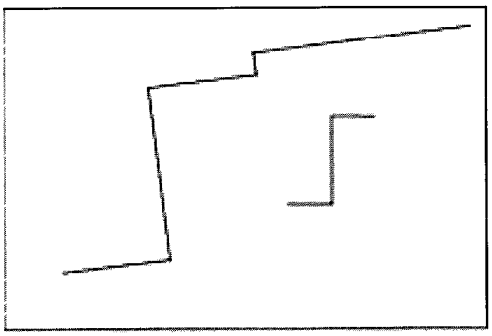
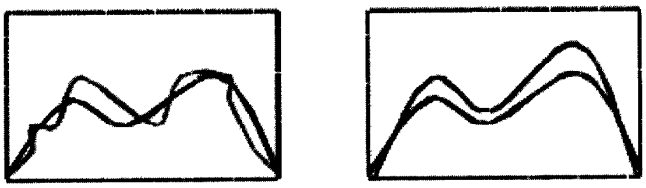


Figure 11. Template examples PSSP

Our ASC method is more general than the one developed by Cohen and Guibas because (1) two arbitrary polylines can be searched for similar por-

tions in each and (2) there are no point limitations on the determination of distances (see the Appendix of this report.)

Another measure of feature shape similarity was defined by Cobb et al. [Cobb, Chung, Foley, Petry, Shaw, & Miller 1998]. The idea of the method is illustrated in the following Figure 12(a) and 12(b).



(a) (b)

Figure 12. Structural differences

The measure of Cobb et al. is based on the following steps:

1. Bring the features to a *standard position* with the start point (x,y)=(0,0) and the end point (x,y)= (1,0) by rotating and rescaling the features. In this way, the first feature is presented as a function P(r), and feature 2 is presented as a function Q(r).
2. *Merge nodes* -- make the number of nodes and their locations on the x coordinate *equal* in both features by mapping all nodes $(p_0,p_1,\ldots,p_n)$ from feature 1 onto the feature 2, and by mapping all nodes $(q_0,q_1,\ldots,q_m)$ from feature 2 onto feature 1. The merged number of nodes is $n + m$ if there are no overlapping nodes in features.
3. Compute the Frechet measure of *distance L_2 between features* as a square root of the integral of squared differences between features using merged nodes:

$$L_2 = (\int_0^1 \|P(r)-Q(r)\|^2\, dr)^{1/2} = (\sum_{i-1}^{n+m} \int_{r_i}^{r} \|P(r)-Q(r)\|^2\, dr)^{1/2}$$

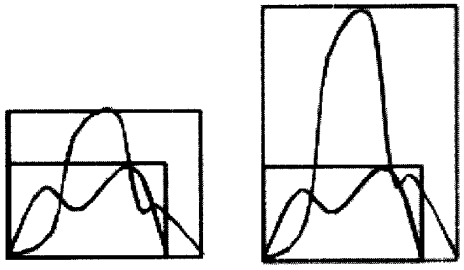
4. Compute the normalized distance between features, L_2 (D_1+D_2), where D_1 and D_2 are lengths of the features in the standard position.

This measure is applicable for the following situations:

1. Features can be arbitrarily rotated and have different sizes.
2. Ratios of scales for x and y are similar for both features. Figure 12(a) shows this case and Figure 13(b) shows the significant difference in ratios in two images indicated by dotted rectangles. In Figure 13(a) Feature 1 is taken from the image where the ratio of width and height of the feature bounding box is 0.5 and feature 2 is taken from the image 2 where the ratio is 0.6.
3. Some structural differences are not important. Figure 12(a) shows structural differences that are not captured by the measure; gray lines

in both Figure 12(a) and Figure 12(b) may give the same similarity with the black line.

4. Each feature is in the bounding box defined by the start and end points (see Figure 12(a). Note this method is not applicable for features shown in Figure 12(b).)



(a) (b)

Figure 13. Examples of scale ratios

(a) (b)

Figure 14. Bounding boxes for standardized features

Our ASC method descried earlier is applicable for both conflation cases presented in Figure 14. It captures the structural differences between two lines shown in Figure 12(b) using angles.

While the method of Cobb is satisfactory for similar features with similar lengths, it lacks the ability to determine partial matches of polylines. By using the Frechet measure of distance L_2 between features it cannot accurately compare structures that are very different and will not work at all with others such as illustrated in Figure 14(b).

Carswell et al. [Carswell, Wilson & Bertolotto, 2002] have developed a composite similarity metric to aid in the location of "image-objects" in images. They combine weighted topology, orientation, and relative-distance similarity measures for the image components Q and compare it with an image scene I. For our earlier example of finding a house with an outdoor swimming pool across a road from a barn and silo, the "image-object" would consist of these five elements with appropriate topology, orientations, and distances defined. Their similarity measure provides a percentage match between the sum of these features Q and similar ones in the image I.

The individual similarity measure weights can be varied to emphasize the relative importance of different image properties. For this example, one

could weight the topological relationships strongly, the relative-distance measure moderately and the orientation similarity metric the least.

This approach should give similar results to one that applies algebraic invariants by combining individual features into composite features and then proceeding with the conflation/co-registration process in a normal manner with loosened distance matching constraints.

5. GENERALIZATION: IMAGE STRUCTURAL SIMILARITY

It is critical to measure the quality of the whole registration/conflation process. We described the issue of matching individual features from two images using the concept of structural equivalence of individual features. Now this concept is used as a base to define structural similarity of two images. No single numeric measure is sufficient to cover the complex variability of images and matching contexts/situations. Thus, we introduce a set of measures that is a more adequate measure of the quality of registration/conflation.

We start with the simple situation with one-to-one feature matches. Every feature in image 1 has a single match in image 2 and the same is true for the image 2. Let $n=(n_1,n_2,...,n_m)$ be a vector of structural similarities defined above for all m pairs of matched features. This vector provides a measure of the structural similarity of two images. We can compute $max(n_1,n_2,...,n_m)$, $min=(n_1,n_2,...,n_m)$, $average(n_1,n_2,...,n_m)$ and the variety of moments of the distribution of $(n_1,n_2,...,n_m)$ and use them as similarity indicators if m is a large number. The difference $max(n_1,n_2,...,n_m)$ - $min(n_1,n_2,...,n_m)$ can be used as an indicator of variability in structural similarity. These indicators can also serve as change indicators that are important for image change detection tasks. High values of elements of vector n mean that there is no change in the image 1 during the period between taking image 1 and image 2.

Now we can consider the situation with subsets of features. Every feature in image 1 has a single match in image 2, but some features in image 2 have no matched feature in image 1. This situation is possible when there is a new development in the area. Again $n=(n_1,n_2,...,n_m)$ is a vector of structural similarities for all m pairs of matched features, but these pairs do not contain all the features in image 2. Similarly we can compute $max(n_1,n_2,...,n_m)$, $min=(n_1,n_2,...,n_m)$, $average(n_1,n_2,...,n_m)$ and other characteristics listed above. However, now even if we have high values of elements of vector n we cannot say that there is no change during the period between taking image 1 and image 2. Let m_1 and m_2 be the number of features in Image 1 and

Image 2, respectively. Then there is a vector n with m_1 elements. A new indicator, m_1/m_2 shows the ratio of common features in two images that can be added. Similarly, sets of measures for more complex situations can be introduced.

The structural similarity definitions can be augmented with definitions of geometric similarity, based on metric distances. We may need to transform one image to the other one before computing geometric similarity. The transformation can be based on features that have been matched.

Below we summarize a version of our shoulder-based method to measure a *structural difference* in curvilinear features in the following way:

Step 1: Compute L, the length of the line from its start to the middle point D along the line.

Step 2: Compute the lengths of two "shoulders", S1 and S2, that is S1 the length of the straight line [T, D] between the start point T and the middle point D.

Step 3. Compute ratios R1=S1/L1 and R2=S2/L2, where L1=L2 I is the half feature length computed along the feature. If a ratio R1 is close to 1 then the first part of the feature is close to the straight line, similar with R2. Find ratios that are in the threshold limit.

Step 4. Repeat steps 1-3 for subfeatures [T, D] and [D, E] recurrently for their subfeatures until all ratios are equal to 1 (straight line).

Theorem. For any polyline with n nodes, step 4 must be repeated no more than $n \log n$ times to get all ratios equal to 1.

Proof. This can be shown by considering an extreme case, where after the finding middle point D, only the single end points will be in the right "shoulder." This means that the n-1 nodes are in the part between the start node T and M including node p_{n-1}. We know that the polyline between node p_{n-1} and the end node $E = p_n$ is a straight line $[p_{n-1}, E]$. Above we assumed that M is between them, thus [D, E] is a straight line and a part of the longer straight line $[p_{n-1}, E]$. That is the ratio R_2 is equal to 1 for [D, E] and Step 4 took only one iteration for the subfeature between D and E.

Now we assume that the same extreme case is true for the left subfeature from point T to point D and subsequently for all its subfeaures. That is we need to repeat step 4 only n times for this extreme case. If our previous assumption is not true and we have more than one node in the subfeature between D and E, we can repeat such binary search process at most for $\log n$ times to come to the situation with a single straight line. Having total n nodes it may take $n \log n$ loops with Step 4 in the worst case.

6. CONCLUSION

This chapter describes a technique of image correlation that does not rely on geometrical or topological invariants or on identifying points in common with known coordinates. This technique determines relative scales and orientations and corresponding points by analyzing linear features identified in each image and fit with a polyline. While the example presented is from a research area at hand, there is nothing intrinsic to this method that ties it to the spatial imaging of the earth. It could as easily be applied to any set of overlapping images from any discipline and to images produced of a dynamic scene at different times.

7. ACKNOWLEDGEMENTS

This research has been supported by a University Research Initiative grant from the US National Geospatial-Intelligence Agency (NGA) that is gratefully acknowledged.

8. EXERCISES AND PROBLEMS

1. Draw a road network with four nodes (road intersections) and 6 roads. Define an algebraic system that would represent this road network.

   Tip: A graph representation is a special case of an algebraic system. Represent a road network as a graph described by the predicate P(x,y,r) that is true if two network nodes x and y are connected by road r. Construct predicate P for your road network.

2. Develop an algebraic representation of your six roads from exercise 1 by building matrixes similar to matrix shown in Table for angles between road segments.

3. Combine algebraic representations from exercises 1 and 2 and draw another road network that would satisfy the combined algebraic representation. Analyze the differences between two road networks. For instance, could they be different representations of the same road network but produced using different sensors in different locations?

9. REFERENCES

Brown, L. A Survey of Image Registration Techniques, ACM Computing Surveys,vol.24 (4), pp. 325--376, 1992

Carswell J., Wilson, D., Bertolotto, M., Digital Image Similarity for Geo-spatial Knowledge Management, in Advances in Case-based Reasoning, Springer-Verlag, Berlin, 2002

Cobb, M., Chung, M., Foley, H., Petry. F., Shaw, K., and Miller, H., A rule-based approach for the conflation of attributed vector data, GeoInformatica, 2/1, 1998, 7-36

Cohen S., Guibas, L., Partial Matching of Planar Polylines under Similarity Transformations. Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 777-786, January 1997

Kovalerchuk B., Sumner W., Algebraic relational approach to conflating images, In: Algorithms and technologies for multispectral, hyperspectral and ultraspectral imagery IX. Vol. 5093, International SPIE Military and Aerospace symposium, AEROSENSE, Orlando, FL., Apr.21-25, 2003, pp. 621-630.

Kovalerchuk B., Sumner W., Curtis. M., Kovalerchuk, M., Chase, R Matching image feature structures using shoulder analysis method, In: Algorithms and technologies for multispectral, hyperspectral and ultraspectral imagery X. SPIE Defense and Security symposium, Orlando, FL., Orlando, FL., April 11-16 2004

Mal'cev A.I. Algebraic Systems, Springer-Verlag, New York, 1973

Neapolitan R., Naimipour K. Foundation of algorithms using C++ pseudocode. Jones and Bartlett Publ., 1998

Zitová T., Flusser J., Image registration methods: a survey, Image and Vision Computing. 21 (11), 2003, pp. 977-1000

Chapter 20

ALGORITHM DEVELOPMENT TECHNOLOGY FOR CONFLATION AND AREA-BASED CONFLATION ALGORITHM

Michael Kovalerchuk and Boris Kovalerchuk
Central Washington University, USA

Abstract: This chapter presents a technology of conflation algorithm development with a wide applicability domain. The sequence of steps starts from vague but relevant expert (or just human) concepts and going toward an implemented conflation algorithm. The generic steps are illustrated with examples of specific steps from the development history of an area-based "shape size ratio" conflation algorithm. The fundamental "shape size ratio" measure underlying the algorithm has rather strong invariance property, including invariance to disproportional scaling. The implemented algorithm has been integrated into the ArcMap GIS Software as a Plug-in.

Key words: Algorithm development technology, imagery registration, conflation, geospatial feature, invariants, disproportional scaling, structural similarity, area ratio method, ArcMap Plug-in.

1. INTRODUCTION

The overall goal of this chapter is to present an **algorithm development technology for conflation (ADTC)**. The concept of conflation was identified in [Cobb et al., 1998; Jensen, Saalfeld et al., 2000; Doytsher et al.,2001; Edwards, Simpson, 2002] and in Chapters 17-19. This task is an expansion of the imagery registration task [Brown, 1992, Zitová, Flusser, 2003, Shah, Kumar, 2003; Terzopoulos et al, 2003; Wang at al., 2001;]. The technology is described in terms of its desirable characteristics and the actual scope of applicability of individual algorithms. In this approach, a category of images that an algorithm can work with should be identified and a formal

description of images in the category should be provided. The concept of monotonicity is used as a tool to help analyze and identify the scope of the algorithm coverage.

The **motivation** to develop an ADTC beyond the development of an *individual algorithm* follows from three observations:

1) Conflation is an ill-posed problem from a mathematical viewpoint. Thus, there is no chance to develop a single "magic" formal algorithm that will solve this ill-posed problem for all possible pairs of images to be conflated.
2) There is a chance to mimic a reasonable human practice in manual image conflation for specific categories of images as an automatic computer algorithm.
3) Human practice and expertise is fragmented, an individual imagery analysis may not work with some specific categories of images and may not provide any input for formal algorithm development.

These observations explain why a technology for algorithm development is an attractive alternative to the unrealistic "magic" algorithm. Having said this we do not want to fall into another trap, i.e., to end up with thousands of "mimic" algorithms with very narrow scopes. This leads us to the requirement that a "mimic" algorithm should be rather **invariant** to rotation, translation and disproportional scaling, and able to work with data of different types and modalities (multispectral, SAR and others), resolutions and noise level.

To build a technology we record and analyze a real sequence of algorithm development. This sequence starts from a very vague idea and ends up with a formal algorithm implemented. Below in Table 1 we provide a free recording of a real sequence of algorithm development.

Table1. Algorithm development recording

A human expert asks himself the question: "What does it mean for two images to be of the same thing?"
I remember my manual conflation process of two images with two lakes. There is a smaller one next to the big one in one image. Look for the big lake in the second image. Aha, there is a smaller lake next to the big one on the second image too. The two lake areas must be the same.
I need to develop a formal method for calculation and comparison of lake size ratios from this general observation. I recall that in my previous project a simple floodfill-like single color algorithm has been developed and used. It breaks up the image into shapes and calculates pixel counts. Maybe I can use it here too. In the floodfill-like algorithm, I set the first pixel as located in the upper left corner. The next unmarked pixel of the same color is joined to the flood area and the process continues until all adjacent pixels are tested. This process sets a flood area as the first shape. After recording all shapes I can compare their sizes in the two images by setting formal comparison relations between them as size ratios.

In essence this is a top to bottom approach where finally the basic pixel count is taken as a measure of lake size to be compared. Thus, Table 1 pro-

vides some information, but it is not systematic. Why should we focus on size ratio not on other parameters? Having only such random records we will be limited in the future to produce a general technology.

Therefore, our first task is finding a systematic way to record an analyst's experience and observations. The systematic way we propose consists of the several steps described below. The first one is that an expert writes a *list of features* he believes are relevant and *free statements* about them such as presented in Table 2.

Table 2. Feature list recording

Other features can be: (i) three or more "lakes" are on the each image, (ii) lakes of different sizes, (iii) unique features (iv) linear closed contours and so on. Relevance of the features can be measured, for instance, by abilities to build an affine transformation of one image to another based on these features.

Going through various ways of formalizing asymmetric features the expert records his/her experience (see Table 3).

Table 3. Feature formalization and algorithm implementation recording

Asymmetric features can be measured by the number of pixels to the top and bottom of center of the shape or the left and right of center. Coded a rather simple algorithm for calculating this "asymmetry percentage". While coding and debugging the function CalcShapeTopBottomAssymmetryPercent() which is based on partial pixel counts (shape sizes), came to the idea of using a ratio of shape sizes as a unique feature, remembering the manual experience of a lake conflation case. To match shapes, decided to collect data on all shapes in the image, and calculate all shape sizes (pixel counts) and a matrix of their ratios. For each row of ratios, trying to find the matching row of ratios in the second image, came to the idea and calculation of matching scores. If eight ratios in the row of ratios of shape 1 in image 1 are the same (within a small threshold for comparing doubles) as the eight ratios in the row for shape 3 of image 2, then the match score of shape 1 to shape 3 is 8. Then remembering that 3 points are enough for linear conflation, the 3 shapes with highest match scores were picked for the transform calculation. The center points of the shapes already calculated for the asymmetry algorithm were used as the 3 points for finding (calculating) the affine transform.

These observations give an impulse to more abstract thinking that includes getting 14 parameters that intuitively seem relevant (unique features, etc). These 14 parameters (after some refinement discussed below in section 3.2) are presented in Figure 2. Then the expert develops relatively easy to calculate pixel-based image measures such as asymmetry percentage, aspect ratio and others. These measures are intended to be indicators of parameter values. Next the expert assesses the application area (image types) for the suggested measure and concludes that it is sizable enough to pursue the method further.

2. STEPS OF THE ALGORITHM DEVELOPMENT TECHNOLOGY

The technology contains several steps described in this section in general terms and some steps are detailed in further sections with examples.

At the **first step** an imagery analyst generates a **set of binary parameters** $\{P\}$ potentially valuable for conflation. Each parameter has only true/false values, but at this stage no formal procedure set up to assign these values (it can be a vague expert opinion).

These parameters are very preliminary and vague. Parameters $\{P\}$ can be present in both images I_1 and I_2 or be comparative for two images. For instance, Parameter P_1 can state that a "unique feature" exists in image I_1. Thus, P_1 can be a binary indicator with two value 0 (false) or 1(true). Using the predicate notation this parameter can be written as $P_2(I_1)=1$ if a unique feature exists. The concept of "unique feature" P_2 is not formalized at this stage.

Parameter P_2 can indicate that scales $g(I_1)$ and $g(I_2)$ of two images I_1 and I_2 differ by no more than 2 times and can be written in a predicate notation as

$$P_2(I_1,I_2) = 1 \Leftrightarrow g(I_1)/g(I_2) < 2.$$

The concept of scale is formalized at this stage.

Another *informal intention* in generating parameters is to get parameters $\{P\}$ that would be *invariant* to rotation, translation and disproportional scaling to be able to generate a robust conflation method. It is not a parameter selection criterion at this stage, because often a parameter can be made invariant after some corrections.

The **second step** is to evaluate preliminary **sufficiency** of the set of parameters and if it is sufficient, attempt to find subsets of parameters that can be sufficient. This informal step intends (1) to narrow the set of parameters for further work if there are too many parameters or (2) to expand the set of parameters if it seems insufficient to avoid inconsistent and unreliable conflation solutions. A set $\{P_{is}\}$ of the potentially sufficient parameters selected from all parameters $\{P_i\}$ identified in step 1 is a result of this step.

The **third step** is to select some **promising subset** of parameters from the set of the potentially sufficient parameters $\{P_{is}\}$ defined on the step 2.

This step is still informal. It is motivated by the intent to review parameters $\{P_{is}\}$ from a variety of viewpoints. One of them is the already mentioned invariance of parameters to translation, rotation and scaling. Others could be the ability to *formalize*, to *measure* and to *compute* each parameter P_i. The result of this step is to select a set $\{P_{isp}\}$ of promising parameters from all potentially sufficient parameters $\{P_{is}\}$ identified in step 2.

The **fourth step** is to attempt to formalize selected parameters from scratch or by using an existing library of formalized components. For instance, "unique feature" parameter was formalized based as an area ratio using an already developed pixel count measure.

This is a critical step of transforming the whole conflation process from a pure expert realm to more automated area with the long-term intent to produce a completely automatic conflation algorithm. The creative nature of this step and its unpredictability are well-known. Steps 1-3 are intended to increase chances for success in this step.

The result of this step is a set $\{P_{ispf}\}$ of the formalized parameters selected from all promising sufficient parameters $\{P_{isp}\}$ identified in step 3.

The **fifth step** is to analyze formalized parameters $\{P_{ispf}\}$ for invariance. This step permits mathematical analysis of the parameter and computational experiments with computing each parameter P_i for differently modified images $\mu_1(I), \mu_2(I), \ldots, \mu_n(I)$:

$$P_i(\mu_1(I)),\ P_i(\mu_2 I)), \ldots,\ P_i(\mu_n(I))$$

and checking that these values are the same.

Obviously mathematical analysis can be more difficult conceptually, because there is no procedure like the one described above for computational experiments. However, mathematical analysis can provide stronger invariance conclusions. Computational experiments can only confirm invariance for the transformations $\{\mu\}$ tested, but the mathematical proof can cover all possible transformations. As we show below this was the case with the mathematically proven statement that the parameter "area ratio" is an invariant.

The **sixth step** is to develop a conflation algorithm based on formalized invariant parameters. Even though the parameters are formalized, this step is still not formal. For some parameters it can be straightforward, but for some parameters it can be a very creative process. The "area ratio" invariant parameter is an example of the relatively straightforward algorithm derivation.

The **seventh step** is to develop an algorithm for finding invariant parameters in images.

The **eighth step** is to determine *algorithm domain limits* – determine the types of images on which the invariant algorithm from step 6 conflates well. If the domain is sizable or significant, we have a valuable conflation algorithm for the domain. If the domain is small and insignificant, then go back to previous steps and retrace the path hoping to get a more universal method.

The flowchart in Figure 1 shows the general sequence of steps for the algorithm development methodology. These steps can be looped to get an improved result.

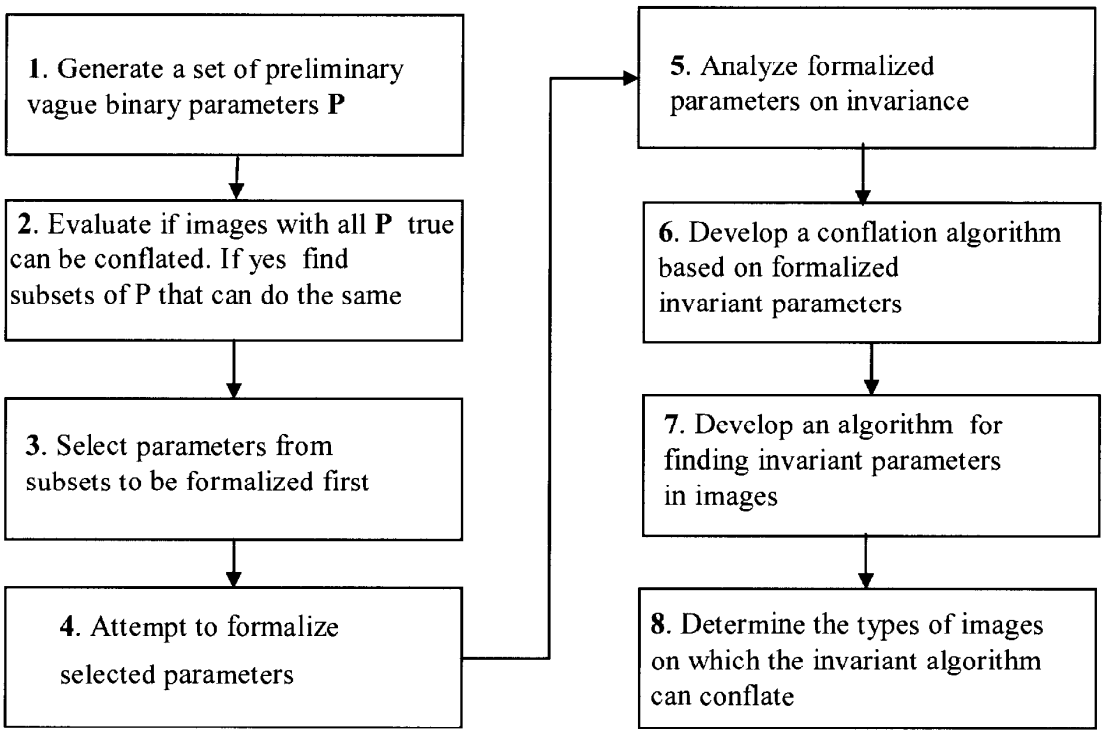


Figure 1. Overall steps of the ADTC technology

3. PARAMETER IDENTIFICATION STEPS

3.1 Generation of parameter set

The identification of parameters is the goal of the first three steps of ADTC. The Imagery Virtual Expert System (IVES) described briefly in chapter 21 includes several tools including a parameter editor that permits recording, editing, and storing parameters. Figure 2 shows the list of 14 parameters suggested for further analysis and recorded in IVES.

These parameters are influenced by the free recording of an expert's algorithm considerations for an algorithm development. In Figure 2 all parameters are binary with only true/false values.

The parameter "Simple unique geometric feature exists" resembles parameter P_1 discussed in section 2 on existence of a "unique feature" in the image. Parameter "Image scales differ no more than two times" is the same as parameter P_2 also described in section 2.

