Technical Report 1420

Attentional Selection In Object Recognition

S. Tanveer F. Mahmood

MIT Artificial Intelligence Laboratory

Attentional Selection in Object Recognition

by

S. Tanveer F. Mahmood

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0010, under Office of Naval Research contract N00014-91-J-4038 and under NSF Grant IR1-8900267. The author was supported by an IBM Graduate Fellowship during part of the thesis work.

Abstract

A key problem in object recognition is selection, namely, the problem of identifying regions in an image within which to start the recognition process, ideally by isolating regions that are likely to come from a single object. Such a selection mechanism has been found to be crucial in reducing the combinatorial search involved in the matching stage of object recognition. Even though selection is of help in recognition, it has largely remained unsolved because of the difficulty in isolating regions belonging to objects under complex imaging conditions involving occlusions, changing illumination, and object appearances.

This thesis presents a novel approach to the selection problem by proposing a computational model of visual attentional selection as a paradigm for selection in recognition. In particular, it proposes two modes of attentional selection, namely, *attracted* and *pay* attention modes as being appropriate for data and model-driven selection in recognition. An implementation of this model has led to new ways of extracting color, texture and line group information in images, and their subsequent use in isolating areas of the scene likely to contain the model object. Among the specific results in this thesis are: a method of specifying color by perceptual color categories for fast color region segmentation and color-based localization of objects, and a result showing that the recognition of texture patterns on model objects is possible under changes in orientation and occlusions without detailed segmentation. The thesis also presents an evaluation of the proposed model by integrating with a 3D from 2D object recognition system and recording the improvement in performance. These results indicate that attentional selection can significantly overcome the computational bottleneck in object recognition, both due to a reduction in the number of features, and due to a reduction in the number of matches during recognition using the information derived during selection. Finally, these studies have revealed a surprising use of selection, namely, in the partial solution of the pose of a 3D object.

Thesis Supervisor: W. Eric L. Grimson

Title: Associate Professor of Electrical Engineering and Computer Science

A little learning is a dangerous thing;
Drink deep, or taste not the Pierian spring;
There shallow draughts intoxicate the brain,
And drinking largely sobers us again.
Fired at first sight with what the Muse imparts,
In fearless youth we tempt the heights of Arts,
While from the bounded level of our mind,
Short views we take, nor see the lengths behind;
But more advanced, behold with strange surprise
New distant scenes of endless science rise!
So pleased at first the towering Alps we try,
Mount o'er the vales, and seem to tread the sky,
Th' eternal snows appear already past,
And the first clouds and mountains seem the last;
But, those attained, we tremble to survey
The growing labors of the lengthened way,
The' increasing prospect tires our wandering eyes,
Hills peep o'er hills, and Alps on Alps arise!

ALEXANDER POPE

Acknowledgments

I would like to express my sincerest thanks to my supervisor Prof. Eric Grimson whose continual support and encouragement has made this thesis possible. He provided valuable guidance, shared interesting ideas, and kept me focused on the problem of selection while allowing sufficient freedom to explore my own ideas. I benefitted from his deep knowledge, experience and financial support, for which I am grateful.

I thank my other committee members, Profs. Ellen Hildreth, Shimon Ullman, and Thomas Lozano-Pérez for reading my thesis draft and providing helpful comments. The discussions I had with Ellen and Shimon during the beginning of my thesis work helped to shape some of my ideas. I also thank Ellen for her careful reading of drafts of my thesis work over the years and providing insightful comments.

During my thesis work, I benefitted from interesting technical discussions with Prof. Tomaso Poggio and Prof. Berthold Horn. My early days at MIT were spent working with Prof. David McAllester and I thank him for getting me started into research here.

I learned a lot about computer vision from my colleagues and members of the welg group Dave Jacobs, Dave Clemens, Todd Cass, Sandy Wells, Kah kay Sung, Dave Beymer, Brian Subirana, Pawan Sinha, and Ronen Basri. Thanks in particular to Dave Clemens for the Grok environment which I used extensively, to Pat O'Donnell for the many times he helped out with the color system and lisp machine bugs, and to Sandy Wells for the immense help during the photo recording of my results.

During my years at the AI lab, I enjoyed the company of Linda Wills, Anita Flynn, my office mates Bonnie Dorr, Sandiway Fong, and later Amnon Shashua, Ivan Bacheldor, and Andy Gavin. Also, Lynn Parker, Manuela Vasilescu, Holly Yanco, Tina Kapur and Aparna Lakshmiratan provided moral support and help with the new baby during the final days of thesis preparation. Jeanne Speckman provided invaluable assistance over the years and Sally Richter was very helpful at the Publications office.

Prior to MIT, Prof. Yegnanarayana at IIT Madras introduced me to signal processing and shaped my research. I also value the respect and encouragement given to me by my teachers at Osmania university.

I owe much to my parents who taught me the value of learning and whose strong belief in my capabilities encouraged me to pursue my career. Finally, I feel fortunate in having Kutub as my husband. This thesis would not have been completed without his constant support, love and encouragement.

To my parents

Contents

Chapter 1

Introduction

1.1 The Selection Problem

Exhibiting some understanding of a scene is important for robots to deal effectively with their environment. This may involve determining which objects are present in a scene or whether a particular object appears in a given scene. Both these operations may involve model-based object recognition, that is, recognizing objects from their stored model descriptions. Most of the approaches to model-based object recognition have used geometric features to recognize objects. Such methods frequently proceed by extracting points or edges as features from both the model and a given image (Figure 1.1) and trying to identify pairings between image data and model features that are consistent with a rigid transformation of the object model into image coordinates. In the absence of any information about the possible location of the object in the scene, all feature pairings are possible candidates and the search for the correct pairings of data and model features becomes combinatorially explosive. Most of this search is fruitless, especially when pairs of features belonging to separate objects are tried, and cannot possibly yield a correct transformation. If recognition systems were provided with information about a set of data features likely to come from a single object, then the search for the matching features can be focused on relevant subsets of features that are likely to yield a correct transformation. Thus in the image of Figure 1.1b, if it could somehow be determined that the region indicated in the circle came from a single object and hence all the features contained within it, then the number of features that need to be looked at would reduce considerably (from 500 to 50 here). This can cause the number of matches that need to be tried to be greatly reduced. To get an indication of the amount of reduction in search, for a reduction in the number of image features from 500 to 50 as in Figure 1.1 and assuming at least 4 pairs of features for matching, the number of possible matches reduces from $O(500^4 50^4)$ to $O(50^4 50^4)$ or by a total of about 10^{12} when there are 50 model features. Further, it is desirable to order these subsets of data features such that the more promising ones, i.e., those that are more likely to point to a single object, are explored first. This can not only increase the likelihood of a good match being obtained earlier, but is also useful when the task is to recognize as many objects as possible in a scene. Recognition systems, therefore, have found the need for a mechanism that can meet these goals, namely, to isolate areas in the image that are likely to come from a single object, and to order these regions such that the more promising ones are explored first. The problem of isolation and ordering of regions likely to come from a single object has been termed the selection problem and has been realized to be one of the key problems in recognition [59, 58]. It was shown in [58] that the expected complexity of recognition (using a recognition method called constrained search) can reduce from exponential to quadratic when all the features (edges in this case) were known to come from a single object. Other recognition methods have also found a need for such information as in the work of [95, 73]. Two kinds of selection can be distinguished, namely, data-driven and model-driven selection. In data-driven selection, the isolation of regions is based solely on the image data and some general a priori knowledge about the scenes, while in model-driven selection, specific information about a model object is used to locate regions in a given image that are likely to arise from this object. The data-driven selection problem occurs when there is no specific information (apart from the features for matching) available about the model object, or when more than one object is being recognized so that model-independent selection is needed, or when the model object is to be indexed from a library of models based on the selected region.

1.2 The Difficulty in Selection

Whether selection is data or model-driven, it is apparent that it differs from the problem of segmentation, where the goal is to partition the image into regions that contain a single object. In selection, it is not essential to isolate regions that totally contain a single object, nor is it necessary to partition the entire image into different object-containing regions. In addition, the isolated regions are only likely to contain the object and not guaranteed to do so. It would seem that these differences should make the selection problem relatively simple. Yet, it has largely remained unsolved. What makes selection so difficult? In the ideal case, if the appearance of the desired object in the scene were known, and objects in the scene were nicely separated and distinguishable from the background, and the illumination conditions were known, then even simple methods that rely on intensity measurements would work well to extract salient groups of features. But in reality the appearance of the object is not known. In addition, illumination conditions and surface geometries of objects present in a scene can cause problems of occlusion, shadowing, specularities, and inter-reflections which are compounded even further by scene clutter. As can be seen from Figure 1.1b, these conditions make it difficult to isolate regions likely to come from a single object based on low-level features such as edges and lines. In data-driven selection particularly, finding a way to extract meaningful structure in a group of features that also points to their likelihood of coming from a single object is difficult. Previous approaches to data-driven selection have regarded it as a problem of grouping data features such as edges, lines and points based on relations such as parallelism, or collinearity, [95, 125], distance and orientation [76], and regions enclosed by a group of edges [25]. Here the observation that it is unlikely that a random collection of features in a scene project to satisfy a relation such as convexity or parallelism in an image is used to infer that the presence of such a relation points to a greater likelihood of such features coming from a single object. This inference is not often valid under occlusions, changes in illumination conditions, and noise in

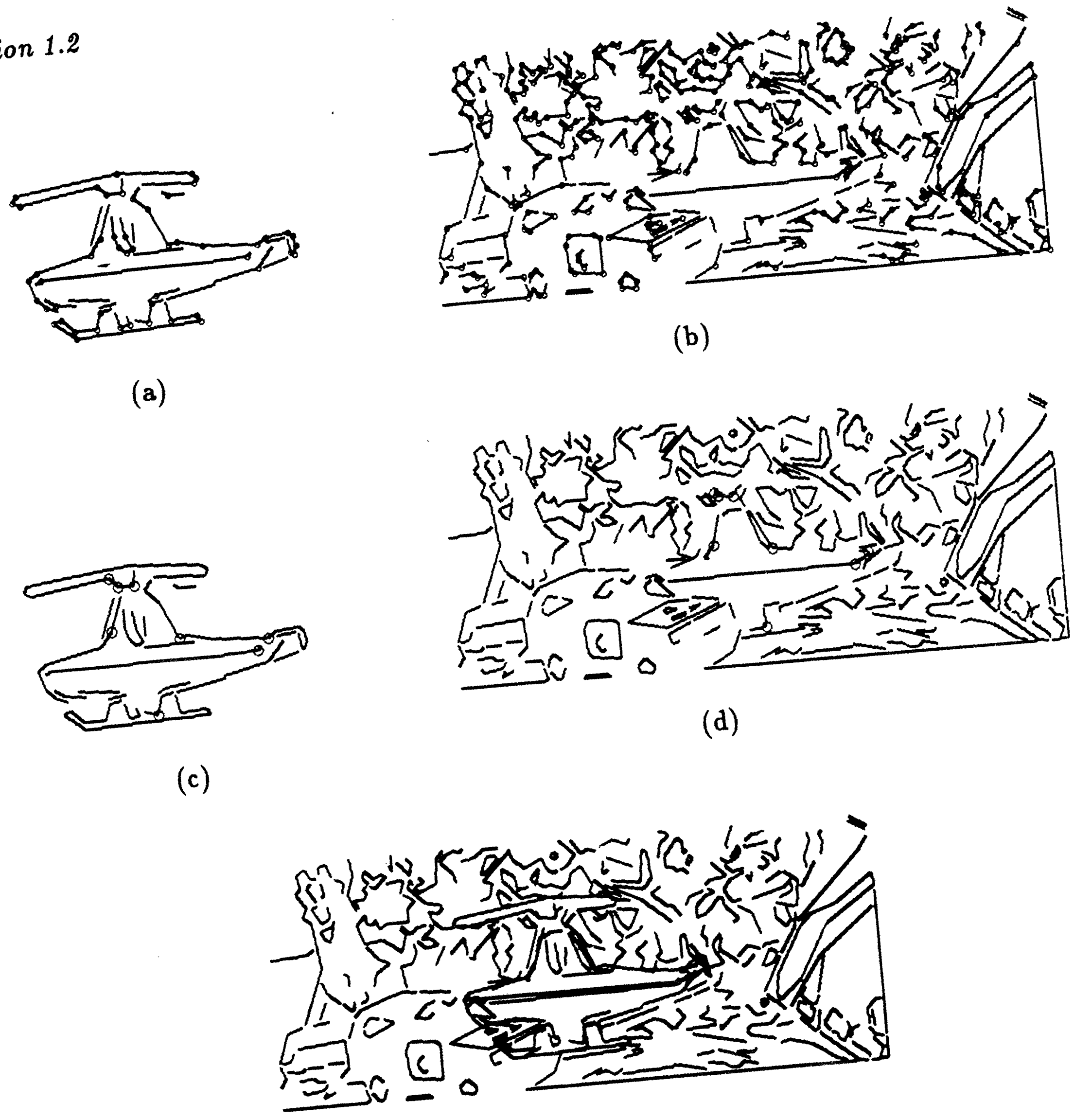


Figure 1.1: *Illustration of object recognition. (a) A view of a model object with corner features (in circles) and edges. (b) A scene in which the object appears. (c)- (d) A set of matching corner features in the model and image respectively that were found to give the correct transformation. (e) The model overlayed on the image of (b) using the transformation computed from the corresponding features shown by the big circles in (c) and (d).*

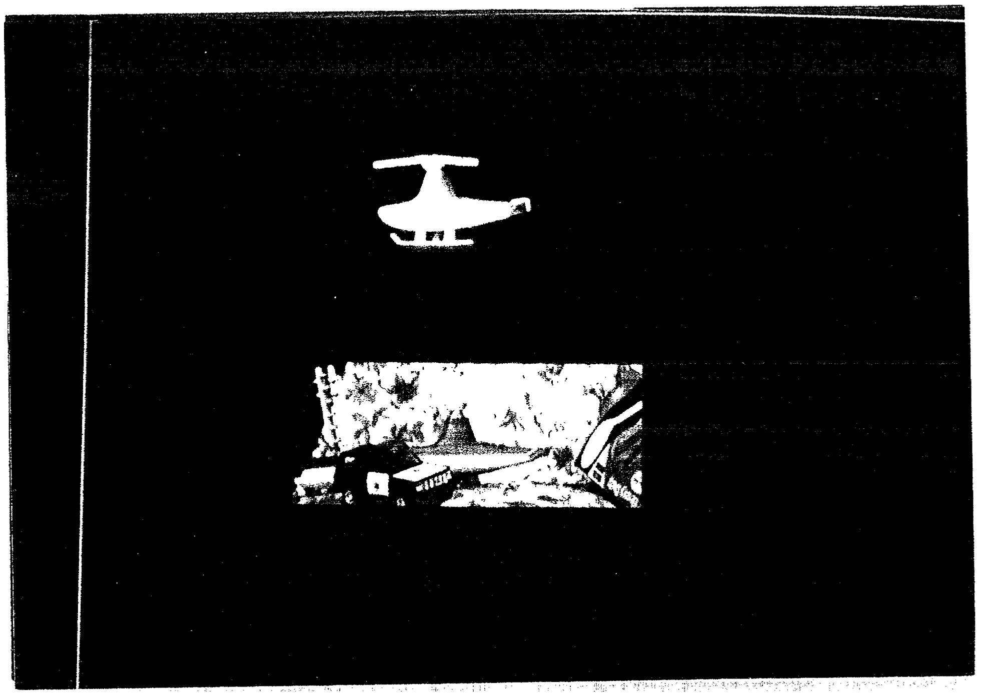


Figure 1.2: *Illustration to show that additional information about the object and scene can help in recognition. Here the model object and scene of Figure 1.1 are shown in color. The color information can help to separate the object's edges from that of the background. But deducing that the colors on the object in the scene of (b) are the same as those in (a) even though they appear different can be difficult.*

the image measurements with the result that most of the groups obtained are often unreliable, that is, they do not come from a single object. Even when a description of the object is available as in model-driven selection, interpreting data features under these conditions is still difficult.

It would appear that some of these problems would be alleviated if we could use more powerful model and image descriptors. For example, if we choose to use the color information in the image for selection, then a single color region is very likely to come from a single object (unless occlusions of similar colored objects occur) and would serve the purposes of data and model-driven selection. But extracting color regions is not easy. Also, finding a description of the color of the model object that remains stable under a variety of imaging conditions is also difficult. For example, deducing that the color of the object instance depicted in the scene of Figure 1.2b is the same as in its model description shown in Figure 1.2a can be difficult.

1.3 Attentional Selection

While it has proven difficult for machines to perform selection, humans, on the other hand, seem to have no such problem. We are able to look at a scene and quickly select some aspect of the scene for further processing. Frequently, such a selection is restricted to single entities or objects in the scene. This ability to perform a selection of the scene to focus the later processing is often attributed to the mechanism of visual attention [146, 70, 117]. The term attention has been interpreted in several different ways by past researchers. It has been described variously as a state of muscular contraction and adaptation, as a pure mental activity, as a limit on the human capability to carry out elementary mental processes [14, 15], and as a spotlight that glues the processing of features [149, 146]. All these views agree though that the end result of utilizing the mechanism of visual attention is to select a certain portion of the scene on which to concentrate future visual processing. Accordingly, visual attention can be viewed as a mechanism that by declaring some aspect of the visual stimulus to be *interesting*, allows the brain to selectively respond

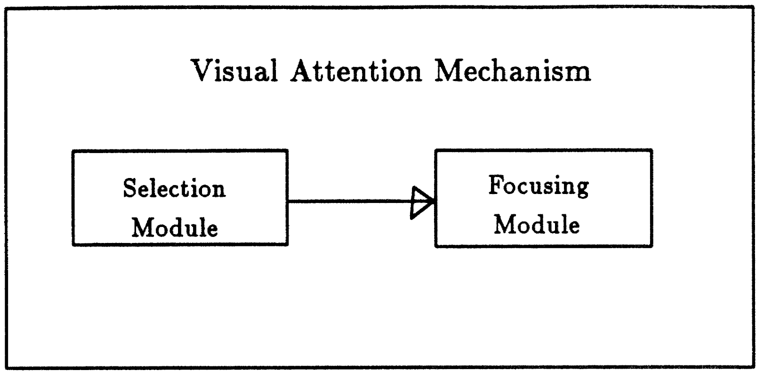


Figure 1.3: *The view of visual attention taken in this thesis. The attention mechanism is regarded as consisting of a selection module and a focusing module. The selection module declares some aspect/region of the scene "interesting" and causes the focusing module to be directed to that region.*

to it. In this view, the attention mechanism is thought of as consisting of a selection module and a focusing module as shown in Figure 1.3. The attentional selection module is one that declares a region of the scene interesting causing the focusing module to be directed to that region. This directing of attention may be deliberate or spontaneous, that is, may or may not be consciously driven. Attention focus can be directed to aspects/regions of a scene that 'pop' out [70], [39], or it can be directed to aspects/regions that are relevant to a task. These two ways of directing the attention focus may be thought of as two modes of attentional selection, namely, *attracted* and *pay attention* modes respectively. Thus in attract-attentional selection, the regions/aspects of a scene selected are those that 'pop' out or 'attract' attention[1], while in pay-attentional selection, the specific task information is used to 'pay' attention to only those object/aspects of the scene that are relevant to the task. The attracted-attention mode of selection has been referred to as pre-attentive processing and is usually considered to precede attention [147, 145, 149]. Here we adopt the view taken in [70] and consider it as part of the entire attention mechanism.

[1]Here the term attention is used colloquially

1.3.1 Attentional selection in object recognition

That visual attention can play a role in object recognition was realized by several researchers [67, 148, 70]. While most work has projected a role for attention as enhancing the detectability of an object (that is being attended to), the idea of using attention to select relevant aspects of a scene to avoid the combinatorial search in recognition was proposed in [70]. The aim of this thesis is to investigate this hypothesis further. Specifically, the aim is to develop a computational model of attentional selection, and use it as a paradigm for selection in object recognition. In particular, we wish to use the attracted-attention and pay-attention modes to serve as paradigms for data and model-driven selection respectively. Thus data-driven selection can be achieved by identifying regions in an image that attract attention (i.e., that are distinctive or salient) with respect to some feature such as color or texture, while model-driven selection can be achieved by paying attention to the model object's features (i.e., using the model features to decide saliency of features in the image). While it is understandable that paying attention to model features can help isolate areas in the image that could contain subsets of data features that are likely to contain a single object (or the specific model object in this case), it is not immediately apparent how locating salient regions can help in serving the goals of selection. Such a choice is, however, motivated by the following considerations. First, it is often observed that an object stands out in a scene because of some distinctive features that are usually localized to some portion of the object. Therefore isolating distinctive regions is more likely to point to a single object than an arbitrary collection of features. Secondly, a distinctive region, if suitably found, can help in limiting the number of candidate models from the library that can potentially match the given data. This is especially true if only a few models in the library satisfy the features that made the data region distinctive. Lastly, it has often been observed that the first objects recognized in a scene are those that attract an observer's attention [70]. Thus ordering the regions by distinctiveness to decide which objects to recognize first seems to be in keeping with this observation. Finally, a number of other approaches have also suggested that selection, at least data-driven, can be performed based on some measure of saliency, such as the structural saliency of curves [135], or saliency defined by local differences in contrast, color, or size [48, 97, 39].

1.4 Roadmap

So it appears that attentional selection is a reasonable paradigm for selection in object recognition both because it can isolate regions likely to come from a (the) single object and because it also orders the selected regions. But how is attentional selection achieved? Previous work on attention in psychological, psychophysical and physiological studies has concentrated more on establishing the capabilities and the existence of the mechanism rather than explaining how selection can be achieved. Psychological studies have put forward theories of attention [145, 14, 44, 125, 36, 117], while psychophysical studies have tried to indicate the effects of attention-mediated behavior as well as the factors that affect attention [84, 42, 108, 51, 148]. For example, such studies have observed that the knowledge of the location of the cue helps in focusing attention faster [42], that an increase in the number of physically similar distracting stimuli decreases the ability to attend [108], and that there is a relation between attention and eye movements [51],[148]. Physiological studies have tried to localize the various regions in the brain where attention can modulate neuronal response. Cells whose response is affected by the state of attention have been found to occur in the visual area V4, area 7, and the infero-temporal cortex (IT) [62, 106, 131, 18]. So while these studies have attempted to establish the existence of the attention mechanism, a few computational models have addressed the question of what constitutes attentional selection [89, 24]. Such models could explain mainly attract-attentional selection, and lacked enough detail to be amenable to an implementation. We begin, therefore, by presenting a computational model of the attentional selection process. This model, presented in Chapter 2, is designed to exhibit both attract and pay-attentional selection using a single framework. The thesis then describes an implementation of the model to serve as a paradigm for data and model-driven selection. Figures 1.4 and 1.5 give a flavor for the selection that can be performed by the system. Figure 1.4a shows an image of a realistic indoor

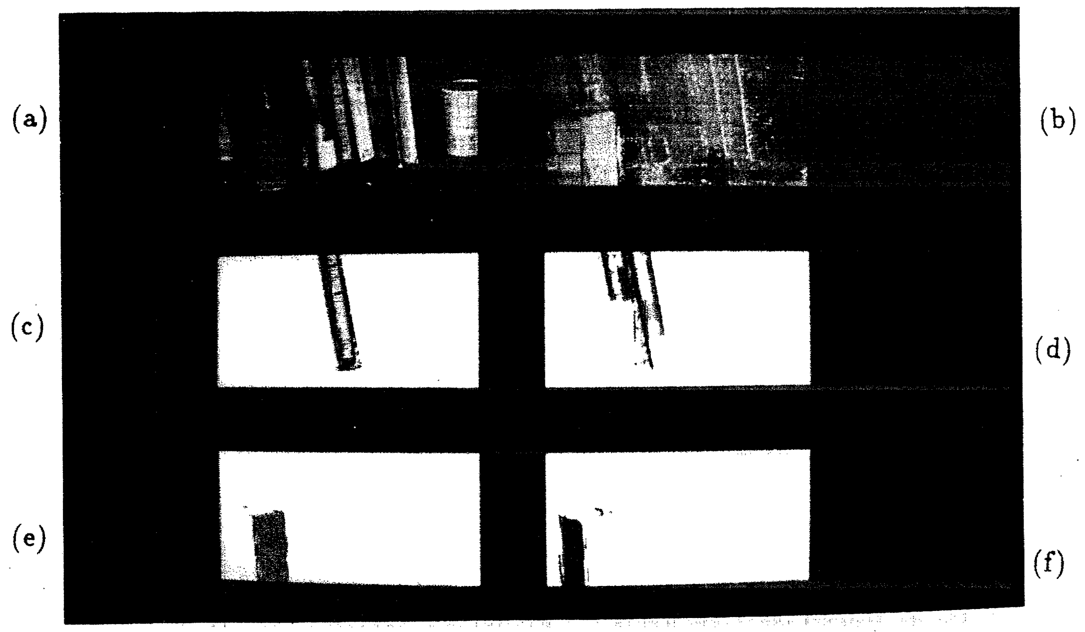
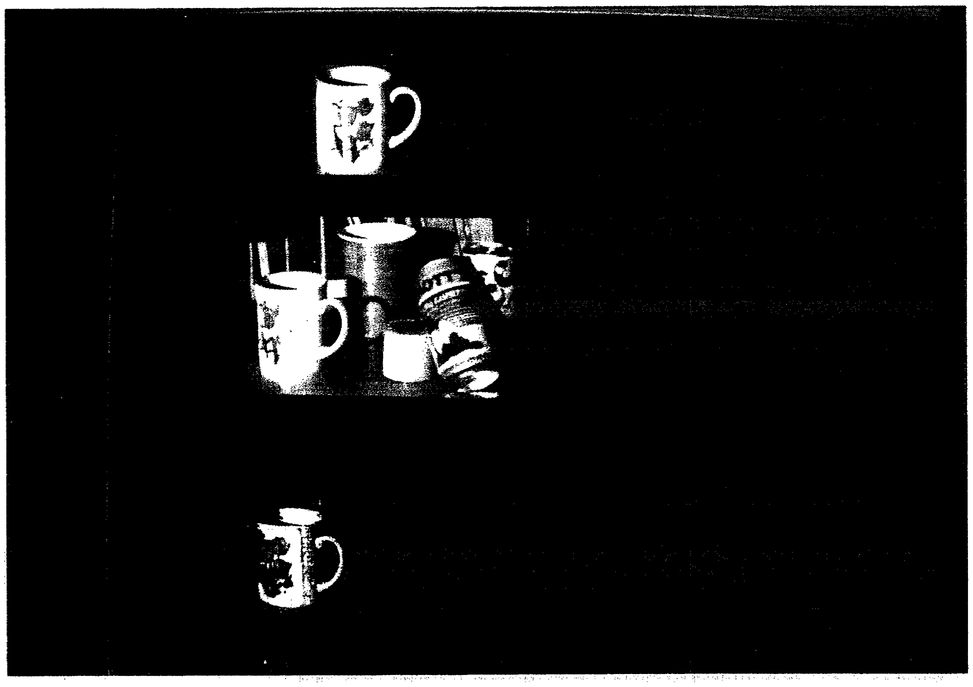


Figure 1.4: *Illustration of data-driven selection that can be performed by our attentional selection mechanism. (a) An image depicting a scene of objects of different materials and having occlusions and inter-reflections. (c)-(f) The four most salient regions detected by the selection mechanism after extracting the various color regions in this image as shown in (b).*

scene with shadows, inter-reflections, and consisting of many types of objects. The four most salient regions found by the system in this image are shown in Figures 1.4b-e. These regions are, therefore, the result of data-driven selection performed by the system. Figure 1.5 shows an example of model-driven selection being performed by our system. Figure 1.5a shows a model object and Figure 1.5b shows a scene in which the model object appears. Finally, Figure 1.5c shows a region of the image that is declared most likely to belong to the model object based on a description of the object drawn from the view shown in Figure 1.5a.