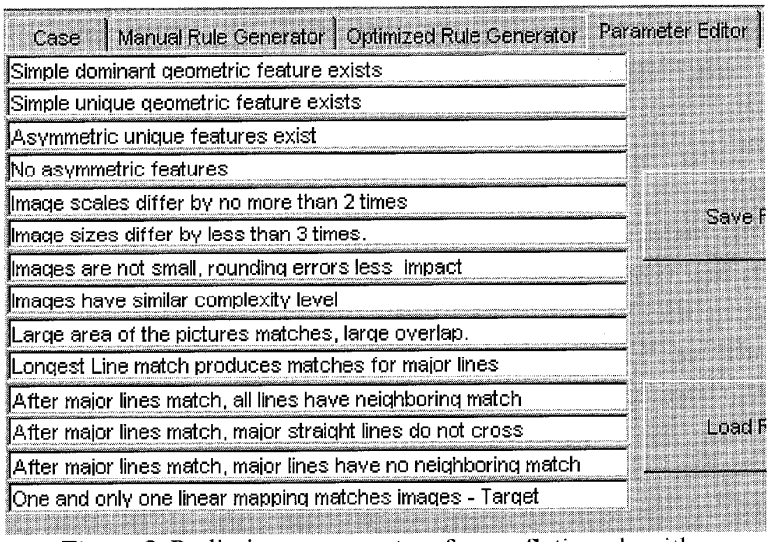


Figure 2. Preliminary parameters for conflation algorithm development recorded in IVES system

3.2 Parameter set minimization

The goal of this step is the further polishing of the set of parameters. At first we want to preliminary evaluate if the set of 14 parameters could be sufficient to solve the conflation problem with a linear (affine) transform. At this informal stage, the answer for this question is simply an expert opinion (yes/no). If the answer is "yes" then we are interested in narrowing this set of 14 parameters to a smaller subset that can be sufficient too. This is also done by asking an expert about subsets $\{P_{is}\}$ of the potentially sufficient parameters selected from all parameters $\{P_i\}$. The simple exhaustive option here is to ask an expert to answer yes/no about all 2^{14}=4096 subsets. Obviously such approach is not realistic and moreover redundant. The monotone Boolean function approach [Kovalerchuk et al., 1996, 2001] is more appropriate here.

The approach has two significant components. The **first stage** is to formulate each parameter in such a way that if the expert answered 'yes" about this parameter then it *increases the chances* that the conflation can be done using this parameter. For instance, the parameter "Simple symmetric feature exists" may or may not indicate increased chances that conflation can be successful. With many similar symmetric features it can be very difficult to make a unique match. This parameter can be negated and reformulated as "Asymmetric unique feature exists". If the expert answers "yes" for this reformulated question then the chances to find a unique conflation matching points and features increase, because we may find unique matching points

and other feature parts that can be uniquely matched with similar elements in another image. In Figure 2 parameters are already reformulated in such a way from their original recording. We will call such reformulation a *positive monotone reformulation.* Thus, the first test for 14 parameters is to ask the expert if he/she agrees that if all 14 parameters are evaluated with "yes" then there are very good chances that two images can be conflated by an affine or more complex transform.

If the answer is "yes" then an attempt to minimize the set of parameters is made. This is the **second stage** of the approach. Figure 3 shows how IVES system supports this stage. The expert is asked to answer if checked parameters received "yes" answers could be it sufficient for successful conflation. If the answer is positive then this subset of parameters will be a new starting point for further decreasing the set of parameters.

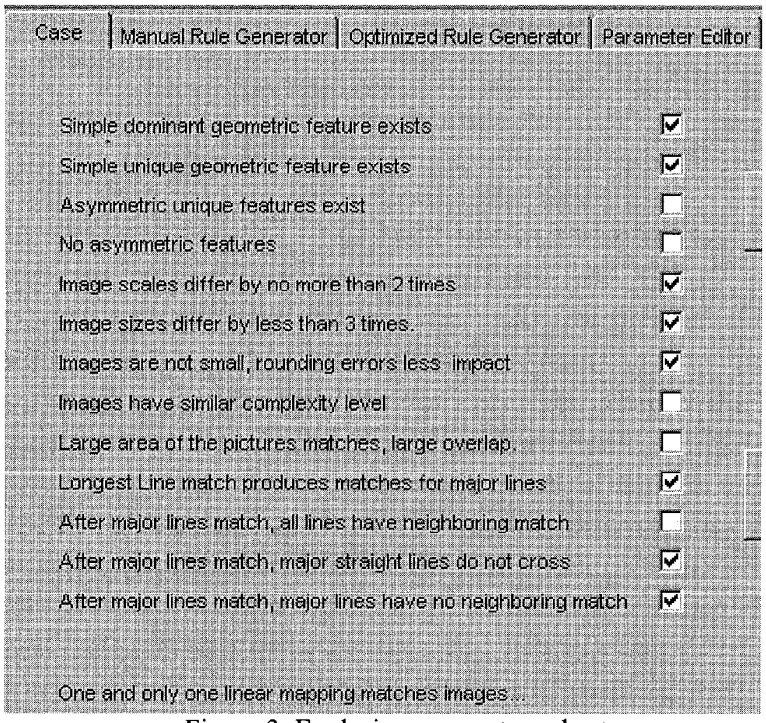


Figure 3. Exploring parameter subsets.

There is no reason to ask the expert about subsets that include all parameters of this subset and some more parameters. That large subset should be sufficient too by the parameter design performed at the first stage described above. This is the property of **monotonicity** of Boolean functions that we exploit. Thus we are only interested in cutting down the number of parameters.

A screenshot in Figure 4 illustrates how IVES system supports this stage. The expert determines if the following checked parameters could be sufficient to have as single linear mapping from one image to another. The expert

answers using yes/no buttons for the current subset of parameters shown in the first column. Previous answers are shown in other columns. The light color (green) indicates positive answers (yes) and the dark color (red) indicates negative answers (no). The sequence of questions is stored in the IVES system in advance. This sequence can be altered by loading another sequence file. Expert's answers are recorded in another file. The question sequence can be optimized by selecting an appropriate sequence file. If nothing is known about potential expert's answers in advance the sequence that will ask the minimal number of questions for the most difficult situation can be used. This best sequence for the worst-case scenario is formalized by the Shannon criterion and is based on Hansel chains [Kovalerchuk et al., 1996]. See also Chapter 16 for more detail.

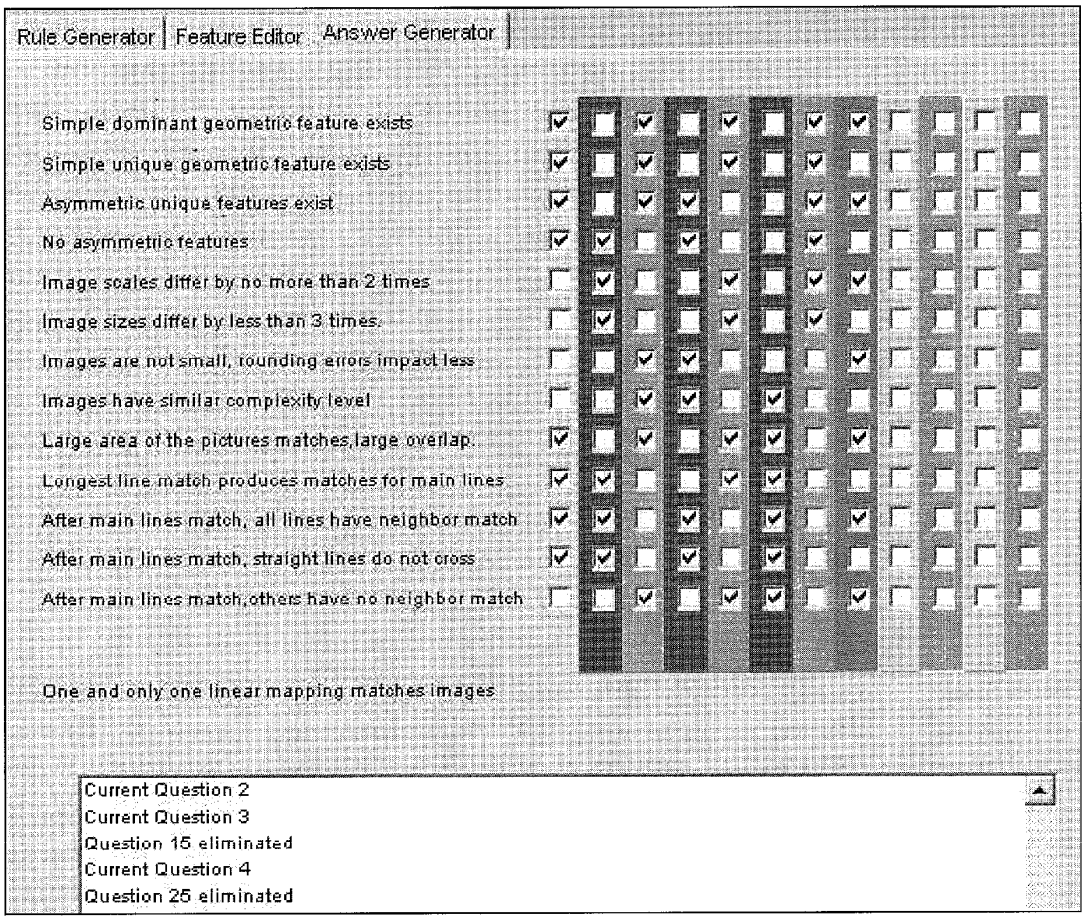


Figure 4. Expert questioning using monotonicity principle. See also color plates.

The most desirable output from this step would be the conclusion that a single parameter out of 14 listed could be sufficient for some sets of images.

3.3 Single parameter selection.

The analysis of expert's answers had shown that the parameter "Asymmetric unique features exist" popped up as a single parameter that could be sufficient for building the algorithm. The motivation for this selection is as follows. If asymmetric features exist and are preferably unique then ambiguity in conflation is less likely. Symmetric features are more likely to cause ambiguity. In the next section we analyze how this parameter can be formalized. We interpret the concept of asymmetric unique feature very generally. Such a feature could be a cluster of three square buildings of different sizes and asymmetrically located to each other. The buildings themselves are not unique, but their mutual location can be unique. Thus we do not require that the feature be a single continuous entity.

4. ATTEMPT TO FORMALIZE PARAMETERS

In this section, we discuss step 4 by analyzing parameter P_3, "Asymmetric unique features exist" selected in the previous section. The first formalization assumes that unique features are represented by *shapes* and uniqueness of shapes is measured by an *aspect ratio.* This aspect ratio is computed as h/w, where h is the height and w is the width of the rectangular bounding box (BB) around the shape. For now we can assume that the sides of BB are parallel to X,Y coordinates. We use notation $P_3(I)$ = true or simply $P_3(I)$ for image I if I contains a shape F with aspect ratio h/w such that there is no other shape F_i with aspect ratio h_i/w_i within 10% of the aspect ratio h/w in the same image A:

$$P_3(I) \Leftrightarrow \forall F_i \quad |h_i/w_i - h/w| > 0.1 h/w$$

Such shape F is considered to be *unique* in image A, but may not be unique in image B that is the same as image A, but transformed somehow. This is the subject of investigation on further steps of the process.

The second formalization for P_3 is based on the concept of area occupied by the shape in the bounding box. In this formalization, $P_3(I)$ =true if less than 60% of the shape' bounding box is occupied by the shape F and its internal shapes is *asymmetric*:

$$P_3(I) \Leftrightarrow > S < 0.6\ hw\ \&\ \text{Asymmetric}(F),$$

where S is the area occupied by the shape. The concept of asymmetric feature should be formalized later as a predicate Asymmetric(F). The simplest

formalization would be just to assume that the already written formalization is acceptable that the shape (with internals) that occupies less than 60% of its smallest bounding box is asymmetric.

It is important to notice that we do not view such formalization as fully formally equivalent to the intuitive concept of asymmetric unique features. Rather, they are partial measures that capture only some aspects of the informal parameter definition.

This parameter can be further specified by requiring that n pixels above the center is 30% greater than the number of pixels below the center and no other shape in same image has a similar asymmetry percentage within a 5% threshold. Example: there is a shape with 40% and there is no other shape between 35% and 45% in the picture. The same characteristic for testing left and right asymmetry can be computed. Any direction can be selected for testing asymmetry.

For unambiguous conflation we may require that all shapes (with internals) occupy less than 60 % of their smallest bounding box. A unique aspect ratio can be a better predictor of asymmetry than areas. Area occupied in the bounding box can be an indicator of convexity and concavity of the shape. An area that is less than a rhombus should be concave somewhere. For conflating images unique features in both images should have similar concavity ratios. If a ratio is close to 0 then the shape is very concave and if it is close to 1 then the shape is very convex. Note some regions can have a ratio far away from 1, but be very convex, e.g., symmetric n-gons. Highly different concavity ratios are indicators against a match.

The **algorithmic procedure** for all these formalizations is the same -- for every shape found in the image search the array of shapes with ratios within 5% of current shape. If no such shapes are found, the current shape is *asymmetric and unique*, P_3=1.

In general the attempt to formalize a selected parameter (termed as step 4 of ADTC) can be viewed as consisting of two substeps:

Step 4.1. *Find a local or global image characteristic* of the image (called an *impact measure*) that may have an true/false impact parameter. As mentioned above "Ratio of pixels to the top and bottom of center of the shape (termed "*Central Asymmetry Percentage*") is an example of an impact measure. The search for impact measures can be performed from scratch or starting from existing impact measures or their components.

The ADTC technology for algorithm development assumes that a system built according to this technology records impact measures in the course of developing conflation algorithms. Then recorded measures are used for developing new impact measures. When the number and diversity of recorded measures will be large enough the system could be able to "learn" new im-

pact measures automatically. The learning is done by modifying and generalizing recorder measures using case-based reasoning approach. Such learning can make development of measures needed for new algorithms easier. This is an essence of the step 4.2.

Step 4.2. *Spread the "Impact Measure"* – modify and generalize the impact measure from step 4.1 to get several similar easy to implement impact measures. For instance, having an impact measure m = "Shape size as pixel-count" new impact measures "Ratio of sizes of two shapes," and "Asymmetric shape" can be developed. For instance these measures can test that $m < 0.6B$, where B is a size of the rectangular bounding box for the shape. Similarly, the measure "Elongated Asymmetric feature" can be developed that is defined as a shape aspect ratio greater than 3.

5. ANALYZE PARAMETER INVARIANCE

Step 5 of ADTC is called *Check Invariance* for short. All suggested impact measures are analyzed for invariance to affine transformations. For instance, the pixel count based ratio of sizes of two shapes mentioned above in step 4 is invariant. See a formal analysis of this impact measure invariance with the invariance theorem below.

Invariance to disproportional scaling (DPS) is one of the most difficult requirements to meet. In Chapter 19 relations between angles and linear segment lengths have been exploited to build conflation algorithms. These relations are relatively robust, that is they do not change for limited DPS. Figure 5 shows a robust DPS situation with the angle relation ">" preserved.

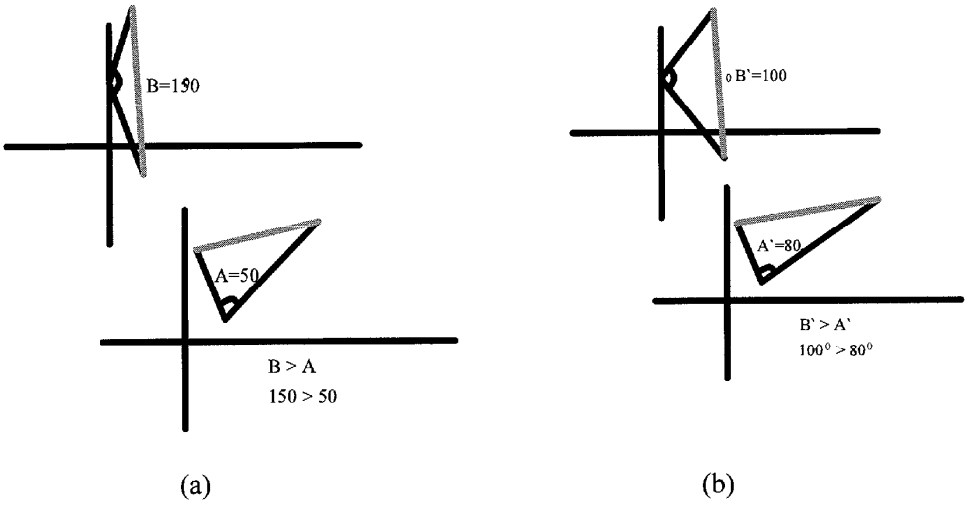


Figure 5. Case of robust invariant angular relation

In Figure 5 (a) angle B is greater that angle A, B>A. This property is preserved under disproportional scaling shown in Figure 5(b), where still B`>A`.

Now we can explore robustness of relations "=" and ">" between angles and areas relative to other disproportional scaling. In Figure 6, area S_1 is equal to area S_2, $S_1=S_2$. In addition, angles A, B, and C, D are equal, A=B, C=D too.

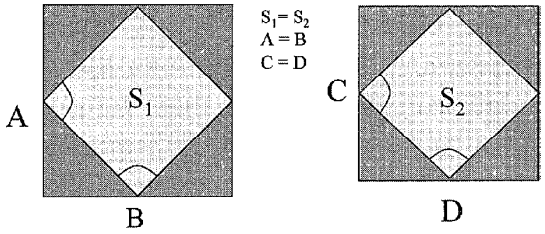


Figure 6. Original image

Figure 7 presents the same image after disproportional scaling, where X coordinate was multiplied by $k_x>1$ and Y coordinate is not changed, $k_y=1$. Relations between angles A and B have been changed, A`< B`. Also relations between angles C and D are changed, C`<D`. In contrast, the relation between areas $S`_1$ and $S`_2$ is not changed, $S`_1=S`_2$.

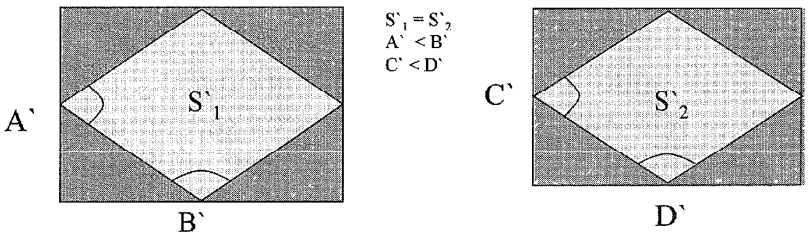


Figure 7. Image after disproportional scaling

This can be proved by noticing that bounding boxes U_1 and U_2 around rhombuses S_1 and S_2 are not changed and each rhombus occupies a half of its bounding box.

More formally we have $U`_1= k_xk_yU_1$, $U`_2= k_xk_yU_2$, and using property $U_1=U_2$, we conclude that $k_xk_yU_1= k_xk_yU_2$ and therefore $U`_1= U`_2$. Next, $S`_1= U`_1/2$ and $S`_2= U`_2/2$ hence $S`_1=S`_2$.

This proof is also valid for the general case when $k_x\neq1$. We can notice that rotation and translation do not change relations between angles as well as between areas. Thus, area relations are not changed under translation, rotation and disproportional scaling.

Above we considered only simple relations "=" and ">" between areas. Other *area functions* can also be invariants. For instance, ratio of areas S_1/S_2 is also invariant under disproportional scaling, because $k_xk_y\,S_1/k_xk_yS_2 = S_1/S_2$.

The consideration above can be converted into a formal theorem statement. Let F be an affine transformation that combines disproportional scaling transformation K with scaling coefficients (k_x, k_y), $k_x \neq 0$, $k_y \neq 0$, translation T and rotation R, $F=K \cdot T \cdot R$, where K, T and R are transformation matrixes.

Let also G_1 and G_2 be two closed regions and $S_1=S(G_1)$ and $S_2=S(G_2)$ be their areas respectively, where S( ) is an operator that computes area of the region G_i.

Theorem: An affine transformation F of the image does not change the relation between area ratios, $S_1/S_2 = S(G_1)/ S(G_2)= S(F(G_1))/S(F(G_2))$, that is F is an **isomorphism** for area ratio $S(G_1)/ S(G_2)$.

The proof follows from the considerations that preceded the theorem. The use of generic algebraic system formalism for describing images is convenient because it permits the analysis of the isomorphism and homomorphism of images based on different image characteristics *uniformly*.

6. CONFLATION ALGORITHM DEVELOPMENT

6.1 Develop conflation algorithm using formalized invariant parameters

In this section step 6 of ADTC is illustrated with the development of an **Area Ratio Conflation Algorithm** (**ARC** algorithm for short). The invariance of the area relations found in the previous section is the base for this algorithm. At first we describe the general concept of the algorithm and then present it in more detail and more formally using the generalized algebraic framework discussed in Chapter 19 for features presented as polylines. Thus in this chapter the algebraic framework is extended for shape-based features.

Two raster images are conflated by finding at least 3 matching uniquely sized regions (areas) in both images and using the center points of those regions as reference points for an affine transform calculation.

There are several possible formalizations of the concept of "matching uniquely sized regions (areas)". In the ARC algorithm, we use the area ratio S_i/S_j as the matching characteristic, because of its invariance shown in the previous section. If two areas in the original image have a ratio, say 0.3, then the same ratio between them should remain the same under any affine transform. Thus in the second image, we can compute areas of regions, their ratios S_i/S_j and search for a 0.3 ratio among them. If only one such ratio was found then centers of these regions give us two tie (control) points for building an affine transform. Finding a third region S_m in the both images with the equal ratios S_i/S_m in both images provides the third tie point needed for an

affine transform. This basic idea is adjusted for the cases where more than one matching triple found. An additional uniqueness criterion is introduced in the algorithm based on the analysis of additional ratios.

Suppose there is an image that contains a large lake of some size and a small lake whose size is ⅓ of the size of the large lake. This size ratio (⅓) is invariant to affine transformations. The ratio precision needs to be adjusted to the scale of least precise image. Ratios ⅓, ½ and ¼ could match 0.336 0.52 0.27 if images are of different scales. The algorithm uses a matching threshold for these cases.

This logic of the algorithm requires: (1) an algorithm for computing area ratios and for matching ratios and (2) an algorithm for region extraction from the image. The first algorithm called the Ratio Algorithm and the second algorithm called Vectorizer are described below. The development of the second algorithm is the goal of the Step 7 of the ADTC technology.

The ratio algorithm starts from a set of regions $\{G_{1i}\}$ for image 1 and a set of regions $\{G_{2i}\}$ for image 2 extracted by the Vectorizer algorithm. The *Ratio algorithm* computes areas for each region in both images, $S_{1i}=S(G_{1i})$, $S_{2i}=S(G_{2i})$ as a number of pixels inside of the region. Next this algorithm computes two matrixes V_1 and V_2. Elements of matrix $V_1=\{c_{ij}\}$ are $c_{ij}=S_{1i}/S_{1j}$ Elements of matrix $V_2=\{q_{ij}\}$ are defined similarly, $q_{ij}=S_{2i}/S_{2j}$. We assume that all areas S_{1i} and S_{2i} are positive.

The matrix representation is important because it permits us to convert the situation to a generic **algebraic system framework**, with algebraic systems $\mathbf{A_k}=<A_k, R_k, \Omega_k>$, where signature Ω_k contains the operator $V_k(a_i,a_j)$ represented as a matrix V_k and handles the conflation problem uniformly. From this point uniformity permits us to use a single and already implemented algorithm to search for matching features in the images. It does not matter for the algorithms in algebraic form whether elements of A_k are straight-line segments, polylines, areas, or complex combinations, or some other features. Elements of A_k also can be numeric characteristics of image components such as a size of region i in image k, S_{ki}.

Example: Let matrix V_1 be computed for regions with areas $S_{11}=6$, $S_{12}=4$, $S_{13}=2$, $S_{14}=1$ (see Table 4) in image 1 and matrix V_2 is computed for areas $S_{21}=4$, $S_{22}=1$, $S_{23}=6$, $S_{24}=7$ in image 2 (see Table 5).

Table 4. Matrix of shape size ratios in Image 1

	$S_{11}=6$	$S_{12}=4$	$S_{13}=2$	$S_{14}=1$
$S_{11}=6$	1	**4/6**	2/6	**1/6**
$S_{12}=4$	**6/4**	1	2/4	**1/4**
$S_{13}=2$	6/2	4/2	1	1/2
$S_{14}=1$	**6/1**	**4/1**	2/1	1

Table 5. Matrix of shape size ratios in Image 2

	S_{21}=4	S_{22}=1	S_{23}=6	S_{24}=7
S_{21}=4	1	**1/4**	**6/4**	7/4
S_{22}=1	**4/1**	1	**6/1**	7/1
S_{23}=6	**4/6**	**1/6**	1	7/6
S_{24}=7	4/7	1/7	6/7	1

The brute force method would search equal ratios in two matrixes excluding the diagonal. There are six equal numbers in these matrixes, which may indicate the match uncertainty. In fact, there are only three numbers that should be considered (only numbers above the diagonal). The numbers below the diagonal are $1/c_{ij}$ of the numbers above the diagonal c_{ij}. This is an unambiguous case, where ratio 6/4 for S_{11}=6, S_{12}=4 is matched to S_{23}=6, S_{21}=4, that is region G_{11} in image 1 is matched to region G_{23} in image 2 and region G_{12} in image 1 is matched to the region G_{21} in image 2. The center of each region is computed as an average of coordinates of all points (pixels) of the region.

Computational efficiency of the algorithm depends on how quickly matrixes V_1 and V_2 will be computed for images with the large number of regions. We can notice that matrix V_1 has only n-1 independent ratios. All other ratios from n^2 ratios are computed from them excluding the diagonal that contains all 1's by definition. The theorem about this is proved below.

It is reasonable to start from these n-1 independent ratios. If all these ratios in A differ from n-1 independent ratios in B then the next n-2 ratios are computed in both matrixes. If they also have no equal values then the process continues for the next n-3 ratios until all elements A from the upper part of A are exhausted.

If some ratios in this process are equal then there is no reason to compute ratios that are derived from them. They will be equal too. Specifically, if $c_{ij} = q_{st}$ and $c_{jk} = q_{tq}$ then we do not need to compute c_{ik} and q_{sq}. They do not add new information and are equal (see proof below). From $c_{ij} = q_{st}$ we can derive that region G_{1i} is matched to the region G_{2s} and region G_{1j} is matched to the region G_{2t}.