The rest of the thesis discusses how such a selection of the scene can be achieved by presenting an implementation of the model of attentional selection restricted to three different features, namely, color, texture and parallel-line groups. Chapters 3, 4, and 5 indicate how data and model-driven selection for object recognition can be



(a)

(b)

(c)

Figure 1.5: *Illustration of model-driven selection that can be performed by our attentional selection mechanism. (a) A model object. (b) A scene in which the object appears in a different pose and is illuminated differently. (c) Region of the scene declared to most likely contain the model object by extracting a color and texture-based description of the object from the view shown in (a).*

performed using these cues individually, which are later combined in Chapter 6 to complete the description of an implementation of the attentional selection model. In Chapter 3, we present a method of representing color information called perceptual color categories as an appropriate representation for the task of selection. These color categories are used to develop a fast region segmentation algorithm that extracts perceptual color regions in images. The color regions extracted form the basis for performing data and model-driven selection. Data-driven selection is achieved by selecting salient color regions as judged by a color-saliency measure that emphasizes attributes that seem to be important in human color perception. The approach to model-driven selection, on the other hand, exploits the color region information in the model to locate instances of the model in a given image. Here region adjacency graph-based descriptions of model and image color regions are used to develop a subgraph matching strategy for model-driven selection. The approach presented tolerates some of the problems of occlusion, pose and illumination changes that make a model instance in an image appear different from its original description.

Chapter 4 presents an approach to data and model-driven selection using texture or pattern on objects as a cue. Here the image is modeled as a short space stationary process and texture information is captured in a representation called the linear prediction (LP) spectrum, to develop a texture region segmentation algorithm that roughly indicates the different texture regions in the image. A measure of texture saliency is then developed to select among the various texture regions. It captures the interplay between regions of different contrast by analyzing their properties such as area, shape, and relative placement. Next, the problem of using the texture or pattern information on a 3D object as a cue to perform model-driven selection is discussed. Here we use the linear prediction spectrum representation again and show that the recognition of the texture pattern on an instance of the model object in an image is possible even under changes in orientation and occlusions. Specifically, we show that as the object undergoes a 3D linear transformation, the linear prediction spectrum of a planar patch texture on the object undergoes a 2D linear transformation that is the inverse of the transformation undergone by the texture. This result is used not only to develop a method of texture recognition by matching the LP spectra of model texture and an isolated image region, but also to recover the transformation. The candidate matching regions are obtained without detailed segmentation, by using a technique called overlapping window analysis. This analysis, under some conditions, guarantees the existence of a window spanning an image region containing only the model texture regardless of its position and orientation. Such a region is sufficient for the recognition of the model texture using the LP spectrum representation.

Chapter 5 examines the property of closely-spaced parallelism between lines on objects and exploits it to achieve data and model-driven selection. Specifically, we present a method of identifying groups of closely-spaced parallel lines in images. Since the end result here is a grouping of lines based on a constraint, this falls into the class of grouping methods for recognition. But the grouping scheme presented generates a linear number of small-sized and reliable groups and meets several of the desirable requirements of a grouping scheme for recognition. Data-driven selection is achieved by selecting salient line groups as judged by a saliency measure that emphasizes the likelihood of the groups coming from single objects. The approach to model-driven selection, on the other hand, uses the description of closely-spaced parallel line groups on the model object to selectively generate line groups in the image that are likely to be the projections of the model groups under a set of allowable transformations and taking into account the effect of occlusions, illumination changes, and imaging errors.

Chapter 6 brings together the work on selection using the individual cues of color, texture and line groups and places it in the context of the attentional selection model proposed in Chapter 2. It presents ways in which these three cues can be combined in both data and model-driven modes to perform an overall selection of the scene. This completes the description of an implementation of the model of attentional selection.

The next few chapters deal with the issue of evaluation of the attentional selection mechanism. Specifically, the pay-attentional selection is evaluated by integrating with a recognition system and recording the improvement in performance on test scenes. Chapter 7 presents details of the recognition system built and its integration with the developed attentional selection mechanism. The conclusion reached from the integration studies is that while attentional selection was originally envisaged as a front-end to a recognition system, information derived during selection could also be used during the matching stage of recognition. The search reduction possible using each of the cues individually and in combination was then explored. These estimates were then borne out in the actual test experiments. These results are presented in Chapter 8. These indicate that attentional selection can reduce the search involved in recognition drastically, by not only reducing the number of features to be examined by removing large portions of the image that are irrelevant, but also reducing the actual number of matches explored during recognition by providing more constraints using information derived from selection. Finally, a surprising use of selection in recognition became apparent from the studies on texture-based selection, namely, that selection can lead to the partial solution of the pose of a 3d object in an image.

1.5 Conclusions

The thesis presents a novel approach to the selection problem in object recognition by proposing a computational model of visual attentional selection as a paradigm for data and model-driven selection. The specific contributions of the thesis lie in the new ways of processing color, texture and parallel-line groups for performing data and model-driven selection using the paradigm of attentional selection, and in demonstrating that such a selection mechanism can drastically reduce the search involved in object recognition. It is hoped that the proposed model of attentional selection can be suitable for other tasks besides object recognition that require a selection of the scene. It is also hoped that the model can give a plausible explanation of the visual attentional selection in the brain.

Chapter 2

A Computational Model of Attentional Selection

In this chapter we present a computational model of attentional selection. We adopt the view that visual attention involves a selection phase followed by a focusing phase as mentioned in Chapter 1. With this view, questions often unresolved about the nature of the attention spotlight such as whether it is serial or parallel [107, 69, 150], external or internal [51, 148], are not of concern as they deal with the focusing phase of attention. Also, the model of attentional selection to be described here gives a conceptual view of the processing that may be involved in the selection phase. Therefore, no claims about a direct mapping of the model into neural architecture will be made.

2.1 The Computational Model

The proposed computational model of attentional selection is shown in Figure 2.1. According to this model, the scene represented by the image is first processed by a set of feature detectors that generate the respective *feature maps*. Some of the features that can be detected (apart from brightness) are: color, texture, depth,

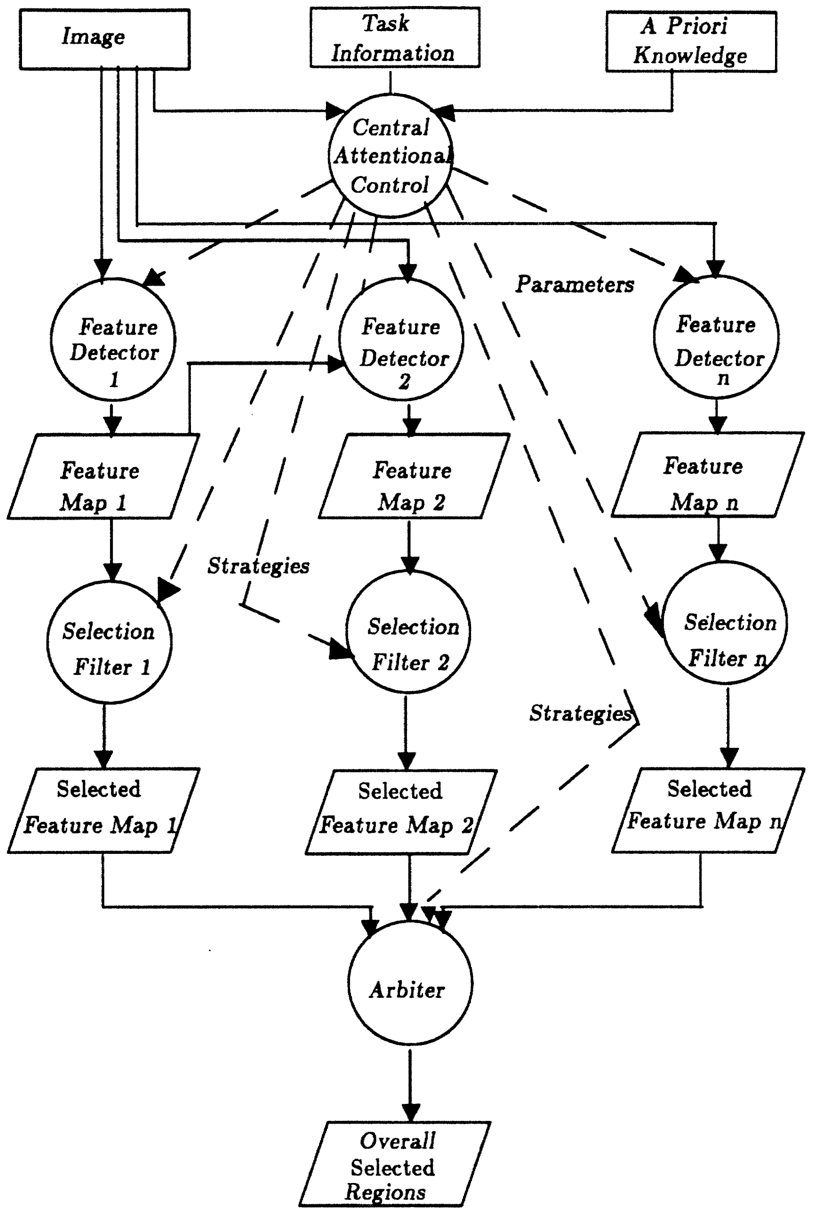


Figure 2.1: *The computational model of attentional selection proposed in this thesis that can demonstrate both attracted and pay attention modes of attentional behavior. The former behavior is achieved by a simultaneous activation of all feature maps using default parameters and procedures. The pay attention behavior, on the other hand, is achieved by a selective activation of feature maps. The two modes are proposed as paradigms for data and model-driven selection in object recognition.*

edges, curvature, contrast, line information such as parallelism, collinearity, length, orientation, and others such as shape, size, symmetry, etc. The feature detectors are algorithms or procedures that are designed to be open ended in that they may have some parameters that usually have a default value but can be set by the task at hand. The parameters can be, for example, the sigma of the Gaussian for an edge detector, or orientation difference thresholds for deciding a set of lines to be parallel in a parallel-line detector. Some of these features form a hierarchy, such as lines and corners abstracted into polygons, an instance of the closure property. The model therefore, allows feature detectors to interact via the medium of feature maps as indicated by the back arrow from a feature map to a feature detector in Figure 2.1. The model also allows more than one feature detector to be used to generate a feature map in the hierarchy. The feature detectors in the model are considered active units as indicated by circles in Figure 2.1 unlike the feature maps which are considered passive units and indicated by rectangles in the figure. The activation of the feature maps is controlled by a central attentional control as indicated in the figure. On inactivation, some of the feature detectors may produce no output at all while others may just let the image through without further processing.

Next, the feature maps are processed separately by *selection filters* that incorporate a set of strategies for selecting regions that are distinctive or salient in the respective feature maps. The result of such a processing is available in the form of individual saliency maps. When feature maps form a hierarchy, not all of them may be processed by selection filters. The strategies are again algorithms for processing information in the respective features and may employ intermediate representations. The choice of a strategy appropriate for a given task is decided by a central control mechanism. The control mechanism itself obtains information to decide the strategy from the task, the image, as well as a priori knowledge about the world.

Once the distinctive aspects of a scene along the respective feature dimensions have been identified, there is a final *arbiter* module that also houses a set of strategies, to decide the most significant aspects of the scene based on all the observed features. Here again, the choice of the strategy is task dependent and is affected by the control mechanism. The set of regions finally selected can then be used for further visual processing, as say, for recognition.

Thus the model suggests that information processing occur in three basic stages: an elementary feature processing stage, followed by a selection based on individual features or cues, and finally, an overall selection of the scene based on a combination of cues. It employs, therefore, three types of representations: feature maps, individual saliency maps, and finally, the overall selected regions map. The latter two maps differ from the elementary feature maps in that they indicate the importance of some image regions over the others so that not every region of the image may be represented. Further, the regions here may map to spatially overlapping regions in the image.

While the above description gives an impression of a passive flow of information, the model incorporates considerable flexibility through the control mechanism that can vary the strategies used to process features at each of the stages. The model also allows top down task level and a priori information to be combined suitably with the bottom up information derived by processing the image along the various feature dimensions. Top-down task level information can influence the processing of the image through the strategies driving the modules. Thus if the task involved searching for a particular color, then the image may be analyzed by a feature detector that looks only for that color rather than extract all the color regions so that the exact form of the maps could vary based on the task. Finally, the model allows for both serial and parallel forms of processing. All the feature dimensions can be processed either serially or in parallel, and within a feature dimension, the image could be analyzed by serial or parallel algorithms.

Either mode of attentional selection can now be demonstrated within the framework of the model as follows: In the *attracted* attention mode, all the modules at all stages are activated and the default strategies residing in these modules are used for choosing both a distinctive region within a map, and the overall most distinctive regions. In the *pay* attention mode, the task dictates what feature maps need to be activated and in what order. It also affects the choice of strategies in both the filtering and arbiter stages. The set of distinctive features chosen for a given scene can thus be different depending on the mode of operation.

2.2 Relation to Other Computational Models of Attention

The above model has some similarities as well as differences with existing computational models of attention [89, 24]. As in other models, the underlying motivation has been to use attention as a mechanism of feature-based selection in scene understanding. Like other models, it is also based primarily on the psychophysical model of visual attention by Treisman [145]. In Treisman's model, incoming information undergoes a preliminary analysis stage that exposes the physical properties such as color, orientation lines, brightness, etc. in feature maps. A filtering mechanism then makes a selection of some part of the scene based on these feature maps. The model emphasizes a preattentive stage where early representations of the image are formed and filtered. This is followed by an attentive stage where attention is focussed on the selected input. In fact, the idea of feature maps to represent the low level processing of information in the image has been suggested for visual processing in general by Marr [102], Barrow and Tenenbaum [6] and Treisman [145] among others. While Marr argued for a primal sketch as a map to expose the image features, Barrow and Tenenbaum had termed the feature maps as intrinsic images. Although building abstraction levels in the form of feature maps helps to expose the information contained in the image along the various feature dimensions, this alone is not sufficient. They must be combined to succinctly represent the inferences made by such processing. In Treisman's model this was hidden in the concept of the selection filter. Later computational models of attention tried to elaborate on the working of the selection filter. In Koch and Ullman's model [89], selection was done by combining the feature maps *somehow* into a single saliency map, and choosing the best (or the most salient) region by a simple winner-take-all mechanism. The feature maps in this model besides representing elementary operations on the image, also recorded the way in which image locations differed from their surrounding locations. Thus each feature map tells how conspicuous a given location is in a given scene. In this sense, it combines the feature map and individual saliency map of our model of attentional selection. The saliency map in Koch and Ullman's model combined information in each individual map into one global measure of conspicuity [153]. A model of attention proposed by Clark and Ferrier [24] later indicated a way of combining the feature maps into a single saliency map by using a linear combination of weighted feature values. That is, each feature map is assigned a weighting function that rates their importance and helps in the selection of relevant features to be incorporated in the saliency map.

The model proposed in this thesis, though similar in spirit to the previous models of attention, differs from them in several important respects. At the outset, it makes explicit the existence of two modes of attentional selection, namely, attract and pay-attentional selection, and provides a framework for exhibiting both modes of selection. While all these models suggested a strict bottom-up processing, by considering pay-attentional selection, this model allows top-down task level information to influence selection, allowing regions that are not necessarily salient on their own to become salient if they are desirable/relevant for the current task. Next, because of the hierarchy of features allowed, the maps need not be retinotopic. For example, in a color region map, the representation may be in terms of the number of regions and their spatial relationship rather than a pixel-wise representation of color regions. Further, because the feature maps are of different types such as color, texture, etc., it is not appropriate to combine them all into a single saliency map, until sufficient abstraction has been built based on these feature maps to allow a common basis for comparison. This is done in the model by building multiple saliency maps, one for each feature, and postponing their combination until the arbiter stage. This not only allows a more flexible way of filtering each feature type as is appropriate for that domain, but also allows a medium for top-down task level information to permeate and affect the selection of the salient regions in each of these feature maps. The combining of feature maps through the arbiter module is thus more general than the linear combination way of combining cues proposed in Clark and Ferrier's model

[24]. Finally, the model separates the control issues from the assembly of feature maps, thus allowing us to analyze separately their individual roles in making up the attentional selection mechanism.

2.3 Performance Evaluation

An interesting question associated with the development of a computational model of attentional selection is how the success of a system implementing the model can be evaluated. That is, how does one evaluate the nature of selection performed by such a system? Evaluating an implementation of attract-attention mode is tricky since it would entail seeing if there is agreement with the selection performed by humans. That is, when asked to describe the most noticeable regions of a scene, do the regions selected by the attract-attention mode of the model coincide with the conclusions of human observers. To test this, psychophysical experiments could be performed in which eye movements of observers are recorded. The density of fixations can be taken to be indicative of the saliency of a region. With this method however, the performance of the system can be judged mostly in cases where the distinctive features detected are localized to a small spatial region. Also, it may not be conclusive since the directing of attention to a region can be independent of eye movements [51]. Even when we cannot evaluate this attention mode by psychophysical experiments, it may still be interesting to see if such an implementation serves some useful purpose. Since our original motivation was to propose attentional selection as a paradigm for selection in object recognition, one way to evaluate the attract-attention mode would be to see if the selected regions satisfy the requirements of data-driven selection in object recognition, that is, do they come from a single object? And are such regions useful for reducing the search in object recognition?

Evaluating the pay-attentional selection is easier since it is tied to a task and requires asking: Did the selection mechanism succeed in selecting regions relevant to the task? By making the task the recognition of a model object description, we can turn the implementation of the pay-attention mode into a model-driven selection mechanism. Thus the pay-attention selection can be evaluated by integrating with an object recognition system and testing the performance of the combined system in reducing the search involved in recognition, as well as the reduction in the number of false positive and negative identifications.

2.4 Implementing Attentional Selection

In attempting an implementation of the model of attentional selection, several issues need to be addressed at practically every level of processing, such as: What kind of features should be chosen? How can they be detected and represented? What are the different strategies that may be used to select distinctive regions in the individual feature maps? How should selection based on different features be combined? Finally, there is the problem of control relating to the activation of various modules and the interaction between feature maps. These problems are nontrivial as seen for example in the problem of deciding which region is the most distinctive one in a map. Thus in a given map, say, a color map, how do we decide which is the most distinctive color? Is it the intensity, the hue or the predominance of a given color that is the deciding factor?

The next few chapters discuss an implementation of the model of attentional selection showing ways in which some of these issues are addressed. Trying to develop an implementation that exploited all the cues that human attentional selection is known to do would be difficult. So the implementation was restricted to using only three cues, namely, color, texture, and parallel-line groups. These provide a fairly rich subset of features that capture the information present in a scene. Next, exploring all possible strategies for both individual saliency detection and overall saliency was again beyond reach. Therefore, we restricted to two kinds of strategies, one meant for extracting saliency based on data alone, the other by making the task information to be the description of a model object for recognition. In doing so we had a restricted implementation of the attentional selection model that also served as a mechanism for data and model-driven selection in object recognition, thus serving our original purpose behind proposing the attentional selection model. The discussion on the implementation to follow in the next few chapters will therefore be presented as ways of achieving data and model-driven selection in object recognition using each of the individual cues, beginning with color.

Chapter 3

Data and Model-driven Selection using Color Regions

In this chapter we address the problem of using color as a cue to perform data and model-driven selection in object recognition using the paradigm of attentional selection. We first motivate the choice of color as a feature to study selection, and outline the requirements imposed by selection on any method of color specification. We then examine some of the existing approaches to color specification in images in the context of selection. Next, we present a method for color specification based on perceptual color categories and show that it leads to a fast color region segmentation algorithm. Later, following the attract-attention paradigm for data-driven selection, a method of finding salient color regions is presented. Finally, a description of model color regions is developed that is used to perform model-driven selection.

3.1 Role of Color in Selection

Color is known to be a strong cue in attracting an observer's attention. Several psychophysical experiments have shown that color is one of the cues that is pre-attentively perceived [147, 146, 39]. Humans also use color information to search for specific objects in a scene. It therefore seems natural to use color as a cue for performing selection in computer vision. But the strong motivation for using color to perform selection comes from the fact that it provides region information and that, when specified appropriately, it can be relatively insensitive to variations in normal illumination conditions and appearances of objects [141]. A color region in the image almost always comes from a single object (unless the object is occluded by another of an identical color) so that data features such as edges and points within a color region form more reliable groups than those obtained by existing grouping methods. This makes such regions useful for data-driven selection. Because objects tend to show color constancy under most illumination conditions, color can be a stable cue to locate objects in scenes, thus making it also suitable for model-driven selection.

3.2 Color Specification for Selection

But how should color be specified? Using the attentional-selection paradigm, both data and model-driven selection require the extraction of color regions so that any method of specifying color should lead to the reliable extraction of such regions from images. For data-driven selection, which merely ranks color regions, it is sufficient to find a method that can distinguish between two adjacent color regions under the current imaging conditions (i.e. under the illumination conditions, surface geometries, and occlusions in the given scene). For model-driven selection, on the other hand, a method of color specification should also be stable, so that the color in the model description can be reliably detected in a scene taken under different imaging conditions. If machines could capture the way humans achieve color constancy, that is, be able to discount spectral variation in the ambient light and assign stable colors to objects, then this would be sufficient for both data and model-driven selection. One way to achieve this color constancy, examined in previous approaches, is by recovering the surface reflectance of objects [91, 101, 68, 55, 151]. When a surface is imaged, the light falling on the image plane (image irradiance) is related to the physical properties of the scene being imaged via the image irradiance equation (see Figure 3.1):

$$I(\lambda, r) = \rho(\lambda, \mathbf{r})F(\mathbf{k}, \mathbf{n}, \mathbf{s})E(\lambda, \mathbf{r}). \tag{3.1}$$

where λ is the wavelength, $\mathbf{r}$ is the spatial coordinate and r is its projection in the image, $E(\lambda, \mathbf{r})$ is the intensity of the ambient illumination, and $\rho(\lambda, \mathbf{r})$ is the component of surface reflectance that depends only on the material properties of the surface (and hence specifies its "surface color"). Finally, $F(\mathbf{k}, \mathbf{n}, \mathbf{s})$ is the component of surface reflectivity that depends on surface geometry, with **k**,**s**,**n** being the viewer direction, the source direction and the surface normal respectively. Since $\rho(\lambda, \mathbf{r})$ is purely a property of a surface, specifying colors of objects by this function has been considered a way of achieving color constancy. But as can be seen from equation 3.1, the recovery of the surface reflectance function when both $F(\mathbf{k}, \mathbf{n}, \mathbf{s})$ and $E(\lambda, \mathbf{r})$ are unknown is an under-determined problem. Most methods, therefore, make some assumptions about either the surface being imaged [91], or about the illumination conditions [101, 68, 55, 151], or both [53] so that enough constraints can be obtained to recover the surface reflectance. In the retinex theory [46, 91] for example, by assuming the surface to be flat, the surface reflectance could be specified by recovering the lightness[1] values in three wavelength bands called channels. The lightness in each channel is recovered by spatial differentiation of image intensity followed by normalization using the average surface reflectance. Other approaches recover the reflectance by assuming it can be specified by a linear combination of a few basis reflectance functions [101, 160, 151]. Here the reflectance is specified by the coefficients of the linear combination.

Methods of recovering the surface reflectance (called surface color recovery methods) work well when the problem is to specify the color of already isolated regions. But when the problem is to extract the color regions in an image, they do not do as well and have to make restricting assumptions such as the mondrian assumption [46] to avoid considering the variation of the spectral reflectance across a surface. A model that takes this variation into account is the dichromatic reflection model for dielectric surfaces [133]. For such surfaces, the reflectance can be expressed as a lin-

[1]Lightness is the estimate of reflectance obtained from the image [66].

ear combination of a surface reflection and a body reflection component. By allowing the coefficients of the linear combination to be a function of position, the variation in the surface reflectance across a surface could be handled. The key observation here was that the variations tend to cluster along the directions of the body and surface reflection components. This observation was exploited by Klinker to develop an algorithm for extracting color regions [88]. Here the filtered components of intensity along three wavelength channels called red, green and blue (to correspond to the filters used in color cameras) as specified by the triple $< R, G, B >$ (called *specific color* henceforth) is expressed as a vector in a 3-dimensional color space called the *rgb-space* with axes standing for the pure red, green and blue components as shown in Figure 3.2. Using the dichromatic reflection model, a color region in the image maps to a T-shaped cluster in this space with colors of pixels crowded along the directions of surface and body reflection components. Color region segmentation was achieved by mapping small portions of the image into color space and analyzing the T-shaped clusters. But separating the T-shaped clusters in color space from adjacent color regions in the image is difficult. Moreover this method required the color of the object to be different from the color of the light source.

Other methods have also tried to specify the color by mapping the image intensity as expressed by the triple of $< R, G, B >$ values as a point in a color space such as the rgb-space and analyzing the clusters in the resulting histograms for purposes of color segmentation [109, 87] and localization [140, 141]. Such clustering approaches suffer from two main problems. First, distance measures that are used to group a set of points in a color space do not really capture the perceptual distance between colors. For example, Figures 3.3b and c show two color patches that are at equal normalized distance in the rgb-space from the color patch shown in Figure 3.3a when they are treated as points in this space[2]. Yet the color patch of Figure 3.3a is perceptually closer in color to the patch in Figure 3.3c than to the patch in Figure 3.3b. The second problem with the clustering approaches can be described as the problem of coordination of information processing in color and image spaces as illustrated in

[2]The normalized distance between two color patches specified by $< r_0, g_0, b_0 >$ and $< r, g, b >$is computed as $\sqrt{(\frac{r_0}{r_0+g_0+b_0} - \frac{r}{r+g+b})^2 + (\frac{g_0}{r_0+g_0+b_0} - \frac{g}{r+g+b})^2}$.

Figure 3.4. Typically, the observation that pixel colors of a colored region all map to a cluster in color space is turned around to infer that isolating a cluster in color space can give a single color region in the image. Clearly this is not true when there is more than one region of the same color. More importantly, while it is true that a color region may map to a single cluster in color space, any cluster in color space, even if obtained from a small region of the image, need not correspond to a single color. This becomes obvious when we observe that any color space (such as *rgb*, *hsv* space [104]), showing the spectrum of visible colors contains boundaries where perceptual color changes occur, so that a cluster in such a space spanning both sides of the boundary does not correspond to a region of single color. When this is not taken into account, a bin in the color space as done in the histogram analysis methods [109, 141], can span different colors. A dramatic example of this can be seen by histogramming the rgb space into 216 equal-sized cubical bins (approximately 0.166^3 in volume)[3]. Figure 3.5a shows a mondrian of color patches created by picking points in the rgb color space that fall into one such bin. As can be seen, the colors within this bin are dramatically different even in hue[4]. A segmentation algorithm based on histogram analysis in color space could potentially treat this bin as a single color (as the bin size is sufficiently small) so that a segmentation of the mondrian image of Figure 3.5a would be as indicated in Figure 3.5b. Thus it appears that boundaries where perceptual color changes occur must be found before any cluster in color space can be interpreted as corresponding to a region in image space. This observation is the basis of our approach to color specification presented next.

3.3 Perceptual Color Specification by Categories

For the purposes of selection, we propose that it is sufficient if a region could be specified by its perceived color, that is, the color perceived by humans looking at an image of the scene, and the effects of artifacts such as specularities could be

[3]The rgb-space was divided into bins of sise 0.166 x 0.166 x 0.166 to give a total of 216 bins to compare closely to the 220 perceptual color categories that we obtained later.

[4]This holds even for another color space such as the hsv color space.

separately accounted, as in [88, 87]. Using the perceptual color, two adjacent color regions would be distinguished if their perceived colors were different, and this is sufficient for data-driven selection. Because objects tend to obey color constancy under most changes in illumination, their perceived color remains more or less the same thus making it sufficient also for model-driven selection. But can perceptual color be quantified at all? In general, several effects such as simultaneous color contrast and color filling, have been known to influence human perception of color [166]. Fortunately, (as we will explain later), these factors are not very critical for selection.

We now present a method for specifying perceptual color using the concept of color categories. The color of pixels in images is described by a triplet <R,G,B> (referred to as *specific color* before), representing the filtered components of image intensity at a point or pixel along three wavelengths corresponding to the red, green and blue filters of the camera. When all possible triples are mapped into a 3-dimensional color space with axes standing for pure red, green and blue respectively, we obtain a color space that represents the entire spectrum of computer recordable colors. Such a color space must, therefore, be partitionable into subspaces where the color remains perceptually the same, and is distinctly different from that of neighboring subspaces. Such subspaces can be called *color categories.* Now, each pixel in a color image maps to a point in this color space, and hence will fall into one of these categories. *The perceptual color of this pixel can, therefore, be specified by this color category.* To obtain the perceptual color of regions from the perceptual color of their constituent pixels, we observe the following. Although the individual pixels of an image color region may show considerable variation in their specific colors, the overall color of the region is fairly well-determined by the color of the majority of pixels (called *dominant color* henceforth). *Therefore, the perceived color of a region can be specified by the color category corresponding to the dominant color in the region.*

Since color categories capture the perceptual color, two adjacent color regions would belong to two different categories, making it possible, as we will show later, to extract the different color regions. This can make category-based specification of color suitable for data-driven selection. For model-driven selection, the color description of the object should remain stable. Since the specific color can change considerably with imaging conditions, it may cause a change in the color category too. But by restricting the illumination changes and by handling effects of artifacts such as specularities and inter-reflections separately, as we will discuss later, the category-based specification can still be sufficient for model-driven selection.