Similarly from $c_{jk}= q_{tq}$ we can derive that region G_{1k} is matched to the region G_{2q}. Equality of c_{ik} and q_{sq} does not add new information because it matches region G_{1i} with region G_{2s} and region G_{1k} is matched to the region G_{2q}, but this match was already established.

The previous consideration was based on sequential fill of lines that are parallel to the matrix diagonal. Tables 6 and 7 illustrate how the third and forth lines above the diagonal (termed the 3rd and 4th *diagonals*) are computed for V_2. The light color shows inputs and the dark color shows output. These computations use the multiplication formula (1).

Table 6. Matrix of shape size ratios in Image 1

	S_{21}=4	S_{22}=1	S_{23}=6	S_{24}=7
S_{21}=4	1	1/4	6/4	7/4
S_{22}=1	4/1	1	6/1	7/1
S_{23}=6	4/6	6/1	1	7/6
S_{24}=7	4/7	1/7	6/7	1

Table 7. Matrix of shape size ratios in Image 1

	S_{21}=4	S_{22}=1	S_{23}=6	S_{24}=7
S_{21}=4	1	1/4	6/4	7/4
S_{22}=1	4/1	1	6/1	7/1
S_{23}=6	4/6	6/1	1	7/6
S_{24}=7	4/7	1/7	6/7	1

To make match more visible we can reorder lines in Table KB according to S_{2i} values starting from the largest value 7 (see Table 8). Now we may notice that two submatrixes with bold frames are the same in Tables 8 and 7.

Table 8 Matrix of shape size ratios in Image 1

	S_{24}=7	S_{23}=6	S_{21}=4	S_{22}=1
S_{24}=7	1	6/7	4/7	1/7
S_{23}=6	7/6	1	4/6	1/6
S_{21}=4	7/4	6/4	1	1/4
S_{22}=1	7/1	6/1	4/1	1/1

Below we discuss the development of a part of the algorithm that finds V_{max}, *a largest common subset in matrixes* V_1 and V_2. If this subset is at least 3×3, i.e., it includes three features and centers of those features are not located on the same straight line then an affine transform can be found between images 1 and 2. If the common subset is much larger than 3×3, then it creates a higher level of confidence that conflation of two images is not accidental.

In this chapter, the search for the common part is more general than the search of submatrixes along the main matrix diagonal in Chapter 19. A **submatrix** $\{c_{ij}\}_{i,j=k,k+1,\ldots,m}$ used in Chapter 19 is formed by a set consecutive indexes from k to m. The **matrix subset** is formed by the set of indexes that may have gaps, e.g., T={1,2, 5,7,9}, that is $\{c_{ij}\}_{i,j\in T}$. Note that by reordering matrix rows and columns, we transform a matrix subset to a submatrix, but this may not speed up the search of the common subset, because the second matrix can be ordered differently.

An algorithm generates all subsets of V_1 of size 3×3 and all subsets of V_2 of size 3×3. We denote them as V_1 (T_r) and V_2 (T_e), where T_r and T_e are index sets that identify three features that form 3×3 subset. Then similarity of every pair V_1 (T_r) and V_2 (T_e) is tested using the Euclidian distance (D):

$$D(V_1(T_r),V_2(T_e)) = \sum_{i,j \in T_r} (c_{ij} - q_{\varphi(i),\varphi(j)})^2$$

and finding a pair with smallest distance that is below the threshold L, where the mapping φ matches features in two images:

$$D(V_1(T_r), V_2(T_e)) \rightarrow \min$$

$$D(V_1(T_r), V_2(T_e)) \leq L.$$

6.2 Proofs

Lemma. If $c_{ij} = q_{st}$ and $c_{jk} = q_{tq}$ then $c_{ik} = q_{sq}$,
Proof. The proof is based on

$$c_{ij} \cdot c_{jk} = c_{ik}. \quad (1)$$

This follows from the definitions of c_{ij}, c_{jk} and c_{ik}, $c_{ij}=S_i/S_j$, $c_{jk}=S_j/S_k$ and $c_{ik}=S_i/S_k$, where

$$c_{ij} \cdot c_{jk} = (S_i/S_j)(S_j/S_k) = S_i/S_k = c_{ik}.$$

Now using $c_{ij} = q_{st}$ and $c_{jk} = q_{tq}$ in (1) we will get

$$c_{ik} = c_{ij} \cdot c_{jk} = q_{st} \cdot q_{tq} = q_{sq} \quad \#.$$

The use of formula (1) is shown graphically in Table 9 where elements c_{23} =2/4 and c_{34} =1/2 in light cells produce element c_{24}=(2/4)(1/2)=1/4 in the dark cell. Similarly cells are marked in Tables KL above for computing element c_{13}.

Together elements c_{13} and c_{13} form a line that is parallel to the diagonal and contains n-2 elements, where n is the size of the matrix. In this example n=4. The last element of the matrix A is c_{14}. The computation of this element from elements c_{13} and c_{34} is illustrated in Table 10. All these tables indicate an important property that is formulated as a theorem below.

Table 9. Matrix of shape size ratios in Image 1

	S_1	S_2	S_3	S_4
S_1	1	4/6	2/6	1/6
S_2	4/6	1	2/4	1/4
S_3	6/2	4/2	1	1/2
S_4	6/1	4/1	2/1	1

Table 10. Matrix of shape size ratios in Image 2

	S_{11}=6	S_{12}=4	S_{13}=2	S_{14}=1
S_{11}=6	1	4/6	2/6	1/6
S_{12}=4	4/6	1	2/4	1/4
S_{13}=2	6/2	4/2	1	1/2
S_{14}=1	6/1	4/1	2/1	1

Theorem: if all elements $c_{j\,j+1}$ (j=1,2,…,n) of $n\times n$ matrix A are given then all other elements of A can be restored.

Proof. We can omit computing elements of V_1 under diagonal, all these elements are $c_{ij} = 1/c_{ji}$ for elements above the diagonal. This directly follows from the definition of elements of V_1. The further proof is provided by designing an iterative process that computes all other elements of V_1.

Step 1. Compute all elements $a_{j\,j+2}$ by using elements $a_{j\,j+1}$ in formula (1):

$$c_{j\,j+1} \cdot c_{j+1,\,j+2} = c_{i\,j+2}.$$

Step 2. Compute all elements $c_{j\,j+3}$ using formula (1) and elements $c_{i\,j+2}$ computed in Step 1:

$$c_{j\,j+1} \cdot c_{j+1,\,j+2} = c_{i\,j+2}.$$

General Step k: Compute all elements $c_{j\,j+k}$ using formula (1) and elements $c_{i\,j+k-1}$ computed in Step k-1:

$$c_{j\,j+k-1} \cdot c_{j+k-1,\,j+k} = c_{i\,j+k}.$$

Repeat Step k until j+k-1< n that is the whole matrix V_1 is exhausted.

6.3 Develop algorithm for finding invariant parameters in images

Finding invariant parameters constitutes step 7 of ADTC. ARC algorithm uses area ratios as invariant parameters for conflation. To be able to run this algorithm we need to find/extract regions, compute their areas and centers

for matched ones. This work is done by a Vectorizer algorithm that sharpens images and finds regions using a flood-fill method from computer graphics [Angel, 2000] that starts from a seed point and looks recursively at colors of adjacent pixels including diagonal neighbors until all neighboring pixels of the same color are found. The set of these pixels is considered a single region. The number of the pixels in the region is considered its area/size. A set of all extracted regions is ARC algorithm input.

After conflation is done by the ARC algorithm the **conflation quality** can be evaluated by visual inspection of the conflated images and by a computational procedure based on the absolute and relative difference between matched regions.

The difference of two regions is XOR (exclusive OR) of pixels of regions G and G`. The **absolute difference** of regions G and G`, $\Delta(G,G`)$ is computed as the number of pixels in the difference of regions G and G`:

$$\Delta(G,G`)=S(XOR(F(G),G`))$$

and the **relative difference** of the regions is

$$\rho(G,G`)=\Delta(G,G`)/(S(G)+ S(G`)).$$

The total difference between three matched regions {G} and {G`} of images Im1 and Im2 found by ARC algorithm is

$$\mu(\{G\},\{G`\})= \mu(G_1,G`_1) + \mu(G_2,G`_2)+ \mu(G_3,G`_3).$$

Similarly the relative difference of matched regions is

$$\rho(\{G\},\{G`\})=[\Sigma_{i=1,2,3}\Delta(G_i,G`_i)] /[\Sigma_{i=1,2,3}(S(G_i)+ S(G`_i))].$$

The maximum of relative difference is 1 and the minimum is 0.

7. DETERMINE CONFLATABLE IMAGES AND ALGORITHM LIMITATIONS

Determining conflatable images and algorithm limitations constitutes step 8 of ADTC. Pixel based size ratio methods work well on images which are feature-rich enough to have at least 3 uniquely sized regions in both images. Some known limitations are:

1) Regions need to be completely contained in the images, not partially cut off by the image edge.

2) Images with a large number of colors may require pre-sharpening.

If only a half of the shape (e.g., lake) is present in one of the images and the image border goes through the lake, the region size ratios would not match and another region would have to be used for conflation. If another region is not found in the image, then the image is too *feature-poor* for successful conflation with this region size ratio method. A line-based method may be able to conflate the cut off lake based on its coast line shape. Thus, line-based conflation methods described in Chapter 19 need to be run on region-poor cases. Below we summarize known limitations that are solvable by other methods:

- region poor because of cut off (line-based methods may work);
- lack of unique regions – many equal sized of regions (line-based methods may work);
- smooth color transition (pre aggregate pixels into larger regions and then run a region-based method). Note that in this case differences in aggregation color thresholds may preclude a match.

There are also limitations that are not solvable by other methods:

- no regions or other features at all, e.g., two featureless pieces of desert;
- all regions are the same, this case is theoretically ambiguous given current data;
- heavy darkening or lightening can melt together two regions.

It seems that the ARC algorithm domain of images with three or more regions of different sizes completely contained in the images and with a modest number of colors is sizable and significant. Also the Area Ratio Conflation Algorithm (ARC) described in this Chapter a good approximation for the whole conflation problem in a sizeable area of images. It is important that applicability of the algorithm to particular images can be tested in advance.

8. SOFTWARE AND COMPUTATIONAL EXPERIMENT

The following series of screenshots demonstrate the initial implementation of the Area Ratio Conflation Algorithm (ARC) for raster images, implemented as an ArcMap Plug-in. ArcMap is the central application in ArcGIS Desktop software developed by ESRI. It supports all map-based tasks including cartography, map analysis, and editing [ArcGIS, 2004].

User actions needed to use the Plug-in include loading the two images into ArcMap and clicking a "Conflate" button. The Plug-in allows the algo-

rithm to run in visualized or non-visualized mode based on the users selection. The screenshots also demonstrate that the algorithm is capable of handling disproportionate scaling.

In the following screenshots (Figures 8-14) we consider the conflation of various sections of an aerial photo and a topographic map of the same area with three lakes.

The screenshot in Figure 8 shows two images before conflation is applied. The image on the left is a preprocessed aerial photo rotated 90 degrees and stretched 2 times in the y direction, thus being *disproportionally* scaled.

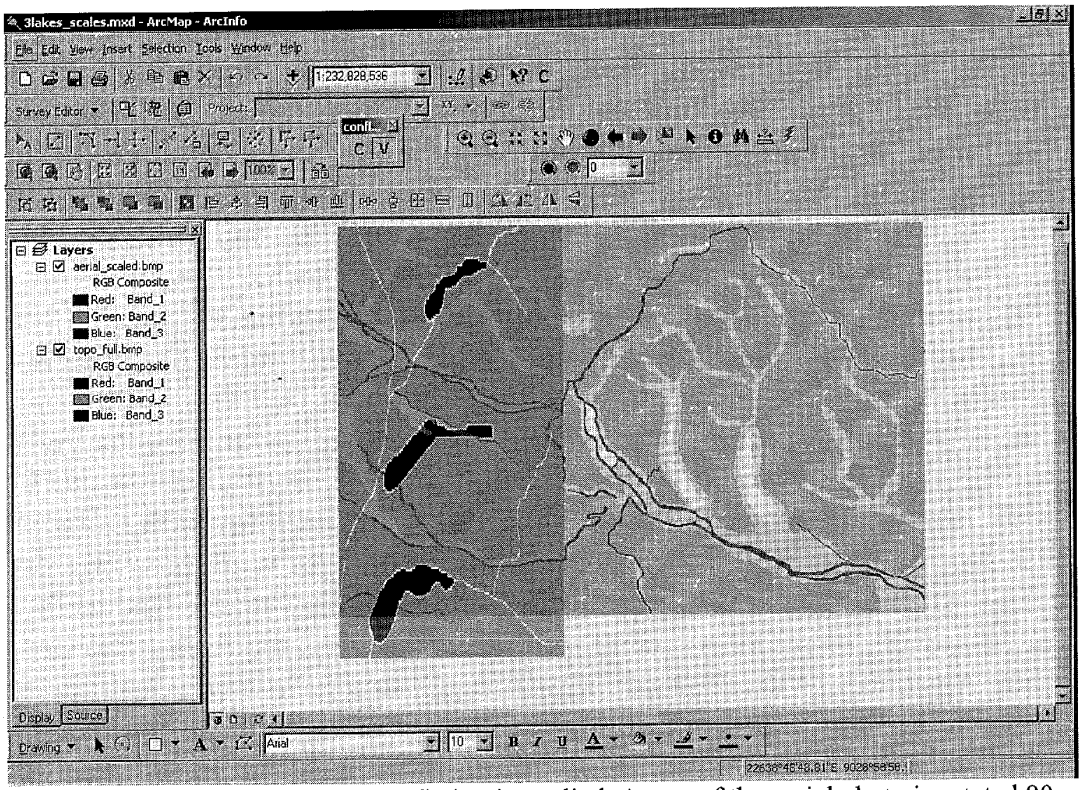


Figure 8. Two images before conflation is applied. A part of the aerial photo is rotated 90 degrees and stretched 2 times in the y direction, thus being disproportionally scaled.

The shape extraction stage in shown in Figure 9 for both images from Figure 8. One image has 6 shapes, the other has 23 shapes. Shapes 3, 4, 6 in image 1 are the 3 lakes, and shapes 14, 13, 15 are the 3 lakes in Image 2. Shape 3 matches shape 14, shape 4 matches 13, and shape 6 matches shape 15. In the visualization the top left corner of the shape label corresponds to the Center Point of the shape with that number. In this case the size ratios of the 3 lakes are used to automatically match up the images. The Center points of the lake shapes are used to calculate the transform needed for conflation. The program makes this determination automatically.

Figure 10 shows the two images with matched features shown with rotation, translation and scaling applied.

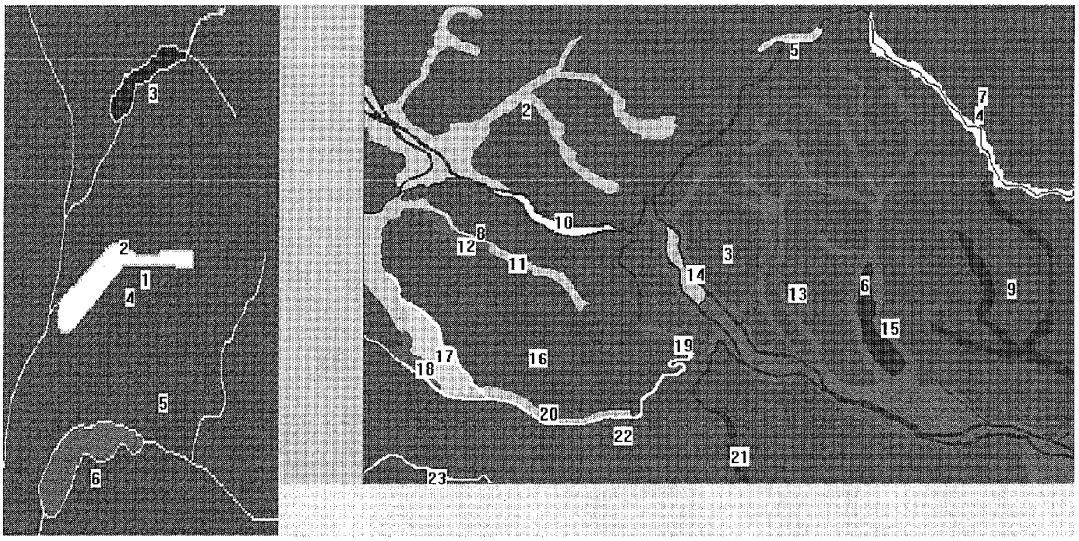


Figure 9. Two images at shape extraction stage. One of the images is scaled disproportionally. See also color plates.

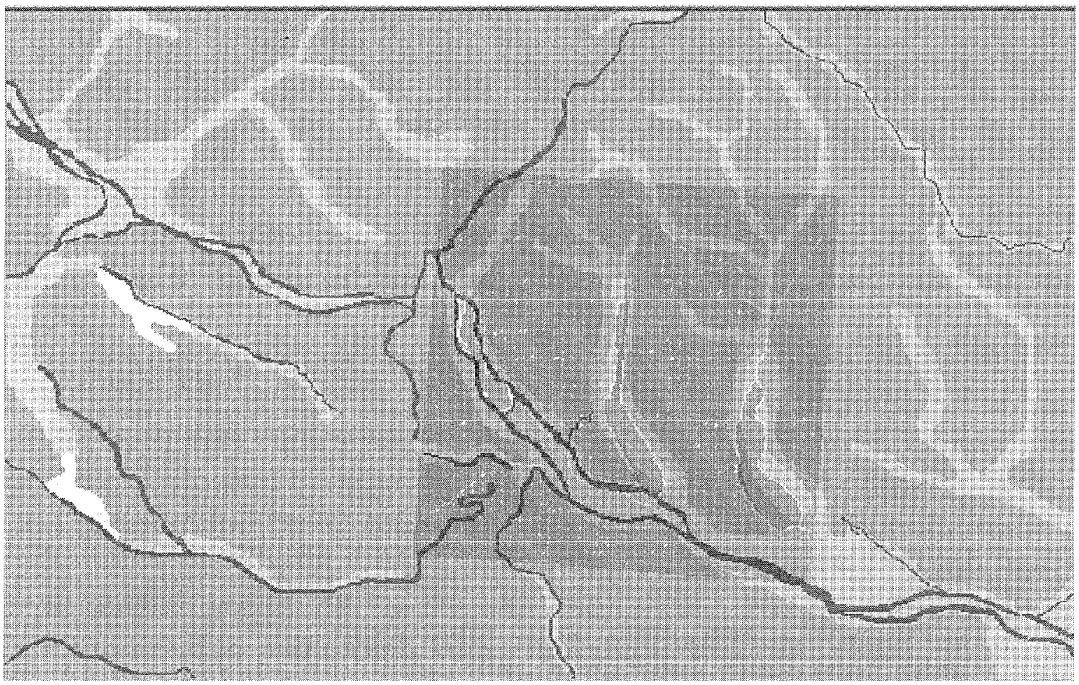


Figure 10. Two images with matched features are shown after rotation, translation and scaling are applied

Figure 11 shows the two full images of the sharpened aerial photo and the topographic map before conflation. The aerial photo is rotated 90 degrees.

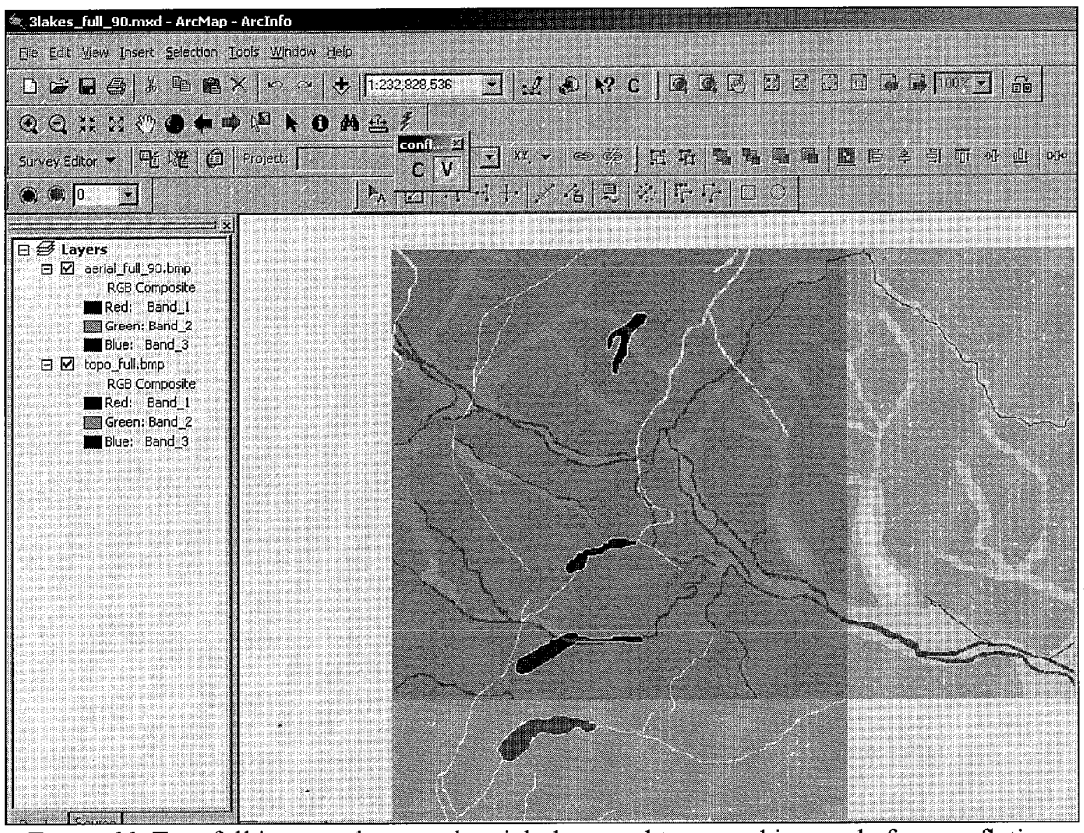


Figure 11. Two full images sharpened aerial photo and topographic map before conflation. Aerial photo is rotated 90 degrees.

Figure 12 shows the visualization of the shape extraction and shape matching stage of the conflation of the two full images of the sharpened aerial photo and the topographic map. No scale difference is present here.

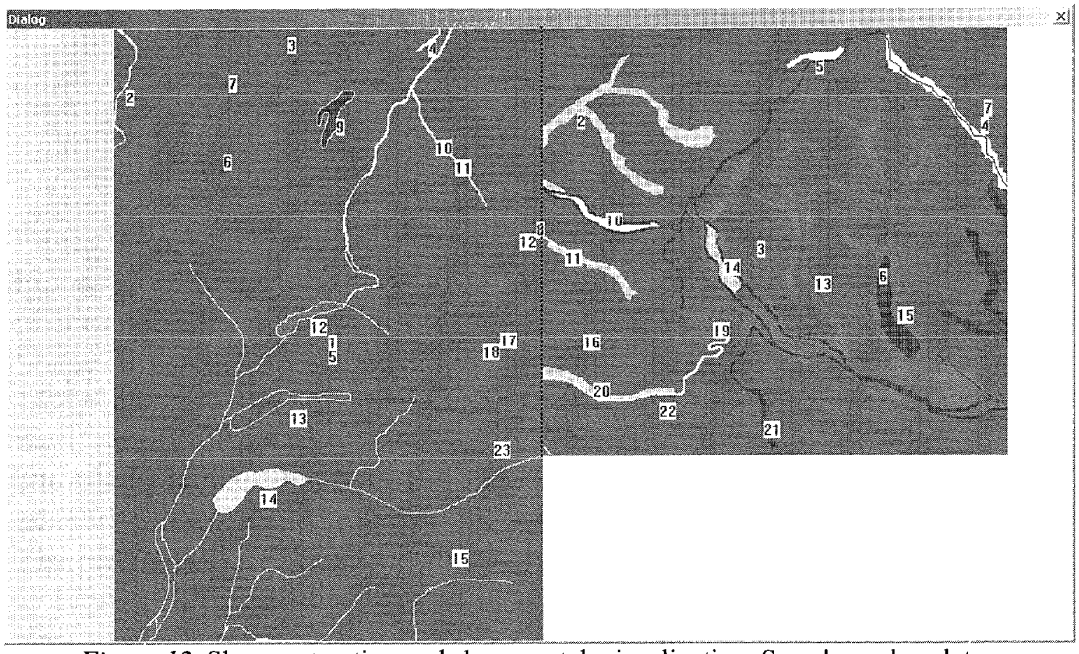


Figure 12. Shape extraction and shape match visualization. See also color plates.

The next screenshot (Figure 13) shows the two complete images after the conflation algorithm has been applied to the full-size original images.

Figure 14(a) shows the two smaller images, which are parts of the aerial photo and topographic map before conflation. Figure 14(b) shows the results of conflation of these smaller images, which are parts of the aerial photo and topographic map.