The category-based specification of perceptual color (of pixels or regions) is a compromise between choosing the specific color (which is extremely unstable with respect to changes in illumination conditions, etc.) and surface color (whose recovery is hard). Also, since the perceptual categories depend on the color space and are independent of the image, they can be found in advance and stored in, say, a look-up table. Finally, a category-based description is in keeping with the idea of perceptual categorization that has been explored extensively through psychophysical studies [10, 19, 129]. These studies concluded that although humans can discriminate between several thousand nuances of colors, psychophysically we seem to partition the color space into relatively few distinct qualitative color sensations or categories [139, 49].

3.3.1 Categorization of color space

The above discussion argued for the viability of an approach that recovers a color to within a category. Before this can be turned into a computational method of color recovery one needs to address the issue of how such categories may be found. Previous work on color categorization involved experiments of naming the color using a limited vocabulary, or identifying colors using the Munsell color charts [166]. But for computational color recovery, we need a way to convert the camera recordable red, green and blue components of colors into computer recordable perceptual color categories. This was done by performing some rather informal but extensive psychophysical experiments that systematically examined a color space and recorded the places where qualitative color changes occur, thus determining the number of distinct color categories that can be perceived. For this, the hue-saturation-value

δh	δs	δv	number of bins	number of categories
5	0.1	0.1	7200	220

Table 3.1: *Parameters used for the quantization of color space.*

color space was used as it specifies a given color in terms of its hue, purity and brilliance - attributes that have been found to give a perceptual description of color [80]. The details of these experiments are described in Appendix A and will not be elaborated here, except to mention the following. The entire spectrum of computer recordable colors (2^{24} colors) was quantized into 7200 bins corresponding to a 5 degree resolution in hue, and 10 levels of quantization of saturation and intensity values (see Figure 3.6). The color in each such bin was then observed by displaying a mondrian (a uniform color patch) of that color on a monitor screen and observing it under dark room conditions with appropriate monitor calibration. From our studies, we found about 220 different color categories were sufficient to describe the color space. The color category information was then summarized in a *color-look-up table.* Although it is true that a finer level of quantization would have yielded more categories, a smaller set is actually more useful since it gives a reasonably coarse description of the color of a region thus allowing it to remain the same for some variations in imaging conditions. In fact, by the above method we can also determine which categories can be grouped to give an even rougher description of a particular hue. This was done and stored in a *category-look-up table* to be indexed using the color categories given by the color-look-up table.

3.4 Color Region Segmentation

The previous section described how to specify the color of regions, after they have been isolated. But the more crucial problem is to identify these regions. In this section, we show that the perceptual categorization principle can be used to determine which pixels can be grouped to form regions in an image. If each surface in the scene were a mondrian, then all its pixels would belong to a single color category, so that by grouping spatially close pixels belonging to a category, the desired segmentation of the image can be obtained. But real surfaces being hardly mondrians, it is rare that pixels of a region from such surfaces all belong to the same color category. They could show considerable variation in color with bright and dark pixels intermixed, and with possibly spurious pixels also being present. We now analyze some of the color variations across an image that can result from imaging a colored surface in the scene.

3.4.1 Variation of color across an image of a 3D-surface

In this section we use some assumptions to show that the color variation across an image of a surface is mostly in intensity. The image irradiance equation given earlier as equation 3.1 and repeated here for convenience as

$$I(\lambda, r) = \rho(\lambda, \mathbf{r})F(\mathbf{k}, \mathbf{n}, \mathbf{s})E(\lambda, \mathbf{r}) \tag{3.2}$$

relates the light falling on the image plane (image irradiance) to the physical properties of the scene being imaged. Although the image irradiance equation assumes that all surfaces in a scene reflect light governed by a single reflectivity function, we can easily reinterpret this equation to represent image irradiance of a single surface. Under the assumption of a single light source, the surface illumination $\mathrm{E}(\lambda, \mathbf{r})$ can be separated as a product of two terms $\mathrm{E}_1(\lambda)$ and $\mathrm{E}_2(\mathbf{r})$, and since $F(\mathbf{k}, \mathbf{n}, \mathbf{s})$ is a function of position $\mathbf{r}$ it can be expressed as $\mathcal{F}(\mathbf{r})$. Then the image irradiance equation can be re-written as

$$I(\lambda, r) = \rho(\lambda, \mathbf{r})\mathcal{F}(\mathbf{r})E_1(\lambda)E_2(\mathbf{r}). \tag{3.3}$$

The surface reflectance and hence the resulting appearance of a surface is determined by the composition as well as the concentration of the pigments of the material constituting the surface. For most surfaces, the composition of the pigments can be considered independent of their concentration so that the spectral reflectance $\rho(\lambda, \mathbf{r})$ can be written as a product of two terms $\rho_1(\lambda)$ and $\rho_2(\mathbf{r})$. Note that this assumption is less restricting than the assumption of homogeneity that has been used before [68]. With this simplification, (and grouping the product of terms dependent on λ and $\mathbf{r}$ separately) the image irradiance equation becomes

$$I(\lambda, r) = H(\mathbf{r})L(\lambda). \tag{3.4}$$

Now, if we consider the filtered version of this signal, i.e., the image irradiance in three channels, say the red, green and blue channels with their associated transfer functions $h_R(\lambda), h_G(\lambda), h_B(\lambda)$, the specific color at each pixel location r is specified by the triple <R(r),G(r),B(r)> where

$$\begin{aligned}
R(r) &= \int_0^\infty I(\lambda, r)h_R(\lambda)d\lambda &= H(\mathbf{r})\int_0^\infty L(\lambda)h_R(\lambda)d\lambda &= H(\mathbf{r})R_1 && (3.5)\\
G(r) &= \int_0^\infty I(\lambda, r)h_G(\lambda)d\lambda &= H(\mathbf{r})\int_0^\infty L(\lambda)h_G(\lambda)d\lambda &= H(\mathbf{r})G_1 && (3.6)\\
B(r) &= \int_0^\infty I(\lambda, r)h_B(\lambda)d\lambda &= H(\mathbf{r})\int_0^\infty L(\lambda)h_B(\lambda)d\lambda &= H(\mathbf{r})B_1\ . && (3.7)
\end{aligned}$$

This shows that under the given assumptions (which include non-homogeneous surfaces), the color of a surface can vary only in intensity. In practice, even when the separability assumption on reflectance is not satisfied, or there is more than one light source in the scene, the general observation is that the intensity and purity of colors are affected, but the hue still remains fairly constant. In terms of categories, this means that different pixels in a surface belong to *compatible categories*, i.e. have the same overall hue but vary in intensity and saturation. Conversely, if we group pixels belonging to a single category, then each physical surface is spanned by multiple overlapping regions belonging to such compatible color categories. These were the categories that were grouped in the category-look-up-table mentioned in Section 3.3.1. The next section describes how these concepts can be put together to give a color image segmentation algorithm.

3.4.2 Color region segmentation algorithm

The algorithm for color image segmentation performs the following steps. (1) First, it maps all pixels to their categories in color space. (2) It then groups pixels belonging to the same category, (3) and finally it merges overlapping regions in the image that are of compatible color categories.

1. Mapping pixels to categories: This is done by a simple indexing of the color-look-up-table by the color of the pixel specified in terms of its hue, saturation, and brightness components. These components can be derived from the specific color as described in [52]. This step takes time = O(N) where N is the size of the image.

2. Grouping pixels of same category: The image is divided into small non-overlapping bins of fixed size, (say, 8x8) and the color categories found in the bins are recorded. The size of the bin can be chosen based on expectations about the average size of color regions found in natural scenes. Each bin thus has a list of color categories summarizing the pixel color information in the bin. Neighboring bins that contain a common color category can be grouped to give a connected component representing an image region of that color category. Since a bin has several color categories, it belongs to several connected components that overlap. The actual grouping algorithm we used is a sequential non-recursive labeling algorithm that simultaneously assembles all the overlapping connected components using the category description in the bins. This algorithm is an extended version of the labeling algorithm for binary images described earlier [66], and uses the union-find data structure to efficiently merge category labels into connected components taking time = $O(k^2 M)$ where M = number of windows, and k = maximum number of categories present in the window (= O(1) for small window-sizes, eg., 8 x 8). The resulting labels are propagated back to the pixels to give the precise boundaries of color regions of single color categories. The color of the region is then specified by the color category and specific color that is the dominant color in the region as described in Section 3.3.

3. Merging overlapping regions: The general problem of determining which regions overlap in the image can be a computationally intensive operation as it involves determining which polygonal regions intersect and finding their regions of intersection. But by using the bin-wise representation of connected components, we can detect and combine overlapping regions with greater ease. From the discussion in Section 3.4.1, a shaded region maps to categories in color space that are compatible, i.e., have the same overall hue. The categories that are compatible are available from the category look-up-table described in Section 3.3.1. To find all such regions

that have compatible categories and overlap in image space, the algorithm examines each window of the image to see if it contains the interior portions of regions of compatible color categories. Such overlap regions are grouped as in Step 2. This step again takes $O(k^2M)$ time. Finally, the window-level color labels are propagated back to the corresponding pixels to give an accurate localization of the color region boundaries.

The algorithm for color image segmentation thus makes only a constant number of passes through the image, each being linear in the size of the image.

3.4.3 Handling specularities

The above algorithm segments the image into regions according to their perceived color. As we described before, this is sufficient for data-driven selection. But for model-driven selection such a description needs to be augmented with the knowledge of artifacts that occur in the image such as specularities, shadows, or inter-reflections. Such artifacts can cause a model region to appear fragmented. For example, a sharp streak of specularity on the surface can cleave its image into two regions. If these artifacts could be identified and corrected, this can improve the effectiveness of a color-based model-driven selection system. We now discuss how one of these artifacts, namely, specularities, can be handled once the color regions have been isolated. Specularities are present in regions produced by objects in the scene having shiny surfaces, such as metallic objects and dielectrics. These specularities have a central bright portion that appears white in most illumination conditions (bright sunlight, day light, tube light) and tapers off near the specularity boundary merging into the rest of the body color. Such specular regions and their adjacent colored regions when projected into a color space form characteristic clusters such as the skewed T described by Klinker et al. [86] and mentioned in Section 3.2. These clusters can, therefore, be analyzed to detect and remove highlights using the method described in that paper.

3.4.4 Results

Figures 3.7- 3.9 demonstrate the color region segmentation algorithm. Figure 3.7a shows a 256 x 256 pixel size image of a color pattern on a plastic bag. The folding on the bag and its plastic material together give a glossy appearance in the image as can be seen in the big S and Y. The result of Step-2 of the algorithm is shown in Figure 3.7b, and there it can be seen that the glossy portions on the big blue Y and the red S cause overlapping color regions. These are merged in Step 3 and the result is shown in Figure 3.7c. As can be seen in the figure, the algorithm achieves a fairly good segmentation of the scene for such surfaces. Figure 3.8 shows another image consisting of colored pieces of cloth with the textured region having several small colored regions within it. The results of the algorithm (Figure 3.8c) show that even such colored regions can be reliably isolated. Finally, Figure 3.9a shows an image of a realistic indoor scene with shadows, inter-reflections, and consisting of many types of objects. The different color regions found in this image are re-colored and shown in Figure 3.9b. Notice in the segmented image of Figure 3.9b that adjacent objects of the same perceptual color are merged (grey books). This is to be expected because the grouping of regions is based on color information alone. More examples of color region segmentation will be seen in later chapters when color-based selection is actually used in a recognition system.

3.5 Color-based Data-driven Selection

The segmentation algorithm described above gives a large number of color regions. Some of these may span more than one object, while some come from the scene clutter rather than objects of interest in the scene. It would be useful for the purposes of recognition to order and consider only some of these regions so that by isolating data subsets from such regions, the search can be focused on key groups of features thus excluding much of the scene clutter. Using the paradigm of attract-attentional selection, the color regions can be ordered by their saliency, i.e., by how distinctive they appear. The method of color-based selection, therefore, is to extract color regions from the image, order them based on a measure of color-saliency and then select a few most salient regions to be given to any recognition system. In this section we first describe a measure of expressing color saliency, and then examine the utility of salient-region selection in recognition.

3.5.1 Finding salient color regions in images

In trying to express distinctiveness, one encounters the question: Is distinctiveness expressible at all? In general, any judgment of distinctiveness has both a sensory and a subjective component. Thus for example, while most of us can perceive brighter colors more easily than duller colors, the judgment of which of two hues of the same brightness and saturation are more salient can be subjective. The aim here is to focus on the sensory component of distinctiveness and hence extract properties of regions that are general enough to be perceived by most observers. Accordingly, we propose that the saliency of a color region be composed of two components, namely, *self-saliency* and *relative saliency*. Self-saliency determines how conspicuous a region is on its own and measures some intrinsic properties of the region, while relative saliency measures how distinctive the region appears when there are regions of competing distinctiveness in the neighborhood.

In order to develop such a measure for color-region saliency, one must ask the following questions: What features in regions determine their saliency? How can they be measured to reflect our sensory judgments? Finally, how can they be combined to give the saliency measure? We now address these questions and derive a measure of color-saliency.

Features used for measuring self and relative saliency

Since the saliency of a color region depends on the region features used, they must be carefully selected. Such features should be: (i) perceptually important, (ii) easily measurable, and (iii) fairly general, to avoid subjective bias.

1. Color: The color of a region is an intrinsic property and affects a region's self-saliency. It is specified by (s(R),v(R)), where s(R) = saturation or purity of the color of region R, and v(R) = brightness, and $0 \leq$ s(R),v(R) ≤ 1.0. The hue of colors is not considered, to avoid subjective bias.

2. Region size: The size of a region is again an intrinsic property and affects its self-saliency. It is chosen as a feature based on the observation that regions that are either very small in extent, or that are large enough to cover the entire field of view, do not often attract our attention. Also, very large regions can potentially span more than one object, making them unsuitable for selection. The size feature is expressed by the normalized size r(R) = Size(R)/Image-size.

3. Color contrast: The color contrast a region shows with its neighbors affects its relative-saliency. The rationale behind choosing color contrast is that even if a region has an interesting intrinsic color, it may not be distinctive if all its neighbors also have equally interesting colors, unless it shows the greatest contrast. It is difficult to express color contrast in a numerical measure that can account for the variations in an observer's judgment with the conditions of observation, size, shape, and absolute color of the stimuli [166]. In the color contrast measure we chose, we augmented an empirical color difference formula to predict the observed color differences, with the knowledge of the hues of the colors derived from their categorical representation. Specifically, the following difference formula d(C_R,C_T) was used to measure color difference between two color regions R and T with specific colors $C_R = (r_0, g_0, b_0)^T$ and $C_T = (r, g, b)^T$ as:

$$d(C_R, C_T) = \sqrt{\left(\frac{r_0}{r_0 + g_0 + b_0} - \frac{r}{r + g + b}\right)^2 + \left(\frac{g_0}{r_0 + g_0 + b_0} - \frac{g}{r + g + b}\right)^2} \quad (3.8)$$

As this measure does not explicitly take into account the hues of the colors, the color category-based representation is used to ascertain whether the hues of the two regions are different, and then the extent of difference is judged using $d(C_R, C_T)$ in such a way that the contrast between regions of different hue is emphasized.

This allows the measure to handle simultaneous color contrast to some extent. The measure is given by c(R,T) below:

$$c(R,T) = \begin{cases} k_1 d(C_R, C_T) & \text{if R and T are of same hue} \\ k_2 + k_1 d(C_R, C_T) & \text{otherwise} \end{cases} \tag{3.9}$$

where $k_1 = \frac{0.5}{\sqrt{2}}$ and $k_2 = 0.5$, so that $0 \leq c(R,T) \leq 1.0$.

4. Size contrast: The size contrast is a feature for determining relative saliency and is chosen because it determines if a region is mostly in the background or in the foreground. The size contrast of a region R with respect to an adjacent region T is simply the relative size (area) and is given by

$$t(R,T) = min\left(\frac{\text{size(R)}}{\text{size(T)}}, \frac{\text{size(T)}}{\text{size(R)}}\right) \tag{3.10}$$

.

Since a region R has several neighboring regions in general, the color contrast $c(R)$ and size contrast $t(R)$ of a region R are measured relative to a *best* neighbor T_{best} for each region, so that $c(R) = c(R, T_{best})$, and $t(R) = t(R, T_{best})$. T_{best} is the neighboring region that is ranked the highest when all neighbors are sorted first by size, then by extent of surround, and finally by contrast (size or color contrast as the case may be).

Combining features for self-saliency:

To determine self-saliency from the chosen features, they are weighted appropriately to reflect their importance. The self-saliency measure chosen emphasizes purer and brighter colors over darker and duller colors by choosing the weighting functions for saturation and brightness as $f_1(\text{s(R)}) = 0.5\text{s(R)}$, and $f_2(\text{v(R)}) = 0.5\text{v(R)}$ respectively. The size of a region is given a non linear weight to deemphasize both very small and very large regions as they do not often attract our attention. The corresponding weighting function has sharp as well as smoothly rising and falling phases determined by the breakpoints t_1, t_2, t_3, t_4 as shown in Figure 3.10a and the equation below.[5] Here n stands for the region size r(R).

$$f_3(n) = \begin{cases} -\frac{ln(1-n)}{c_1} & 0 \leq n \leq t_1 \\ 1 - e^{-c_2 n} & t_1 < n \leq t_2 \\ s_2 - c_3 ln(1 - n + t_2) & t_2 < n \leq t_3 \\ s_3 e^{-c_4(n-t_3)} & t_3 < n \leq t_4 \\ 0 & \mathrm{t}_4 < n \leq 1.0 \end{cases} \tag{3.11}$$

where $\mathrm{t}_1 = 0.1$, $\mathrm{t}_2 = 0.4$, $\mathrm{t}_3 = 0.5$, $\mathrm{t}_4 = 0.75$, $\mathrm{s}_1 = 0.8$, $\mathrm{s}_2 = 1.0$, $\mathrm{s}_3 = 0.7$, $\mathrm{s}_4 = 10^{-3}$ and $c_1 = -\frac{ln(1-t_1)}{s_1}, c_2 = -\frac{ln(1-s_1)}{t_1}, c_3 = -\frac{(s_2-s_3)}{ln(1+t_2-t_3)}, c_4 = -\frac{ln\frac{s_4}{s_3}}{(t_4-t_3)}$ and n = size of region R = r(R).

Combining features for relative saliency

Once again, the chosen features are weighted appropriately to determine relative saliency. The color contrast is weighted linearly by a function f_4(c(R)) = c(R), to emphasize regions showing greater contrast. The relative size is exponentially weighted by a function $f_5(t(R)) = 1 - e^{-12t(R)}$ to favor situations in which a region and its best neighbor have approximately the same size.[6]

Finding self and relative saliency

Once the various features determining self and relative saliency are appropriately weighted, they reinforce each other so that the self and relative saliencies can be given by simple additive combinations of their individual features. The self-saliency of a region R denoted by SS(R) is given as f_1(s(R)) + f_2(v(R)) + f_3(r(R)). Similarly, the relative saliency of the region R, RS(R) is given by f_4(c(R)) + f_5(t(R)). Finally,

[5]Such a function along with the thresholds and rates of change was empirically derived from informal psychophysical experiments performed using color regions of various sizes. These experiments are described in Appendix B.

[6]Once again this function was obtained by performing informal psychophysical experiments.

the overall saliency of a region R is expressed by a linear combination of self and relative saliency as SS(R) + RS(R), using the following rationale. Any combination method should be flexible enough to allow a region to be declared salient if it shows good contrast (i.e., high relative saliency) even though it may not be interesting on its own. Conversely, a region that is interesting on its own but fails to become interesting in the presence of neighboring regions should not be chosen. On the basis of these observations alone, nonlinear combining methods such as (SS(R) * RS(R)) or max(SS(R), RS(R)) are not suitable. If a region is both interesting on its own as well as in the presence of other regions in the scene, then it must be given more importance. All three criteria are satisfied when the two saliency components are linearly combined. The color saliency of a region R is therefore given by

$$\text{Color-saliency}(R) = f_1(s(R)) + f_2(v(R)) + f_3(r(R)) + f_4(c(R)) + f_5(t(R)). \quad (3.12)$$

The saliency measure described above does not completely take into account all the perceptual effects of simultaneous color contrast, color-filling, etc. Because such effects do not greatly undermine a region that is already very outstanding (very salient), and because saliency is being used to rank the regions, we have ignored these effects.

The color regions in the image can now be ordered using the saliency measure and a few most significant regions can be retained for selection (called salient regions, henceforth). The number of salient regions to be retained can be determined when the selection mechanism is integrated with a recognition system to perform a specific task, and is therefore left unspecified here.

Results

We now illustrate the ranking of regions produced by the color saliency measure derived above. Figures 3.9c- 3.9f show the four most distinctive regions found by applying the color-saliency measure to all the color regions extracted from the scene shown in Figure 3.7a. Similarly, Figures 3.7d- 3.7f, 3.8d- 3.8f, 3.11c- 3.11f, show the few most salient regions found in their respective scenes. In the experiments done so far, the color-saliency measure was found to select fairly large bright-colored regions that showed good contrast with their neighbors, appeared perceptually significant and came mostly from single objects.

3.5.2 Use of Salient Color-based Selection in Recognition

Data-driven selection based on salient color regions is primarily useful when the object of interest has at least one of its regions appearing salient in the given scene. In such cases, the search for data features that match model features can be restricted to the salient regions, thus avoiding needless search in other areas of the image. By selecting salient color regions, we obtain a small number of groups (a region is itself a group), containing several features. It was shown in [27] that such large-sized groups are useful for indexing, i.e., to determine which regions from models in a library could correspond to a given group. But when the task is to recognize a single object, it is desirable to have small-sized groups. For this, existing grouping techniques can be applied to the data features found within the color regions to obtain reliable small-sized groups.

We now estimate the search reduction that can be achieved with such a selection mechanism. Let (M,N) = total number of features (such as edges, lines, etc.) in the model and image respectively. Let (M_R, N_R) = total number of color regions in the model and image respectively. Let N_S = number of salient regions that are retained in an image. Let g = average size of a group of data features, within a model or image. Let (G_M, G_N) = number of groups formed (using any existing grouping scheme) in the model and image respectively. Finally, let G_{Ni} be the number of groups in the salient image region i. Using a method of recognition called the alignment method [71], at least three corresponding data features are needed to solve for the pose (appearance) of the model of a rigid object in the image. If no selection of the data features is done, then the brute-force search required to try all possible triples is $O(M^3N^3)$. If selection is done by only grouping methods (i.e., without color region selection), then the number of matches that need to be tried is $O(G_M G_N g^3 g^3)$ since only triples within groups need to be tried. But as we mentioned before, grouping methods often make mistakes, so that not all groups contain features belonging to a single object. In at least one such study [25] out of the 150 or so groups isolated, about 83 groups actually came from single objects. Most of the remaining 67 groups would not yield any consistent match and would represent fruitless search. Consider the case when grouping of data features is done within all the color regions. With this, the grouping is more reliable, and also, the number of groups is smaller (as groups straddling regions are not considered), so that the overall effect is to reduce the search. For example, with M = 200, N = 3000, g = 7, and G_M = 30, G_N = 430 (these numbers are typical of indoor scenes), the search reduction assuming 70% reliability in simple grouping to > 95% reliability in grouping within color regions is $\approx 0.25 * 10^9$ which is a considerable improvement. Consider next when grouping is restricted to salient color regions. The number of matches further reduces to $O(\sum_{j=1}^{N_S} G_{Nj} G_M g^3 g^3)$, since only the groups in the salient regions need be tried.

To obtain an estimate of the number of matches and time taken for matching in real scenes when color-based selection is used, we recorded the number of regions (obtained by applying the segmentation algorithm of Section 3.4.2), and the number of data features within regions in some selected models and scenes (Figures 3.12 and 3.9a show typical examples of models and scenes tried). The regions were ordered using the color saliency measure and the four most salient regions were retained. Then search estimates were obtained using the above formulas, and assuming a grouping scheme that gives a number of groups within regions that is bounded by

$$\frac{\text{the number of features in a region}}{\text{average size of the groups in a region}}.$$

This is a good bound on the number of groups produced using a simple grouping scheme such as grouping 'g' closely-spaced parallel lines in the region that will be described in Chapter 5. The result of such studies is shown in Table 3.2. As can be seen from this table, the number of matches is always smaller when salient color regions are used for selection.

S.No	M	N	M_R	N_R	No selection		Only grouping		Salient color + grouping	
					Num. matches	Time	Num. matches	Time	Num. matches	Time
1.	229	1170	1	18	$1.92\text{x}10^{16}$	610yrs	$6.52\text{x}10^{8}$	11min	$3.37\text{x}10^{8}$	5min
2.	507	2655	2	20	$2.4\text{x}10^{18}$	77,341yrs	$3.22\text{x}10^{9}$	54min	$1.32\text{x}10^{9}$	22min
3.	124	2655	2	20	$3.57\text{x}10^{16}$	1131yrs	$8.05\text{x}10^{8}$	13min	$3.3\text{x}10^{8}$	5min
4.	507	2247	2	14	$1.48\text{x}10^{18}$	46,884yrs	$2.72\text{x}10^{9}$	46min	$7.8\text{x}10^{8}$	13min

Table 3.2: *Search reduction using color-based data-driven selection. The last column shows the match time when color-based data-driven selection is combined with grouping. The color-based selection is done by choosing the four most salient regions. Here $g = 7$, Time per match = 1 microsecond, and the grouping method is as described in text.*

3.6 Color-based Model-driven Selection

The previous section described a data-driven selection mechanism that was meant for an object of interest having some salient color regions. This will not be of much help when the object of interest is not salient in color (but salient in some other domain, say texture) or is not salient at all. In such cases, the color description of the model can be used to perform selection. We now describe one such color-based model-driven selection mechanism. Here, given a color-based description of a model object, the task is to locate color regions that satisfy this description. The use of model information to constrain the matching of model features to image features is not new. Several model-driven search restriction techniques such as generalized Hough transforms [75], heuristic termination [59], and focal features have evolved [8, 2, 9]. The emphasis in these methods was on geometric constraints that can prune the search space during the matching stage of recognition. The approach we present here, on the other hand, emphasizes some global relational information about model color regions to prune the search space prior to matching. It also provides possible correspondences between model and image regions. Such a correspondence can further reduce the complexity of recognition because the search for pairing model features to data features can be restricted now to these corresponding regions rather than all image regions. Color information in the model object has been used before to search for instances of the object in the given image of a scene [141, 164]. These approaches represent model and image color information by color histograms and perform a match of the histograms. Such approaches usually cause a lot of false positive identifications, and do not explicitly address some of the problems that arise in going from a model object to its instance in a scene. Also, since they do not supply correspondence between model and image regions, they are not as useful for reducing the search in recognition.

In order for any scheme for model-driven selection to be effective for reducing the search in recognition, it must meet two requirements: (i) it must be sufficiently selective to avoid many false positive selections that cause needless search for matches, and (ii) it must be sufficiently conservative to avoid many false negatives, causing recognition to fail when it should have succeeded. A selection scheme can make false negatives if it does not adequately take into account the various problems that arise in going from a model object to its image in the scene. An object may not appear the same in the scene as it does in the model, because it has undergone pose changes, or because it is occluded, or its colors appear different in the current illumination conditions. In addition, artifacts such as specularities, inter-reflections, and shadows may also cause changes in the appearance of the object. So how can a model-driven selection mechanism meet these two apparently conflicting requirements? We now describe an approach to model-driven selection that meets some of these requirements. It makes a particular choice of model description and assumes that this is made available to it for selection. Since this model description affects the way our approach formulates the color-based model-driven selection problem, it is described first.

3.6.1 Model Description

The color region information in the model (in an image or view of the model, that is) is represented as a region adjacency graph (RAG)

$$M_G = < V_m, E_m, C_m, R_m, S_m, B_{rm}, B_{sm} > \tag{3.13}$$

where V_m = color regions in the model, E_m = adjacencies between color regions, $C_m(u)$ = color of region u $\in V_m$, $R_m(u,v)$ = relative size of region 'v' with respect to region u. $S_m(u)$ = size of region u, and B_{rm} = a bound on the relative size of regions given by R_m, and B_{sm} = a bound on the absolute size of regions given by S_m.

The above description exploits features of regions that tend to remain more or less invariant in most scenes where the model appears. Even though the specific color of the model changes under variations in illumination conditions and pose changes, the perceived image color remains more or less the same if we restrict to commonly occurring lighting conditions such as sunlight, daylight, and overhead room (tube) light(s). In such cases, when the color of a model region is specified by its color category, then either the category remains the same (for small specific color changes) or it can change to a compatible color category, i.e. be predominantly of the same hue. Since the compatible categories are available in the category look-up table as described in Section 3.3.1, this variation in color can be taken into account during matching. Similarly, the adjacency information between two color regions tends to remain more or less invariant in the different appearances of the object, as long as the two regions are visible in the given image and there are no occlusions. Finally, the relative size of regions is preserved under changes of scale. But it can undergo considerable changes if the pose of the object changes, say when a region goes partially out of view. The bound on the relative size changes in each pair of adjacent regions, B_{rm} indicates the extent of pose changes that a selection mechanism is expected to tolerate. Relative size changes can also occur due to occlusions. By placing some loose bounds on the absolute size changes as given by B_{sm}, the model description restricts the changes that can be tolerated in the presence of occlusions. For size changes in a region that go beyond the bounds, that region will be considered no longer recognizable, and then the selection will have to depend on the evidence for other model regions in the image.

This description is fairly rich and has some structural information about color regions that can be used to restrict the number of false positives during selection, and some constraints on the relative and absolute size changes that can be used to restrict the number of false negatives made by the selection mechanism.

The above model description is assembled as follows. A model description specifies a color view, that is, a range of 2D views of the model in which one or more of the color regions described in the model are visible. If the model has some views showing an entirely different set of color regions, then they must be specified as separate color views. For each color view, a central model view is chosen and its color regions are extracted using the segmentation algorithm described in Section 3.4.2. Then their color, absolute size, relative size and adjacency information are recorded. By noting the relative size changes in different model views constituting a color view, the bounds B_{rm} on the relative sizes of regions are obtained. The bound on the absolute size changes are loosely set to disallow very small or very large scale changes.

Finally, the model color region description gives a way to analogously organize the color region information in the image as an image region adjacency graph as $I_G = < V_I, E_I, C_I, R_I, S_I >$, where each term has a meaning analogous to $< V_m, E_m, C_m, R_m, S_m >$ respectively.

3.6.2 Formulation of the color-based model-driven selection problem

In this section we will formulate the color-based model-driven selection problem as a type of subgraph matching problem. Given the image region adjacency graph, the model object, if present in the scene represented in the image, will form a subgraph in I_G. The location strategy can be regarded as the problem of searching for suitable subgraphs that satisfy the model description. Any such subgraph $I_g =< V_g, E_g, C_g, R_g, S_g >$ such that $||V_g|| \leq ||V_m||, ||E_g|| \leq ||E_m||$, has associated with it a node correspondence vector $\Upsilon = \{(u_m, u_g)|\forall u_m \in V_m, u_g \in V_g \cup \{\perp\}, \{\perp\}$ is a null match$\}$. Although there are an exponential number of such subgraphs, not all of them correspond to model RAG. From the model description a set of unary and binary constraints could be derived (as is described later) that make only some subgraphs feasible. A feasible subgraph is, therefore, a subgraph that has all its nodes satisfying unary and binary constraints. For model-driven selection, since it is desirable to have at most one image subgraph matching the model RAG, we can select from among these subgraphs, a subgraph(s) that in some sense best satisfies the model description. Here we formulate color-based model-driven selection as the problem of choosing a feasible subgraph(s), I_g that minimizes the following measure:

$$\mathrm{SCORE}(I_g) = \left(1 - \frac{\|V_g\|}{\|V_m\|}\right) + \frac{2\sum_{\forall (u_g,v_g)\in E_g, \Upsilon(u_m)=u_g, \Upsilon(v_m)=v_g} R^2_{mg}(u_m, v_m, u_g, v_g)}{\|E_m\|}. \tag{3.14}$$

where $R_{mg}(u_m, v_m, u_g, v_g)$ expresses the change in the relative size when adjacent model regions (u_m, v_m) are paired to corresponding image regions (u_g, v_g) and is given by $R_{mg}(u_m, v_m, u_g, v_g) = \frac{|R_m(u_m,v_m)-R_g(u_g,v_g)|}{\max(R_m(u_m,v_m),R_g(u_g,v_g))}$. SCORE($I_g$) emphasizes rewards for making as many correspondences as possible as indicated by the first term, called Match(I_g), and penalties for a mismatch of the relative size, as indicated by the second term, called Deviation(I_g), which measures the mean square deviation of the relative sizes. Since the subgraphs are all feasible, the deviation accounts for occlusions and pose changes in a more refined way than the binary constraints alone. Another advantage of this measure is that it can be incrementally computed from individual region matches, so that a branch-and-bound search formulation can be used to reduce considerably the search involved in finding the best subgraph (i.e. the one with the lowest score). Finally, the above formulation is based on the hypothesis that at least one of the regions in the isolated subgraph corresponds to a model region. It is also designed primarily to locate single instances of the model object in the image. More instances can be found after removing the regions in the found instance from the image RAG.

3.6.3 A color-based model-driven selection mechanism

A color-based model-driven selection mechanism was built using the above formulation. The mechanism essentially uses a search strategy to find the best subgraph. The result of selection is the correspondence vector associated with the best subgraph. The search strategy used the following constraints to restrict the search among feasible subgraphs.

1. Unary constraints: The color and absolute region size information provided in the model description were used to develop unary constraints on these features. The color $C_g(u_g)$ of an image region u_g is said to match the color $C_m(u_m)$ on a model region u_m if these colors belong to the same category or compatible categories (described in Section 3.3.1). With this scheme, brighter colors (of a given hue) in the model could potentially match to darker colors of the same overall hue in the image, thus accounting for a simple lowering in illumination levels. The bounds on the absolute size provided by B_{sm} (see equation 3.6.1) act as loose size constraints to rule out some clearly absurd scale changes (such as, say, a 100 fold increase in the smallest model region implying a blowup of the model outside the image bounds).

2. Binary constraints: The adjacency (as well as non-adjacency) and relative size information provided in the model were used as binary constraints to prune some impossible subgraphs. Specifically, the lack of adjacency in model regions is a powerful constraint, because two adjacent regions in the image cannot correspond to two regions that are not adjacent in the given color description (assuming a rigid model)[7]. Two adjacent regions in the model may, however, not appear adjacent in a given image due to occlusion. A simple analysis of occlusions could rule out several false matches in such cases (such as, say, discarding a match if the area spanned by the occlusion within a rectangle enclosing the candidate non-adjacent image regions far exceeds the combined size of the corresponding adjacent model regions). The bound on the relative sizes served as another binary constraint. The bound B_{rm} was used to constrain possible matches by requiring $R_{mg}(u_m, v_m, u_g, v_g) \leq B_{rm}(u_m, v_m)$.

[7]Notice here that the search is for a given color view of the model.

3. Searching for the best subgraph

The search for the best subgraph (i.e. the subgraph that minimizes the value of SCORE in equation 3.14), can in principle, be done by an exhaustive enumeration of subgraphs. But with the algorithm described below, the search required is reduced to a large extent. The algorithm used is essentially a variation of the branch and bound interpretation tree (IT) search [59], with the major difference being that no verification is done when the search reaches a leaf node (as the task is selection and not recognition). Each level of the search tree represents a possible match for a model region (this includes a null match), so that the depth of the search tree is fixed by the number of nodes in the model RAG. The unary constraints are checked *a priori* to prune the breadth of the search tree. A subgraph in the image RAG that is a potential match for the model RAG is represented by a path in the IT. The value of SCORE is updated at each node as $\text{SCORE}_{i+1} = \text{SCORE}_i - \frac{1}{\|V_m\|} + \frac{2R^2_{mg}}{\|E_m\|}$. By keeping the lowest value of SCORE so far, search can be cut off below any node with a Deviation(I_g) value greater than the lowest SCORE value. In practice, the unary and binary constraints prune the search tree considerably so that the average number of full paths (up to the leaves) explored are few ($\approx$ 50). Finally, after an instance of the model region has been found in the image, the selected area is removed and the search repeated on the resulting image RAG to look for more instances of the model object.

3.6.4 Results

The result of using color-based model-driven selection are illustrated in Figures 3.12 and 3.13. Figure 3.12a shows a model object, and its color description obtained by using the color-region segmentation algorithm of Section 3.4.2 is shown in Figure 3.12b. Here the background was removed by a simple threshold on intensities. This description is used to create a model RAG which is shown in Figure 3.13c. Figure 3.12c shows a scene in which the model object occurs. The scene shown has several other objects with one or more of the model colors. Also, the model appears in a different pose here, being rotated to the left about the vertical axis and illuminated from above. Figure 3.13e shows the result of applying the unary color constraints. For example, the big blue glass matches the small blue flowers based on color alone. Next, the unary constraint on absurd size changes are used to prune the possibilities and the result is shown in Figure 3.13f. Finally, the subgraph with the lowest value of SCORE is shown in Figure 3.13g. As can be seen from this figure, a region containing most of the model object has been identified even though the color image segmentation was not perfect (notice the small streak above the white rim of the cup that merges with the book in the background). More examples of color-based model-driven selection will be found in Chapter 8 when it is used in a recognition system.

3.6.5 Search reduction using color-based model-driven selection

The color-based model-driven selection mechanism provides a correspondence of model regions to some image regions. The matching of model features to image features can be restricted to within corresponding regions, and this reduces the number of matches that need to be tried for recognition. To reduce the search further, conventional grouping can be performed within the selected color regions, as described in Section 3.5.2. To estimate the search reduction in this case, we continue with the analysis done in that section. Let N_l be the number of solution subgraphs given by the selection mechanism, and let I_k represent one such subgraph with the number of nodes = N_k. Let (G_{u_j}, G_{v_i}) = the number of groups in region u_j of the solution subgraph I_k, and region v_i of the model RAG that corresponds to u_j as implied by the correspondence vector Υ associated with I_k. Then assuming, as before, the average size of the group = g, the number of matches that need to be tried are $O(\sum_{k=1}^{N_l} \sum_{j=1}^{N_k} G_{u_j} G_{v_i}.g^3.g^3)$. To compare this kind of selection with pure grouping we can take some typical values of these numbers. Letting M = 200, N = 3000 (where M and N are the model and image features (line segments here)), $g = 7$, $G_M = 30$, $G_N = 430$, $G_{u_j} = 8$, $G_{v_i} = 5$, $N_l = 5$, $N_k = 5$, we have the number of matches with grouping alone to be $O(G_M G_N g^3 g^3) \approx 1.56 * 10^9$, and using model-driven color-based selection with grouping, the number of matches become

M	N	M_R	N_R	Objects	N_l	N_k	No selection		Only grouping		Model-driven selection	
							Num. matches	Time	Num. matches	Time	Num. matches	Time
786	3268	5	30	20	1	(3)	1.69×10^9	530000yrs	6.15×10^9	103min	4.55×10^7	45sec
83	3078	1	20	14	3	(1,1,1)	1.67×10^{16}	528yrs	6.2×10^8	11min	1.7×10^8	3min
507	2655	2	20	14	2	(2,1)	2.4×10^{18}	77,341yrs	3.22×10^9	54min	3.72×10^8	6min
507	2247	2	14	6	1	(2)	1.48×10^{18}	46,884yrs	2.72×10^9	46min	3.16×10^8	5min

Table 3.3: *Search reduction using color-based model-driven selection. The last column shows the match time when model-color-based selection is combined with grouping. Here $g = 7$, Time per match = 1 microsecond, and the grouping method is as described in text.*

$\approx 1.25 * 10^8$. Assuming 1 microsecond as time per match this corresponds to reduction in match time from 26 minutes to $\approx$ 2 minutes. By trying several models and images of scenes where they occurred, we recorded the average number of subgraphs generated by the model-driven selection mechanism. The search estimates were obtained using the above formula for model-driven selection with grouping, and the formulas for other methods mentioned in Section 3.5.2. The results are shown in Table 3.3. The bound on the number of groups in a region was the same as used in Section 3.5.2. As can be seen from the table, the number of matches using correspondence between model and image color regions is always lower. A curious feature to note from the table is that it takes less number of matches (and hence lesser time) for a more complex model (entry 1 in Table 3.3) containing several color regions, than for a simple object with fewer regions (entry 2 in Table 3.3). This is understandable since, with a large number of regions, the constraints are stronger and hence the false matches are fewer.

Discussion: The above studies estimated the search reduction without actually integrating the selection mechanism with a recognition system. Moreover, the estimated search was based on the assumption that there were no false negatives given by the selection mechanism. Since the best match rather than an exact match is found, the selection mechanism is less likely to make false negatives. However, the best subgraph may not always correspond to the object if there is another object which can be described by a similar RAG. In such cases, the model object is usually in the best few subgraphs so that the false negatives can still be reduced by retaining such subgraphs.

3.7 Summary

In this chapter we have shown how color can be used as a cue to perform both data and model-driven selection. Unlike other approaches to color, we have used the intended task to constrain the kind of color information to be extracted from images. By restricting the tolerable illumination conditions to commonly occurring lighting conditions (daylight, room light), the category-based color specification was found sufficient for selection. The perceptual categorization of colors enabled fast color image segmentation. This color description of the image formed the basis of data and model-driven selection. A saliency measure was then developed to rank the color regions to perform data-driven selection. Lastly, an approach to model-driven selection was presented that exploited a description of model color regions to locate instances of the model in the image.

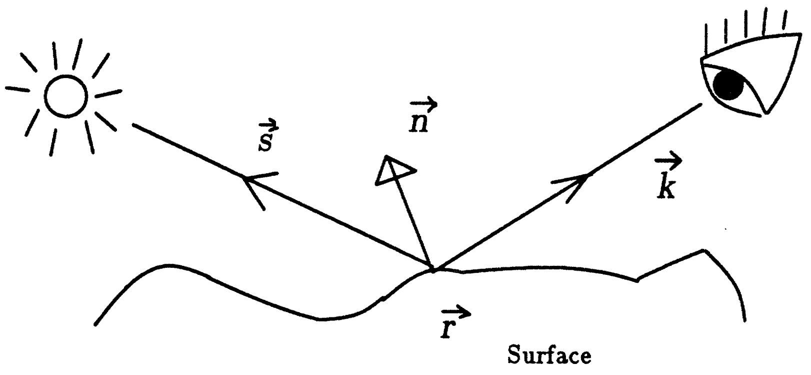


Figure 3.1: *Illustration of the factors contributing to the image irradiance at a point in the image. The surface reflectance, the surface geometry relative to the illumination source and observer direction, and the characteristics of the light source all contribute to the radiance from a surface.*

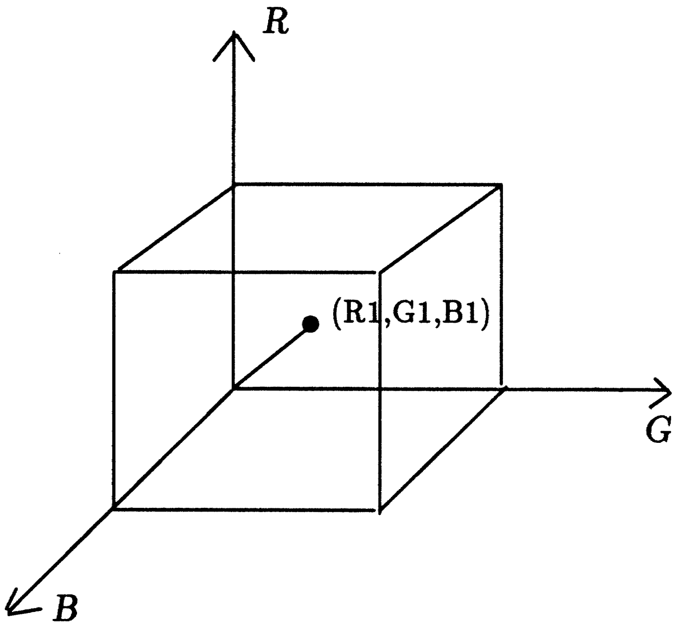


Figure 3.2: *The rgb-color space. The axes here stand for the pure red, green, and blue components of intensity obtained using the respective filters.*

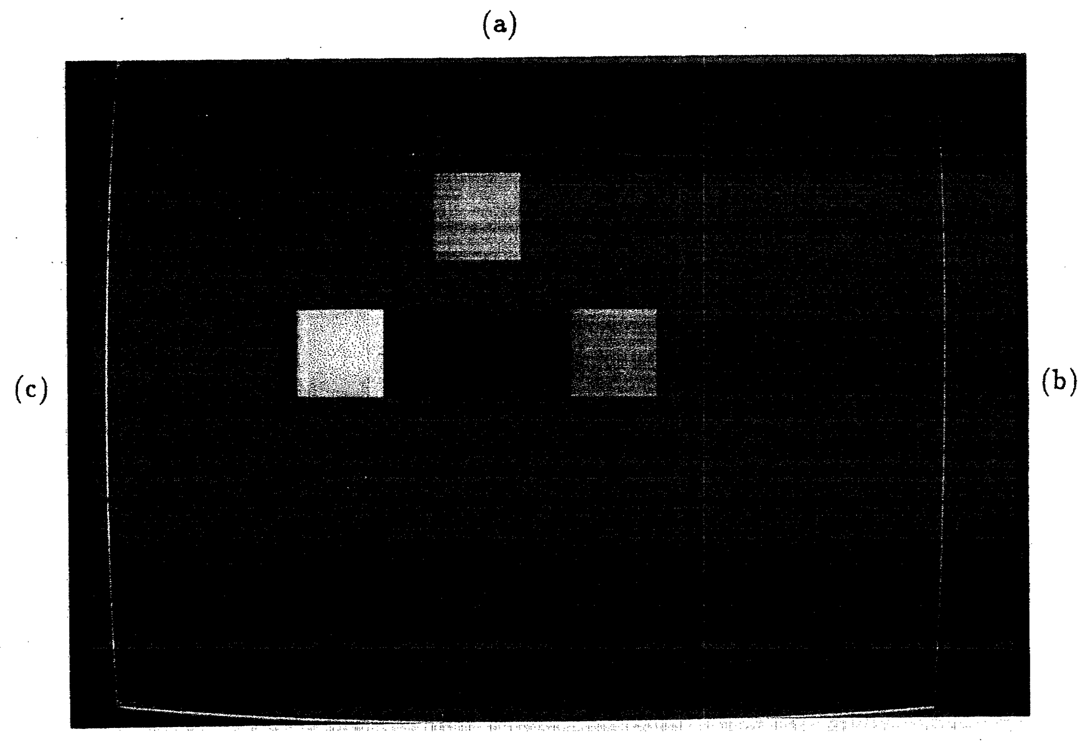


Figure 3.3: *Illustration to show that distance measures in color space do not capture the perceptual distance between colors well. (a) a color patch specified by the triple of rgb values* $< 229, 114, 124 >$. *(b) - (c) two color patches specified by the triples* $< 178, 125, 147 >$ *and* $< 229, 174, 134 >$ *respectively. These patches are equidistant (a distance of 0.1003) from the patch in (a) using the normalized distance measure described in text.*

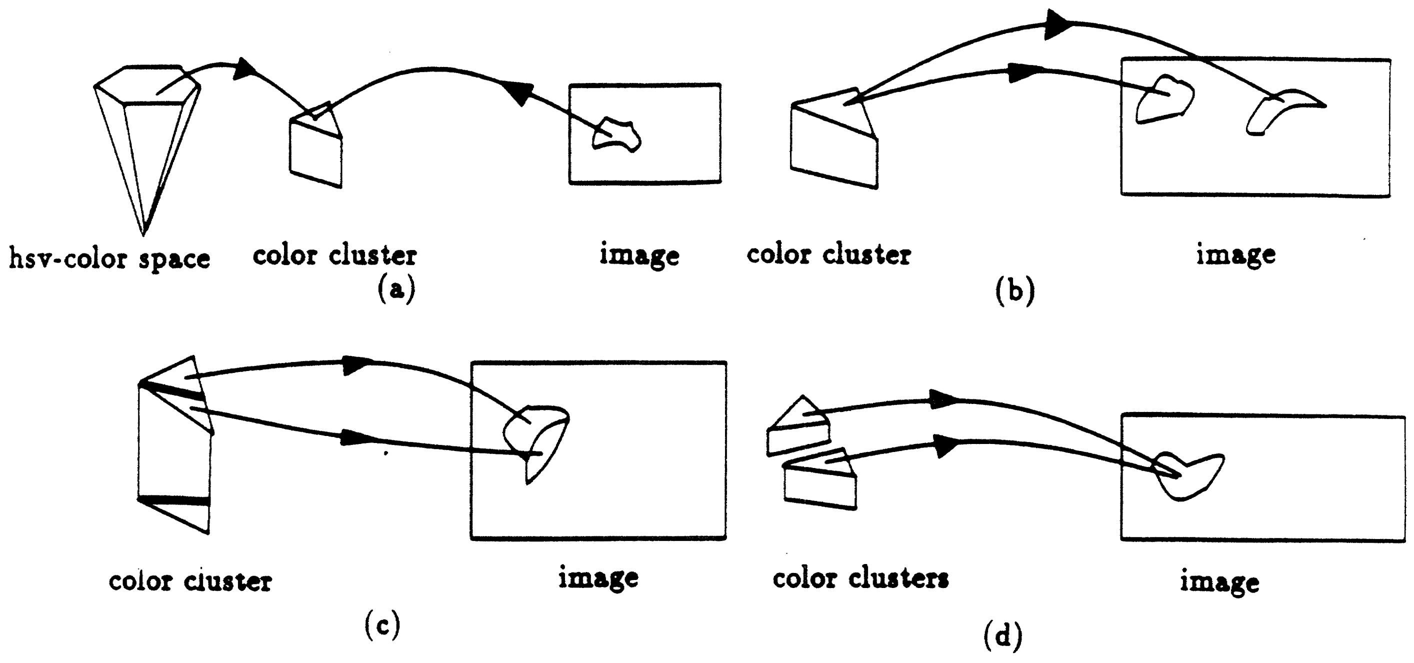


Figure 3.4: *Illustration of the problem of coordination of grouping between color and image spaces. (a) A uniformly colored image region corresponds to a cluster of a single color in color space. (b) A cluster in color space need not correspond to a single region in image space. (c) A cluster in color space need not be of a single color. (d) A smoothly shaded surface may correspond to several clusters in color space.*

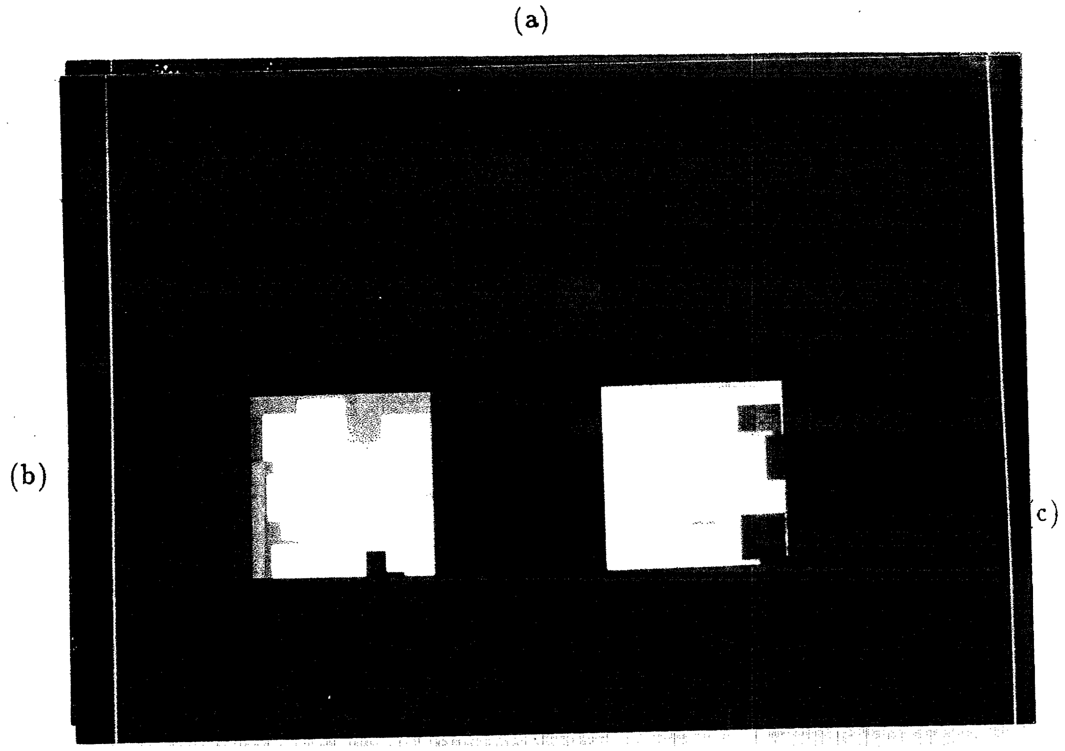


Figure 3.5: *An illustration to show that arbitrarily quantizing a color space as done in histogramming a color space can cause perceptually different colors to be grouped in a single bin. (a) A synthetic image generated from several patches whose colors belong to a single bin obtained by dividing the rgb-space into small-sized bins of dimensions 0.166* x *0.166* x *0.166. (b) The segmentation that would be produced using the histogramming approaches that use such bins. (c) The segmentation produced by our algorithm using perceptual categories. The segmented regions are shown recolored in both figures.*

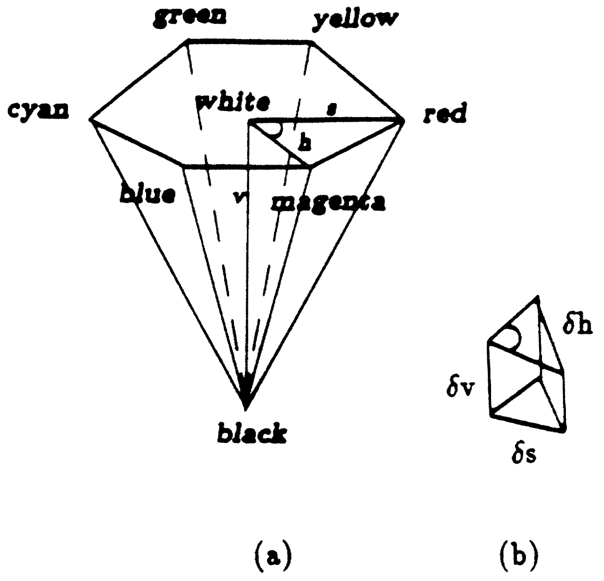


Figure 3.6: *Illustration of the quantization of the hsv-color-space. (a) Hsv-color model. (b) A cell of the quantized color space using the categorization data shown in Table 3.1.*

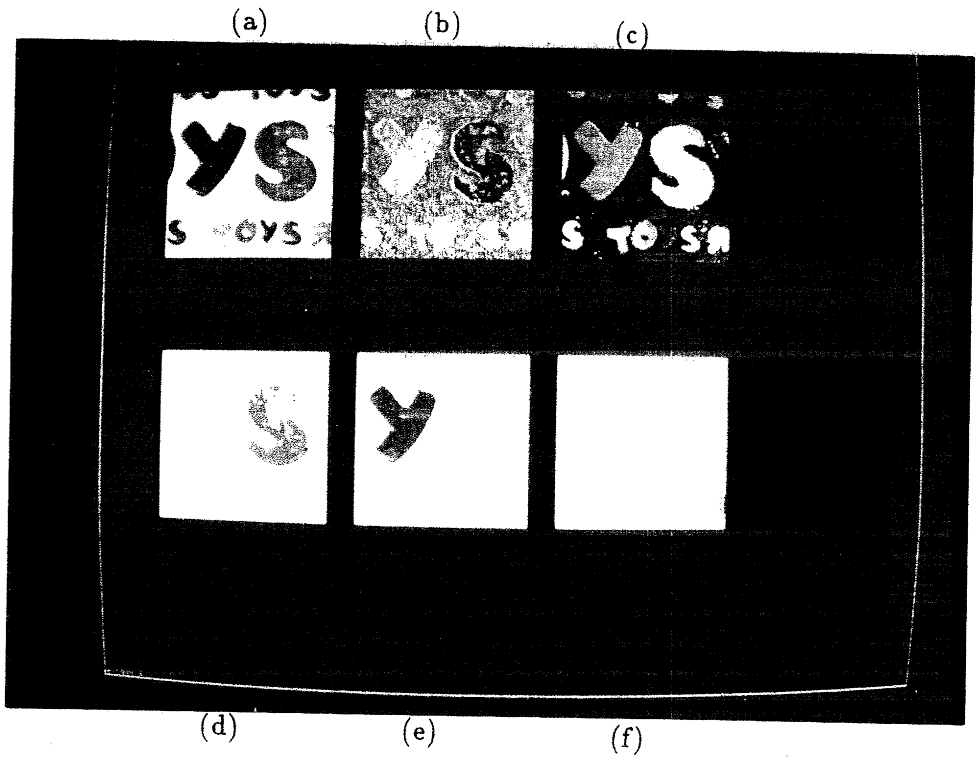


Figure 3.7: *Illustration of color region segmentation and saliency. (a) Input image consisting of regions of 3 different colors: red, green and blue against an almost white background. (b) Result of step 2 of algorithm with regions colored differently from the original image to show segmentation. (c) Final segmentation of the image of (a). (d) — (f) The three most distinctive regions found using the color saliency measure of Section 3.5.1.*