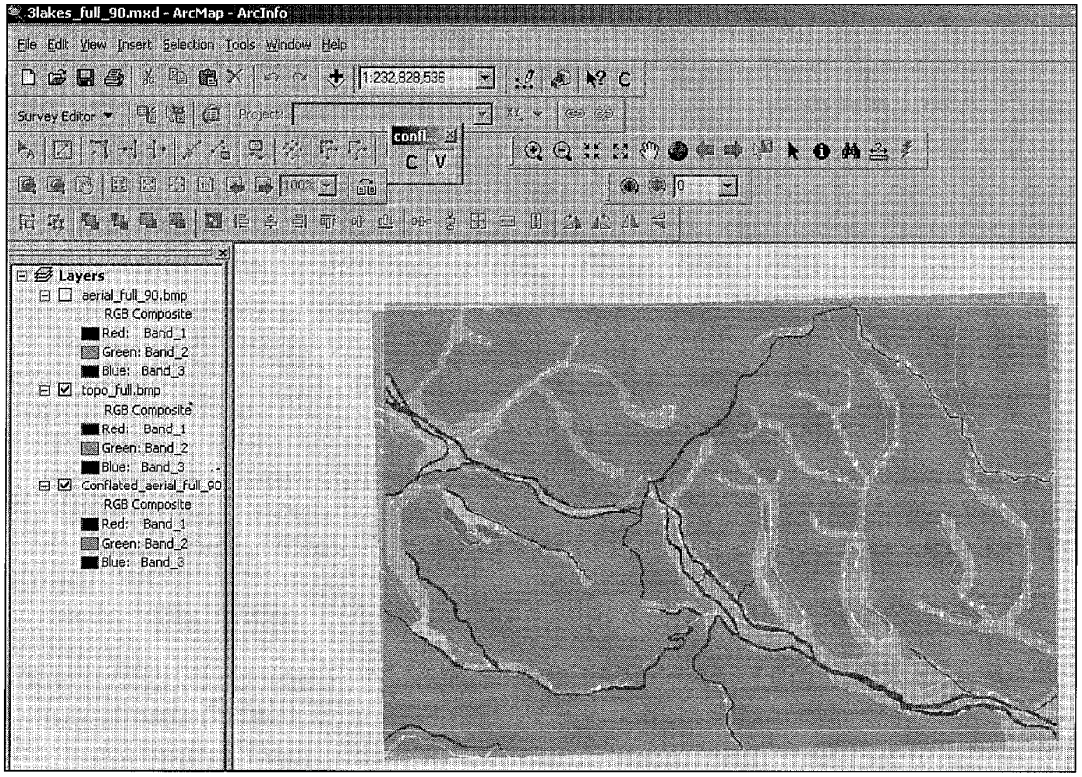
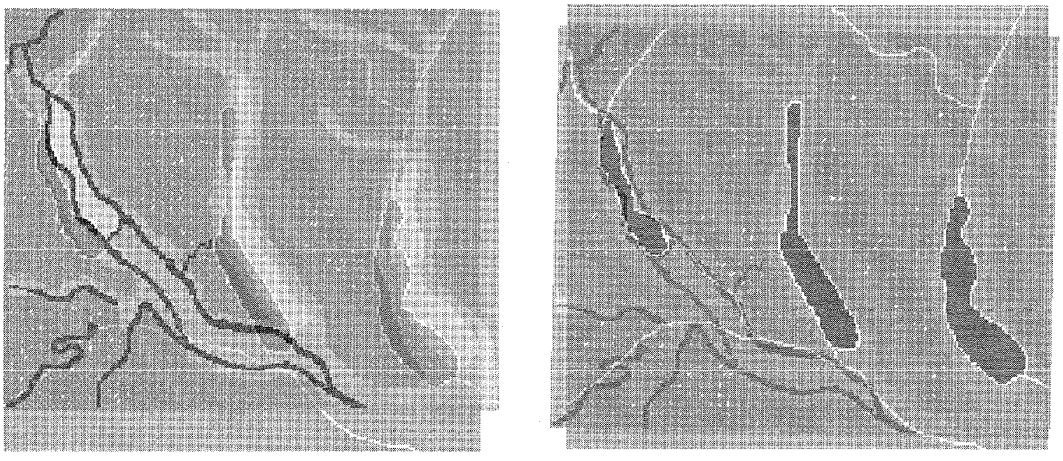


Figure 13. Two complete images after match applied to whole original images.



(a) Before conflation. *(b)* Results of conflation

Figure 14. Conflation of two smaller images, which are parts of the aerial photo and topographic map. See also color plates.

9. CONCLUSION

Human practice and expertise is fragmented, an individual imagery analyst may not work with some specific categories of images and may not provide sufficient input for formal algorithm development. A technology for algorithm development is designed for such situations to integrate a collective expertise of imagery analysts. The chapter presented an algorithm development technology for conflation (ADTC) that goes beyond the development of an individual algorithm. ADTC is described in terms of its desirable characteristics and the actual scope of applicability of individual algorithms. The concept of monotonicity is used as a tool to help analyze and identify the scope of the algorithm coverage. The ADTC technology contains eight steps described in this chapter.

The ADTC technology was illustrated with the ARC algorithm that was developed in accordance with ADTC. The experiments showed that the *ARC algorithm* could handle rotation, translation, and scaling (including disproportional scaling). The ARC uses *area ratios* that are generally invariant to these transformations. The experiments also showed that the method could handle the images with a significant number of features.

10. ACKNOWLEDGEMENTS

This research has been supported by a University Research Initiative grant from the US National Geospatial-Intelligence Agency (NGA) that is gratefully acknowledged.

11. EXERCISES AND PROBLEMS

Advanced

1. Design a conflation algorithm using ADTC technology

2. Evaluate invariance of your algorithm for affine transforms.

3. Evaluate the scope of your algorithm

4. Write code that implements algorithm ARC.

5. Write code that implements your algorithm designed in exercise 1.

6. Compare ARC and your algorithms conceptually and in computational experiments.

12. REFERENCES

Angel E., Interactive computer graphics: a top-down approach with OpenGL, Addison-Wesley, 2000

ArcGIS, ESRI, 2004, http://www.esri.com/software/arcgis/about/desktop.html

Brown, L. A Survey of Image Registration Techniques, ACM Computing Sur-veys,vol.24 (4),pp. 325--376, 1992, citeseer.nj.nec.com/brown92survey.html

Cobb, M. Chung, M,.Foley, Petry. F., Shaw, K., and Miller, H., A rule-based approach for the conflation of attributed vector data, GeoInformatica, 2/1, 1998, 7-36.

Doytsher, Y. Filin, S., Ezra, E. Transformation of Datasets in a Linear-based Map Conflation Framework, Surveying and Land Information Systems, Vol. 61, No. 3, 2001, pp.159-169.

Edwards D., Simpson J. Integration and access of multi-source vector data, In: Geospatial Theory, Processing and Applications, ISPRS Commission IV, Symposium 2002, Ottawa, Canada, July 9-12, 2002, http://www.isprs.org/ commission4/proceedings/pdfpapers/

Jensen J., Saalfeld, A., Broome, F., Price, K., Ramsey, D., and Lapine, L. Spatial Data Acquisition and Integration, 2000, NSF Workshop GIS and Geospatial Activities. Accessed June 2001 http://www.ucgis.org/research_white/data.html

Kovalerchuk, B., Triantaphyllou, E., Despande, A.S, and Vityaev, E. Interactive Learning of Monotone Boolean Functions, Information Sciences, Vol. 94, issue 1-4, 1996, pp. 87-118.

Kovalerchuk, B., Vityaev E., Ruiz J.F., Consistent and Complete Data and "Expert" Mining in Medicine, In: Medical Data Mining and Knowledge Discovery, Springer, 2001, pp. 238-280.

Shah, M., Kumar, R., (Eds.) Video Registration, Kluwer, 2003

Terzopoulos D., Studholme, C., Staib, L., Goshtasby A., (Eds) Nonrigid Image Registration, Special issue of Computer Vision and Image Understanding Journal, vol. 89, Issues 2-3, pp. 109-319, 2003

Wang, J., Chun, J., and Park, Y.W. GIS-assisted image registration for an onboard IRST of a land vehicle. Proc. SPIE Vol. 4370, p. 42-49, 2001

Zitová B., Flusser J., Image registration methods: a survey, Image and Vision Computing. 21 (11), 2003, pp. 977-1000

Chapter 21

VIRTUAL EXPERTS FOR IMAGERY REGISTRATION AND CONFLATION

Boris Kovalerchuk, Artemus Harper, Michael Kovalerchuk, and Jon Brown
Central Washington University, USA

Abstract: The unique human expertise in imagery analysis should be preserved and shared with other imagery analysts to improve image analysis and decision-making. Such knowledge can serve as a corporate memory and be a base for an imagery virtual expert. The core problem in reaching this goal is constructing a methodology and tools that can assist in building the knowledge base of imagery analysis. This chapter provides a framework for an imagery virtual expert system that supports imagery registration and conflation tasks. The approach involves tree strategies: (1) recording expertise on-the-fly and (2) extracting information from the expert in an optimized way using the theory of monotone Boolean functions and (3) use of iconized ontologies to built a conflation method.

Key words: Imagery virtual expert, ontology, knowledge base, rule generation optimization, monotone Boolean function, registration, conflation.

1. INTRODUCTION

The goal of *imagery registration* is providing geospatial coordinates to the image. The goal of the *imagery conflation* is correlation and fusion of two or more images or geospatial databases. "The process of transferring information (including more accurate coordinates) from one geospatial database to another is known as 'conflation'" [FGDC, 2000]. Typically, the result of the conflation is a combined image produced from two or more images with: (1) matched features from different images and (2) transformations that are needed to produce a single consistent image. Note, registration of a new image can be done by conflating it with a registered image.

Such a way of registration can be useful if there is a lack of reliable metadata that provide registration directly. This consideration motivates us to concentrate on the conflation task in this paper. Recently the conflation has been viewed as a *matching technique* that fuses imagery data and preserves inconsistencies (e.g., inconsistencies between high and low resolution maps, "best map" concept, [Edwards, Simpson, 2002]). This approach tries to preserve the pluralism of multisource data. The traditional approach [USGS, 1998] uses an "artistic" match of elevation edges. If the road has a break on the borderline of two maps then a "corrected" road section starts at some distance from the border on both sides and connects two disparate lines. This new line is artistically perfect, but no real road may exist on the ground in that location (see Figure 1).

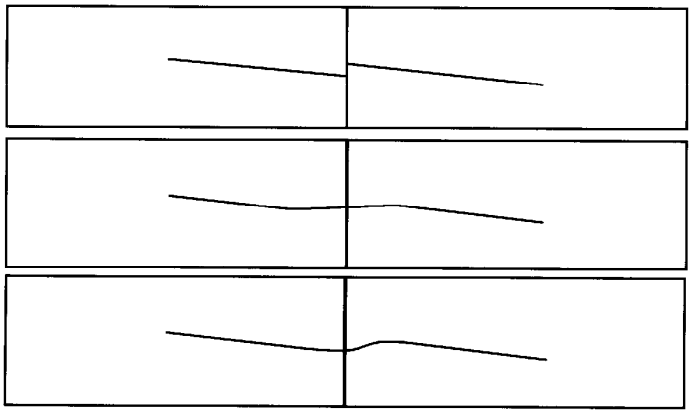


Figure 1. Initial mismatch and "artistic" conflations

Why design virtual experts for conflation? Can the conflation problem be solved by designing a sophisticated mathematical procedure without relying on an expert's knowledge? In essence, the conflation problem is a *conflict resolution* problem between disparate data. Inconsistencies in multisource data can be due to scale, resolution, compilation standards, operator license, source accuracy, registration, sensor characteristics, currency, temporality, or errors [Edwards, Simpson, 2002]. The conflict resolution strategies are highly *context and task dependent.* Dependency of conflation from a specific task is discussed in Chapters 17, 18 and 19.

In solving a conflation problem, experts are unique in extracting and using non-formalized context and in linking it with the task at hand (e.g., finding the best route). Unfortunately, few if any contexts are explicitly formalized and generalized for use in conflating other images. It is common that the context of each image is unique and not recorded. For example, an expert conflating two specific images may match feature F1 with feature F3, although the distance between features F1 and F2 is smaller than the distance between features F1 and F3. The reasoning (that is typically not recorded)

behind this decision could be as follows. The expert analyzed the whole image as a context for the decision. The expert noticed that both features F1 and F3 are small road segments and are parts of much larger road systems A and B that are structurally similar, but features F1 and F2 have no such link. This conclusion is very specific for a given pair of images and roads on these images. The expert did not have any formal definition of structural similarity in this reasoning. Thus, this expert's reasoning may not be sufficient for implementing in an automatic conflation system for conflating other images. Moreover, informal similarity the expert used for one pair of images can differ from similarity the same expert will use for two other images.

There are two known approaches to incorporate context: (1) *formalize context* for each individual image and task directly and (2) *generalize context* in the form of expert rules. In the first approach, the challenge is that there are too many images and tasks and there is no unified technique to for context formalization. The second approach is more general and more feasible, but in some cases may not match a particular context and task, thus a human expert needs to take a look.

2. SHORTCOMINGS OF PREVIOUS ATTEMPTS TO DEAL WITH THE SUBJECT

Currently, even large knowledge bases cannot answer many questions, which are in their scope. The real world is too dynamic, uncertain, and complex for even modern knowledge bases. The conflation/registration problem is an example of such a real world problem. DARPA's program "High-Performance Knowledge Bases" [HPKB, 1996] that started in 1997 has set up the critical size barrier for large knowledge bases around the *10,000 axiom/rule* limit. At that time, DARPA's goal was to build technology, which will scale up to *100,000 axiom/rule* knowledge base systems [HPKB, 1996]. The DARPA program "Rapid Knowledge Formation" [RKF, 1999] formulated new requirements that include parallel entry of knowledge by teams of 25-50 individuals (end users) for test tasks such as crisis management and battlespace understanding.

According to DARPA using High Performance Knowledge Base technology, a 5-person team can create knowledge at a rate of *40 axioms* per hour and *100K* of axioms per year. After that, DARPA stated a new goal: the creation of new knowledge at a rate of *400 axioms* per hour. Next, DARPA identified the criterion of *comprehensiveness* of the knowledge base at the level of *a million axioms*. The PARKA project research team at the University of Maryland extracted *125913 assertions* (facts) from CIA World Fact book pages on the World-Wide Web using a *web robot* [VLKB, 1998].

Thus, a million axioms can be comparable with encoding knowledge from eight books like the CIA World Fact book. Next, notice that these assertions are not rules, but *facts* such as "economy_imports#tajikistan#$690 million {1995}," that is, Tajikistan's imports were $690 million in 1995. In addition, those assertions have been extracted from the written text using a web robot. For the conflation task, there is *no text available* to use as a source for a web robot. This means that we need to build such written sources and extract rules from experts directly.

To understand what the million-axiom size of the knowledge base means in more detail we need to clarify the concept of axiom used by DARPA. The same DARPA source provides an example of the axiom:

$\forall x, p_1, p_2$ vehicle(x) $\Leftrightarrow$ physical_object(x) and self-propelled(x) and can(move(x), p_1, p_2).

This is a relatively simple statement with three basic statements combined using AND operator. For complex tasks with interdependent attributes, axioms can involve more than ten statements connected by the AND operator. Respectively the time for extracting these rules can be much longer. They also can be much less trivial and certain as in conflation problems.

Let us consider the rate and quality of knowledge base development reached in 1999 in the High Performance Knowledge Base program [RKF, 1999]. The maximum number of axioms was 90,000 per 10 months and the smallest number of axioms was 2300 per seven months. Depending on the domain and the problem, 90K may be not enough or 2300 may be enough. Also note that $90{,}000 < 2^{17}$, which means that for designing a complete knowledge base with 17 binary attributes we need even more axioms.

Building comprehensive virtual experts. Let us illustrate the important question of completeness and comprehensiveness of the knowledge base. A knowledge base will be called *complete* if for a given set of attributes the knowledge base can generate an answer for every combination of values of these attributes. We will say that a knowledge base has *comprehensive coverage* if a set of attributes of rules in the knowledge base covers most of attributes used in the domain. For instance, we may include in the knowledge base all the attributes used in NIMA's Vector Product Format (VPF) to get a comprehensive coverage. Nevertheless, this knowledge base may not be complete because rules cannot produce answers for many questions formulated as AND combinations of VPF attributes.

However, there are some *positive examples* of a complete knowledge base. For instance, in medical imaging [Kovalerchuk, et al., 1996, 2001] having 11 binary attributes of X-ray images (mammograms) entered into the knowledge base for a particular patient the knowledge base should output

one of the target values: "highly suspicious for malignancy" or "probably benign." Another target with the same 11 binary features for the same patient will be biopsy (should biopsy be done or not). Both of these questions are life critical questions (more than 100,000 die each year of breast cancer in the US). An axiom "mined" from the experienced radiologist may look like the following:

If variation in shape of calcifications is marked AND the number of calcifications is between 10 and 20 AND the irregularity in shape of calcifications is moderate THEN the case is highly suspicious for malignancy.

The common way to extract such rules is to ask an expert to write down rules that the expert uses and to provide software for converting the rules to computer-readable knowledge base (KB) form. However, it is unlikely that the expert will enter complex rules involving, say 12 attributes. Often it is above a human's capabilities to keep in mind more than 5 to 9 attributes for analysis. Later testing may show that the rule with only 3 attributes is wrong for some cases and the knowledge base should be refined. The refinement can take years and for life critical applications the systems should not be used before the process of cleaning rules will successfully finish. The problem is that this process can be exponential in time. For instance, having 14 binary attributes we may search among 2^{14} =16384 potential rules like the rule shown above.

To avoid refining and testing the knowledge base for years, we need to be sure that the set of rules is complete enough from the beginning. Asking the expert does he believe that 10 rules he entered are complete may not be the right choice. We need to be sure that the rest of potential 16384-10=16374 rules are not rules at all. Thus, DARPA's design time should also measure both rules included in the knowledge base and the rules *rejected.* If we know that something is not a rule, this is also useful knowledge. There is a big difference between a rejected rule and a rule unconfirmed by the expert or not tested against independent data yet. DARPA's current goal of 1,000,000 axioms, would correspond to a design of a complete knowledge base with less than 23 binary attributes. More exactly, in the worst case scenario for 22 attributes we may need to record 1,144,066 axioms (using formula from Hansel Lemma [Kovalerchuk et al, 1996, 2001]), which exceeds the 1,000,000 axioms. To find all these axioms (rules) we may need to search in a much larger set of possible axioms, which is 2^{23}.

Let us note that the problem of designing a complete knowledge base above the limit of 20 attributes (10^9 potential rules) is especially critical when knowledge is not presented in any printed form (book, articles) and should be extracted from an expert as a sole knowledge body. This is the case of the virtual expert for imagery conflation/registration problem.

3. GOALS AND IVES SYSTEM ARCHITECTURE

The goals of this chapter is to determine how to build an **Imagery Virtual Expert System (IVES)** for imagery analysis, create tools to capture imagery specific information and knowledge for IVES, and create tools to foster intelligent consultation with IVES.

We discuss specific tasks and methods that represent three views of the system: an imagery analyst view (end user view), knowledge engineer's view (system support view), and tools view (developer's view).

The support of an **imagery analyst's view** means defining *imagery analysis* from the Decision Making Level and to the Subpixel Level, implementing analysis tools, providing *quality assurance* tools, and specific tools for imagery *conflation.*

The support of a **knowledge engineer view** includes providing tools for *discovering imagery analysis rules* including recording conflation process and discovering conflation rules.

The support of **a developer view** means providing conceptual and algorithmic base for building methods and software for *mission-specific solutions* that include:

- new *optimized rule extraction procedures* using monotone Boolean functions;
- a contradiction analysis method for extracted rules and for *deconflicting* rules obtained from different sources;
- a recording method for capturing expert knowledge *on-the-fly*;
- Image-DAML (*I-DAML*) language, DAML ontologies, and agents

IVES architecture contains three integrated components:

- Analysis and Recording Tool (ART),
- Multi-Image Knowledge Extractor (MIKE), and
- Joint Outline Notator with icon markup (JON).

The general design of IVES system is presented in Figure 2. The complete system design contains seven components. The first component- Interactive on-the-fly recording of expert's *actions* during registration/conflation serves as major source of the raw information for rule generation. It is also useful as a quality control tool of the analyst's conflation results (a kind of airplane "black box"). An interactive optimized *rule generation procedure*

based on the theory of monotone Boolean functions (TMBF) is intended to speed up direct generation rules by the imagery analyst. These two components are implemented in Java as a web portal.

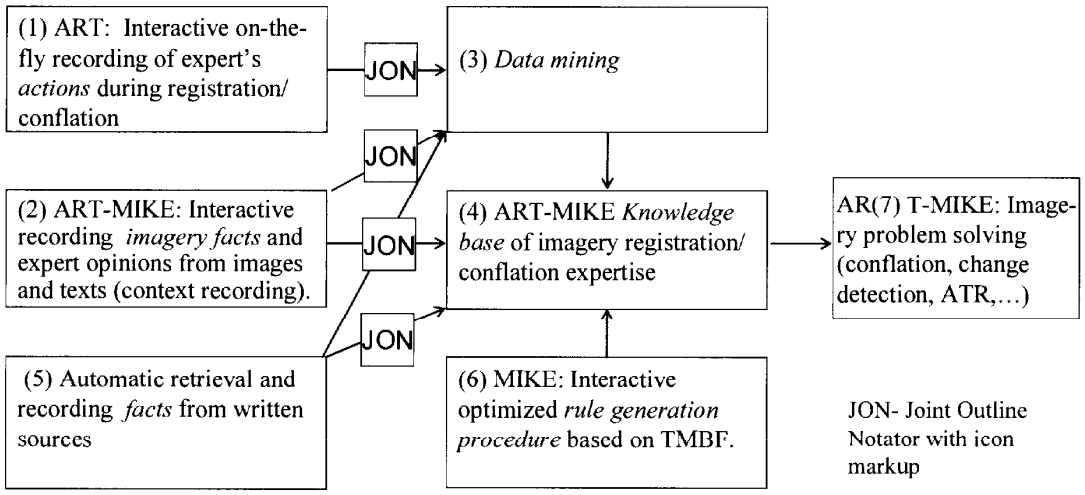


Figure 2. Imagery Virtual Expert System (IVES) architecture

These components are described in more detail in the following sections. The data mining component (block 3) is designed to generalize the record of an expert's actions. The results of such generalizations are conflation rules. The major problem for successful mining of rules from recordings is that the system actually records lower level expert's actions, such as rotation, scaling, translation. Mining these lower level rules may not be very beneficial, thus the system provides a tool for the expert to record an identification of upper level categories such as "selecting main feature", "conflating main features", etc. The system design includes recording expert's actions and mined rules in XML format, for rules it is RML (rule markup language).

Automatic retrieval and recording of facts from written sources (Block (3)) is designed to fulfill functions similar to PARKA project [VLKB, 1998]. At this moment, we do not consider this source as a main source to fill the KB with facts related to registration and conflation of images, but potentially it can provide useful facts for the KB. In contrast, interactive recording of imagery facts from images and texts (Figure 2, block (6)) can be one of the major sources of conflation related facts right now. The reason is that such recording provides *context* for conflation and registration tasks.

4. INTERACTIVE ON-THE-FLY ANALYSIS AND RECORDING

The IVES system component called the Interactive on-the-fly **Analysis and Recording Tool (ART)** is implemented as a web portal that allows an expert to conflate images while having the knowledge presented by conflating these images recorded on-the-fly. A part of IVES is also implemented as an ArcMap Plug-in. ART currently allows the user to load a set of images and conflate these images using basic scaling, translation and rotation tools.

The user can view these images overlapped and change the opacity of the images for conflating. The system provides facilities for marking up the sections of the images using various shape tools. Each of these markups can be named by the user using a basic name of his/her choice or by choosing from a list of predefined terms from a variety of ontologies. Such marking permits to build the bridge between the image and the domain that will describe the image in a deeper context.

Currently, the system ontology base includes three ontologies:

- DAML-OWL Geofile ontology [DAML Geofile, 2001],
- DAML-OWL CIA World Fact book ontology [DAML WFB, 2002],
- DAML-OWL NIMA Feature and Attribute Coding Catalogue [FACC].

Other ontologies also can be loaded. The Geofile ontology consists of about 70 terms on the low level and 6 terms on the top level of the tree next to the root. The World Fact book ontology consists of about 190 terms on two levels and the Feature and Attribute Coding Catalogue ontology consists of 540 terms on the low level, 59 terms on the next level and 8 terms on the level on level next to the tree root. ART supports tree view of these ontologies with the following functionality: browsing, editing (adding new terms and deleting terms), expanding and shrinking tree view, adding icons to terms and drugging icons to images.

Selected parts of images can be marked up with terms from these ontologies (see block 1 in Figure 3). There is also a detailed list of actions the user has performed that can be undone and redone to any point. A basic magnifier is available for taking a detailed look at the image. Any of these markups and conflations can be applied to multiple images to allow two (or more) images already conflated to be conflated to a third image or simply zoom in all images. All of these actions are recorded, can be presented in a human readable form and are available for playback.

ART includes three categories of tools: *basic tools*, *conflation tools* and on-the-fly user *action recording tools*. Basic and conflation tools allow the user to load a set of images, and then conflate them using scaling, translation

and rotation and affine control points and shape based conflation tools. These tools provide a foundation for interactive tools on-the-fly recording of expert's actions is implemented as a web portal. Thus the system allows an expert to conflate images while having the expert knowledge presented by conflating these images recorded on-the-fly.

The user can view these images overlapped and change the opacity of the images for conflating. The system provides facilities for marking up the sections of the images using various shape tools. Each of these markups can be named by the user using a basic name of his/her choice or by choosing from a list of predefined terms from one of the ontologies, e.g., DAML-OWL Geofile ontology. There is also a detailed list of actions the user has performed that can be undone and redone to any point. A basic magnifier is available for taking a detailed look at the image. Any of these markups and conflations can be applied to multiple images to allow two (or more) images to be conflated. All of these actions are recorded, can be presented in a human readable form and are available for playback. Figure 3 shows a conflation sample with user action recording using ART.

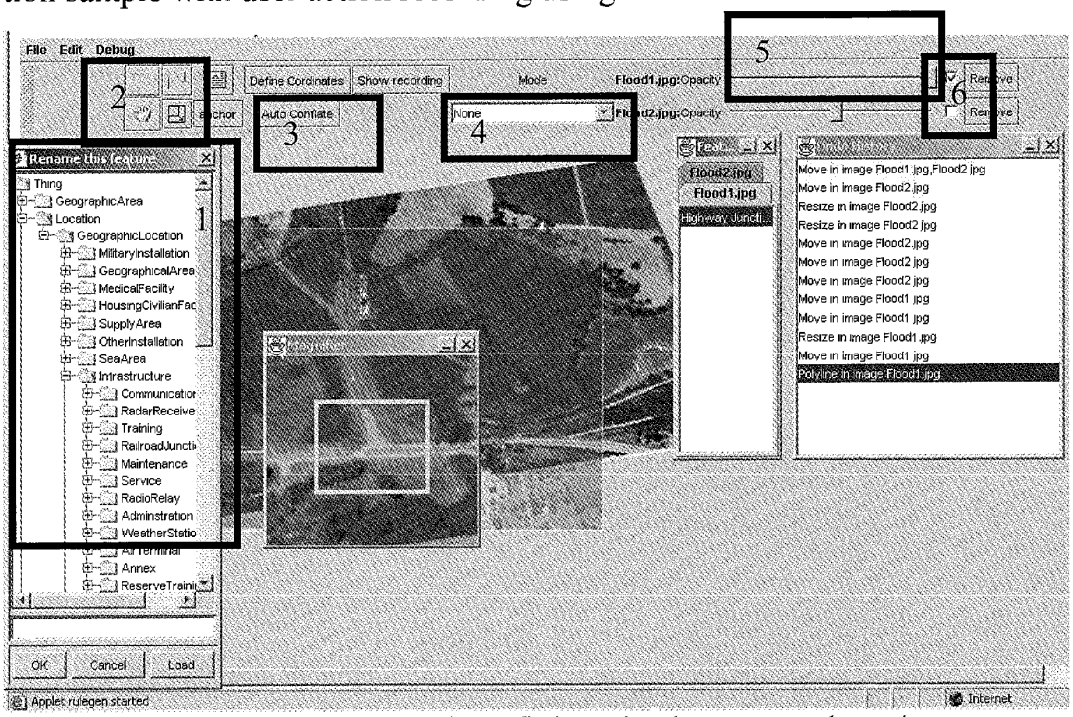


Figure 3. Image of sample conflation using the case recorder tool

The list below presents comments to numbered items shown in Figure 3:

1. The Markup tools can be used to mark major features on the image and name them, these names can be defined in a variety of ontologies.
2. The move, resize, and rotate tools are basic operations used to conflate images

3. The Auto Conflate tool allows the user to choose 3 points on two images and using transformations, match those points together (and hopefully the images as well).
4. The Show recording button allows the user to see how the conflation was broken up via the current mode that was used in various segments of the conflating.
5. The Opacity slider is used to set the transparency of the images, allowing one image to be seen through another.
6. The checkboxes are used to select the image(s) that will receive the operations such as move or draw polyline. This allows a user to resize or rotate both images together, or once two images have been conflated together, a third image can be brought in and conflated against the other two together.

5. MULTI-IMAGE KNOWLEDGE EXTRACTOR

5.1 Components and architecture

A **Multi-Image Knowledge Extractor** (**MIKE**) assists an imagery analyst in rule extraction and recording. The common way to extract rules is asking an expert to write down rules and providing software for converting the rules to computer-readable knowledge base (KB) form. The major problem with this straightforward approach is that:

- Typically experts have limited time available for rule entering, refining, testing and debugging.
- The refinement time can grow exponentially with adding more attributes.
- Experts are unlikely to enter complex rules because it is difficult to keep in mind more then 7±2 attributes.
- In life critical applications, the process of rule refinement and debugging has to finish before the system is used.

The complete IVES system design contains seven units shown in Figure 2 above. Unit (1) serves as major source of the raw information for rule generation. Unit (2) supports interactive optimized rule generation. The rule generation contains five steps depicted also in Figure 4:

- Interactive recording characteristics to be used as arguments of rules;

- Interactive optimized rule generation based on the theory of monotone Boolean functions;
- Recording test cases for testing rules;
- Testing rules against test cases, and
- Recording rules to the knowledge base of imagery registration/conflation expertise.

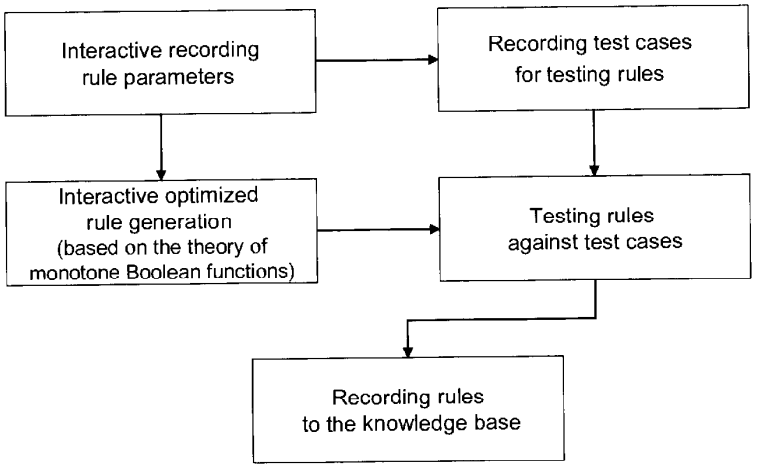


Figure 4. Interactive rule generation

5.2 Interactive optimized rule generation and testing

To model expert knowledge an interactive optimized rule generation mechanism is implemented in MIKE. It is based on the theory of monotone Boolean functions [Kovalerchuk et al., 1996, 2001] where a set of binary vectors represent combinations of image characteristics an inputs of monotone Boolean functions. Previously approach has been successful in the medical application [Kovalerchuk et al., 2001]. The medical imagery analyst (radiologist) was asked only 40 questions and a complete set of rules (out of potential 2048 questions) has been extracted in just 30 minutes.

Prototype web-based medical expert consultation system has been created. Similarly, a prototype web-based conflation imagery virtual expert has been created and available for the research and education purposes from the book website at http://www.cwu.edu/~borisk/bookVis.

The logic of *rule generation* based on the theory of monotone Boolean functions is as follows. Assume that the analyst identified n rule arguments (parameters). An example of parameters shown in Figure 5 is for the task of creating expert rules to judge that two images can be conflated by using a single affine linear transformation). Figure 5 contains 14 parameters such as:

(1) a simple dominant geometric feature exists, (2) a simple unique geometric feature exists, (3) an asymmetric unique features exist and so on.

Now the imagery expert can be asked if all these 14 arguments (parameters) are true for a pair if images, would he/she conclude that two images can be conflated with a single affine linear transformation that most likely is unique. If the answer is "yes," then it will be encoded as 1. Taking into account that all 14 arguments are Boolean too, we matched a Boolean vector (11111111111111) to (1). We can consider a subset of 14 parameters and ask the expert the same question about these subset of parameters. For instance, we can ask about situation represented by the vector ((11011101110111), where the third "0" indicates that we do not require that parameter #3 should be true. The total number of subset (and questions) could be 2^{14}.

Assume that we built a system asking an expert only some of these 2^{14} questions. Then if in the real conflation case we have a situation (11011101110111) that was not asked about then the *incomplete knowledge base* does not provide the answer and the conflation task can not be solved even if this is the solvable situation. The theory of monotone Boolean functions allows us to avoid asking 2^{14} and still generate a **complete set of rules**. A specific example of rules build using MIKE system is described in Chapter 20.

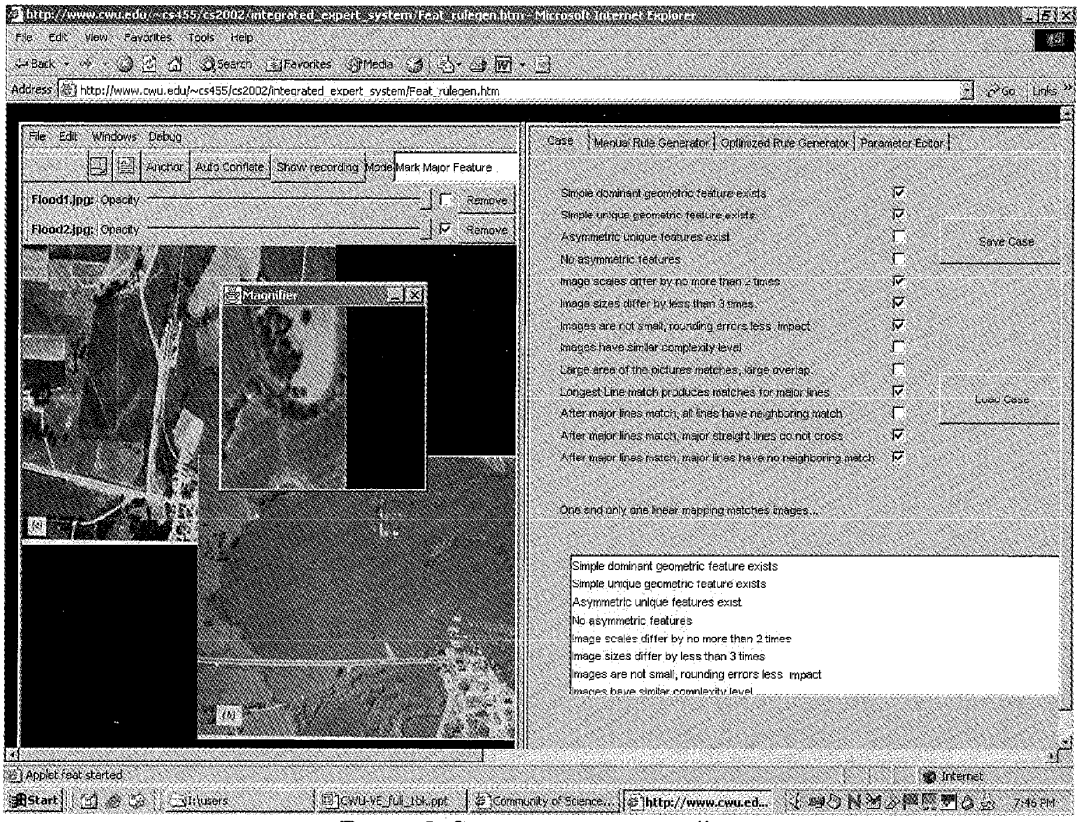


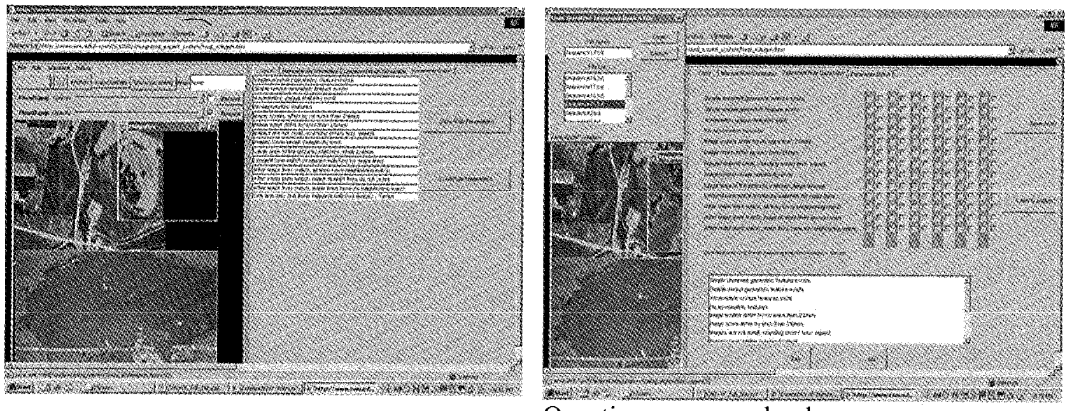
Figure 5. Case parameter recording

Two fundamental ideas permit to accomplish this optimization of the expert interview process: (1) dynamic sequence of questions – each further question depends on the expert's answers to previous questions, (2) rule arguments are designed in such way that the property of monotonicity is fulfilled.

Monotonicity means that if the expert believes that in the situation presented by the vector v =(11011101110111) there is no way to have a single linear affine transformation then for every situation that has some "1" in v substituted by "0" the answer should be negative too. This means that we can eliminate a large number of questions to the rational expert.

Expert interactively defines the rules that are encoded by the system of binary vectors. Next rules are tested against a database of test cases for which the answer is known in advance. In image conflation, the known cases can be the cases that are actually *georeferenced* and the *correct transform* can be computed from reference points. This would be testing against georeferenced data.

Initial steps. The screenshots of MIKE system are shown in Figures 6 and 7 below. Figure 6(a) shows recording parameters and Figure 6(b) shows loading the sequence of the questions to be asked from an expert, imagery analyst using the property of monotonicity.



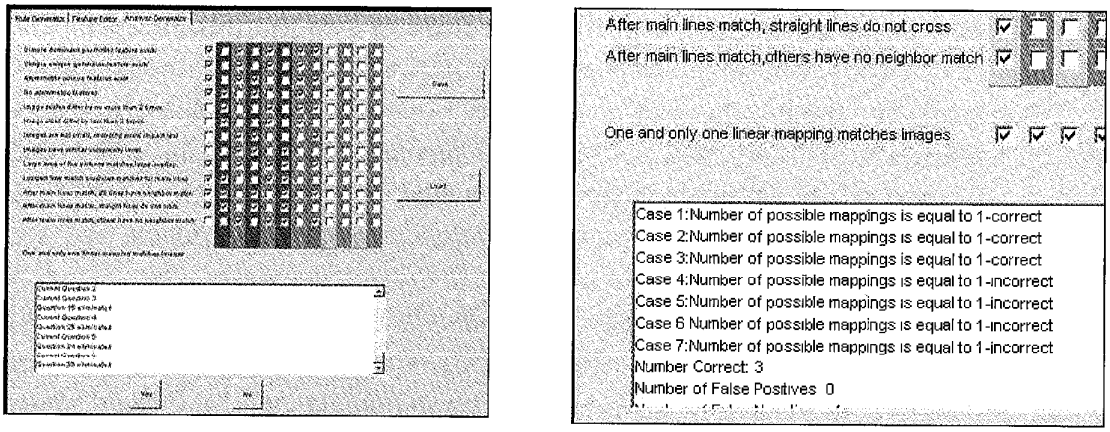
Parameter editor

Question sequence loader

Figure 6. Parameter recording and loading question sequence

Defining rules. In Figure 7 (a), each column is a question asked from the expert that is used by the system to build conflation rules in the form if <conditions for Image1 and Image 2> then it is highly possible that <one and only one (unique) affine transform that conflates images 1 and 2 exists>. Colored columns are questions already answered by the expert (green columns indicate "yes" answer and "red" column indicate "no" answers). The current question is shown in the first (left-most) column. The expert presses "yes" or "no" buttons for the current question as determination of green or

red answer. The system after coloring the answer shifts all columns to the right and shows a new question in the first column to the expert.



(a) Defining Expert Rules (b) Testing rules against image set

Figure 7. Defining expert rules and case parameter recording

Sequencing questions. The question sequence is loaded to the system as a text file as shown in Figure 6(b). A knowledge engineer can select another file that provides another sequence of questions. In this way the sequence that is most appropriate for the given expert and problem cab be used. The system is not simply show the next question from the file but generates it *dynamically* that is the next question can be eliminated if answer for that question can be derived from expert's previous answers. Elimination is based on the powerful principle of monotonicity described in this chapter. The question log is presented in the bottom of the screenshot in Figure 7(a). It is updated after each answer and shows the index of the question that was eliminated from the question sequence presented to the expert.

Automated rule testing. The screenshot in Figure 7(b) shows how the rule R that contains some of 14 parameters checked (i.e., their conjunction) is tested against a set of test image pairs $T=\{<I_i,I_j>\}$. The value $R(I_i,I_j)=1$ is interpreted as a forecast that images I_i and I_j are most likely uniquely conflatable by an affine transform.

Each pair in T is associate with "ground truth" binary flag, $f(I_i,I_j)$ which is interpreted that images I_i and I_j are really uniquely conflatable by an affine transform, that is one and only one affine transform between I_i and I_j exists in their common parts within a reasonable error limits.

The log window in Figure 7(b) shows that rule R was correct only partially. There are pairs of images where R forecasts differ from $f(I_i,I_j)$ values known for T. Pairs of images from T may have no any affine transforms or have several different affine transforms within acceptable error level. Note that accuracy of R was evaluated using the judgment that parameters are correctly evaluated.

6. ICONIC MARKUP IN IVES

Iconization of images permits moving image analysis from the realm of details to *simple visual pattern recognition.* Dynamically composed icons displays *user oriented relevant data.* Grouping icons into a meaningful organization similar to storyboards for movie direction is useful for reviewing one or more complex image summaries in iconic form. All together this technique enhances *readability and comparability of images.* Dynamic icon interaction moves complex analysis towards *Drag and Drop icon simplicity.*

Supplying an analyst with an iconic markup system presents significant advantages for IVES by increased speed for the imagery analysis process and by more intuitive understanding of analysis elements over more typical markup systems such as simple polygons. The iconic markup architecture provides the following functionality:

- Image iconization;
- Scalable iconic summaries;
- Iconic storyboards;
- Aggregate icons;
- Dynamic icon interaction and
- Customizable icons.

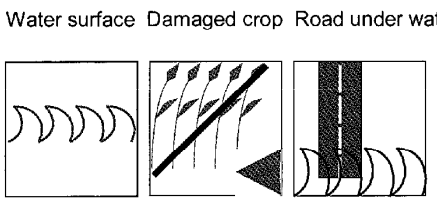
Icons can be operated on to compose, customize, change application area, and change non image data. A scalable solutions using iconic summaries of images and analyses compresses large volumes of information and make them more manageable for analysis.

This is a *customizable framework* – iconic summaries and notation can be designed to fit the analyst's focus, needs and working style. Tools such as Semantic Zooming, Icon Editing and Icon Libraries aid in assuring that the user can tailor the system to his/her needs.

These capabilities are described in more detail in Chapters 8-10 with application for text annotation by icons, but this technique is equally applicable to images. An iconic notation system enhances standard markup with icons and other icon based mechanisms used to identify aspects of the markup region, and other related info. If icons are added to the markup, there are instances where an iconic summary is sufficient to give an overview and access to details about the image without the need to present the image. This offers significant space savings and can result in enhanced comparisons over large numbers of images. Figure 8 illustrates a combination of a standard marking the area of interests (AOIs) using rectangles and polygons with iconic annotation.

Multi-image feature correlation.

Detailed markup



Iconic summary of image

Figure 8. Iconic notation

A rectangular markup in top left image in Figure 8 shows the flood area and the flood icon in the corner. The smaller rectangular markups indicate road and crops under flood. After two images have been conflated the marked up areas are automatically transferred to another image taken before the area was flooded. Such transfer can help to estimate damage and plan rescue operations. A detailed markup shown on the left identifies the flood area in more detail. The iconic summary in the bottom of Figure 8 can be read as "a flood area with crop damage and a road under flood". An analyst can review such annotations before looking actual images in detail especially after conflation. This can be done faster than work with images that contain much more information and majority of which may not be relevant.

7. ICONIC ONTOLOGICAL CONFLATION

General Iconic Conflation is implemented as an extension of ART-MAKE software. This software: (1) loads images, (2) loads ontologies including iconized ontologies, (3) edits ontologies, (4) marks up images using a selected ontology, (5) marks up images with icons associated concepts in ontologies (each images can be annotated using a mixture of ontologies), (6) generates ontological iconic annotation of images to be able to compare and

conflate images on conceptual ontological level, (7) stores marked up images in a database, loads marked up images, runs iconic conflation to find matched images and conflates images using ontological similarity measure. This architecture is depicted in Figure 9.

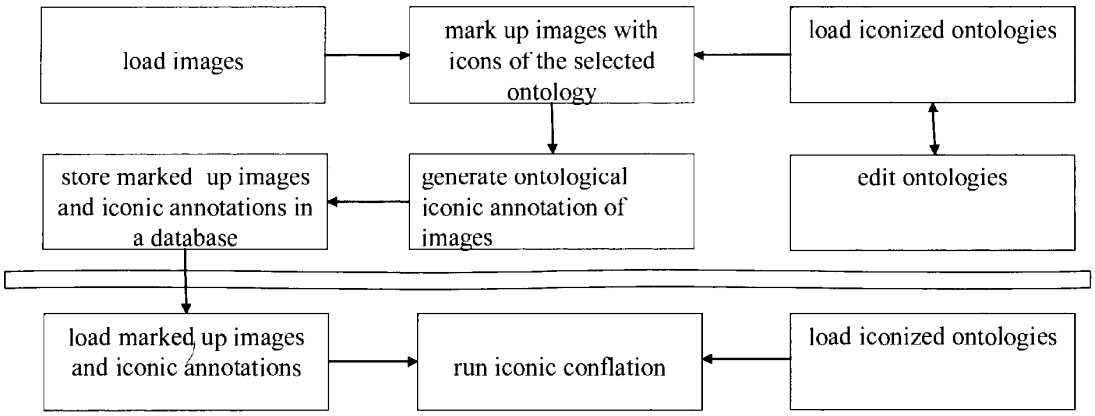


Figure 9. Iconic conflation architecture

7.1 Flood case

Below we describe a conflation task of two images taken from [Lillesand, Kiefer, 1987]. In essence these images cover the same area, but one of them covered by water as a result of flood. The flood area is almost half of the image. We assume the scenario where person A annotates image flood1 independently of person B who annotates image 2 using the same or different ontologies. Because persons A and B do not coordinate image annotation process, they can pick up different terms from ontologies to mark the same spot in two images. This is a typical situation in GIS where the same object may have several individual and group manes associated.

We start with *Person A.* We assume that person A's only goal is to annotate the image with useful icons so it can be later used as a tool in conflation. This person even may not possess skills to accomplish conflation.

Screenshots in Figure 10 show the initial steps of the iconic conflation process after opening a blank workspace of ART, loading an OWL Goefile ontology and a raster image. The tree based ontology is marked up with icons. Later these icons appear in the iconic annotation of the images. Since Geofile didn't come with icons, synthetic ones are used instead. For simplicity of presentation the ontology is iconized with numeric icons. Any other icon can be loaded including icons located on the web using its URL. These icons can raster images or SVG files.

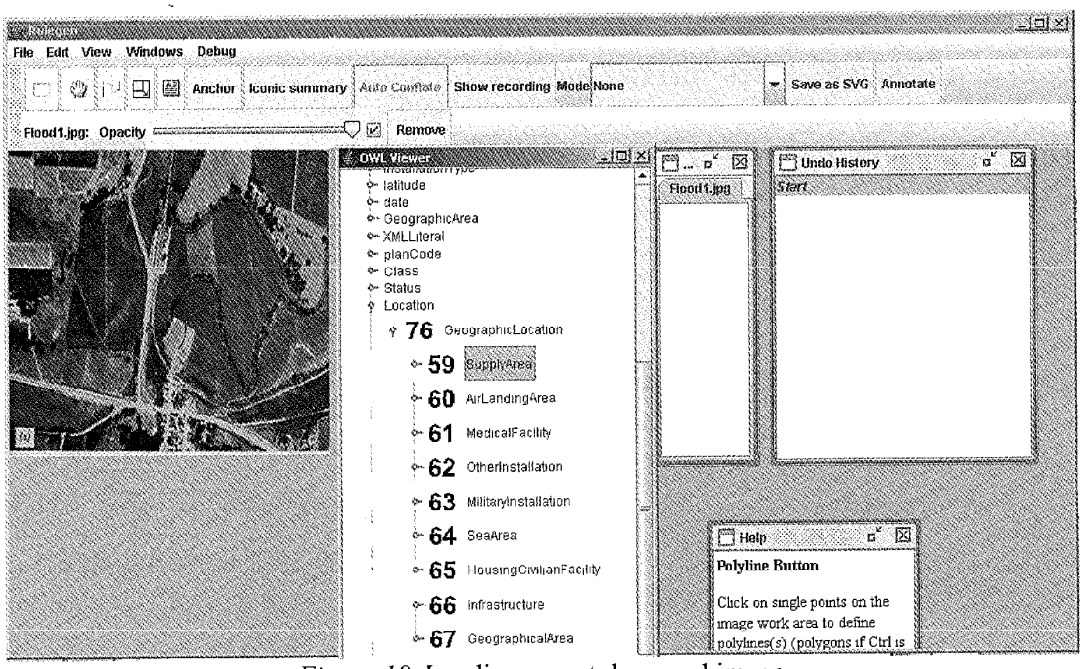


Figure 10. Loading an ontology and image

The user is now ready to drag icons onto the image in order to mark it up. This is the starting situation for the creation of the bridge between the image and the domain information in the form of DAML-OWL ontology when an image and an ontology are loaded. Figure 11 shows the next step where several numeric icons have been drugged to markup the appropriate locations in the image by the user annotator.

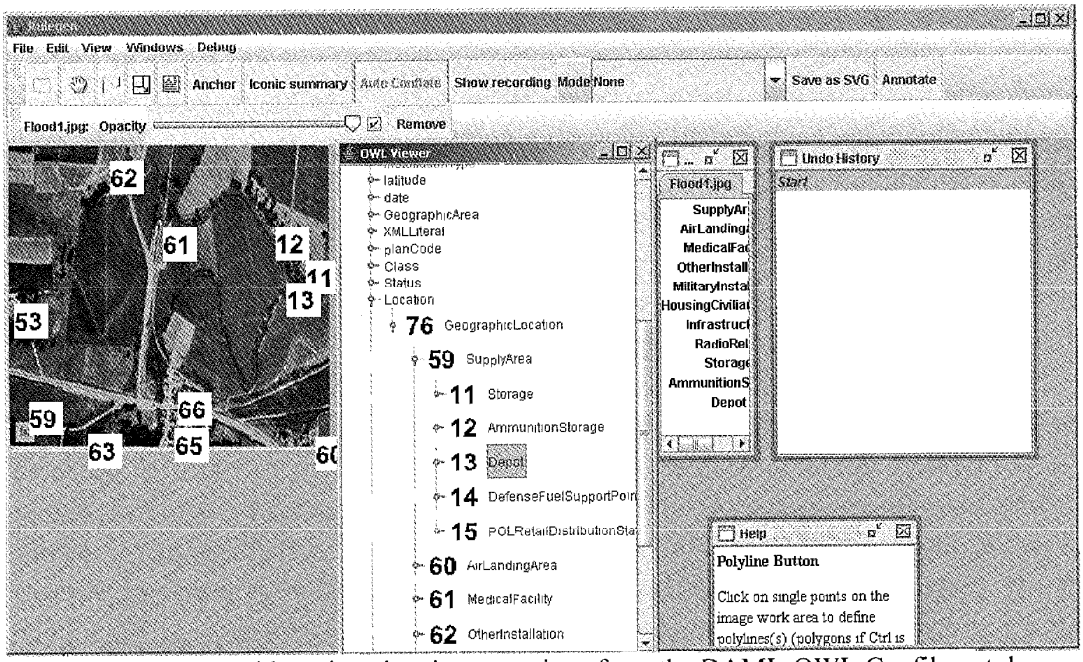


Figure 11. Image with various iconic annotations from the DAML-OWL Geofile ontology (in the middle).