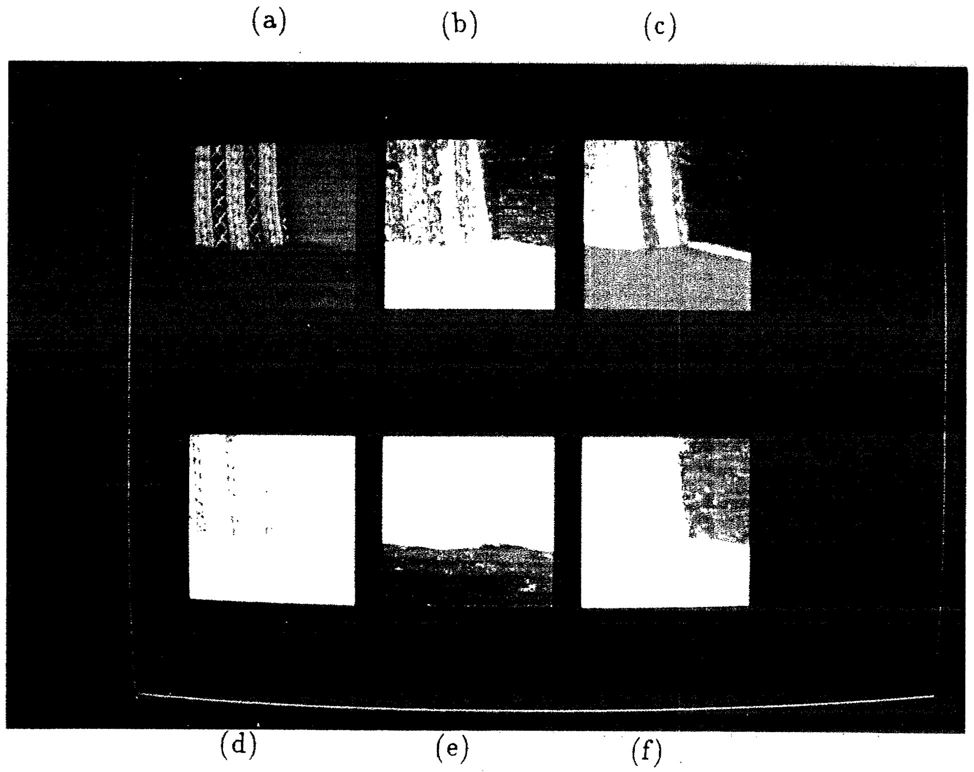


Figure 3.8: *Illustration of color region segmentation and saliency — Another example. (a) Input image of a set of colored cloth materials. (b) Regions obtained at the end of Step 2 of the algorithm (before merging overlapping regions). (c) Final segmented image suitable recolored to show the segmented regions. (d) - (f) The three most distinctive regions found using the color saliency measure given in Section 3.5.1.*

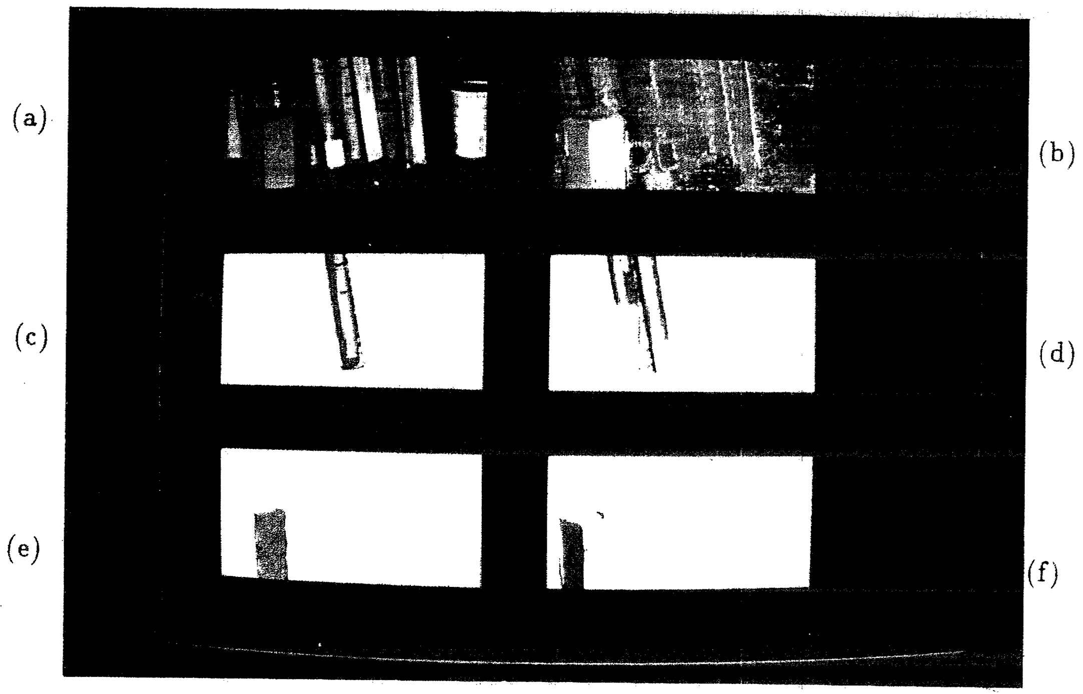


Figure 3.9: *Illustration of color region segmentation and color saliency – Another example. (a) Input image depicting a scene of objects of different materials and having occlusions and inter-reflections. (b) Segmented image using the color region segmentation algorithm. (c)–(f) The four most distinctive regions detected using the color-saliency measure. The white portion in the red book appears so because of the white background.*

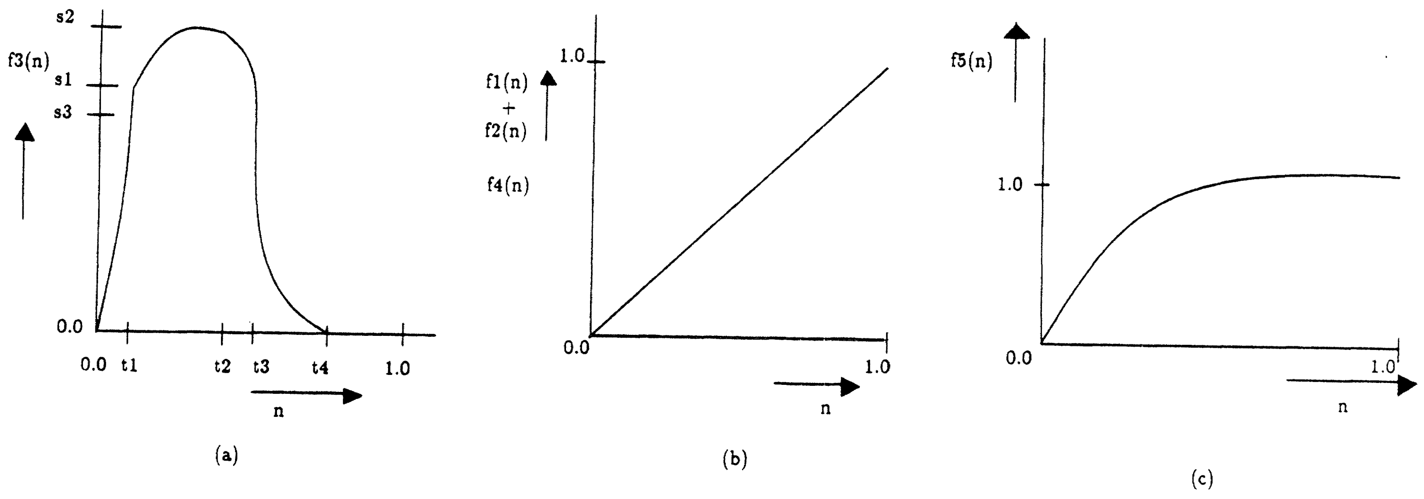


Figure 3.10: *Graphs of weighting functions used in devising the color-saliency measure.*

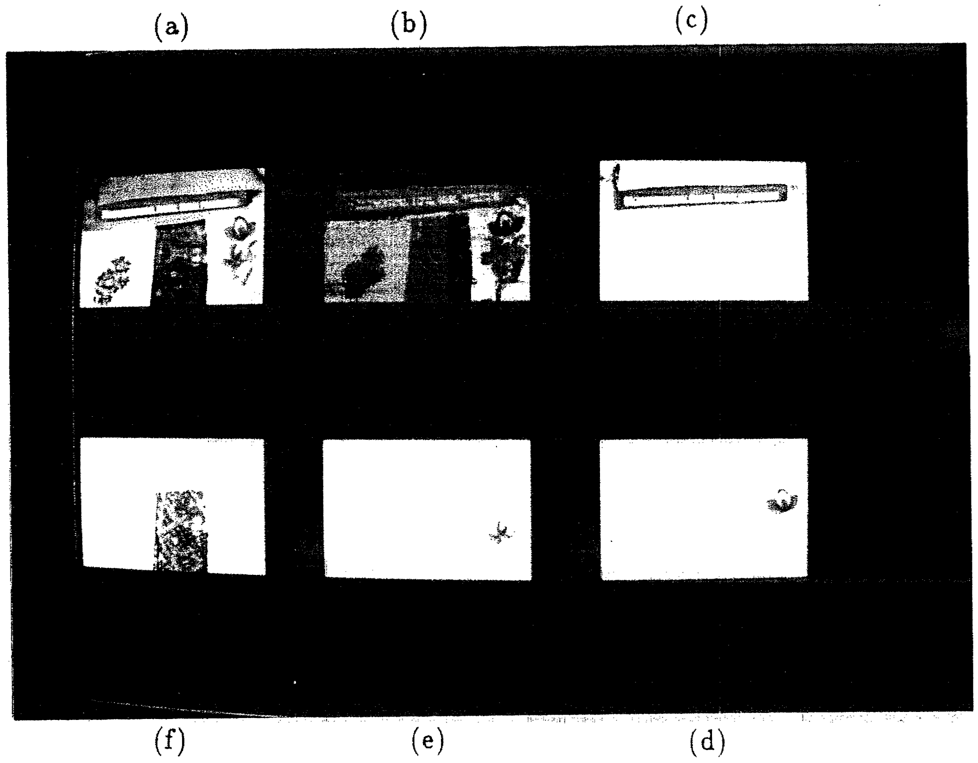


Figure 3.11: *Illustration of color region segmentation and color saliency — Last example. (a) Input image depicting a scene of different kinds of objects (cloths and polished book). (b) The color regions extracted from (a) using the color region segmentation algorithm. (c) — (f) The four most distinctive regions detected using the color saliency measure.*

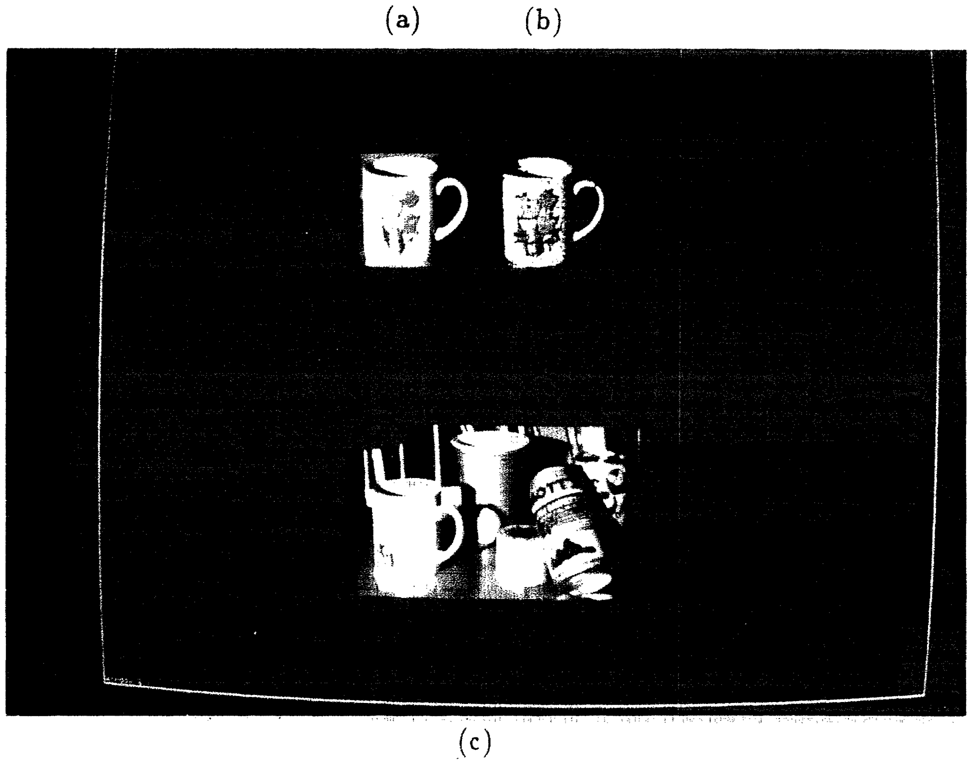


Figure 3.12: *Illustration of model-driven selection using color. — Model and scene. (a) The object serving as the model. (b) Its color regions extracted using the color segmentation algorithm. (c) A cluttered scene in which the object appears.*

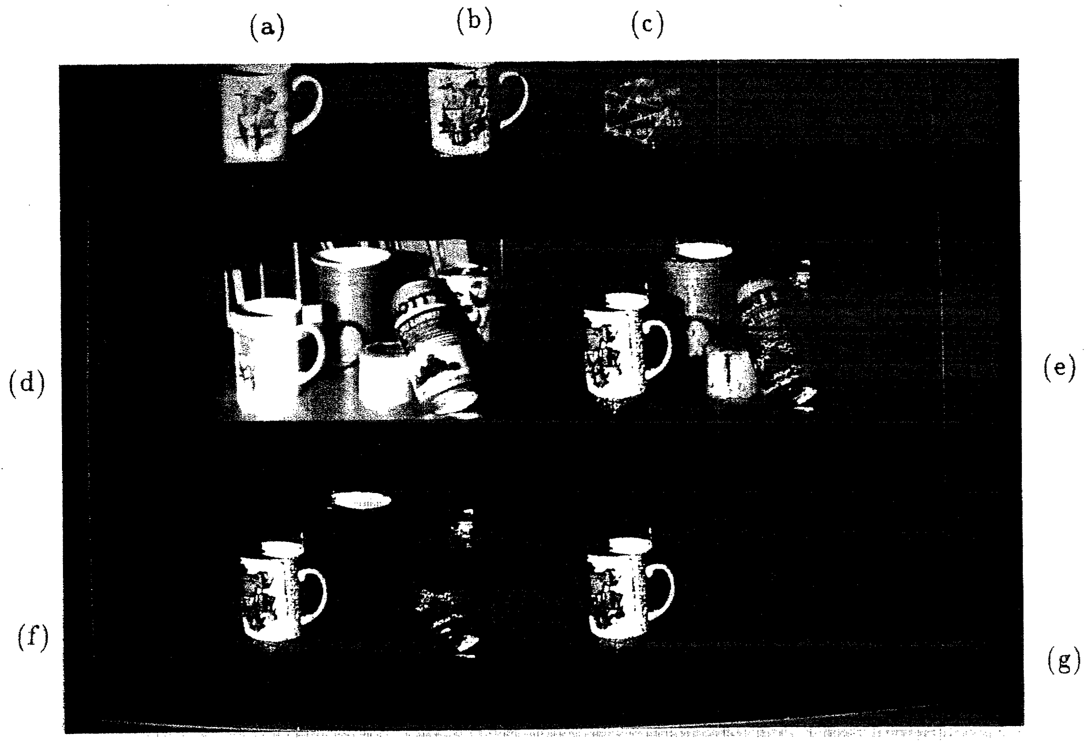


Figure 3.13: *Illustration of color-based model-driven selection.(a) The object serving as the model. (b) Its color regions extracted using the color segmentation algorithm. (c) The color region adjacency graph description of the object of (a). (d) A scene containing the model object of (a). (e) Regions selected based on unary color constraint. (f) regions of (e) pruned after using the unary size constraint. (g) Regions corresponding to the best subgraph that matched the model specifications.*

Chapter 4

Data and Model-Driven Selection using Texture Regions

The previous chapter indicated how data and model-driven selection could be performed using color. In the absence of color information, or in cases where color is insufficient to distinguish between objects, texture or the grey-level pattern on an object can be used to perform a selection of the scene. In this chapter we present an approach to data and model-driven selection using texture, using again the paradigm of attentional selection. We begin by discussing why texture is a good cue for selection, and outline the requirements imposed by selection on any method of texture analysis. An approach to texture region segmentation that satisfies these requirements is then presented. It employs a representation of texture called the linear prediction spectrum, and a model of the texture image formation process to roughly isolate texture regions. These regions are subsequently ranked by a measure of texture saliency in accordance with our approach of extracting salient regions for data-driven selection. Next, we show that the linear prediction spectrum is also a suitable representation for model-driven selection as it can solve for the pose of the texture on a model object when it is present in a given image. Finally, we discuss the utility of the texture-based selection mechanisms for object recognition.

4.1 Texture in Selection

4.1.1 Role of Texture in Selection

Texture or pattern on objects in a scene often attracts an observer's attention. Several psychophysical experiments have shown that texture like color, is one of cues that is pre-attentively perceived [147, 39, 78, 79]. We often also remember interesting patterns on objects and use them to search for such objects in a scene. It therefore, seems natural to use texture as a cue to select aspects of a scene. A textured region, if reliably isolated, almost always comes from a single object (or a single part of a scene) and can serve as a reliable grouping method for data-driven selection. This reliability can be enhanced when only a few salient texture regions are considered. Also, such salient texture regions can form fairly large groups of edges, making them suitable for indexing into a library of models [26]. These features make texture a useful domain to perform data-driven selection. Because the texture pattern on an object appears more or less the same, under most illumination conditions, and for most poses of objects in scenes it can, if appropriately described, be a stable cue to search for instances of a model object in an image of a scene. This makes texture suitable also for model-driven selection.

In order for texture to be useful for data-driven selection, a good representation of texture is needed that can lead to the reliable isolation of the various texture regions in an image. Reliable isolation, however, does not require precise localization of boundaries of different texture regions in an image. For data-driven selection, it is sufficient if we could tell roughly if one portion of the image is sufficiently different from the adjacent portions with respect to textural information.

For model-driven selection, on the other hand, a stable description of the texture on the model and a good matching strategy is required that would account for the illumination and orientation changes, occlusions, etc., that make an instance of the model object in a given image appear different from its original description. Finally, for both data and model-driven selection using texture, computational feasibility should be of concern as selection is only meant to be a preprocessing stage to recognition. We now review some of the existing methods of texture descriptions in light of these requirements.

4.1.2 Texture description methods

The problem of texture description has been addressed before for a variety of applications including texture segmentation [110, 144], classification [163, 161, 122], and synthesis. These methods can be classified based on whether they examine the spatial domain, the frequency domain, or both, for extracting textural features. Spatial domain methods often model either stochastic or structural textures. A structural texture can be characterized by a set of primitives called textons together with a placement rule for the textons to yield the overall texture. A stochastic texture on the other hand, does not show easily identifiable primitives, and no apparent deterministic placement rule. For structural textures, the texture descriptors are formed by extracting textons and computing their density and other statistics [78, 158]. But the reliable isolation of textons for the purpose of segmentation of such textures has been found to be difficult. For stochastic textures, a number of techniques are available such as those that are based on correlation [22] and grey-level co-occurrence methods [96, 127], that extract features from grey level information and compute statistics such as first and second order moments, entropy, etc. For most textures, grey-level-based statistics are insufficient to distinguish between textures since it is possible to produce two textures that obey similar laws of grey-level placement but could give rise to perceivably different textures. Further, such methods are usually sensitive to changes in spatial extent, and illumination conditions.

A number of other approaches have used parametric descriptions of textures for purposes of texture synthesis, segmentation, and classification [137, 21, 103, 33, 50]. These include autoregressive models [137, 21, 103, 64, 121, 82], auto-regressive moving average models [142], Gibbs random field models [35], Gaussian markov random field model [20], and Markov random field models [30, 81, 11]. They differ in the way spatial dependencies between intensities at neighboring pixels are exploited. Some of these models have interesting properties. For example, a rotation-invariant autoregressive model was presented in [83], that could have implications for texture recognition. However, these models have been mostly applied to aerial texture and textures from the Brodatz album [16]. They have not been applied to the kind of pattern-like textures on 3D objects that occur in natural scenes, and so their applicability is not well-known. Later we will examine one such model of texture and examine its relevance for selection.

A number of techniques are available that exploit the Fourier domain to describe textural information [45, 94, 1, 43, 4]. Some of these attempt a multi-channel decomposition of textural information in the Fourier domain as was done using Gabor filters [13, 99, 114]. Other methods extract features based on the power spectrum of the texture [94, 1]. Frequency domain methods have some advantages. In particular, the spectrum-based methods can capture important textural information as peaks in localized portions of the Fourier spectrum [3, 4, 94]. They also possess the capability to handle spatial extent differences and minor illumination changes [94]. In such methods, spatial extent differences can be expressed as effects of windowing the data[1], an operation that distorts the spectrum but mostly retains the identity and location of spectral peaks. Similarly, smooth intensity changes (i.e. those that can be approximated by a delta function in the Fourier domain) affect the amplitude of the peaks more than their location and identity. Some internal variations in textures can also be tolerated by these methods since these may appear as low amplitude spurious peaks. These features make Fourier descriptions of textures suitable for both data and model-driven selection. However, their main disadvantage seems to be the expense in computing the 2D Fourier transform, and the difficulty in designing good power spectral difference measures, for subsequent analysis of textures [94, 1].

Joint spatial/frequency approaches to texture also exist. They exploit representations such as the Wigner distribution [122, 123, 124], the wavelet transform [100], that indicate the frequency content in localized regions in the spatial domain. However, such representations are prohibitively expensive sometimes requiring, frequency information per image pixel to be recorded.

[1]We will see this in detail in Section 4.2.1.

4.1.3 Approaches to texture segmentation

Even though some of the above methods of texture description capture important properties of texture, using them to reliably extract the various texture regions in the images has been found to be difficult. This is because such texture description methods are good provided the region being analyzed is of one texture. However, when the problem is to find these texture regions, methods that analyze the image by applying such description measures to arbitrary regions (containing more than one texture) do not yield good segmentation. Existing approaches for analyzing a texture image use texture descriptors such as the ones described above and use methods such as region growing [3, 23, 119, 22], clustering, thresholding, relaxation labeling, etc. [28, 113], to group the various textured regions. Region growing methods [23, 119] typically compute some textural properties over some window in the image and grow such seed regions by grouping adjacent windows with similar textural properties. For such methods there is no clear choice of window size for all textures. A window that is too small may not be able to capture textural features, while a large window could potentially span more than one texture. Others use a split and merge scheme for segmentation by measuring the similarity of textures over hierarchically organized window sizes [113]. Clustering approaches try to find clusters in some feature space with features assembled for each pixel in the image and assign image pixels to the closest cluster based on the feature vector at that pixel [28, 132]. Such methods often require some knowledge of the number of different textures in the image. Thresholding approaches similarly identify modes in a histogram of some texture feature computed from image pixels and assign pixels to the most representative mode of the histogram with each mode taken to represent a texture category. Since neighborhood dependence of pixels is not exploited in such schemes, the resulting segmentation often tends to be spotty. Finally, estimation theoretic methods assign a region type to each pixel based on analyzing the likelihood of a given set of texture regions arising in a given image, with the region type assignment constrained to minimize some measure of fit [12, 142, 35]. Such methods often need some supervision on the choice of initial regions to characterize textural information, and also some knowledge of the total number of textured regions in images.

Thus most of the existing approaches to texture region segmentation tend to be computationally expensive and result in problems of over or under segmentation without additional supervision.

4.2 An Approach to Texture Region Segmentation

We now present an approach to texture region segmentation that uses a model of the texture image formation process to arrive at a way to analyze the image. The segmentation is then performed using a texture description that combines the advantages of spatial and frequency domain approaches to texture.

4.2.1 Texture image representation

Most approaches to texture make an implicit assumption that the textured image comes from a stationary stochastic process [1, 63]. Since each textured region in the image usually differs sufficiently from other regions, the stationarity assumption over the entire image is not generally valid. Our approach to texture segmentation is based on the following model of texture image formation. The texture image is modeled as having been generated by a short-space stationary stochastic process. That is, the characteristics of the process generating the image are assumed to be stationary over a region of space characterizing a certain texture (hence short-space). The implications of the short-space stationary assumption is that a signal (texture image here) should be analyzed in portions rather than in its entirety. This is reflected in the Fourier representation of such signals, called the short-space Fourier transform given by [124, 93]:

$$F(n_1, n_2, \omega_1, \omega_2) = \sum_m \sum_l s(m,l) w(n_1 - m, n_2 - l) e^{-j\omega_1 m} e^{-j\omega_2 l} \tag{4.1}$$

where $s(m,l)$ is the short-space stationary signal (image), $w(m,l)$ represents a window of finite duration N_1 x N_2, and (n_1,n_2) stand for the shift of the origin of the window. Like its 1D counterpart, the short-time Fourier transform [120], the short-space Fourier transform (SPFT) also has a moving window interpretation. That is, the SPFT can be interpreted as the Fourier transform (FT) of a portion of the signal $s(m,l)$ defined by the current position of the window $w(m,l)$ (given by the shift (n_1,n_2) as shown in Figure 4.1). Thus the SPFT captures the frequency variations in the short-space stationary signal over a short window duration. From this it is apparent that the SPFT captures both spatial and frequency variations in the texture image.