The next step is to record the annotated image 1. The user can store annotated images and after that ART-MIKE software no longer has just a raster image to work with, but knows the location of various key features via iconic annotation. Finally the annotation is committed to a database where it can be later pulled for reference data. Person A's goal for this image is done.

Now we move onto person B. Person B just received a photo of a flooded area (shown in Figures 12 and 13 next to the first image), and needs to know what was beneath that water area.

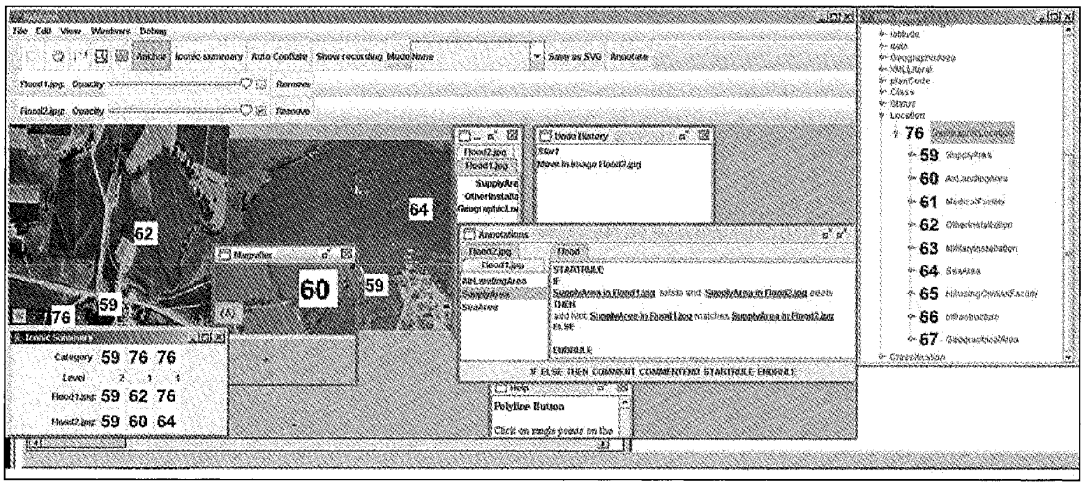


Figure 12. Annotated images and iconic conflation

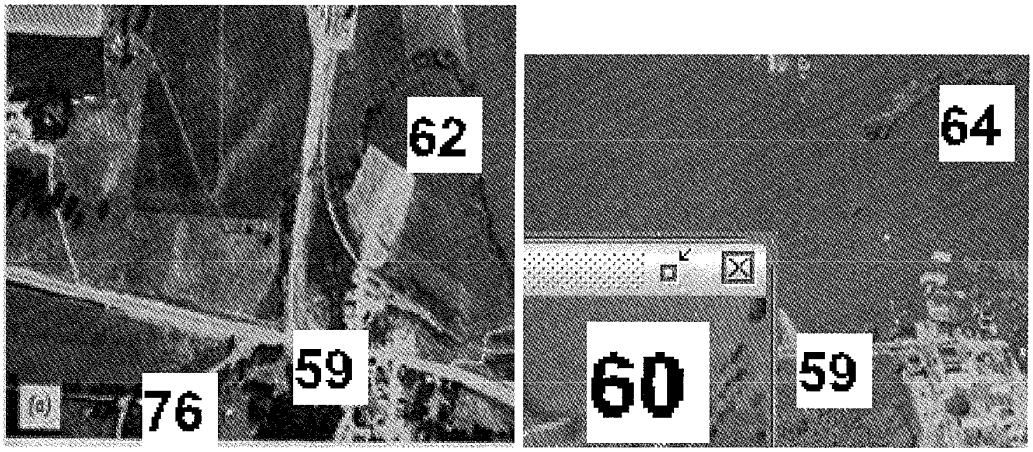


Figure 13. Loaded annotated images

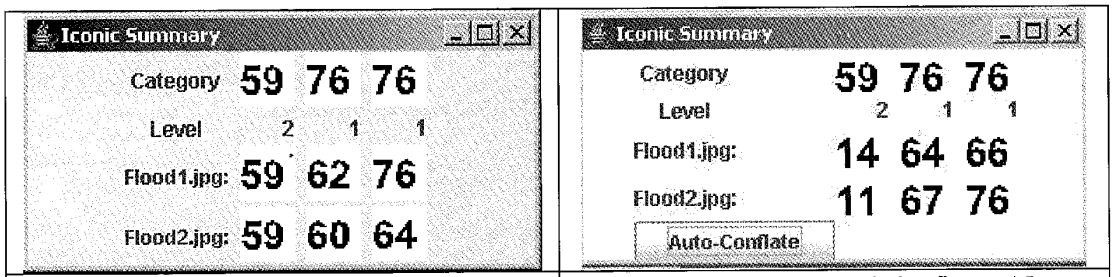
These steps show the completion of one task (annotating images) and the start of another task (**conflating images**). Expert requests the system for a suggestive match. The software tries to determine a match based on the similarities of the annotations in each image and shows the results for the expert to evaluate.

An automated approach to match up iconic annotations from one image to another is based on similarities of ontological concepts in the ontology

tree. The similarity of two annotations (annotated features), F_1 and F_2 is measured by *upper matching category (UMC)* in the ontology that is:

(1) the node itself if both nodes are the same,
(2) one of the nodes, if one of the nodes is ancestor for another, and
(3) a closest ancestor for both annotations otherwise.

For instance, in the iconic summary in the bottom in Figure 12 (see also Figure 14 (a)) concept #62 (other installation) and concept #60 (air landing area) have concept # 76 (geographic location) as their closest ancestor. Concepts #76 and #64 (sea area) have #76 as a common closest ancestor and concept # 59 (supply area) appeared in both images has itself as UMC. The match level for each pair of these iconized concepts is the level of UMC in the ontology tree. For instance, node #76 is a root (level 1) and node 59 belongs to the next level 2.



(a) Ontological match for figure 13 (b) Ontological match for figure 15

Figure 14. Matching ontological icon categories

Figure 15. Loaded images marked up differently

Thus, we have three matched points (identified by icon matched icon locations) in two images and an affine transform can be run to conflate them.

Now we can assume that two other persons C and D marked the same images as shown in Figure 15 below. Figure 14 (b) shows ontological match of these mark ups and Figure 16 shows the result of their conflation using these match.

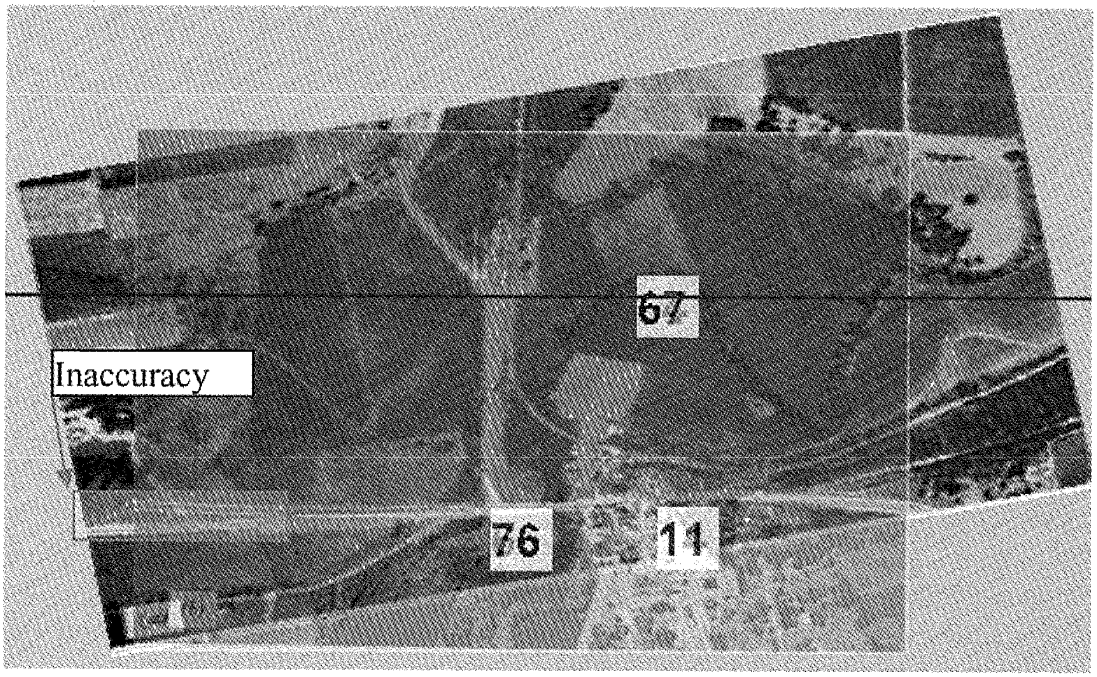
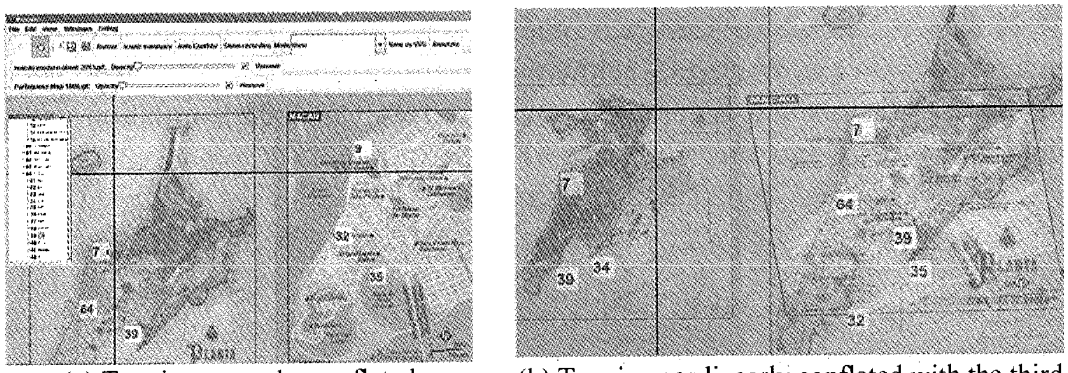


Figure 16. Result of affine conflation based on ontological match and its accuracy

7.2 Historic maps case

The same iconic ontological approach have been used to conflate historic maps using the same DAML-OWL Geofile ontology with terms: City (7), Operating Area (9), Bay (32), Port (35), Dock (39) AND Sea area (64). As can be seen below simple manipulation of a bitmap might not be enough to get everything matched up. In these cases the maps are hand drawn and are horribly inaccurate. Attempts on using scaling, translation, and rotation will fail, but ontological matching can be still correct. Figure 17(a) shows two images to be conflated: (1) modern Macau map, 2003 (on the left) and historic Portuguese map, 1889, on the right from the collection of the Library of the Congress [Macau, 2003]. Figure 17(b) shows that two images conflated and the third map is not conflated yet with two already conflated. This figure indicates the need for non-linear transformation.



(a) Two images to be conflated

(b) Two images linearly conflated with the third image on the left to be conflated

Figure 17. Two images to be conflated [Macau, Library of the Congress, 2003]. See also color plates.

7.3 Algorithm for finding best ontological match for conflation

The algorithm consists of seven steps:

Step 1. Collect all icons that mark up images to be conflated.

Step 2. Identify ontology terms {t} that matched to icons and location of terms in the ontology tree for found icons.

Step 3. Compute matrix M = $\{m_{ij}\}$, where m_{ij} is a similarity measure between terms t_i and t_j in the ontology tree. Values m_{ij} are computed for every pair of terms (t_i, t_j) where each term t_i is from image 1 and each term t_j is from image 2.

Step 4. Finding the smallest element for every row of matrix M and ordering the set of these elements, L.

Step 5. Select three smallest elements from L and test that they are from different rows and columns (DRC) in M. If this is the case then it is considered that the best match has been found between terms in two images. For instance, let these three elements be m_{23}, m_{45} and m_{17} then matched terms are (t_2, t_3), (t_4, t_5) and (t_1, t_7) and locations of respective icons are used for identifying matching affine transform between images.

Step 6. If the first three elements fail DRC test then test other triples until DRC triple will be found or the set L is exhausted.

Step 7. If W is a set of DRC triples found then find a triple $<m_{ij}, m_{dk}, m_w>$ with the smallest sum $m_{ij} + m_{dk} + m_{wv}$ in comparison with other triples. This triple is called a conflation solution.

The measure of similarity m_{ij}, is computed in two alternative ways: (1) as an *upper matching category (UMC)* that is in essence the level of the closest ancestor (see description in section 7.1) and (2) as a *number of nodes between terms* in the ontology tree. We prefer (1) because the deeper match

level then the higher chances that the match is not accidental. The same number of intermediate nodes used in (2) can be in both shallow and deep match.

8. CONCLUSION

In moving toward the goal of preserving human expertise in imagery analysis and capturing non-formalized context, virtual expert tools have been developed to assist knowledge engineers and image analysts in populating the knowledge base of the virtual expert. The first tool records an imagery analyst's actions on the fly, assists the analyst in marking up imagery with iconized ontology terms and provides an ontological image conflation. The second tool generates expert rules by questioning the imagery analyst and minimizing questioning time using the theory of Monotone Boolean functions. These tools are implemented as a web portal using Java.

Future work for a knowledge engineer includes developing tools for discovering imagery analysis rules, implement tools for recording conflation process, and developing tools to apply imagery analysis rules to conflation.

9. ACKNOWLEDGEMENTS

This research has been supported by a University Research Initiative grant from the US Geospatial-Intelligence Agency that is gratefully acknowledged.

10. EXERCISES AND PROBLEMS

1. Select two aerial photos from the web of the same area but with different spatial resolution. Design an iconic annotation for these images and provide justification for your chose of icons, spatial features and ontology terms.

2. Build an ontology that will fit images you used in exercise 1. It should be a tree with three or more levels.

3. Build a sequence of questions based on four binary parameters x_1, x_2 x_3, x_4 starting from the simplest question that contains only parameter x_1 to be loaded to the knowledge extractor MIKE.

Tip: MIKE accept s binary vectors; the question with x_1 can only be encoded as 1000. Justify your sequence assuming monotonicity of expert's answers.

11. REFERENCES

DAML GeoFile ontology, http://www. daml.org/ 2001/02/geofile/geofile-ont , 2001

DAML WFB ontology. http://ontolingua.stanford.edu/doc/chimaera/ontologies/world-factbook.daml, 2002

Edwards D., Simpson J. Integration and access of multi-source vector data, In:Geospatial Theory, Processing and Applications, ISPRS Commission IV, Symposium 2002, Ottawa, Canada, July 9-12, 2002, http://www.isprs.org/ commission4/proceedings/ pdfpapers/ 269.pdf

FACC, NIMA Feature and Attribute Coding Catalogue (FACC), USIGS-CDM-E, 2001. http://www.usgs.gov

FGDC, Federal Geographic Data Committee FGDC-STD-999.1-2000, NSDI Framework Transportation Identification Standard -- Draft. www.bts.gov/gis/fgdc/introduction.pdf

HPKB, High-Performance Knowledge Bases, DARPA, 1996, http://www.darpa.mil/iso/documents/hpkb/baa96-43pip.html

Kovalerchuk, B., Triantaphyllou, E., Despande, A.S, and Vityaev, E. Interactive Learning of Monotone Boolean Functions, Information Sciences, Vol. 94, issue 1-4, 1996, pp. 87-118.

Kovalerchuk, B., Vityaev E., Ruiz J.F., Consistent and Complete Data and "Expert" Mining in Medicine, In: Medical Data Mining and Knowledge Discovery, Springer, 2001, pp. 238-280.

Lillesand T., Kiefer, R., Remote sensing and image interpretation, John Wiley and Sons, NY., 1987.

Macau: A selection of cartographic images, http://lcweb2.loc.gov/ammem/gmdhtml/macau /macau.html, . The Library of Congress, 2003

RKF, Rapid Knowledge Formation, DARPA, 1999, http://www.darpa.mil/iso/RKF/ KFOverview.ppt

USGS, Digital Elevation Model Standards, USGS, 1998, http://rockyweb.cr.usgs.gov/nmpstds/demstds.html

VLKB, Very Large Knowledge Bases, 1998, http:// www. cs.umd.edu/projects/plus/ Parka/parka-kbs.html

Color Plates

Chapter 2

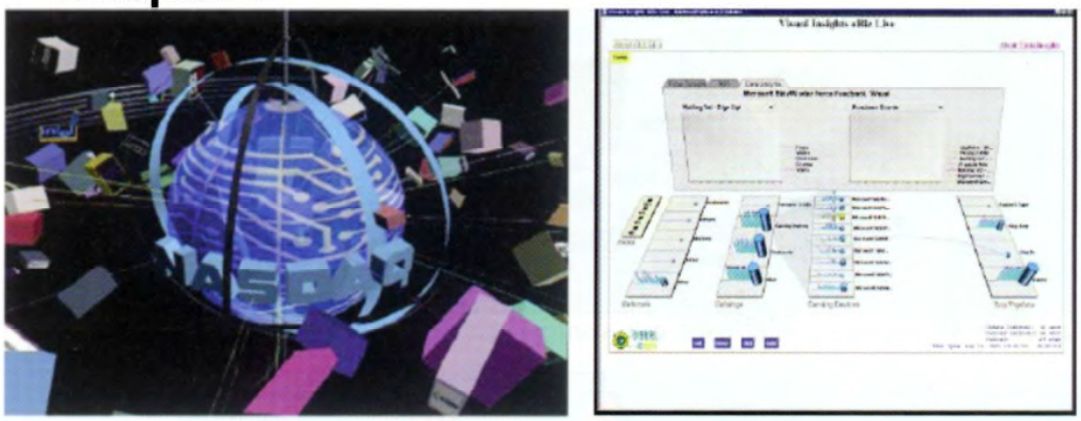


Figure 1. Information visualizations for presentation and branding. Left NASDAQ display and Right: Visual Insights' eBizLive product for showing website activity

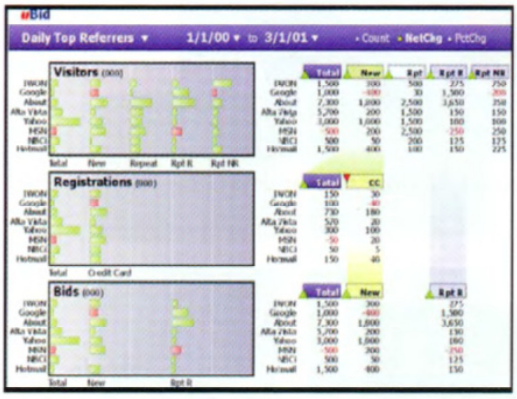


Figure 2. Executive Dashboard courtesy of Bill Wright.

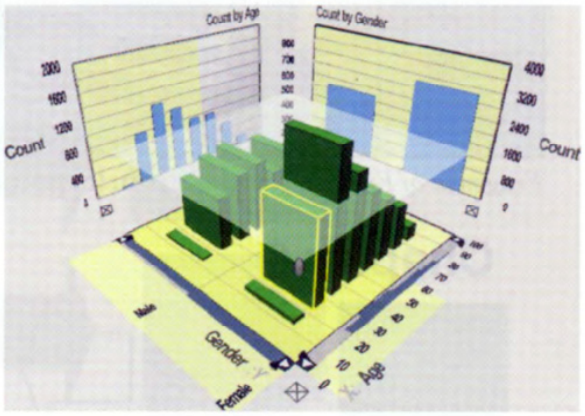


Figure 3. Real-time 3D Visual Report courtesy of Visual Insights

Figure 4. Advizor 2000 Visual Discovery and Analysis Tool

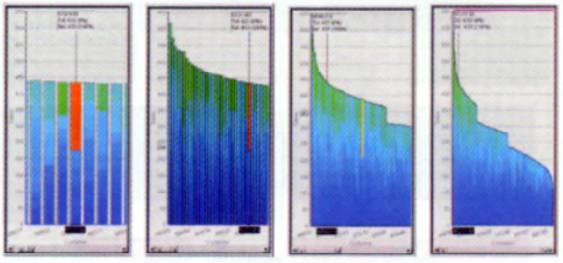


Figure 5. Bar chart scalability is increased by using levels of rendering detail and a red overplotting indicator at the top of the view. Scalability in this case facilitates locating and then focusing attention on particular bars.

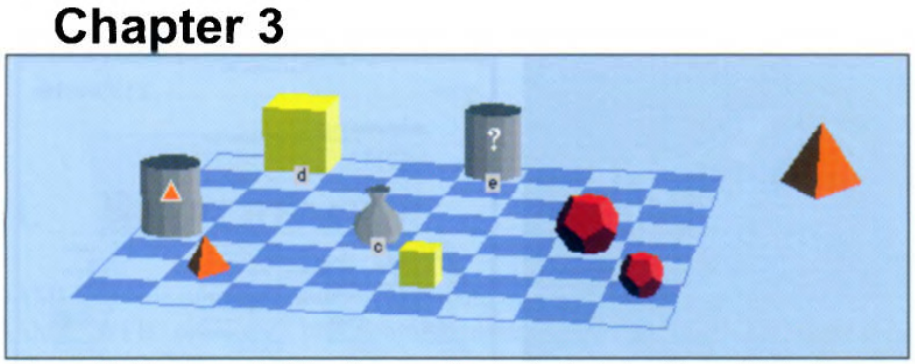
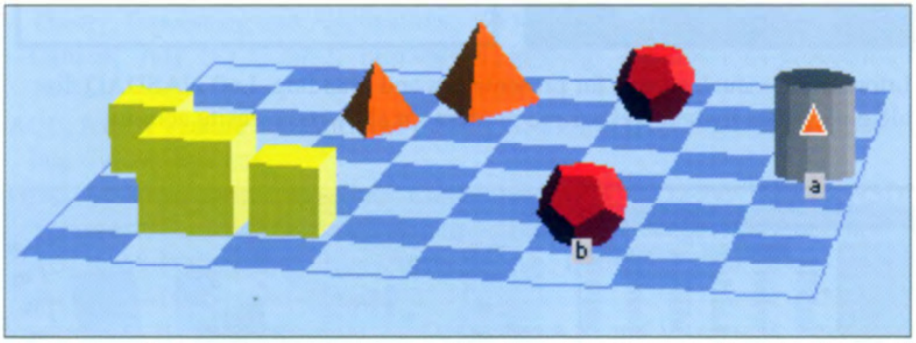


Figure 11. Hyperproof [http://www-csli. stanford.edu/hp/Hproof2.html]



Fragment of Figure 12. Task diagram [http://www-csli.stanford.edu/hp/Hproof3a.html]

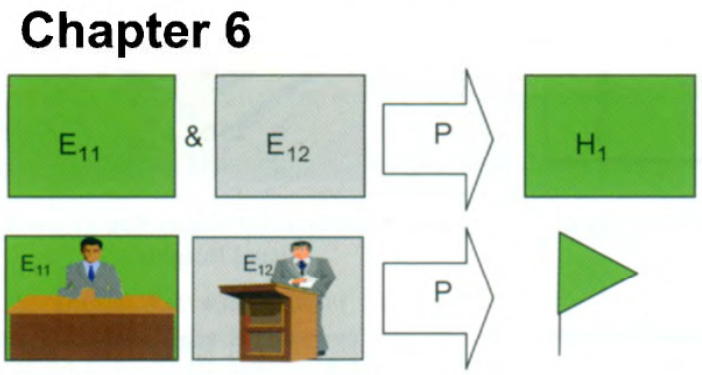


Figure 2. Traditional and iconic visualizations of rule R_1

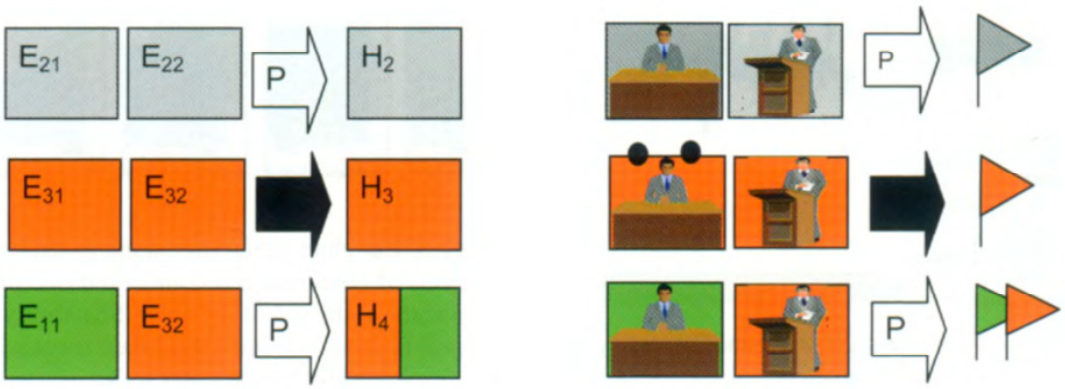


Figure 3. Traditional and iconic visualizations of rules

Fragment of Table 3

Fragment of Tables 4 and 8.

	Legend 1	*Legend 2*
H_1		
H_2		
H_3		
H_4		

	Icon		*Icon*
E_{11}		E_{32}	
E_{12}		E_{41}	
E_{21}		E_{42}	
E_{31}			
E_{111}		E_{112}	

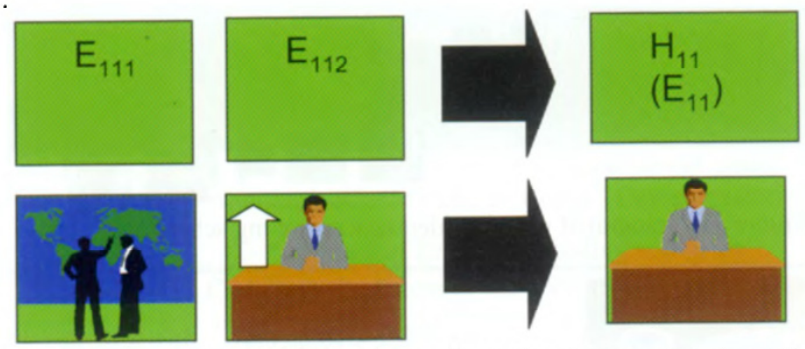


Figure 4. Comparison of two visual reasoning alternatives

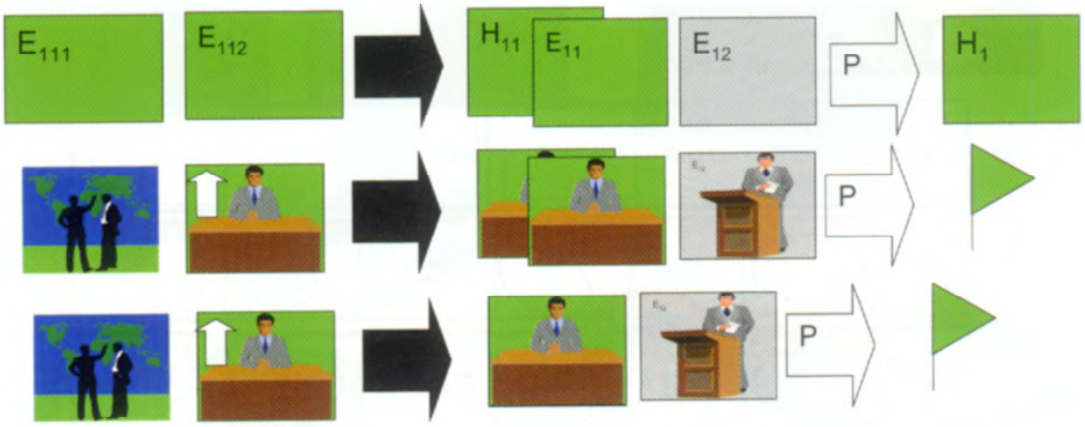


Figure 5. Reasoning chains

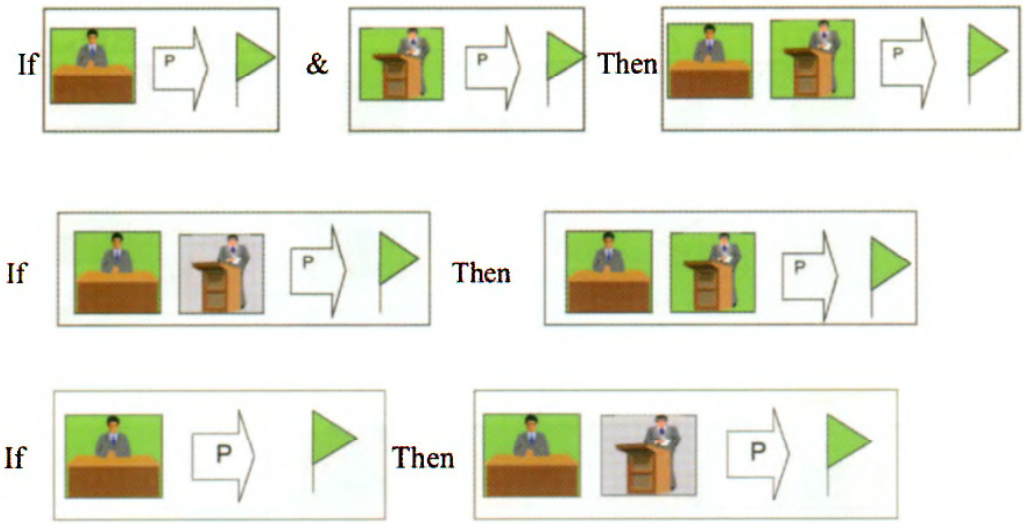


Figure 6. Visual reasoning rules

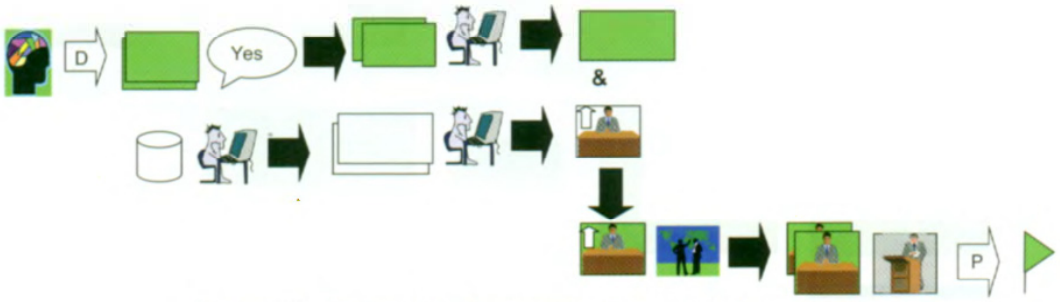


Figure 11. Integrated visual evidentiary reasoning scheme

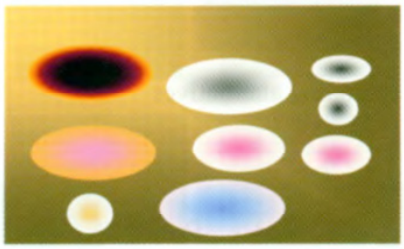


Figure 12. Signal ellipses

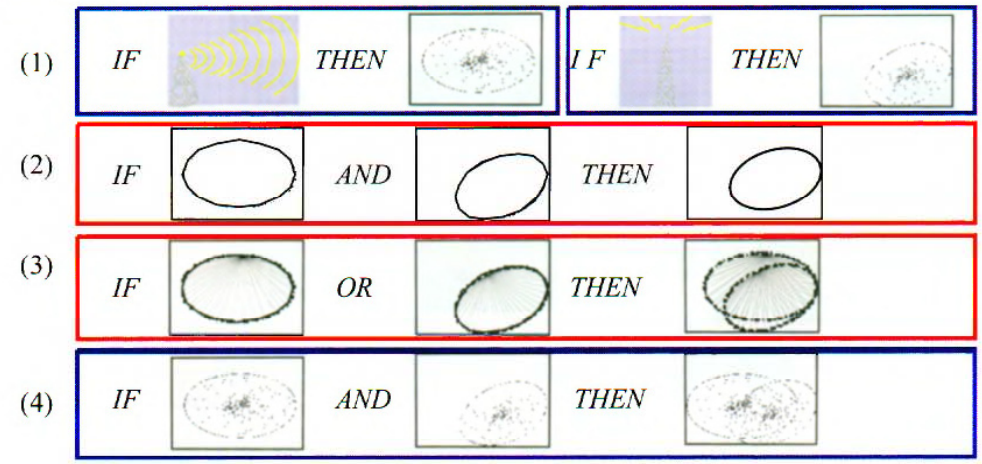


Figure 13. Visual rules with signal ellipses

Chapter 8

Figure 1. Pieter Bruegel's painting "Blue cloak" (1559), oil on oak panel, 117 x 163 cm. (with permission from Staatliche Museen zu Berlin - Gemaldegalerie, Berlin)

Fragment of Table 1. Encoding text in art

Compressed content of the text: metaphor proverb and icon		
Big fish eats little fish 	He catches fish with his hands 	Merged picture: Big fish eats little fish while he catches fish with his hands 

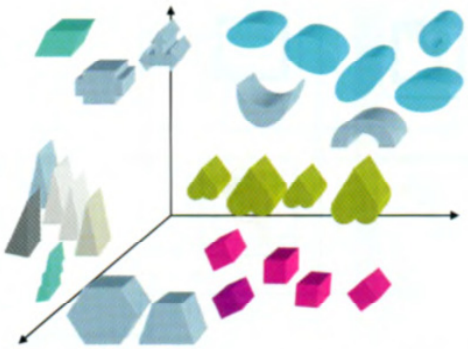


Figure 8. Examples of low-level visual correlations based on glyphs

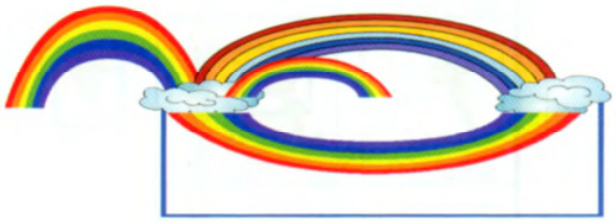


Figure 10. Rainbow correlation

(a) Magic lens (b) Linked panels

Figure 12. High-level geospatial visual correlation for objects of different levels of resolution.

Chapter 9

Table 8. Examples of Bruegel composite icons

Base icons			*Composite icons*	
Truck	Damage	Uncertainty	Damaged truck.	Truck damage Questionable

	message	template	security	ID
message				
template				
security				
ID				

Table 10. Composite icon generation.

Chapter 10

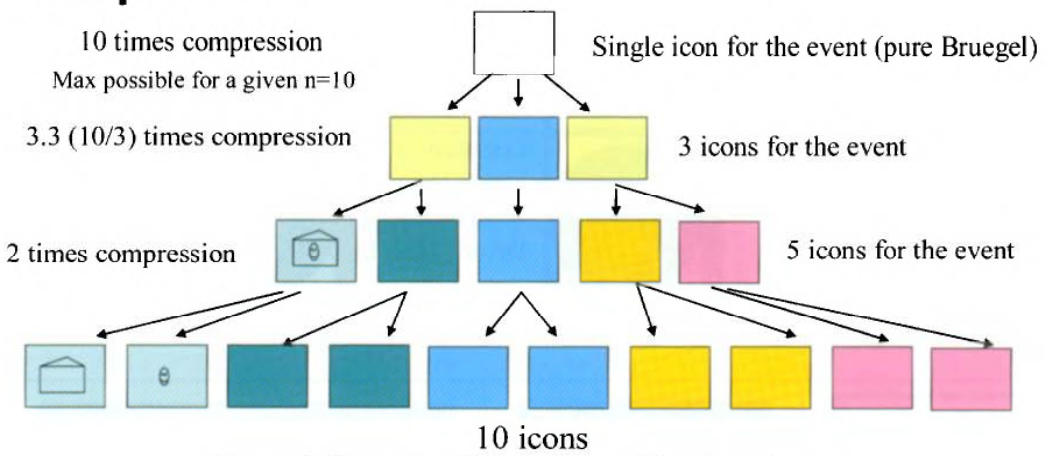


Figure 3. Dynamics of compression of iconic sentence.

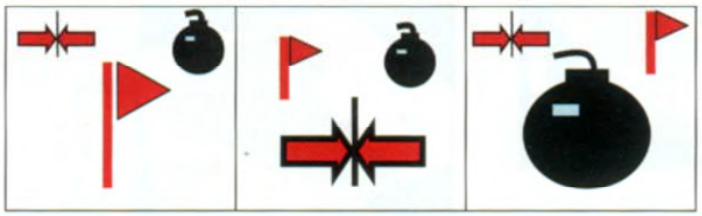


Figure 6. Icons with user defined weighting

(a) The slashes across the lower right indicate that this is a small band (blue scale) of armed rebels (or terrorists).

(b) This icon describes a fairly large number of civilian targets (blue scale) were hurt pretty badly (red scale) by some action.

Figure 9. Possible icons for MUC concepts

	the basic frame with red relevance scale and		civilian		terrorist (without the gun he can be just a rebel)
	target modifier (iconel)		stage of execution (accomplished)		stage of execution (threatened)
	perpetrator, highly relevant (yellow mark on green)		land vehicle		land transport vehicle

Figure 10. Bruegel icon examples: base icons

Location: icon type 1. Red shows country's location.	Location: icon type 2. Lens shows a small country.	location: icon type 3.The flag indicates a country.
Location icon with city modifier	Location icon with town modifier	Location icon with department modifier

Figure 14. Bruegel icon examples: location icon types

Figure 15. Bruegel case studies.

Chapter 11

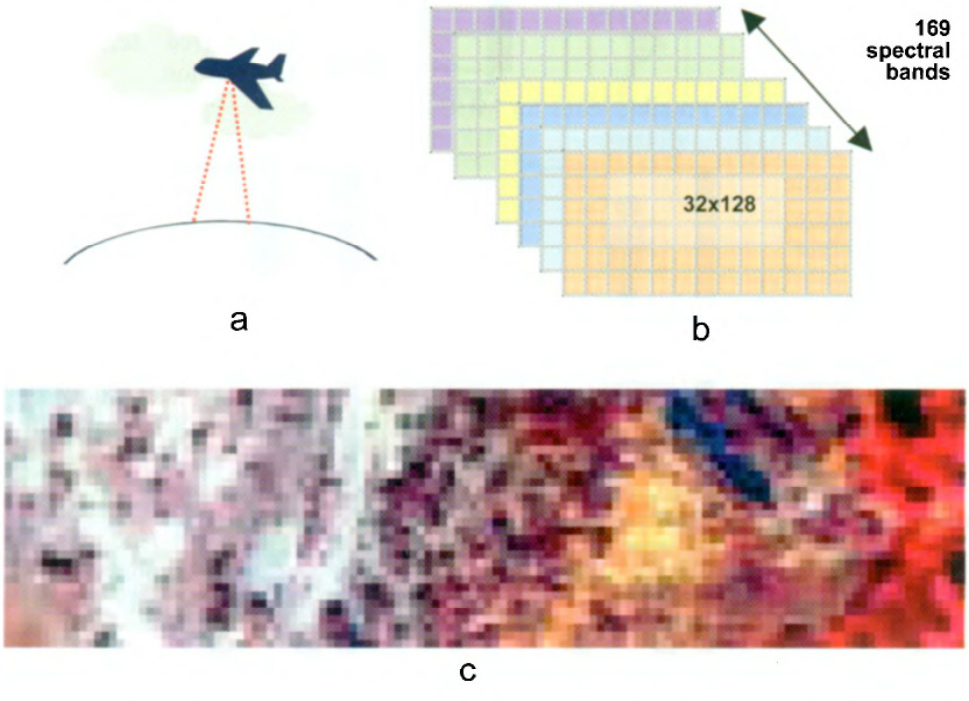


Figure 1. a) An illustration of the operation. b) A sketch of a hyperspectral image set with 169 spectral bands ranging from very short to very long bands. c) A color infrared (CIR) imagery of the semi-desert area in Eastern Washington.

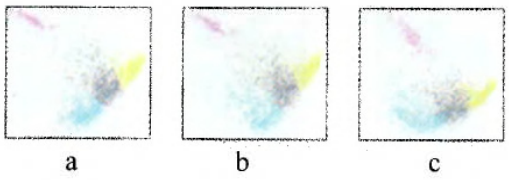


Figure 5. Scatterplots generated by MDS using document vectors with sizes equal to a) 200, b) 100, and c) 50 terms.

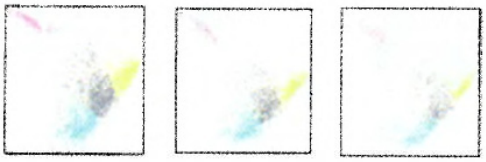


Figure 6. Scatterplots generated by MDS using a) 3298, b) 1649, and c) 824 document vectors.

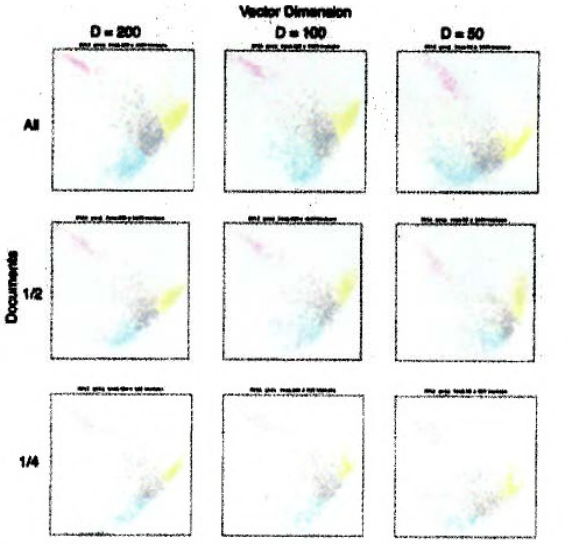


Figure 7. A scatterplot-matrix demonstrates the impact of reducing document vectors (row) versus reducing vector dimensions (column) using classical scaling technique.

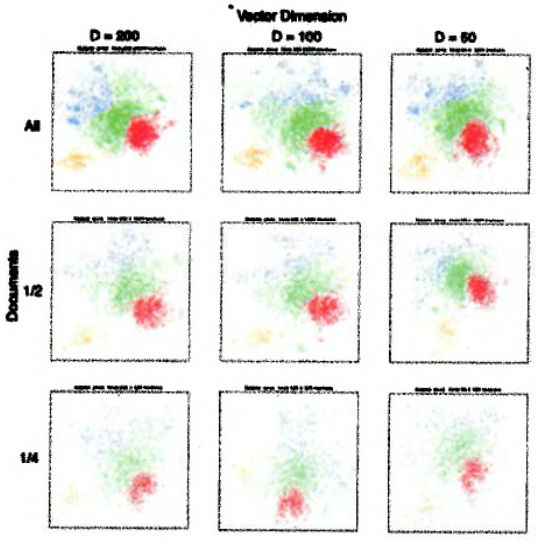


Figure 8. A scatterplot matrix demonstrates the impact of reducing document vectors (row) versus reducing vector dimensions (column) using the Sammon Projection technique.

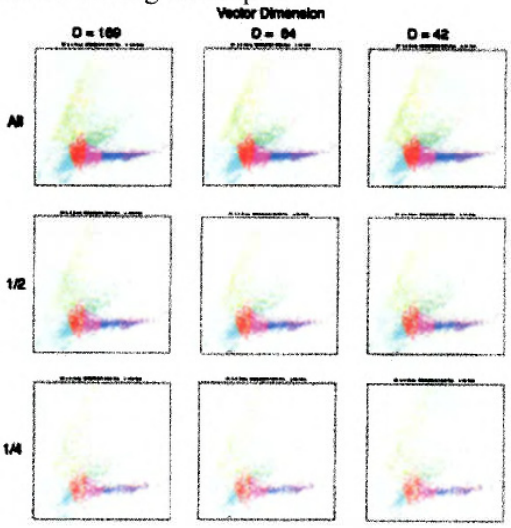


Figure 9. A scatterplot matrix demonstrates the effects of reducing pixel vectors (row) versus reducing vector dimensions (column) using remote sensing imagery.

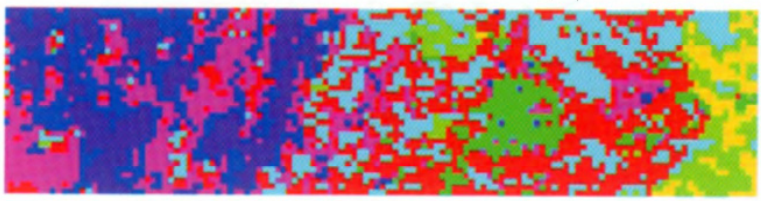


Figure 10. Colors generated by the scatterplot clusters clearly identify different features of the images shown in Figure 1c

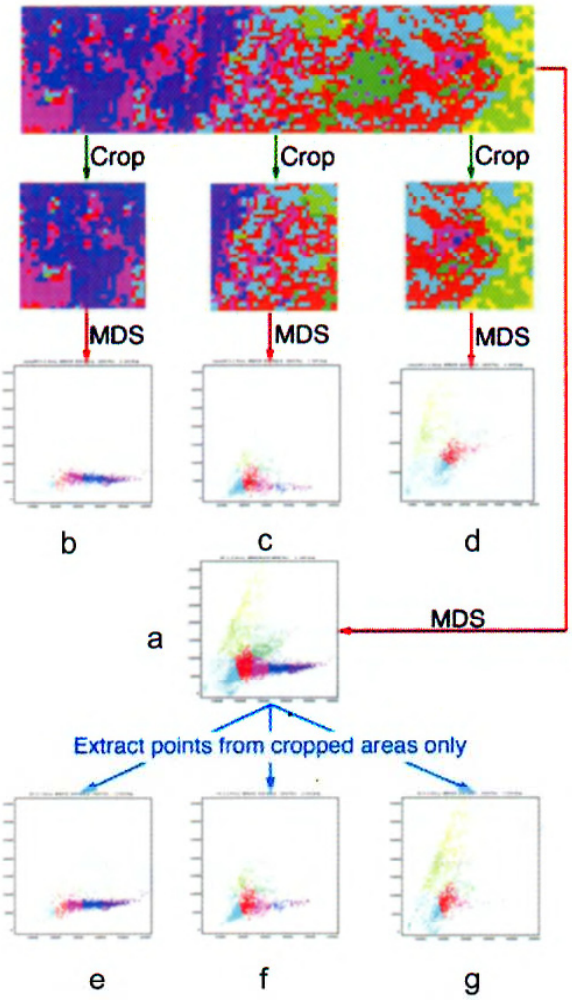


Figure 12. Scatterplots.

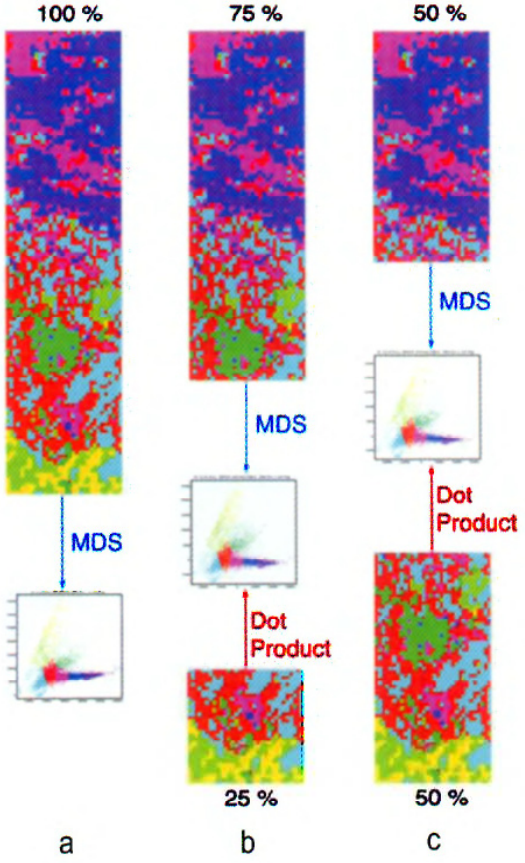
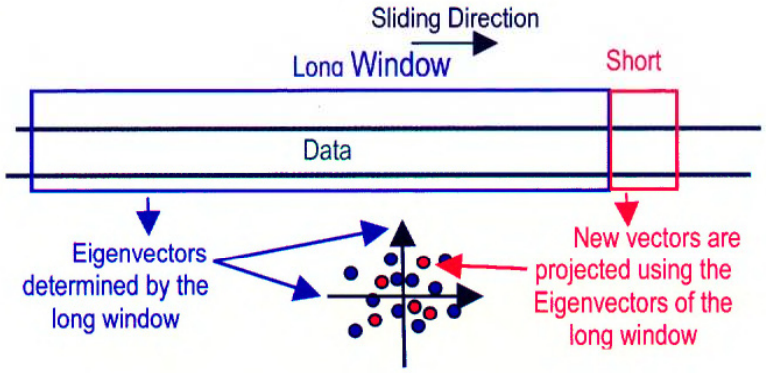


Figure 14. Eigenvectors of the scatterplots

Figure 13. An illustration of our multiple sliding window design in visualizing data streams



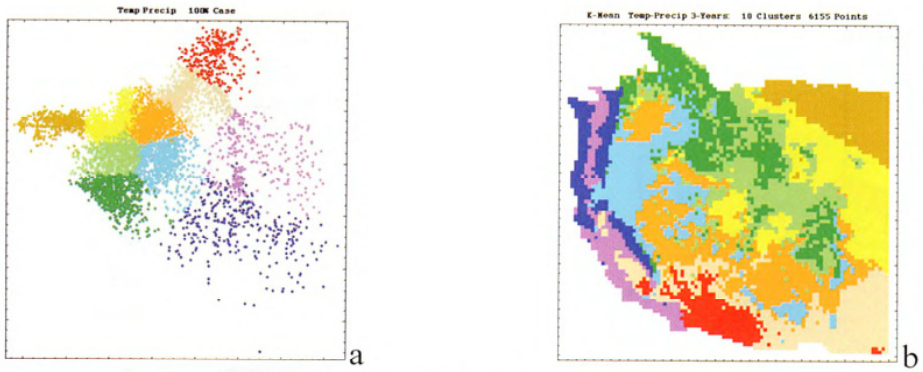
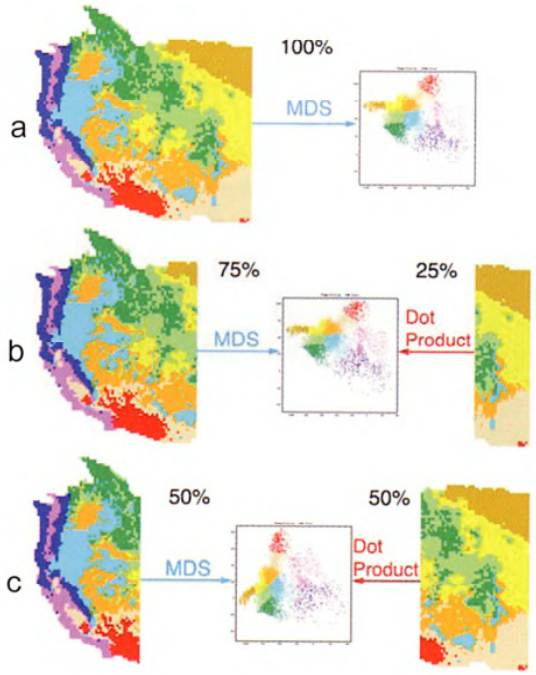


Figure 15. a) A scatterplot of 6,155 hydroclimate vectors divided into 10 clusters. Each cluster is represented by a unique random color. B) Corresponding cluster colors are projected to the map position

Figure 16. The Eigenvectors of the scatterplots



Chapter 14

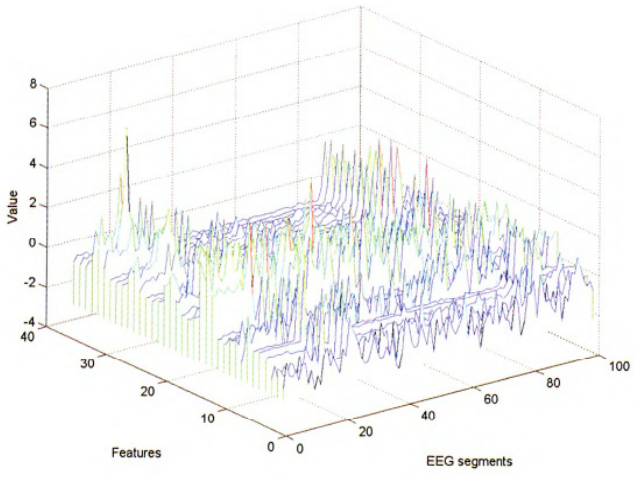


Figure 4. Fragment of EEG containing 100 segments presented by 36 features in which the EEG-viewer recognized five artifacts.