How does representing a texture image by its SPFT lead to a method of texture segmentation? The sliding window interpretation of the SPFT suggests a way to analyze the texture image. We can slide a window over different portions of the image and notice differences in the SPFTs of adjacent windows. Specifically, we can look at the power spectrum of the SPFT of each window and identify any difference in the high energy peaks of the respective power spectra. This is based on the following rationale. The SPFT for a given window describes the Fourier transform and hence the Fourier power spectrum of the textured region trapped by the currently sliding window, albeit distorted due to convolution with the window's transform. That is, if a window spans a texture region $s(m,n)$, its SPFT is the FT of $s_1(m,n) = s(m,n)w(m,n)$. If an adjacent window spans the same textured region, we expect the SPFT's of the two windowed regions to be similar since the two windowed regions are two portions of the same texture region $s(m,n)$, even though the window displacement is different in the two cases. However, if the adjacent window spans another texture region, i.e., contains a boundary, then the signal within that window $s_2(m,n)$ can be written as $s(m,n)w_1(m,n) + s'(m,n)w_2(m,n)$ where

$$w_1(m,n) = \begin{cases} 1 & \text{over the portion containing s(m,n)} \\ 0 & \text{elsewhere} \end{cases} \tag{4.2}$$

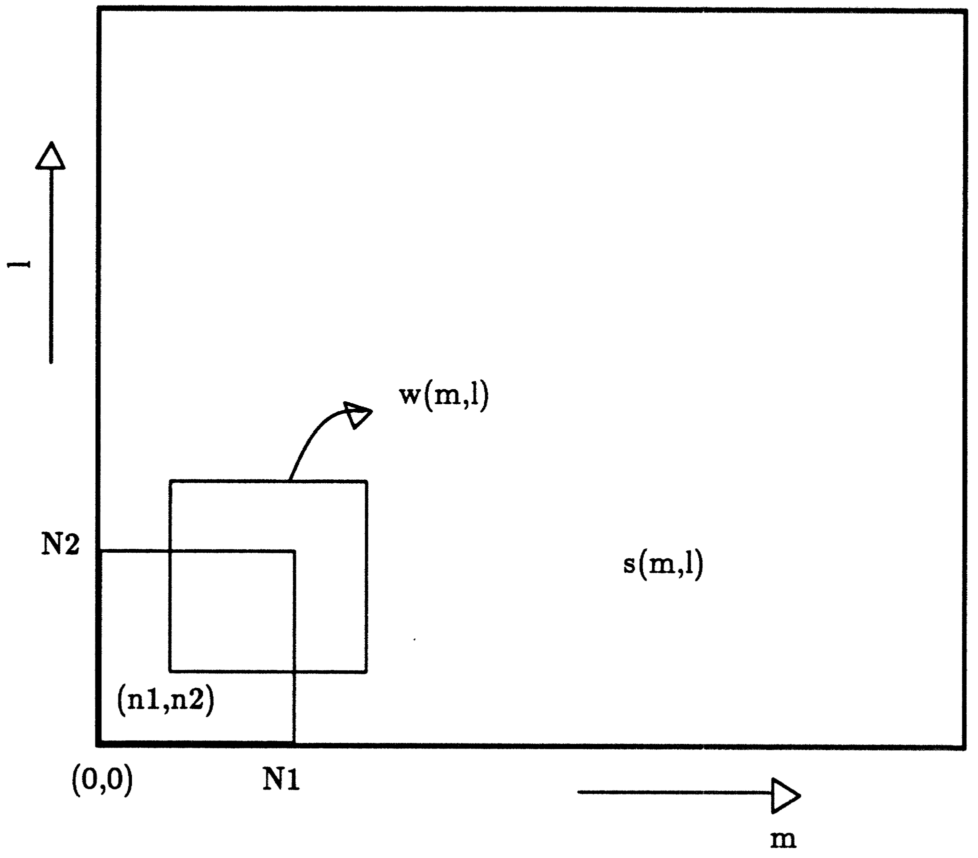


Figure 4.1: *Illustration of moving window analysis for computing the short-space Fourier transform.*

and

$$w_2(m,n) = \begin{cases} 0 & \text{over the portion containing s(m,n)} \\ 1 & \text{elsewhere} \end{cases} \quad . \tag{4.3}$$

Thus $s_2(m,n)$ can be thought of as consisting of truncated windowed versions of the two texture regions $s(m,n)$ and $s'(m,n)$[2]. The SPFT of the new windowed region is the sum of the Fourier transforms of the portions $s(m,n)w_1(m,n)$ and $s'(m,n)w_2(m,n)$. Such a FT's spectrum will, in general, appear different from the spectrum of the earlier window, both in the location and the amplitude of high energy peaks (depending on the amount of overlap between the two transforms). A similarity/difference detector could, therefore, pick up such differences and postulate a boundary between adjacent regions.

Thus by doing a moving window analysis and looking at spectral peaks in the SPFT spectra of adjacent windowed regions, we can predict boundaries between regions. Later, we show how such boundary fragments can be combined together to yield a texture segmentation of the image. But first, we address the issues of the choice of the shape, the duration, and the extent of overlap between windows, as they affect the reliability with which the different texture regions in an image can be isolated.

Choice of Window

From windowed Fourier analysis, we know that the shape of the window can introduce distortion in the shape of the transform and hence the power spectrum of the signal. Here we will restrict to rectangular windows for reasons of simplicity, although better windows such as the Hamming, Hanning, and Kaiser windows are available that smooth the distortion produced by the window shape[111].

[2]In case the windowed region has portions of more than two texture regions, then $s'(m,n)$ can represent all the regions other than $s(m,n)$.

Duration of window

Ideally, the least distortion of the Fourier transform of the signal is achieved by a window of infinitesimally narrow bandwidth (i.e. of infinite duration). However, a large duration is not desirable as it can span more than one stationary region[3]. A constraint on the choice of window size for texture image segmentation, therefore, can come from some a priori knowledge about the smallest size of textured region present in an image of a scene. In our experiments, we have chosen window sizes ranging from 32x32 to 64x64. With this choice, the bandwidth of the rectangular window's spectrum along each dimension, given by F_s/N, where F_s is the sampling rate of the image, and N, the extent along that dimension, is $= F_s/64$. This is a sufficiently narrow bandwidth for most applications.

Overlap between windows

The overlap between windows is an important factor as it decides the amount of aliasing in the short-space Fourier transform. This has implications in the validity of the comparison of Fourier spectra in two adjacent windowed regions. The adjacent windows could have a maximum overlap of $N - 1$ x $N - 1$, i.e., could be spaced 1x1 pixels apart. But this would violate the computational feasibility requirement of selection, as an enormous number of windowed regions need to be examined. Fortunately, Nyquist sampling of the SPFT specifies a minimum amount of overlap that must exist between adjacent windows, thus keeping the number of windowed regions to be analyzed computationally feasible. By extending the method described in [120] to 2D, we derive such a minimum overlap as follows. Since the SPFT is a function of space and frequency, the shift in space between two adjacent windows (and hence the overlap between adjacent windows) gives the sampling rate in space. This can be found by fixing the frequency variables to, say, $(\omega_{10}, \omega_{20})$ and examining the resulting SPFT which is now a function of space alone as

[3]A finite duration window also ensures the existence of the SPFT as it makes the resulting sequence absolutely summable.

$$F(n_1, n_2, \omega_{10}, \omega_{20}) = \sum_m \sum_l f(m,l) e^{-j\omega_{10}m} e^{-j\omega_{20}l} w(n_1 - m, n_2 - l) \qquad (4.4)$$

where some of the terms from equation 4.1 are rearranged to reflect a new interpretation of the SPFT. From the above equation, the SPFT for a fixed frequency can be thought as the convolution of the sequence $f(n_1, n_2)e^{-j\omega_{10}n_1}e^{-j\omega_{20}n_2}$ with a filter whose impulse response is given by $w(n_1, n_2)$. In frequency domain, this corresponds to a multiplication of the displaced FT of the signal $f(m,l)$ with the FT of the window. Most windows have a low pass filter response with a narrow passband around zero frequency ($\omega_{10} = \omega_{20} = 0$). Thus the FT of $F(n_1, n_2, \omega_{10}, \omega_{20})$ also is low pass and bandlimited by the window's spectrum. If the bandwidth of the window is B, then $F(n_1, n_2, \omega_{10}, \omega_{20})$ needs a sampling rate of at least $2B$ samples per second according to the Nyquist rate. Since the bandwidth of the rectangular window is about F_s/N in each dimension, the minimum distance between consecutive samples of $F(n_1, n_2, \omega_{10}, \omega_{20})$ should be $1/2B = N/2F_s = N/2$ in terms of the pixel spacing in the textured image. Since the shift between consecutive windows must be at most $N/2$ pixels, it means that the minimum amount of overlap between adjacent windows is at least $N/2$.

Relevance of minimum overlap for texture segmentation

The minimum overlap between consecutive windows was determined to avoid aliasing in the SPFT. But is it necessary to maintain such a minimum overlap in order to make texture segmentation valid? It is true that as long as the underlying textures in any two adjacent windows are sufficiently different, it should show in their respective Fourier power spectra. But to ensure the recovery of all the possible textured regions (i.e. to ensure that there is no under-segmentation), this minimum overlap must be maintained. This can be seen as follows. The SPFT is meant to capture both the spectral and spatial variations in a short-space stationary signal. The adequate sampling rate condition in the space domain indicates that unless this is met, the missing values of $F(n_1', n_2', \omega_{10}, \omega_{20})$ for a fixed frequency location

$(\omega_{10}, \omega_{20})$, at an arbitrary shift (n_1', n_2') are not recoverable from the known sample values at (n_1, n_2). Since this is true for any fixed frequency, it implies that the spectral variations in the transform $F(n_1, n_2, \omega_{1i}, \omega_{2i})$ for the signal in the window with shift (n_1', n_2'), are not capturable at all frequencies $(\omega_{1i}, \omega_{2i})$ (see Figure 4.2 for an illustration of this argument). Thus when spectra of adjacent windows are compared, if the spectrum in the intermediate window (implied by the shift (n_1', n_2')) is different from the previous window, this will not be detected and a potential boundary (and hence perhaps a textured region) may be missed by such a segmentation scheme. On the other hand, if the SPFT is adequately sampled in space (by maintaining the minimum shift), the missing spectra of the intermediate window carries no new information as far as the texture image is concerned and hence can be skipped during comparison of adjacent windows.

Discussion

Thus we see that having a model of texture image formation process gives a way to analyze a texture image (moving window analysis) and also a way to describe the texture information within each windowed region (by its SPFT). A segmentation algorithm could then be designed that predicts boundaries between adjacent regions by comparing their SPFT spectra. Such a segmentation algorithm would not precisely locate boundaries but would still satisfy the rough segmentation requirement for data-driven selection. However, it would not satisfy the computational feasibility requirement as it involves the computation of at least a $2N$ x $2N$ Fourier transform for each analysis window of duration N x N. The next section presents a compact way of describing textural information within an analysis window that leads to a computationally feasible texture segmentation algorithm.

4.2.2 Linear Prediction Spectrum-based description of textures

Consider a texture region $s(m, n)$ in the texture image where m and n stand for the two coordinate axes of the discrete image plane and s is the intensity signal at those

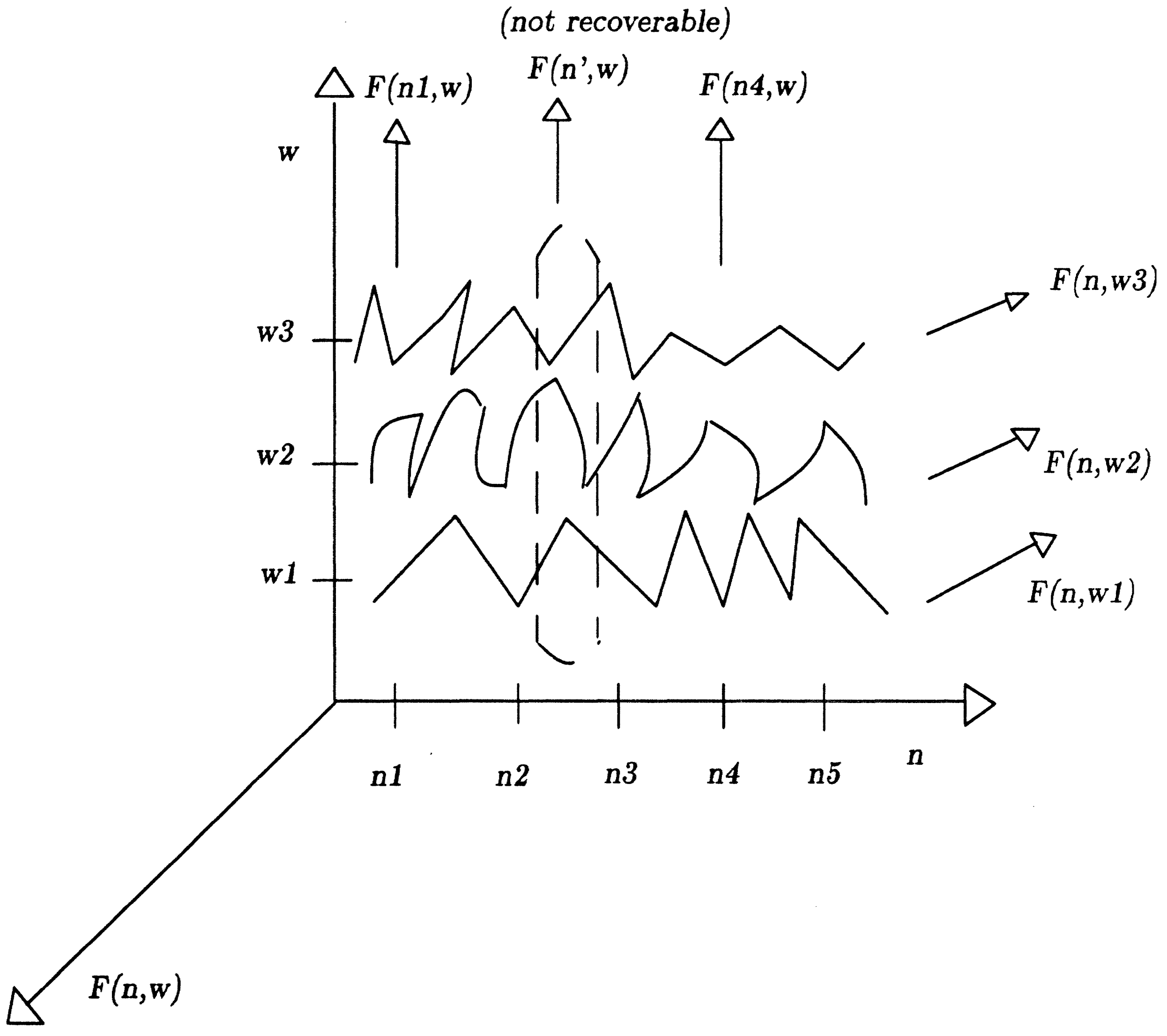


Figure 4.2: *Illustration of the need for minimum overlap for complete texture segmentation. The different waveforms depict short-space Fourier transform evaluated at different ω-coordinates.*

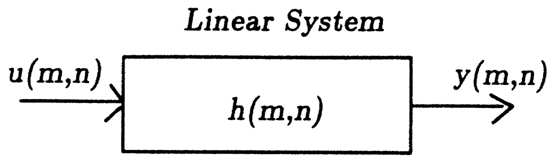


Figure 4.3: *Illustration of linear system analysis. Here a signal $y(m,n)$ is regarded as the output of a black box (the linear systems) that is fed with some unknown input signal $u(m,n)$.*

locations. If the texture region $s(m,n)$ is obtained by using a sampling rate much above the Nyquist rate, then it is well-known that its samples are dependent on each other [74]. One way to model this dependency is to say that the signal sample at a location can be obtained from its neighboring samples in a linear predictive fashion. That is, the 2D texture region $s(m,n)$ can be modeled as the output of a 2D linear system with some unknown input u(m,n) so that the texture can be characterized by the impulse response of this linear system as (see Figure 4.3)

$$s(m,n) = \sum_{k=-P_{(k,l)\neq(0,0)}}^{P} \sum_{l=-Q}^{Q} a_{kl}s(m-k,n-l) + Gu(m,n) \tag{4.5}$$

where a_{kl} are the coefficients of the impulse response of the linear system and are called the autoregressive (AR) model parameters or coefficients. G is the gain of the system, and $(2P+1)$ x $(2Q+1)-1$ represents the order of the AR model, i.e., the number of coefficients of the linear system.

The parameters a_{kl} can be estimated in several ways. One popular way is to minimize the overall prediction error between $s(m,n)$ and its approximation $\hat{s}(m,n)$ given by

$$\hat{s}(m,n) = \sum_{k=-P_{(k,l)\neq(0,0)}}^{P} \sum_{l=-Q}^{Q} a_{kl} s(m-k, n-l) \tag{4.6}$$

using the rationale that since the input u(m,n) is, in general, not known, $s(m,n)$ can only be approximately predicted as a linearly weighted sum of neighboring samples.

The prediction error per sample is then given by

$$e(m,n) = s(m,n) - \hat{s}(m,n) = s(m,n) - \sum_{k}\sum_{l} a_{kl} s(m-k, n-l) \tag{4.7}$$

where the limits in the summation are as given in equation 4.6. The total squared error in such an approximation is given by

$$e = \sum_{m}\sum_{n} (s(m,n) - \hat{s}(m,n))^2. \tag{4.8}$$

This approximation can be made best in the minimum squared error sense by differentiating e with respect to the parameters a_{kl} and setting to zero to yield the following set of linear equations to solve for the AR parameters [98, 41]:

$$\sum_{k=-P_{(k,l)\neq(0,0)}}^{P} \sum_{l=-Q}^{Q} a_{kl} R(k-i, l-j) = R(i,j), -P \leq i \leq P, -Q \leq j \leq Q, (i,j) \neq (0,0) \tag{4.9}$$

where R(i,j) is the autocorrelation function given by

$$R(i,j) = \sum_{m}\sum_{n} s(m,n)s(m+i, n+j), -P \leq i \leq P, -Q \leq j \leq Q, (i,j) \neq (0,0). \tag{4.10}$$

Discussion:

With the linear prediction or autoregressive model of a texture, the texture in a region can be compactly described with a handful of AR parameters. Such AR modeling of textures has been used before for synthesizing several natural textures [21] and is, therefore, a plausible model even for representing natural textures. Also, the estimation of the AR parameters requires the solution of linear equations (Equation 4.9) and the computation of only a few autocorrelation values (Equation 4.10) making it less expensive than taking the Fourier transform. Since the AR computation takes all samples of the signal into account, it tends to average out minor textural variations across a texture and also spatial extent differences, so that the estimation of these parameters is less sensitive to exact spatial extent and minor textural differences. This can be useful for texture segmentation since a given texture region spanning windows can be modeled in these windows by parameters that are very close in value.

Although there are several advantages to representing the textures as AR models, the AR parameter domain itself is not very good for examining differences in the textural content of adjacent windows. It has, nevertheless, been used before for texture segmentation [137, 12]. In [137] for example, a chi square test of similarity was derived from the AR parameters which would mainly detect textural differences in stochastic textures but not in structural textures. In general, tests of textural differences formed from the AR parameters can mainly detect strong differences and cannot account for large variations in illumination and orientation changes, and natural statistical variations. So, following the advantages of the Fourier domain representation mentioned in Section 4.1.2, we now examine a frequency domain interpretation of the AR parameters called the linear prediction spectrum.

Linear prediction spectrum: The transfer function[4] of the linear system in equation 4.5 when expressed in frequency domain (ω_1, ω_2) gives

$$H(\omega_1, \omega_2) = \frac{S(\omega_1, \omega_2)}{U(\omega_1, \omega_2)} = \frac{G}{(1 - \sum_k \sum_l a_{kl} e^{-j\omega_1 k} e^{-j\omega_2 l})}. \qquad (4.11)$$

The linear prediction (LP) spectrum is defined as the power spectrum of the transfer function and is given by

[4]The transfer function of a linear system is the ratio of its output signal to its input signal.

$$L(\omega_1,\omega_2) = \mid H(\omega_1,\omega_2) \mid^2 = \frac{G^2}{\mid (1 - \sum_k \sum_l a_{kl} e^{-j\omega_1 k} e^{-j\omega_2 l}) \mid^2}. \qquad (4.12)$$

The linear prediction spectrum has several important properties that make it a suitable domain in which to examine textural differences. First, it is well known that the linear prediction spectrum approximates the Fourier power spectrum of the signal [98, 41]. This approximation is known to be better at high energy peaks than at low energy peaks, although the approximation is good on the average at all frequencies [98]. Figure 4.5 shows an example of the linear prediction spectrum of a speech signal that illustrates this property. Thus $L(\omega_1,\omega_2)$ highlights high energy peaks in the power spectrum. It is in this sense that the usefulness of LP spectrum for textural comparisons becomes apparent. It can be seen as a means of extracting useful information (high energy peaks) from the power spectrum of the texture in a given window, so that comparing the LP spectra of two adjacent windows amounts to comparing the respective power spectra with only the essential information retained. Figure 4.4 shows examples of textures whose similarities/differences can be easily perceived from their LP spectra. Like the power spectrum, the LP spectrum too cannot distinguish between textures that have the same power spectrum but differ in phase spectrum when the same order AR model is used to parameterize the two textures. Since such textures rarely occur simultaneously in a scene, this is usually not a critical issue.

Another motivation for choosing the LP spectrum as a texture representation comes from its computational feasibility. Although the LP spectrum approximates the power spectrum, it does so in a ratio sense[5]. It was shown in [98] that this results in a smoothed spectrum that can be coarsely sampled (when compared to the power spectrum). Figure 4.5 that shows an example of linear predictive modeling of a speech signal also illustrates intuitively, why a lower sampling of the LP spectrum may be sufficient. A suggested sampling rate has been superposed on this figure which was taken from [98]. This lower sampling rate can also be derived from the definition of LP spectrum in equation 4.12. There the expression $A(\omega_1,\omega_2) =$

[5]That is, it is the spectrum that minimises the ratio of power spectrum to the LP spectrum.

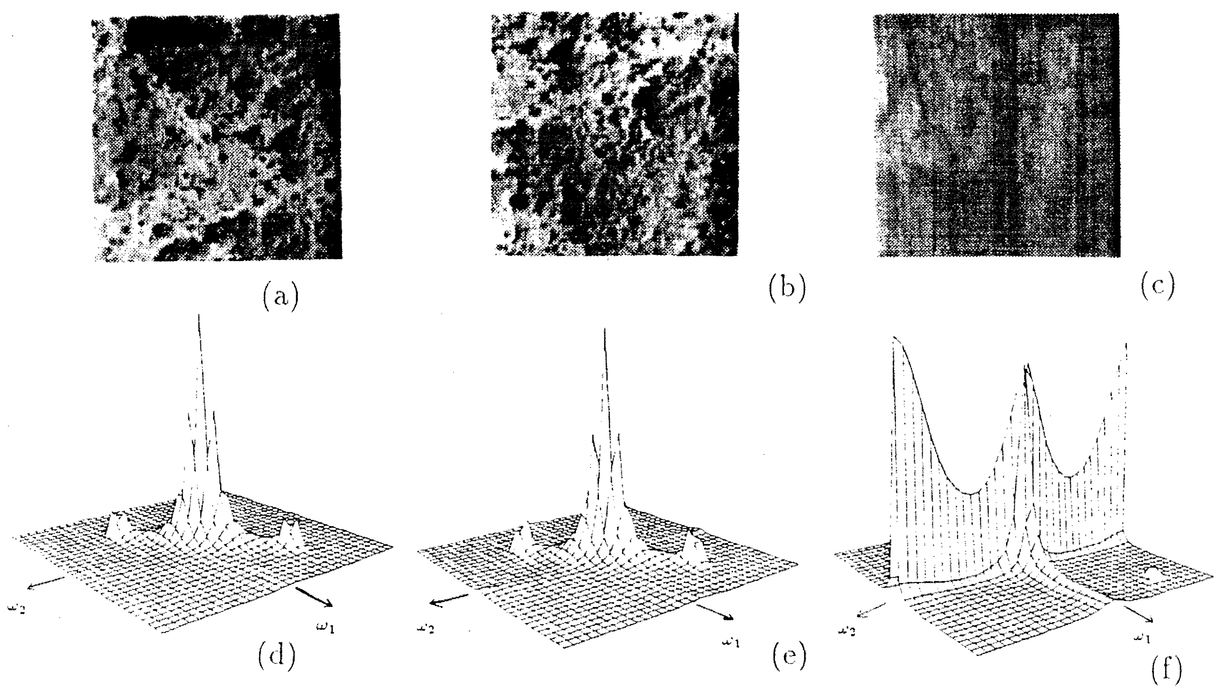


Figure 4.4: *Illustration of the power of LP spectra in highlighting the similarities/ differences between textures. (a) An aerial texture of a collection of trees. (b) Another aerial texture showing a different collection of trees. (c) An aerial texture of a field. (d) The LP spectrum of (a) using a 2x2 non-causal AR model. (e) The LP spectrum of (b) using a 2x2 non-causal AR model. (f) The LP spectrum of (c) again using a 2x2 AR model. The similarities between textures in (a) and (b) is brought out in their respective LP spectra.*

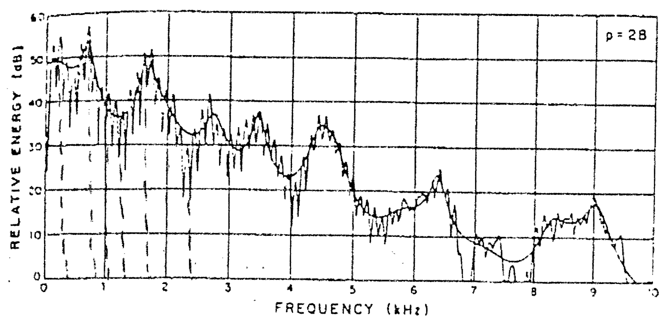


Figure 4.5: *Illustration to show that the LP spectrum can be sampled at a lower rate than the Fourier transform of the signal. The plot shows a 28-pole fit to a speech spectrum and is taken from [98]. A suggested sampling rate of the LP spectrum has been superimposed on the plot.*

$(1 - \sum_k \sum_l a_{kl} e^{-j\omega_1 k} e^{-j\omega_2 l})$ is essentially the (negative of the) Fourier transform of the AR parameters (with a_{00} = -1), and since there are $(2P + 1)$ x $(2Q + 1)$ of them, a sampling rate of $(4P + 2)$ x $(4Q + 2)$ is sufficient for sampling $A(\omega_1, \omega_2)$ and hence the LP spectrum. Thus the LP spectrum can be easily computed using a much smaller point Fourier transform than the power spectrum (that needs $2M$ x $2N$ where M x N is the size of the windowed texture region). From these features it is apparent that the linear prediction spectrum representation is an appropriate domain for analyzing textural differences and hence for texture region segmentation.

4.2.3 Comparison of LP spectra of adjacent windowed regions

We now develop a measure for recording differences between the LP spectra of adjacent windowed regions in a texture image which can then be used in a texture segmentation algorithm. The difference between spectra can be judged by looking at certain properties that capture the essence of the frequency content of the LP spectra of two adjacent windows W_1 and W_2. Since the windows overlap, it is clear that there is a common textured region R_2 in the two windows (see Figure 4.6). For the purpose of segmentation, we would like to know if two adjacent overlapping windows span different textured regions. A simple way to check this would be to see if boundaries exist in the regions of overlap. Since every image region is in the overlap region of two adjacent windows, all textured regions could potentially be discovered via such a boundary detection. If we call the image regions outside the region of overlap in W_1 as R_1 and in W_2 as R_2, the scenarios of distribution of regions R_1, R_2, R_3 we consider are shown in Figure 4.7. That is, either none, one, or two boundaries are possible in the overlapping region with respect to the windows W_1, W_2. Note that the case of no boundary in the region of overlap does not necessarily mean that the texture regions in W_2 is the same as in W_1. They could still have a boundary as shown in Figure 4.8. Such boundaries would be discovered in the previous or next set of overlapping windows shown in that figure. Consider now the entire regions r_1, r_2, r_3 whose windowed portions are R_1, R_2, R_3, and let their Fourier transforms be denoted by F_{r1}, F_{r2}, F_{r3}. Then the Fourier transforms F_{R1}, F_{R2}, F_{R3} of regions R_1, R_2, R_3, are convolutions with shifted windows W_1, W_2 of F_{r1}, F_{r2}, F_{r3}, respectively, and when considered in isolation, would appear as their distorted versions. Some of this distortion is suppressed when only high energy peaks in the respective power spectra are considered in the corresponding LP spectra L_{R1}, L_{R2}, L_{R3}. When the regions (R_1, R_2) and (R_2, R_3), are considered in combination as the signal within windows W_1 and W_2, then the resulting spectrum is the sum of the individual Fourier spectra. When the LP spectra of adjacent overlapping windows are compared, therefore, there may be some each of (including none), the common peaks, the missing peaks, and the spurious peaks (where missing and spurious peaks are decided with respect to one of the windows, say, W_1, as reference). But each of these factors (i.e., the common, missing, and spurious peaks) alone may not be sufficient to distinguish between the two LP spectra as they cannot uniquely resolve the cases (2), (3) and (4) shown in Figure 4.7. In case 2, we would find some each of common, missing, and spurious peaks arising from contributions from R_1, R_2, R_3 respectively, while cases 3 and 4 show instances of common and missing (case 3) or spurious (case 4) peaks. Using common peaks alone, we cannot resolve between all three cases (2,3,4), while with missing peaks we cannot resolve between (2) and (3), and similarly with

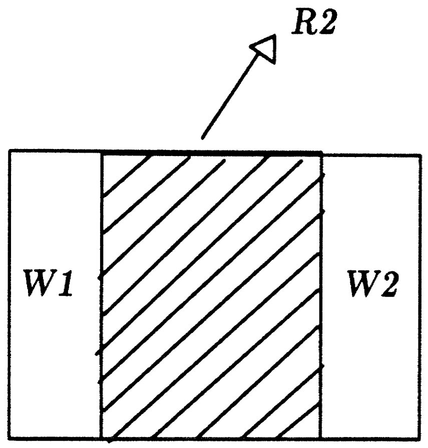


Figure 4.6: *Scenarios of texture regions in the overlapping windows.*

spurious peaks we cannot resolve between (2) and (4). The difference measure we chose, therefore, combines factors computed on each of these three types of peaks and adds them up in a reinforcing way.