Chapter 15

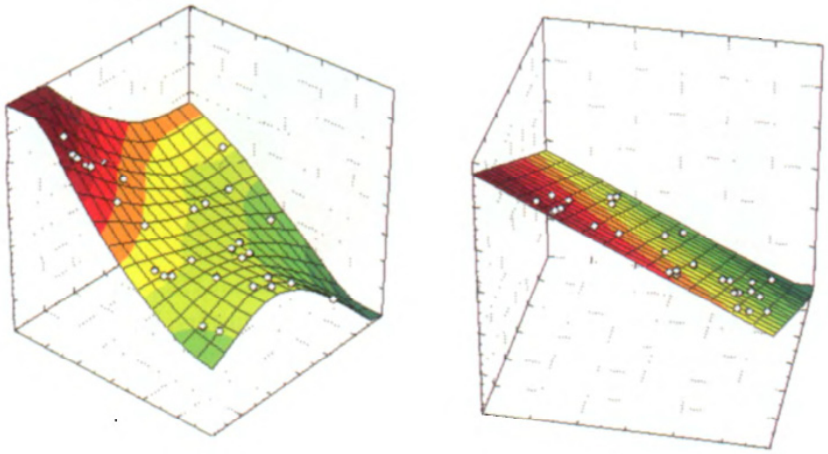


Figure 2. Data visualization: original data on the left and simultaneously rescaled data on the right

Chapter 16

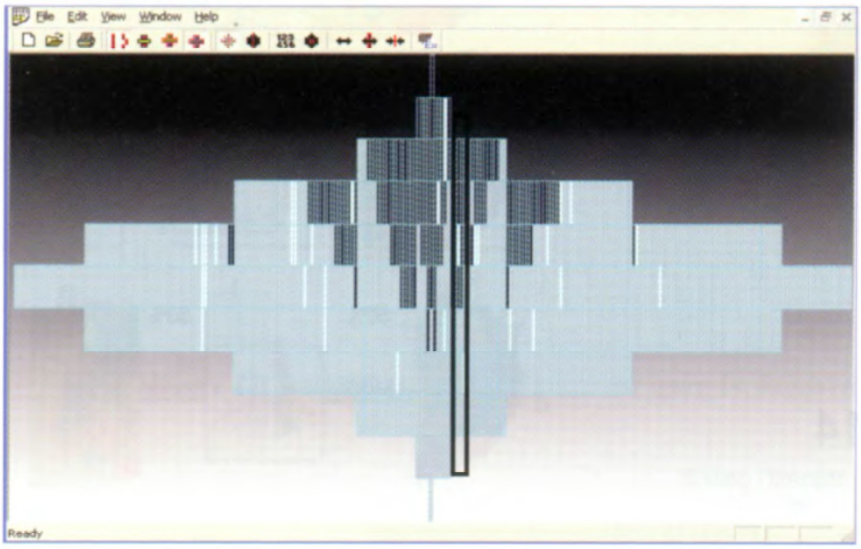


Figure 11. Breast cancer cases visualized using procedure P_3 with cases shown as bars with frames.

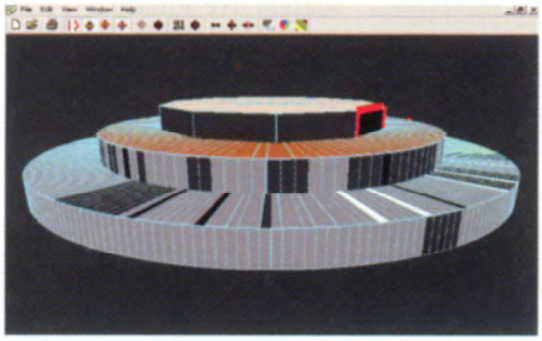


Figure 14. A 3-D version of Monotone Boolean Visual Discovery with vertical and horizontal surfaces used.

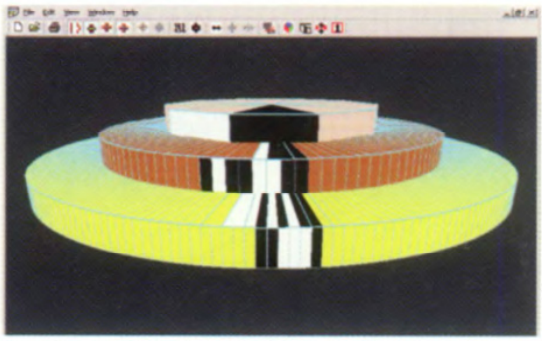


Figure 15. A 3-D version of Monotone Boolean Visual Discovery with grouping Hansel chains

Chapter 18

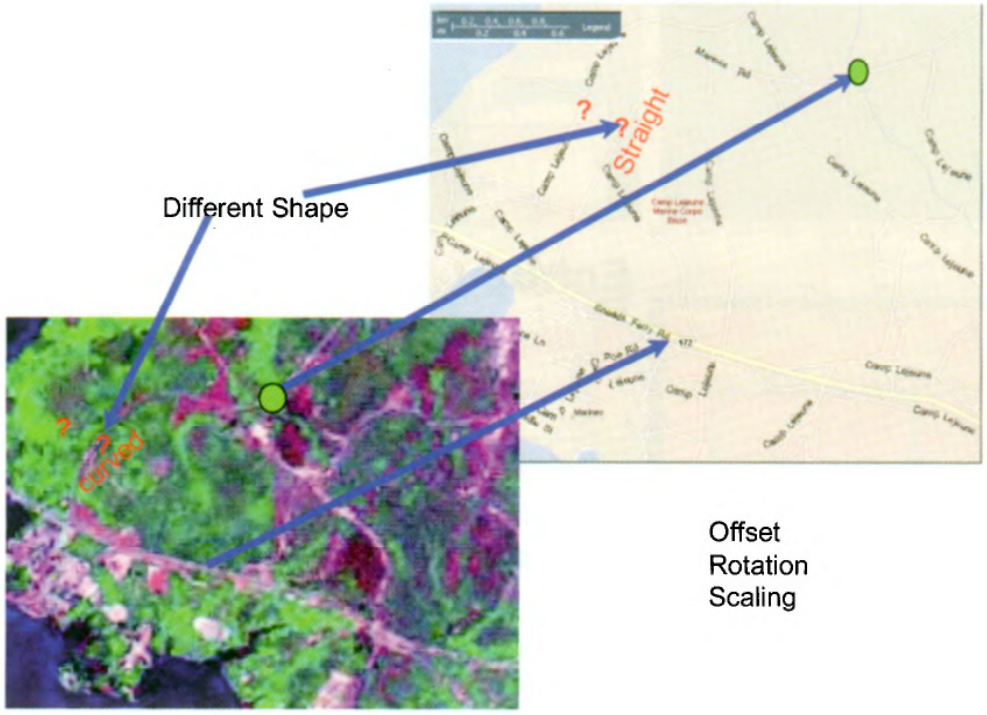
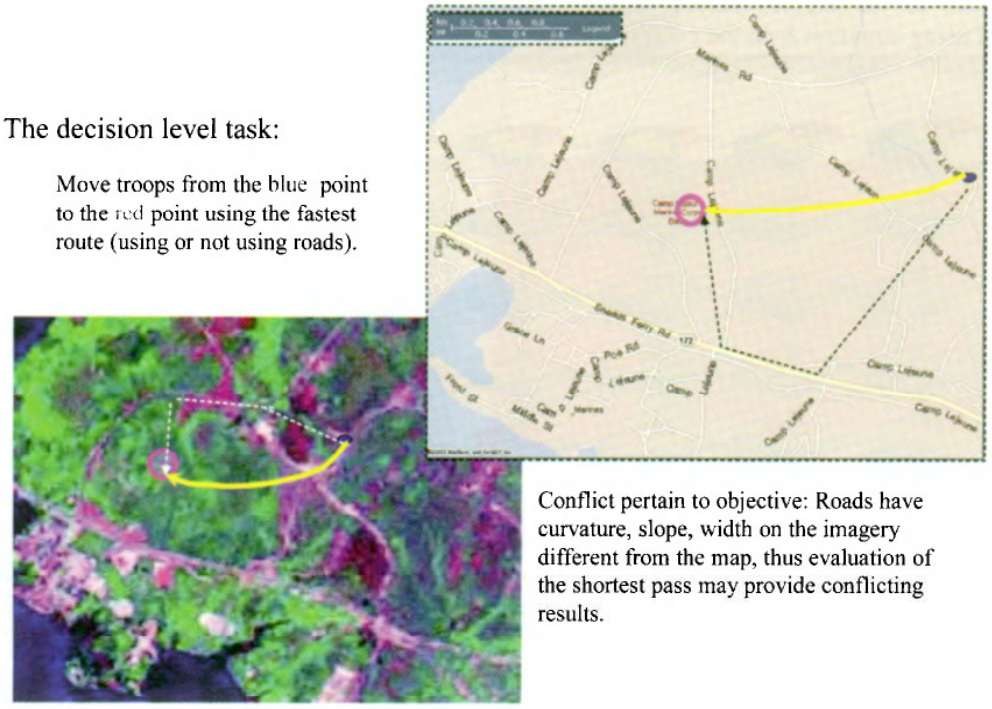


Figure 2. Local inconsistency between imagery source and vector product

The decision level task:

Move troops from the blue point to the red point using the fastest route (using or not using roads).



Conflict pertain to objective: Roads have curvature, slope, width on the imagery different from the map, thus evaluation of the shortest pass may provide conflicting results.

Figure 5. Decision level conflation problem

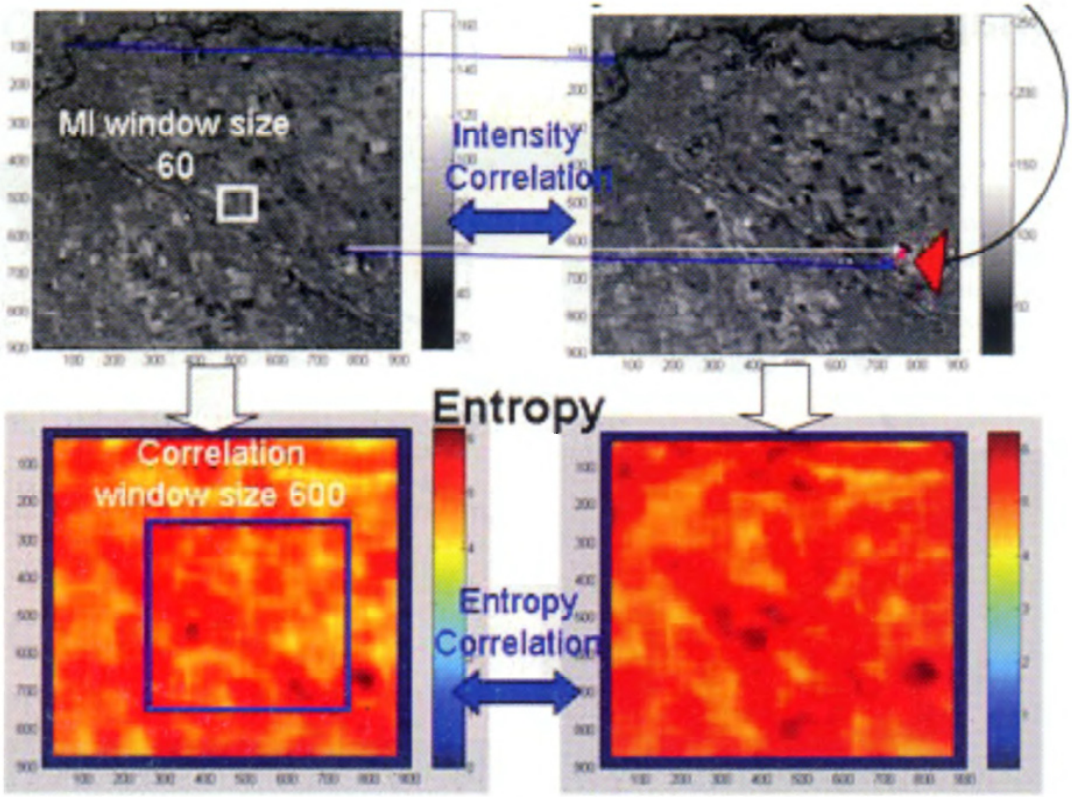


Image (a) Image (b)

Figure 16. Illustration of offset, scaling, and rotation phenomenon at the scale of warping; i.e., 100s pixels. Image (a) and (b) or the time lapsed image pair. Blue vector maps corresponding river bend features. White vector maps the lake. The dark vector maps the registration vector for the selected features. The lower two figures are the displays of entropy transformed using equation 8 on the corresponding intensity imagery pair.

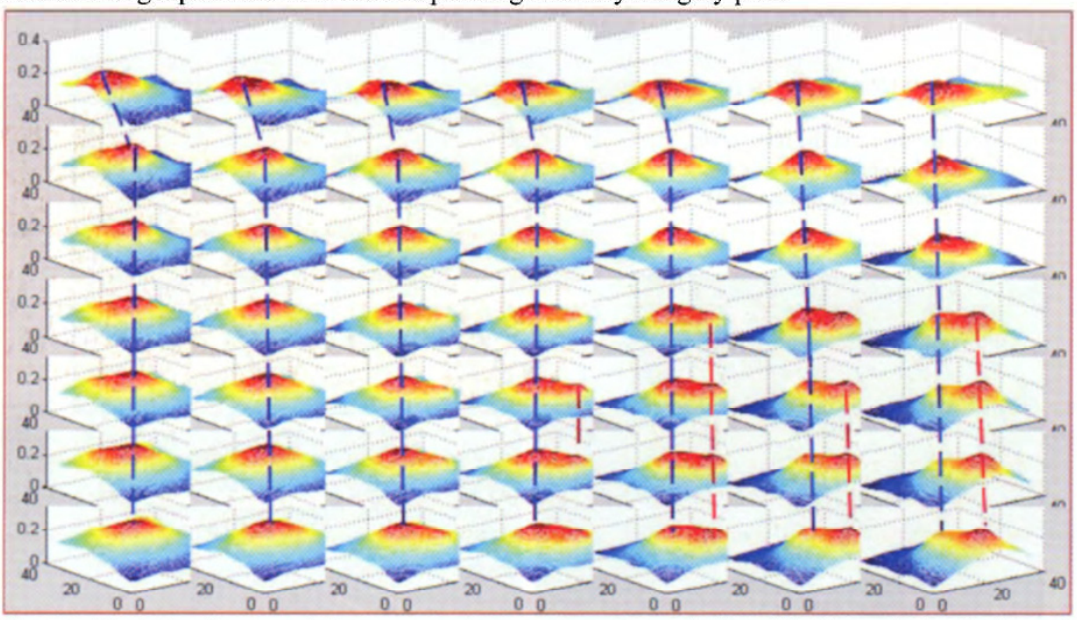


Figure 18. Intensity correlation non-unique and wrong solution. Radiance intensity-based correlation from total of 49 locations centered on a 7x 7 grid over the image pairs in Figure 16. The connection of the correlation peaks does not have the same trend as the registration vector displayed in Figure 16. See also color plates.

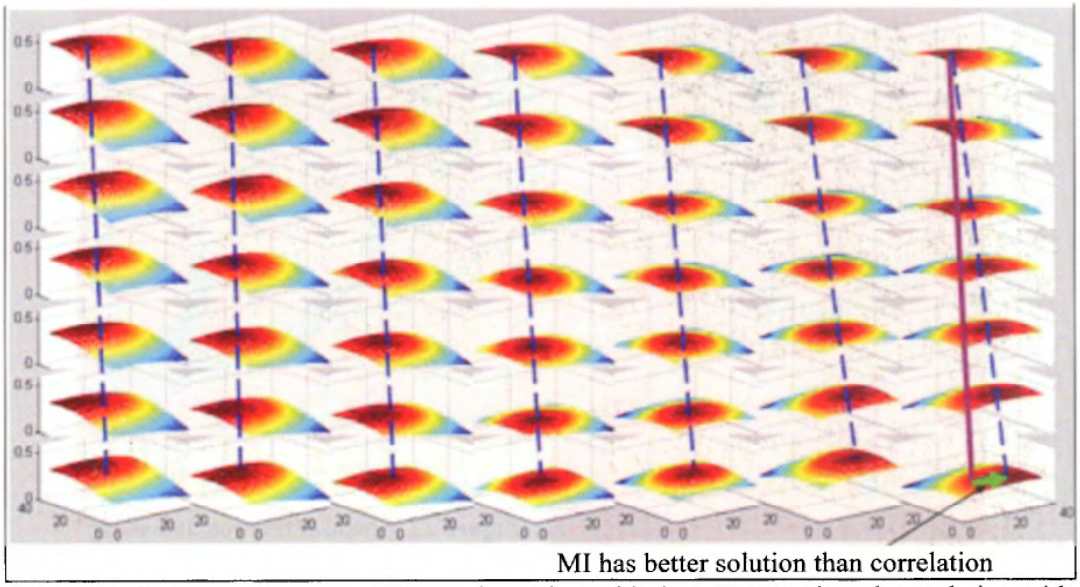


Figure 19. Entropy correlation. The trend matches with data. Entropy-based correlation with the same lay out in 18. The connection of the of the correlation peaks does have the same trend as the registration vector from Figure 16. See also color plates.

Chapter 19

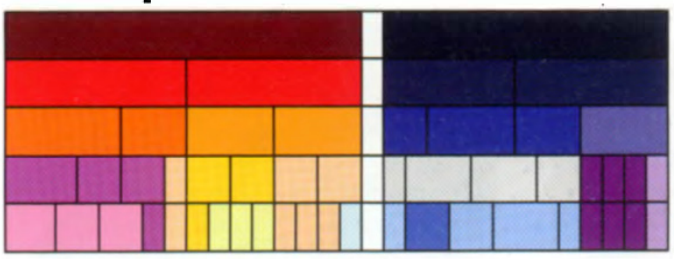
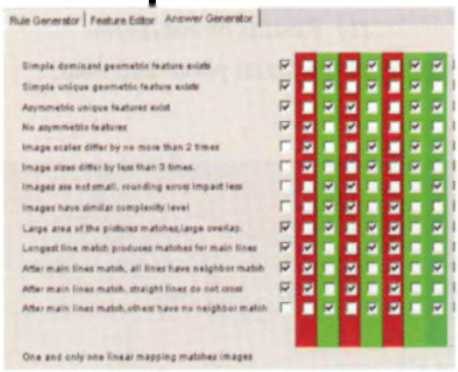


Figure 8. Illustration of structural lengths

Chapter 20



Fragment of Figure 4. Expert questioning using monotonicity principle

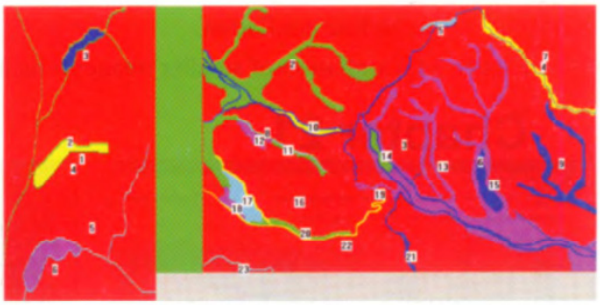


Figure 9. Two images at shape extraction stage. One of the images is scaled disproportionally.

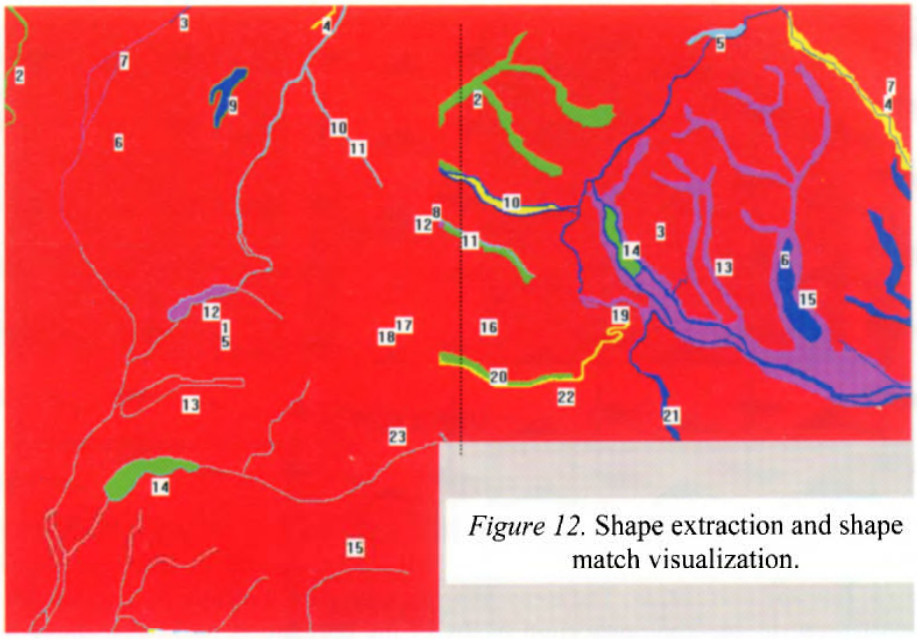
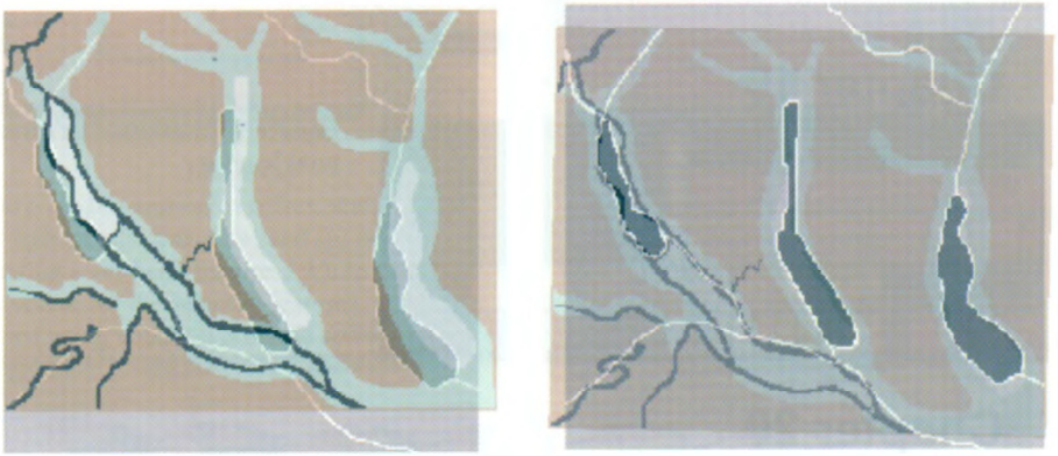


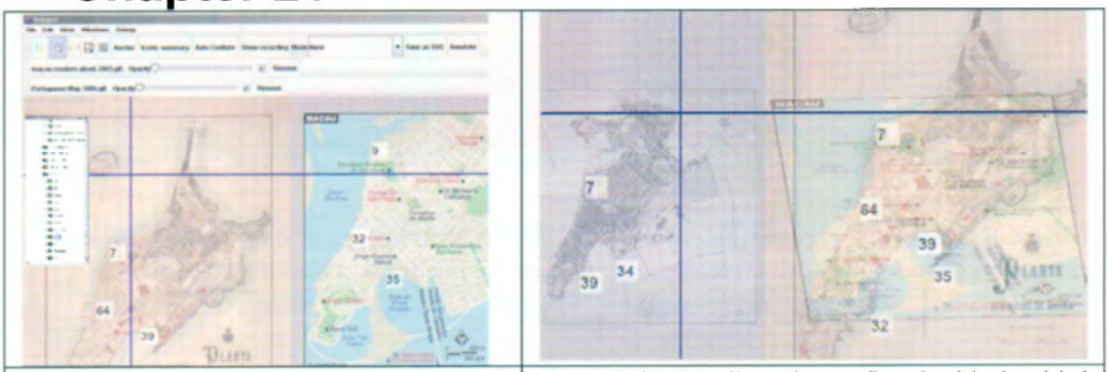
Figure 12. Shape extraction and shape match visualization.



(a) Before conflation. *(b)* Results of conflation

Figure 14. Conflation of smaller images which are parts of the aerial photo and map.

Chapter 21



(a) Two images to be conflated

(b) Two images linearly conflated with the third image on the left to be conflated

Figure 17. Two images to be conflated [Macau, Library of the Congress, 2003]