LP spectrum difference detection measure

Let P and P$'$ denote the set of peaks in the LP spectra of the two adjacent windowed regions W_1, W_2 obtained by taking a 2Px2Q order model for AR estimation in both windows. That is

$$P = \{p_1, p_2, ...p_m\},\ P' = \{p'_1, p'_2, ...p'_n\} \tag{4.13}$$

Let the amplitude of peak p_i be denoted by $V(p_i)$ and its frequency location by $l(p_i)$. Let the set PP' denote the set of common peaks, i.e., when $l(p_i) = l(p_j)$ so that

$$PP' = \{(p_{i1}, p'_{j1}), ...(p_{ik}, p'_{jk})\}. \tag{4.14}$$

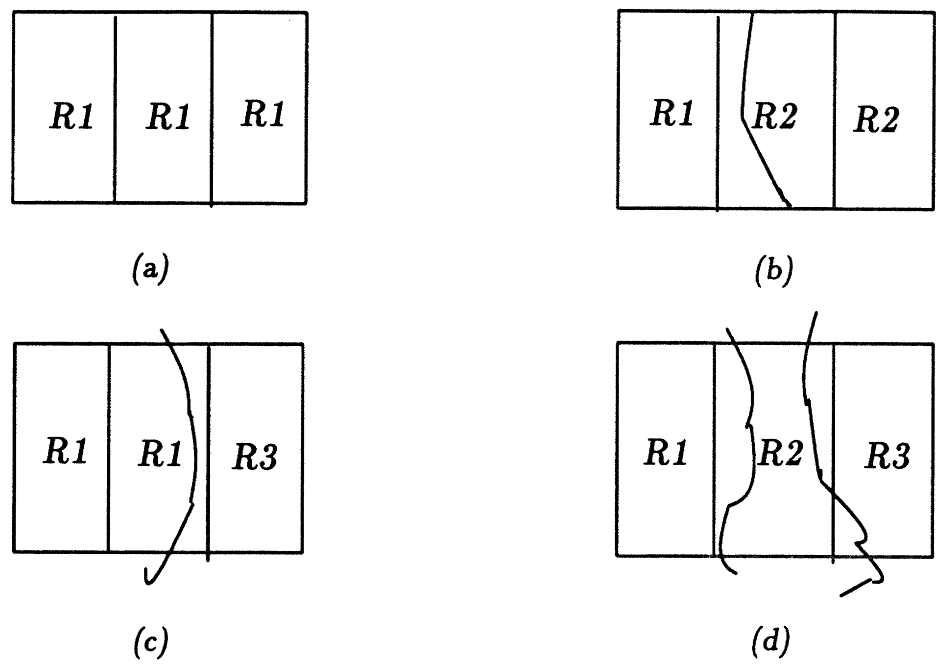


Figure 4.7: *Scenarios of the distribution of texture regions in adjacent overlapping windows showing the possibilities of zero, one or two boundaries in the region of overlap with respect to the windows W_1 and W_2.*

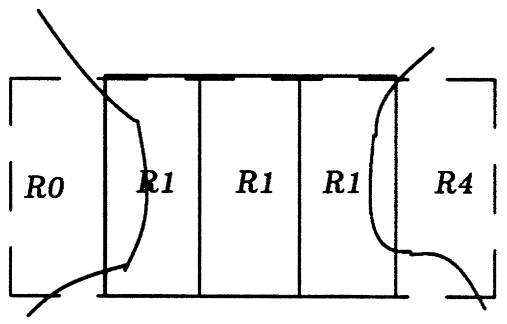


Figure 4.8: *Illustration to show that an absence of boundary in the region of overlap could still indicate a boundary in the region spanned by the overlapping windows.*

Let

$$PM = \{p_{i1}, ...p_{ik}\},\ PM' = \{p'_{j1}, ...p'_{jk}\} \tag{4.15}$$

represent the first and the second of the pairs in the set PP'. Then the set P-PM denotes the set of missing peaks while the set P′-PM′ denotes the set of spurious peaks with window W_1 taken as the reference window.

The difference measure was obtained by recording the following information about each of the common, the missing, and the spurious peaks: (i) their number, (ii) their relative amplitude, and (iii) their respective distributions of locations.

(a) Number of missing peaks: In general, the more missing peaks, the less the certainty that R_1 is present in the second window W_2. Further, if the peaks that are missing are of high energy, then this makes it even more certain. These two factors are reflected in the following choice of function to weigh the contribution from the missing peaks towards detecting the difference between the two LP spectra.

$$Fact_1 = \frac{\sum_{p_i \in P-PM} V(p_i)}{M_p * V_{max}} \tag{4.16}$$

where $V_{max} = max\{V(p_i)\}$, and $M_p = \mid P - PM \mid$. The factor is normalized so that it lies between 0 and 1 with 1 indicating a greater difference between the two spectra.

(b)Number of spurious peaks: Once again, the more spurious peaks, the more likely it is that a new region is present in the second window W_2. Further, this conclusion will be strengthened by the presence of strong spurious peaks. This is reflected in the choice of the second factor as:

$$Fact_2 = \frac{\sum_{p'_j \in P'-PM'} V(p'_j)}{M_{p'} * V'_{max}} \tag{4.17}$$

where $V'_{max} = max\{V(p'_j)\}$ and $M_{p'} = \mid P' - PM' \mid$.

(c) Number of common peaks: When spectra show common peaks, they either indicate that some portion of regions R_1 and R_2 may be present in the second window W_2 or that texture in region R_3 has certain common spectral peaks with R_2 or R_1 or both. A better indication can be obtained by comparing the amplitude of corresponding common peaks. This is expressed in the following factor:

$$Fact_3 = \frac{\sum_{(p_i,p_j^{'})\in PP'} abs\left(\frac{(V(p_i)-V(p_j^{'}))}{max\{V_{max},V_{max}^{'}\}}\right)}{\mid PP' \mid}. \tag{4.18}$$

(d) Location information at peaks: Since the frequency content of the peaks is indicated by their location, the difference in the relative location of ordered peaks (ordered by energy) gives an indication of the difference in the frequency content of the two LP spectra. It can be obtained by recording the difference in the distribution of peaks as follows. Let $P_1 = \{p_{i1}, p_{i2}, ..p_{im}\}$ be the set of peaks in P ordered by their energy. Similarly, let $P_1^{'} = \{p_{j1}^{'}, ...p_{jn}^{'}\}$ be the peaks $P^{'}$ ordered by their energy. Let $r(p)$ denote the rank of peak p in these ordered sets. Let $d(p_i, p_j^{'})$ = Euclidean distance between $l(p_i)$ and $l(p_j^{'})$ such that $r(p_i) = r(p_j^{'})$ and $p_i \in P_1$ and $p_j \in P_1^{'}$. Taking into account the relative importance in the ordering, so that the difference in the location of higher energy peaks between the two LP spectra is given more importance than the difference between low energy peaks, we weight the distance $d(p_i, p_j^{'})$ by

$$w_{ij} = max\{\frac{V(p_i)}{max\{V_{max}, V_{max}^{'}\}}, \frac{V(p_j^{'})}{max\{V_{max}, V_{max}^{'}\}}\}. \tag{4.19}$$

The factor $Fact_4$ records the difference in the distribution of peaks in the two spectra as given below:

$$Fact_4 = \frac{\sum_{(p_i,p_j^{'})\in P_1 P_1^{'}} w_{ij} d(p_i, p_j^{'})}{\mid PP' \mid}. \tag{4.20}$$

LP difference detection measure: Since each of the factors detect a certain type of difference that reinforce each other, the LP difference detection measure is formed by a simple linear combination as

$$Dist(W_1, W_2) = \sum_{i=1}^{4} Fact_i \tag{4.21}$$

The maximum value of the distance is 4 since each of the factors is normalized to lie between 0 and 1. After noting the value of the distance measure for several test windowed regions in texture images, we found a value of difference ≥ 2 as a conservative indication of a boundary. Better thresholds could have been designed, however.

Extensions: The above measure captures the difference in the number, distribution and relative strengths of peaks in the LP spectra of two adjacent overlapping textured regions. In cases where there are only a few peaks in the two spectra, this difference measure can be augmented with factors similar to $Fact_{1-4}$, but computed this time on the significant frequencies around the peaks.

4.2.4 Texture region segmentation using LP spectrum-based difference detection

The texture region segmentation algorithm brings together the ideas of moving window analysis of the texture image and the LP spectrum-based description of textures. It analyzes the image in 3 stages. In the first stage, a rectangular window of size M_w x N_w is moved over the image with an overlap of L_1 x L_2. Within each windowed texture region, the AR parameters are estimated and the LP spectrum is generated. The LP spectra of two adjacent windows is then compared using the measure given in equation 4.21. Since the windows overlap either horizontally or vertically, the boundaries predicted are then drawn vertically or horizontally respectively through the middle of the overlap region between windows. The result of this is a crude segmentation of the scene represented by the image. The difference measure chosen is rather conservative in that it may miss the prediction of certain portions of region boundaries. However, the boundaries that are predicted tend to be correct. In the second stage, the missing portions of boundaries are found by trying possible extensions of existing boundary fragments forward and backward for contour completion[6]. The reprediction of boundaries in such extensions is tried using a higher order AR model. Any remaining missing portions of contour are then ignored. The final stage of the algorithm assembles the various texture regions by performing a depth-first search on the overlapping windows. Specifically, it scans the overlapping windows top to bottom, left to right, and groups adjacent overlapping windows that do not have a boundary between them into one region provided the window being considered does not show a boundary with an already added segment[7]. The result of all three stages is to have the various texture regions approximately isolated with over-segmentation more likely than bleeding of regions (which was removed to a large extent in step 2). Such over-segmentation is more desirable than under-segmentation for selection, since it needs regions likely to come from a single object, even if they are not completely isolated.

Examples: We now illustrate the texture region segmentation algorithm with a few examples. Figure 4.9a shows a scene consisting of three cloth textures. By using a 64 x 64 sized window with overlap of 32 x 32, and comparing the LP spectra of adjacent windows as indicated in the above section, a segmentation of the scene was obtained as indicated in Figure 4.9b. As can be seen from this figure, the three texture regions have been isolated by this segmentation, although their precise boundaries are not well captured. Figure 4.10a shows another scene consisting of a collage of Brodatz textures. The region segmentation of the scene of Figure 4.10a is shown in Figure 4.10b. Some of the regions have been coalesced because of the forced merging done in Step 3 of the algorithm. Finally, Figure 4.11 shows the performance of the algorithm on an image of a scene consisting of textural information in the presence of non-textural scene clutter.

[6]The boundary fragments are extended up to a maximum of two overlapping windows. This was done after noting in several segmentation experiments on test images that the first stage of boundary prediction does not miss prediction of boundaries for more than two consecutive segments.

[7]Such cases occur when there are broken boundaries and the result will then be a forced segmentation into two regions.

(a) (b)

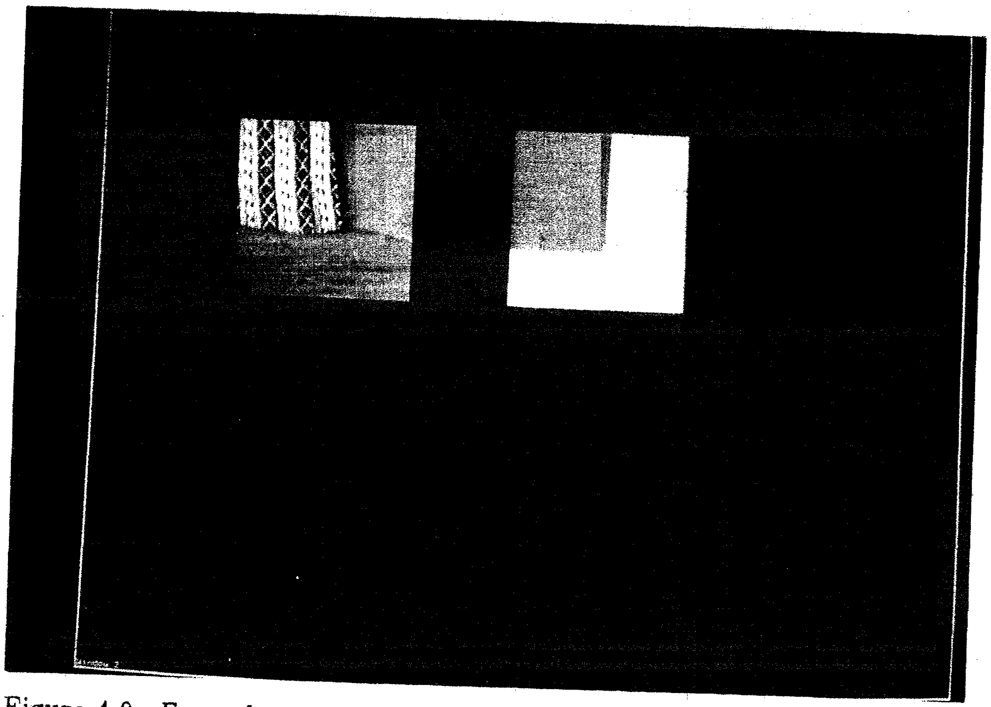
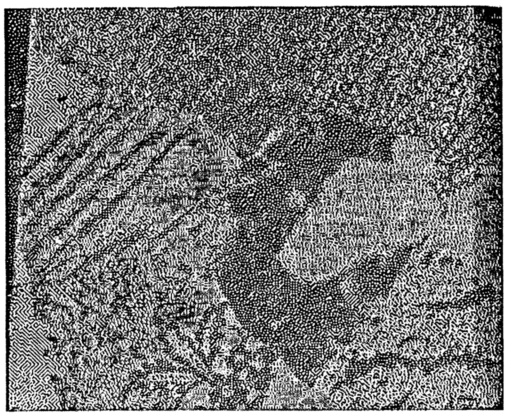


Figure 4.9: *Example of texture region segmentation. (a) A scene consisting of three textured regions. (b) The texture regions found using the region segmentation algorithm.*


(a)

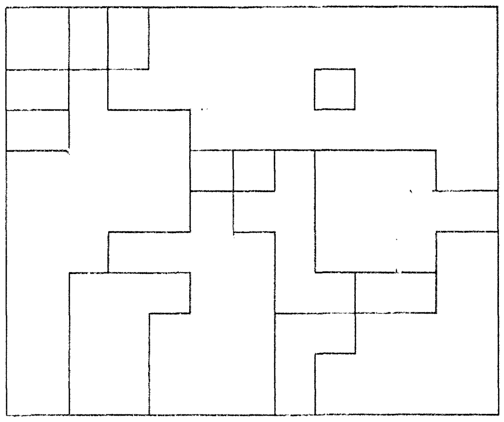

(b)

Figure 4.10: *Texture regions segmentation - Another example. (a) A scene depicting a collage of Brodatz textures. (b) A segmentation of the scene performed by the algorithm. Note the rough nature of the boundaries between various regions.*

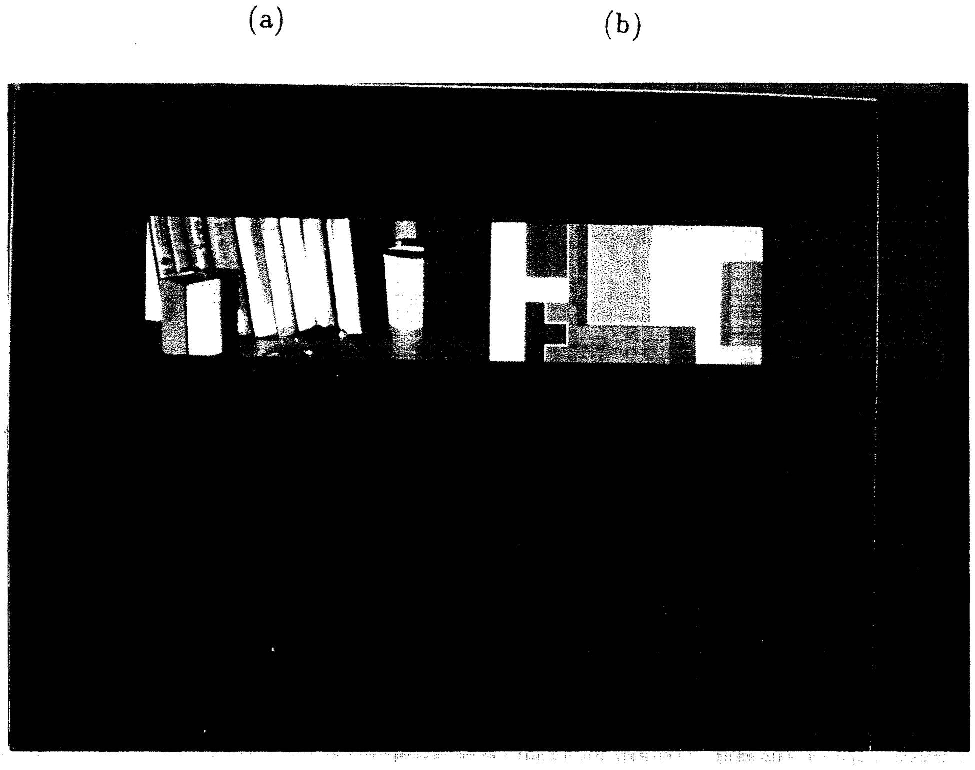


Figure 4.11: *Texture region segmentation - Final example. (a) A scene consisting of several objects with only a few objects carrying texture information. (b) Result of region segmentation performed by the algorithm.*

S.No.	Num. Regions	Num. Regions	Num. Regions	Num. Regions
	Present	Found	Merged	Split
1.	3	3	0	0
2.	7	7	1	1
3.	11	9	2	0
4.	15	17	2	4
5.	15	16	1	2

Table 4.1: *Statistics on segmentation performed by the texture segmentation algorithm on some scenes. The complexity of the scene is indicated by the number of texture regions indicated in Column 2.*

By trying the segmentation algorithm on images of several scenes such as the ones described above, we recorded the number of correct segmentations done by the algorithm. The results are shown in Table 4.1. Here the number of texture regions manually detected in these images are listed in Column 2 and give an indication of scene complexity. Column 3 indicates the number of texture regions found by the algorithm in these scenes. Column 4 indicates the number of texture regions that were merged and column 5 indicates the number of regions that were split. From the table we see that both merging and splitting are possible. The regions that were merged were mainly small regions. Due to the conservative nature of the segmentation algorithm, merging was less common than segmentation, which is more tolerable than undersegmentation for the purpose of selection,

Analysis: Each of the steps of the segmentation algorithm can be done relatively fast. In Step-1, the estimation of AR parameters involves the computation of $O(PQ)$ autocorrelation values taking time $O(MNPQ)$ for the summation in Equation 4.10, followed by a solution of a system of linear equations (Equation 4.9) that take time of $O(P^3Q^3)$ by standard methods such as Gaussian elimination [56]. The Fourier transform of the AR parameters takes time of $O(PQlogPQ)$ while the spectra comparison takes time of $O(M/L_1.N/L_2.N_p)$ where $N_p = max\{\mid P \mid, \mid P' \mid\}$, $\mid P \mid$, $\mid P' \mid$ are the number of peaks in the two spectra, and $M/L_1.N/L_2$ is the number of analysis windows. Thus Step-1 takes time of $O(P^3Q^3 + MNPQ + PQlogPQ + M/L_1N/L_2.N_p)$. The order of complexity of Step-2 is the same as Step-1 except that the AR computation is done on only a fraction of the number of windows. In Step-3, the depth-first search examines each segment (window) at most once while visiting and, for each new segment added, compares it with every added segment, so that it takes time of $O((M/L_1N/L_2)^2)$. Since P, Q are typically chosen to be small ≈ 4, and $M/L_1, N/L_2 \approx 16$, so that the overall complexity of the algorithm is still small.

Discussion

This approach to texture region segmentation exploits good features of existing approaches to texture description and analysis but differs from them in some important respects. The model of the texture image formation process used is more general and applicable to images in which not all of the regions carry texture information. Further, it specifies, in some sense, the correct way to analyze an image consisting of several texture regions. Also, unlike in region growing methods, where the choice of good seed regions that capture the property of a single texture region was not clear, this approach can capture global properties by having a wider analysis window than the usual seed regions, and at the same time, capture local properties (namely, boundaries between adjacent regions) because of overlapping windows. Finally, by performing a rough region segmentation relatively fast, it satisfies the requirement of data-driven selection.

4.3 Texture-based Data-driven Selection

When considering scenes typically analyzed in object recognition, not all regions in an image of the scene carry texture information. Some of them come from the scene clutter or the background rather than texture patterns on objects. A texture segmentation algorithm such as the one described above, when applied to these images, gives regions not all of which are relevant from the point of carrying texture information. Also, imperfect segmentation often results in a large number of texture regions, some of which may span more than one object. For the purpose of recognition, therefore, it would be useful to select only a few key texture regions. Following again the paradigm of attract-attentional selection, we now present a method of choosing salient texture regions.

4.3.1 Texture saliency

As in color saliency, we focus on capturing the sensory component of texture saliency. Thus, for example, we would like to capture the sensing of brightness variations in a texture pattern, but which pattern of variations is more meaningful requires some a priori knowledge and will not be attempted. Some gross region properties such as color and area were sufficient to distinguish between color regions for color saliency. These are not sufficient to distinguish between texture regions for purposes of texture saliency. Texture captures the pattern of intensity variations in a region and, therefore, more detailed properties need to be captured for texture saliency. We now propose a representation for the texture image that highlights such sensory properties that are general enough to be perceived by most observers. A saliency measure is then designed that extracts perceptually significant features from such a representation.

One of the properties of a texture that seems to be sensed often is the contrast present in the grey level variations across the texture. There is also some physiological evidence that retinal ganglion cells and cells in the lateral geniculate nucleus respond to a difference in contrast [80]. Often the sensing of contrast is restricted to observing the relatively dark or bright patches (regions) in a texture against an intermediate grey background. In fact, a representation of the texture that highlights the dark or bright regions against a grey background seems to capture most of the textural information, as can be seen from the examples shown in Figure 4.12. This can also be inferred from a theorem of Curtis and Oppenheim [31] which states that under some conditions, it is possible to uniquely recover a 2D signal (to within a scale factor) from its threshold crossings[8], thus making such threshold crossings a

[8]That is, by recording the locations at which the signal value crosses a certain threshold.

complete representation of the signal. If the texture is convolved with the Laplacian of Gaussian (LOG) function, it often satisfies the conditions stated in their theorem (namely, bandlimitedness and nonfactorability). Now, if the bright and dark regions are obtained by recording the threshold crossings of the convolved image (using positive and negative thresholds as indicated by Voorhees and Poggio [159]), then by the above theorem, the texture (the LOG convolved texture, that is), can be uniquely specified from such bright and dark regions.

We propose, therefore, that the saliency of a texture region can be inferred by analyzing the interplay between the bright and dark regions in the texture against a grey background. To develop a measure of saliency, we need a way to reliably extract the relatively bright and dark regions in the texture, represent them in a suitable form, and analyze their interaction. We now discuss how each of these operations can be done.

Texture representation for deriving saliency

In our approach, the texture is represented by a set of four binary maps. Two of them highlight the bright and dark regions and the other two highlight the background against the dark or bright regions. The bright and dark regions were obtained by convolving the texture image with the Laplacian of Gaussian function, and grouping pixels that cross an appropriately chosen brightness or darkness threshold. The thresholds for obtaining the bright and dark regions were chosen using the method described in [159, 158]. The background regions were obtained by grouping those pixels that failed to satisfy the darkness or brightness thresholds (see Figure 4.13 for an illustration). Thus for example, in a map highlighting the dark regions in the texture, the background includes the white regions as well as regions of intermediate grey level. If the regions being highlighted in a binary map are designated as white regions, and the other regions as black, then each binary map can be treated in a uniform way by analyzing the white and black regions.

Representing the bright and dark region information in separate binary maps allows us to evaluate their individual contributions in determining the overall saliency

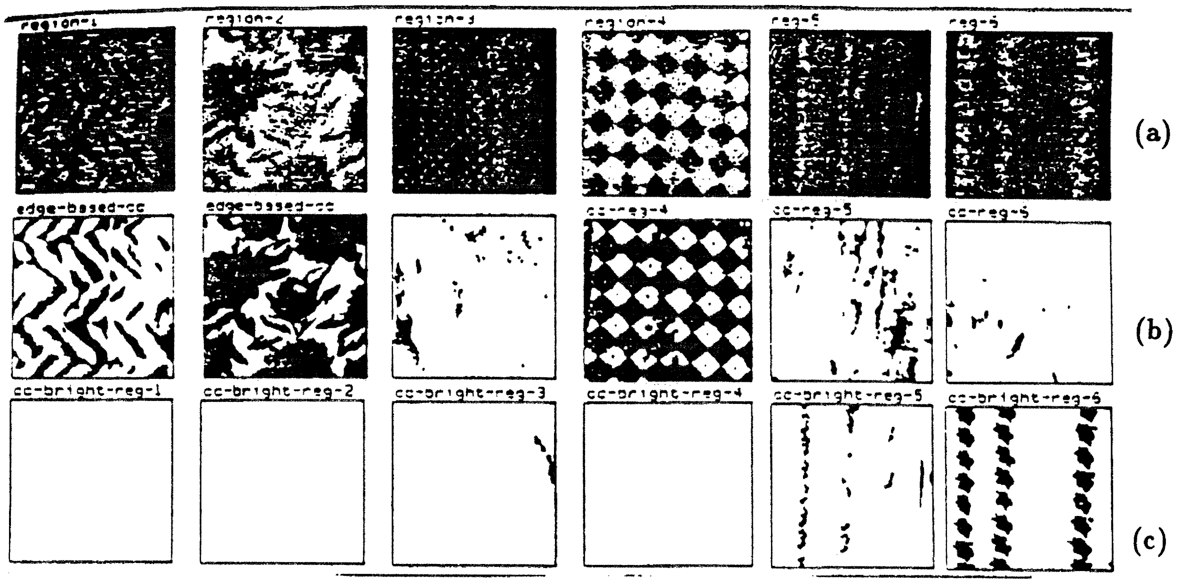


Figure 4.12: *Illustration to show that the bright and dark regions against an intermediate background capture important textural information. (a) Sample textures. (b) Binary maps highlighting the dark regions in the respective textures. (c) Binary maps highlighting the bright regions in the respective textures.*

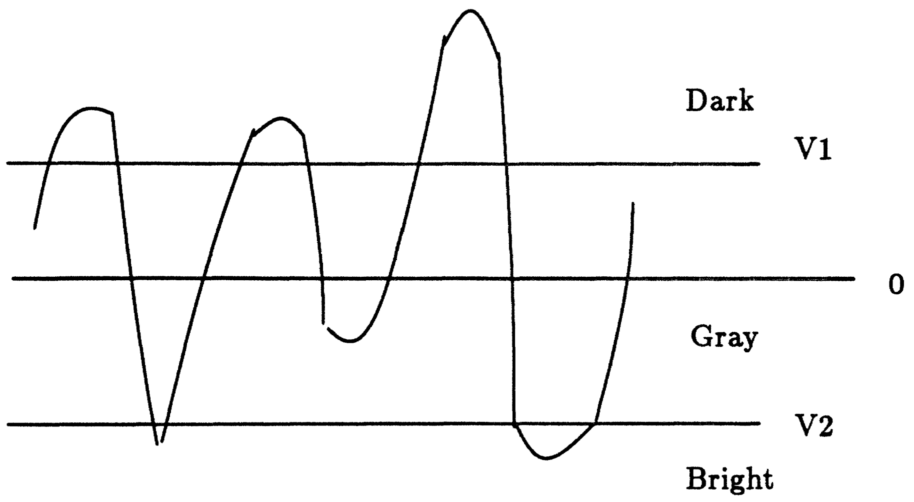


Figure 4.13: *Method of obtaining bright, dark, and intermediate grey regions in a texture. V_1 and V_2 are the darkness and brightness thresholds respectively.*

of the texture. For some textures the bright regions may capture the interesting patterns while for others it may be the dark regions. Figure 4.14 illustrates some of these textures. Figure 4.14a shows a texture for which the dark regions (against background) form interesting patterns, while Figure 4.14c shows a texture for which the bright regions form interesting patterns. Finally, Figure 4.14e shows a texture in which both bright and dark regions are important in determining saliency. In the next section we analyze the binary maps to determine the attributes that determine their individual contributions towards texture saliency.

Attributes of binary maps for determining texture saliency

The interplay between black and white regions in a binary map can be revealed by computing attributes that indicate the shape and sizes of these regions and their arrangement with respect to each other. Specifically, the attributes chosen for analyzing each of the binary maps capture properties of the black regions that lie inside some white region (called holes henceforth). The properties of white regions that lie

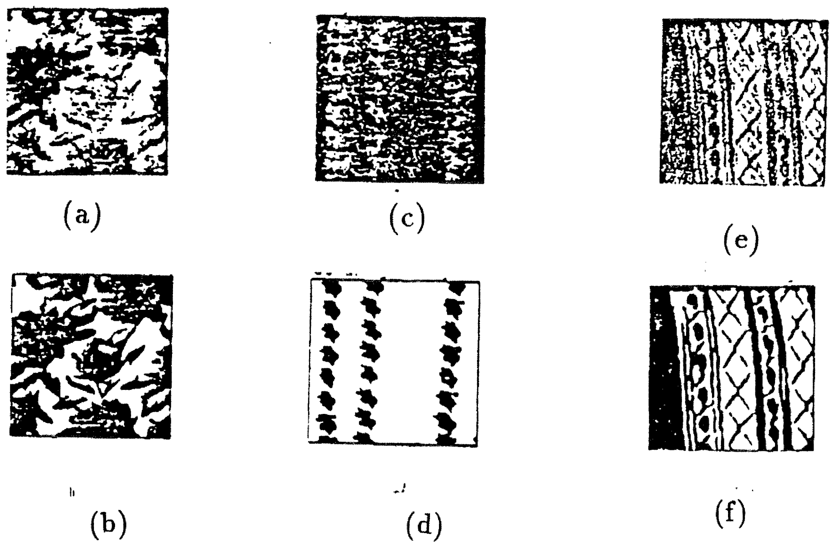


Figure 4.14: *Illustration to show the relative importance of dark and bright regions. (a) A texture where dark regions are dominant as seen from its binary map in (b). (c) A texture where bright regions are dominant as seen from its binary map in (d). (e) A texture where both bright and dark regions are dominant as seen from its binary map in (f).*

inside some black region will be considered in the binary map that has the roles of the black and white regions reversed. Some of the attributes capture global properties of these regions while others try to capture any local subpattern in a texture that may make it interesting. The global attributes we chose were : (i) the number of holes, (ii) shape of the holes, and (iii) the distribution of the holes inside white regions. As attributes that capture local subpatterns we chose: (i) maximum number of holes in a white region and (ii) maximum area occupied by any hole. We now describe the rationale behind the choice of these attributes and ways of measuring them.

Since all the attributes need information about holes, that is, the black regions that lie inside some white region, we first need to determine holes. Here we associate a hole region with the immediately enclosing white region. In doing so, we may restrict the computation of local attributes but not the global attributes. Determining the inside/outside relation (i.e. hole relation) between two regions is a polygon containment problem that takes $O(n\ logn)$time in the general case, and $O(n)$ time when the polygons are nonintersecting, where n is the number of points on the contours of regions being considered. However, by exploiting the fact that the map is binary, the containment test can be done in $O(1)$ time per pair of white and black regions by testing for the inside/outside relationships between rectangles enclosing the two regions as shown in Figure 4.15. With this test, a black region is taken to be inside a white region, if the enclosing rectangle of the black regions is inside the enclosing rectangle of the white region. The enclosing rectangles for all the regions can be found by a single linear scan of the binary image and noticing the top, bottom, left, and right extrema of each region.

1. Total number of holes: This attribute was chosen because it captures the overall density of the pattern. It can be obtained by simply counting the number of holes in all white regions.

2. Maximum number of holes within a white region: A large number of holes in a white region signals that the brightness variations occur rapidly in the texture and may indicate any significant subpattern that make this texture distinctive as can be seen from the examples in Figure 4.16. Since the holes inside a white region are

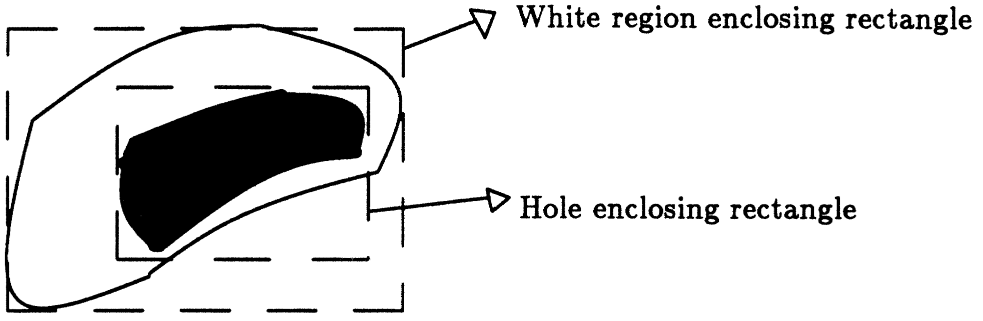


Figure 4.15: ***Hole detection test for binary maps. A black region is considered to be inside a white region if its enclosing rectangle is inside the enclosing rectangle of the white region.***

already known, this attribute can be easily evaluated. As the holes are associated with the nearest enclosing white region, this attribute does not take into account any black regions lying inside white regions that are themselves inside white regions (see Figure 4.17 for an illustration).

3. Area occupied by the holes within a white region: When the black regions within white regions are small in size and large in number, then the combined area of the holes within a white region can give an indication of the intricacy of the texture. To evaluate this attribute we compute the filled areas of black and white regions, that is, the sum of the areas (in pixel counts) of all regions lying inside a region. If we denote the filled areas of all holes in a white region by FA_b, the filled area of the white region by FA_w, and the maximum area (pixel area) of a hole in the white region by A_b, then the expression

$$\frac{FA_b}{FA_w}\left(1 - \frac{A_b}{FA_b}\right)$$

maximized over all white regions is taken as representative of any subpattern that may be present in the texture.

4. Shape of black regions: The shape of the regions is considered based on the rationale that non-smooth contour variations of the regions could create a pattern that is

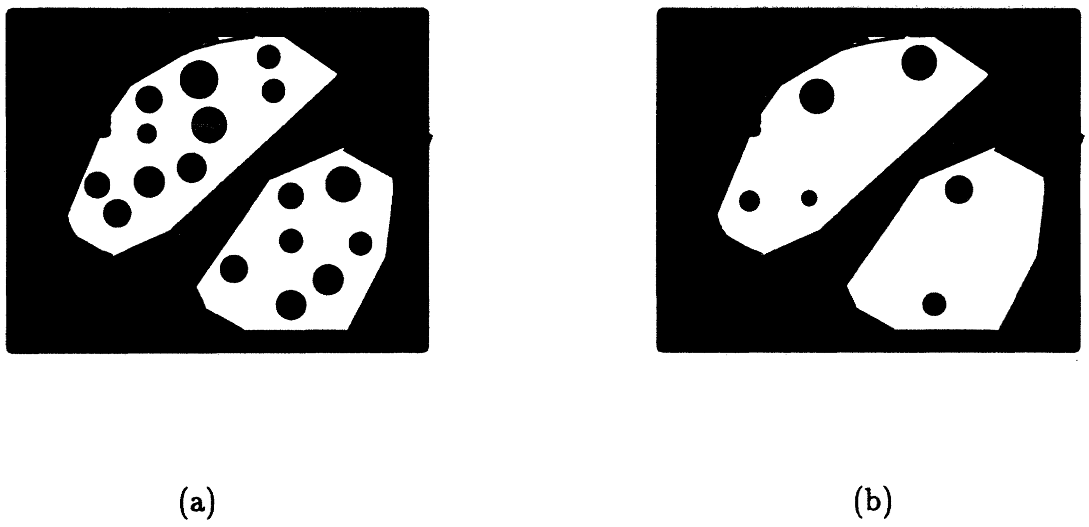


Figure 4.16: *Role of the number of holes in deciding texture saliency. The binary maps of the textures of (a) and (b) differ in the number of holes. The texture shown in (a) appears more salient due to a larger number of holes.*

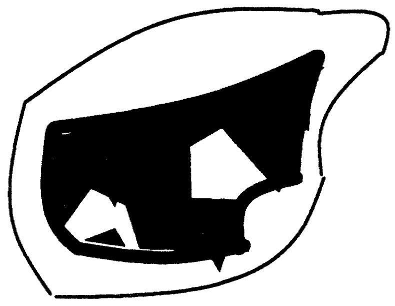


Figure 4.17: *Illustration to show that white regions lying deep inside white regions are not captured in attribute 2.*

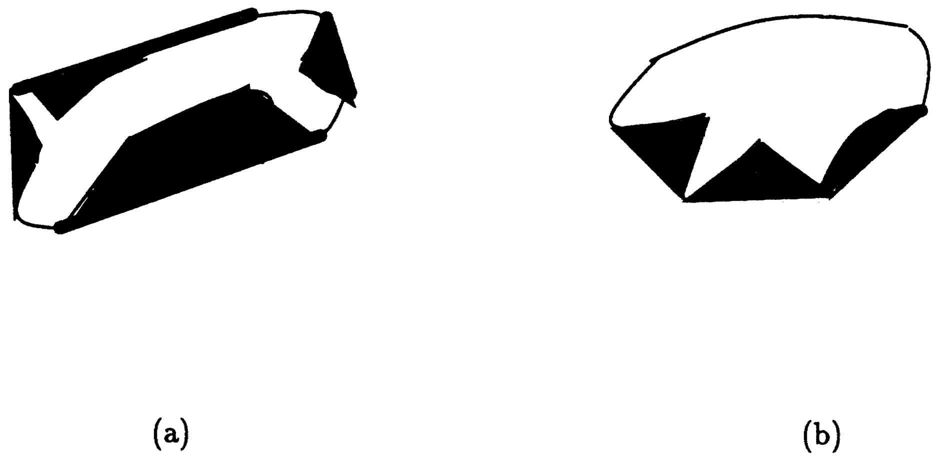

(a) (b)

Figure 4.18: *Illustration of the shapes whose non-smoothness is well-captured by the non-smoothness measure described in text.*

interesting. Detailed shape evaluations are not done, however, in order to avoid subjective bias. The relative deviation of the actual shape of the region from a convex polygonal approximation to the shape gives an indication of non-smoothness because it captures the area of local concavities in the shape of the region. Specifically, the following measure was used to evaluate shape non-smoothness (SS):

$$\frac{\text{convex area - actual area}}{\text{convex area of the region}}.$$

Figure 4.18 shows binary maps of some textures where the non-smoothness of the contours of black regions are well-captured by this measure. A distribution of the SS * relative size of all the black regions (relative to the area of the enclosing white region) is obtained and the value of SS corresponding to the maximum of the distribution is taken to be the representative non-smoothness in the shape of the black regions. The factor of relative size is added as a way of giving importance to the shapes of the bigger and hence more noticeable regions.

5. Distribution of black regions within white regions: When textures have a similar number of black and white regions with similar smoothness of shapes, then the

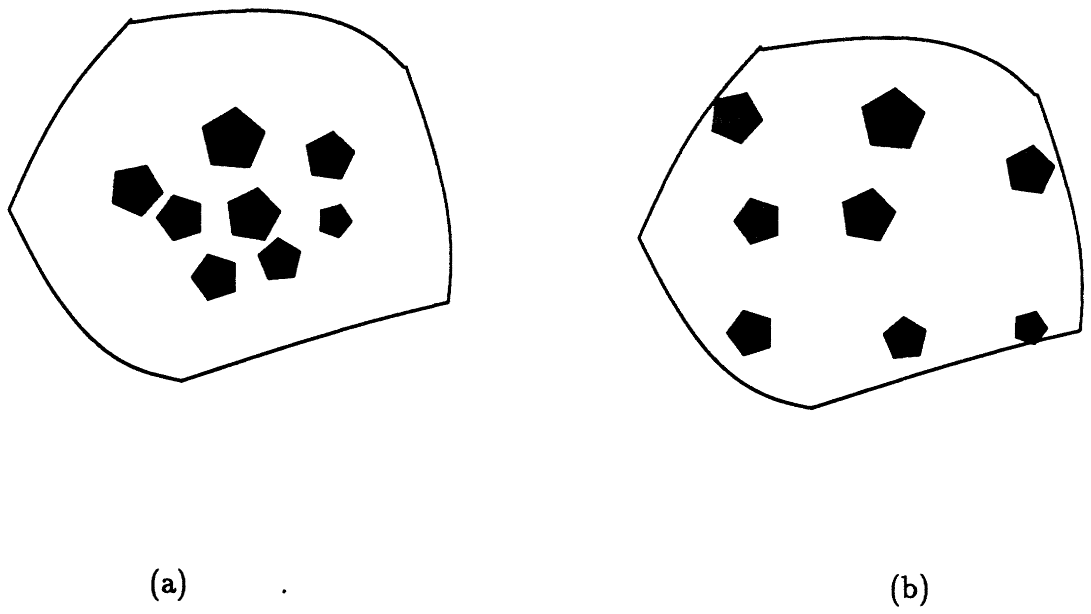


Figure 4.19: ***Illustration of the importance of the distribution of holes in determining texture saliency. The textures shown in (a) and (b) differ in the relative placement of holes and appeal to different extent perceptually.***

distribution of black regions within white regions can become important in deciding saliency as can be seen from the examples in Figure 4.19. Simple measures of inter-region distance such as

$$\frac{\text{area of the convex hull of all black regions} - \sum \text{area of black regions}}{\text{area of the convex hull of all black regions}}$$

do not very well capture the spacing between black regions within a white region. Figure 4.20 shows an example where the uneven spacing between black blobs is not well-captured by the above measure. Perceptually, it appears that only the spacing between certain "facing" regions is considered in getting a sense of the distribution when we look at such binary maps as shown in Figure 4.21. Here the dark lines indicate those facing regions whose inter-region spacing is considered while the broken lines show those facing regions whose inter-region spacing is ignored in perceptually determining the overall spacing between the black blobs. Finally, the curvy lines indicate which regions are not considered perceptually as facing regions at all. In general, it is difficult to characterize the perceptual phenomenon that determines which regions face each other, and the spacing between which of the facing regions is important. Our approach to characterize the inter-region space distribution is based on the following definition of facing regions. Two regions are said to face each other if there is at least a straight line joining a point on the contour of one region to a point on the contour of the other, that does not touch or cross any intermediate region. The closest distance of approach between such facing regions is then given by the length of the shortest such straight line. The algorithm for estimating the distance distribution proceeds in the following steps. First, it determines which regions face each other and obtains the shortest distance of approach for each such pair of facing regions. Then it determines the distance between which facing regions is important by retaining only those distances that form the minimum spanning tree of the shortest distances between regions. Such a tree determines which regions are closer to one another but does not capture all of the perceptually important inter-region distances, as shown in Figure 4.22. Here the relevant distances that are considered by the algorithm are marked by broken lines while the perceptually important distances are marked by solid lines. Once the retained shortest distances are obtained for all the black regions inside white regions, a cumulative distribution of such retained distances is taken. From the cumulative distribution, a representative distance 'd' is chosen such that 90% of the retained distances have value lower than 'd' (see Figure 4.23). This representative distance is chosen as a measure to characterize the distance distribution between the black regions in the binary map.

The facing regions and their closest distance of approach is determined using an extension of the Delaunay triangulation algorithm [118]. Specifically, the contour pixels of all black regions within a white region are pooled together and a triangulation of the space within the white region is done. Then the edges connecting points on the same region are removed and the minimum distance between two regions is recorded by examining the length of the edges emanating from points in one region to the other. The triangulation takes $O(MNlogMN)$ time where $M \gg N$, and M is

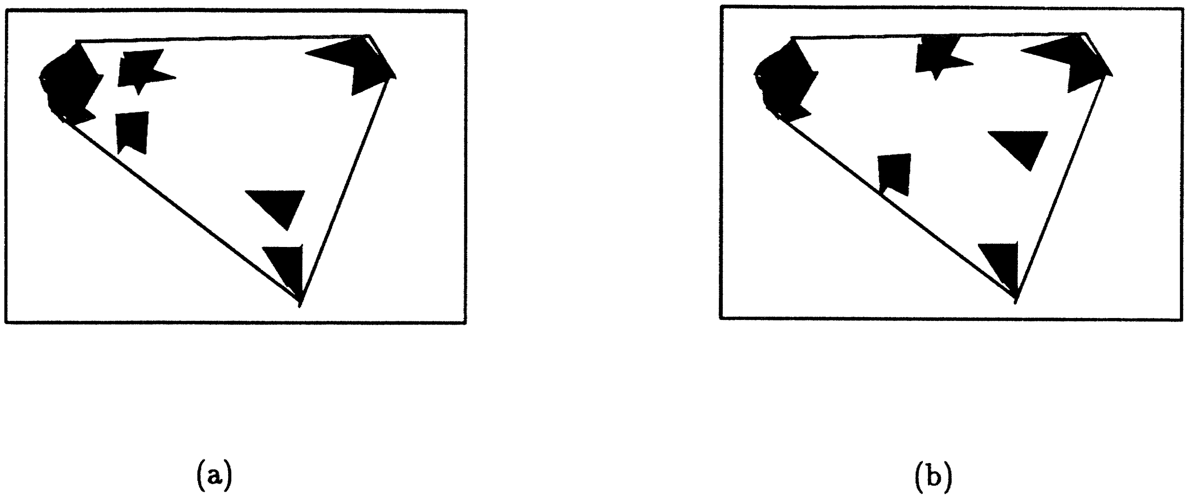


Figure 4.20: *Example to illustrate the inadequacy of the simple measure of inter-region distance given in text. The texture shown in (a) and (b) would have the same value using that measure but differ in the arrangement of holes.*

the number of points on the contour of a black region and N is the number of black regions. The shortest distance to all the facing regions can be found in $O(6MN)$ time, since each point is examined once and the maximum degree of branch from any contour point in the triangulation is 6 [118]. The shortest path from any region to the other is found by using Kruskal's algorithm for generating the minimum spanning tree and takes time $O(NlogN + N^2 A(N))$ where A is the Ackerman's function [29]. In this way, the entire distance distribution can be computed in time linear in the number of contour points of regions.

Discussion: From the above discussion on the attributes to determine saliency of a texture region, it appears that there is some redundancy in the information captured by each of these attributes. For example, in the case where the texture has a large number of relatively small-sized holes in a white region, both attributes 2 and 3 would indicate this feature of the texture. We chose these attributes in spite of the redundancy to account for textures such as the ones illustrated in Figure 4.24, where such attributes are not redundant. Here, Figure 4.24a and b shows two textures that

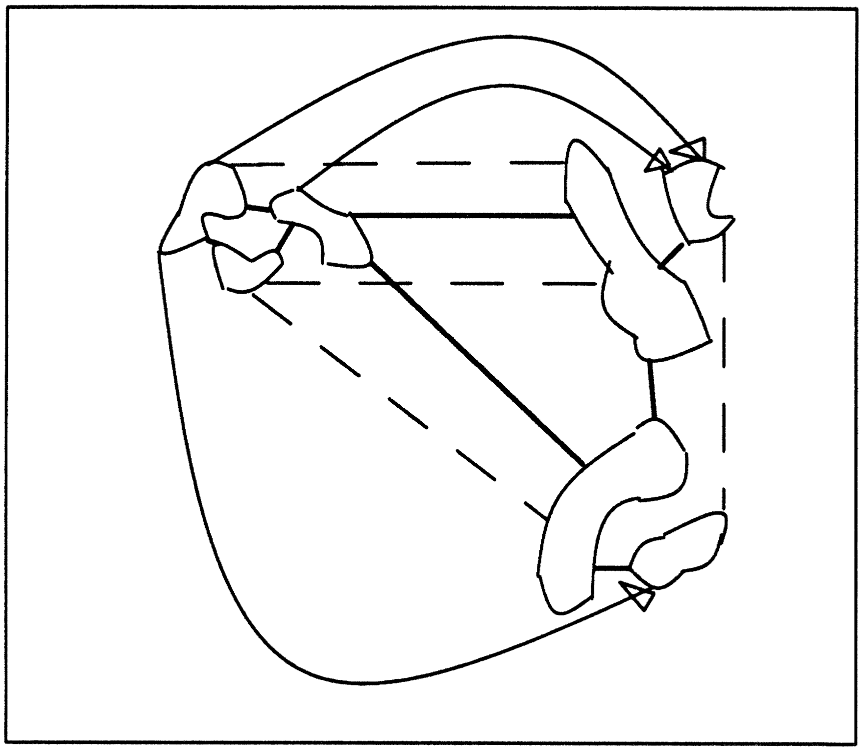


Figure 4.21: *Illustration to show the relevance of the distance between certain facing regions in determining the distance distribution. The solid lines indicate those facing regions whose distance distance are considered while the broken lines indicate facing regions whose distances seem to be ignored in arriving at a perception of inter-region distance. The curved lines indicate regions that don't seem to be facing each other.*

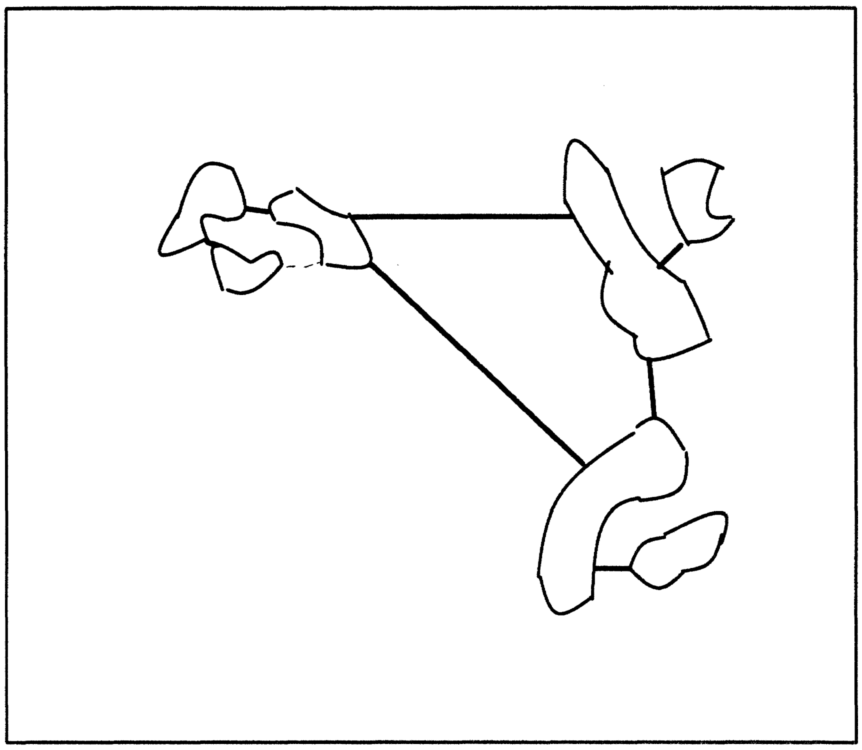


Figure 4.22: *Illustration of the distances considered by the algorithm for distance estimation. The broken lines indicate those distances that are considered perceptually but not by the algorithm.*

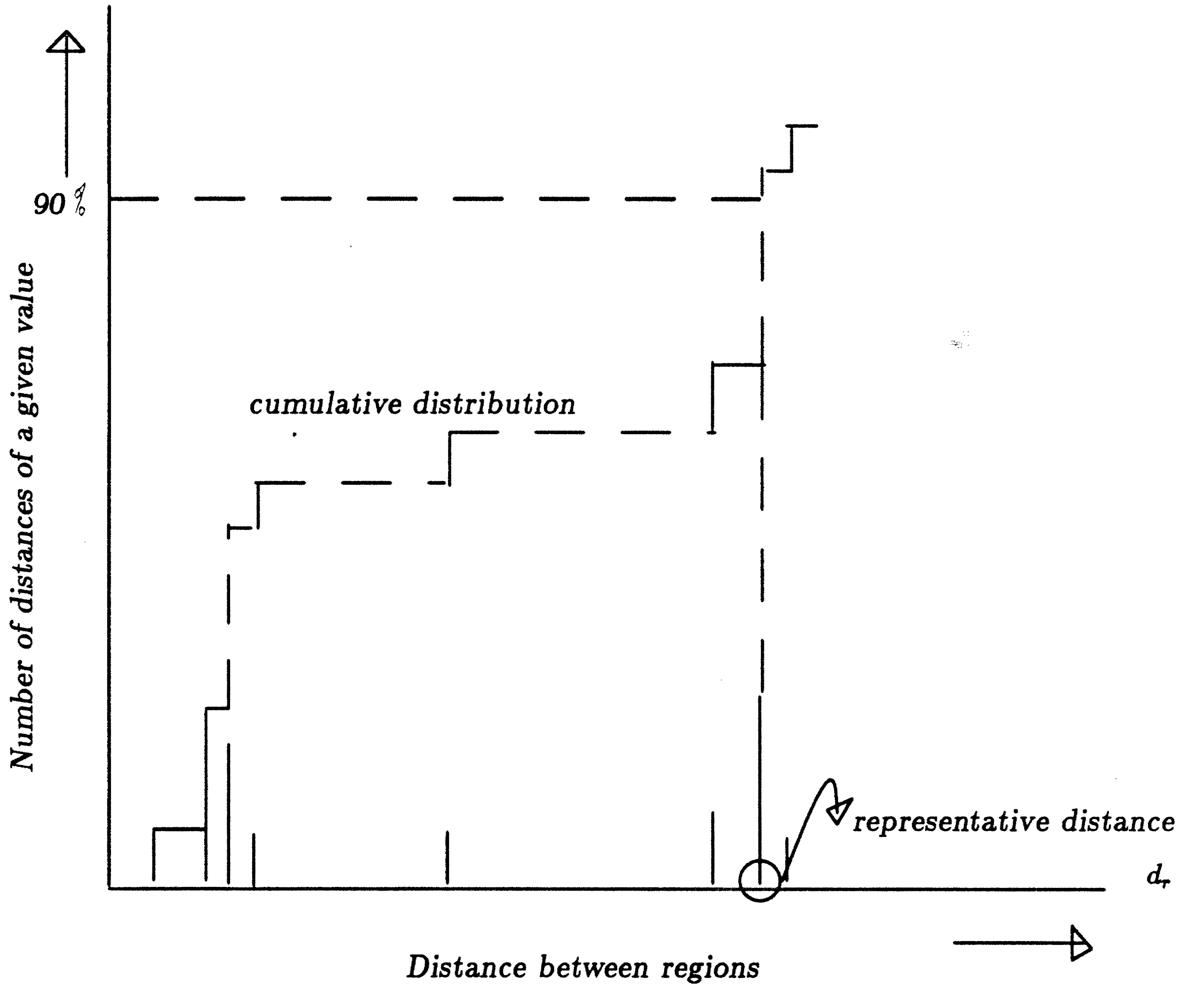


Figure 4.23: *Illustration of the choice of representative distance d_r for characterizing distance distribution.*

are distinguished by the value of attribute 2 (and not by the value of attribute 3) while Figure 4.24c and d shows two textures that are distinguished by the value of attribute 3.

A measure of texture saliency

In this section we derive a measure of saliency by combining the information obtained from an analysis of the black and white regions in the various maps using the attributes described above. We had proposed earlier that the saliency of a texture can be inferred by capturing the perceptual significance of the local contrast variations in the texture. In terms of the binary map representations of texture, we need to choose weighting functions for the individual attributes that reflect their individual contributions in determining the perceptual significance of the interaction between the black and white regions. Capturing the perceptual significance of such an interaction by numerical weighting functions is difficult, so our choice of the weighting functions was based on the grounds of perceptual plausibility, and computational feasibility. Specifically, they were chosen to emphasize the intricacy in the texture pattern in each of the binary maps. We now describe the weighting functions chosen for the attributes in each of the binary maps.

1. Total number of holes: In general, the more the number of holes, the better since this contributes to the intricacy of the pattern. However, the minimum perceivable size of the holes and the distance between them sets an upper limit to the number of such black regions that can be present in a binary map. Here we assign a minimum perceivable size of 4x4 pixels, and a minimum spacing between holes as also 4x4 so that the maximum number of holes that can be still perceived are : size of the texture region/ 8x8. If the number of black regions exceeds this threshold, the pattern is considered too dense to be perceived and will be weighted less after this threshold. Such a weighting is reflected in the following function:

$$f_1(t) = \begin{cases} t/T & 0 \leq t \leq T \\ e^{\alpha(t-T)} & T < t \leq M_t \end{cases} \tag{4.22}$$

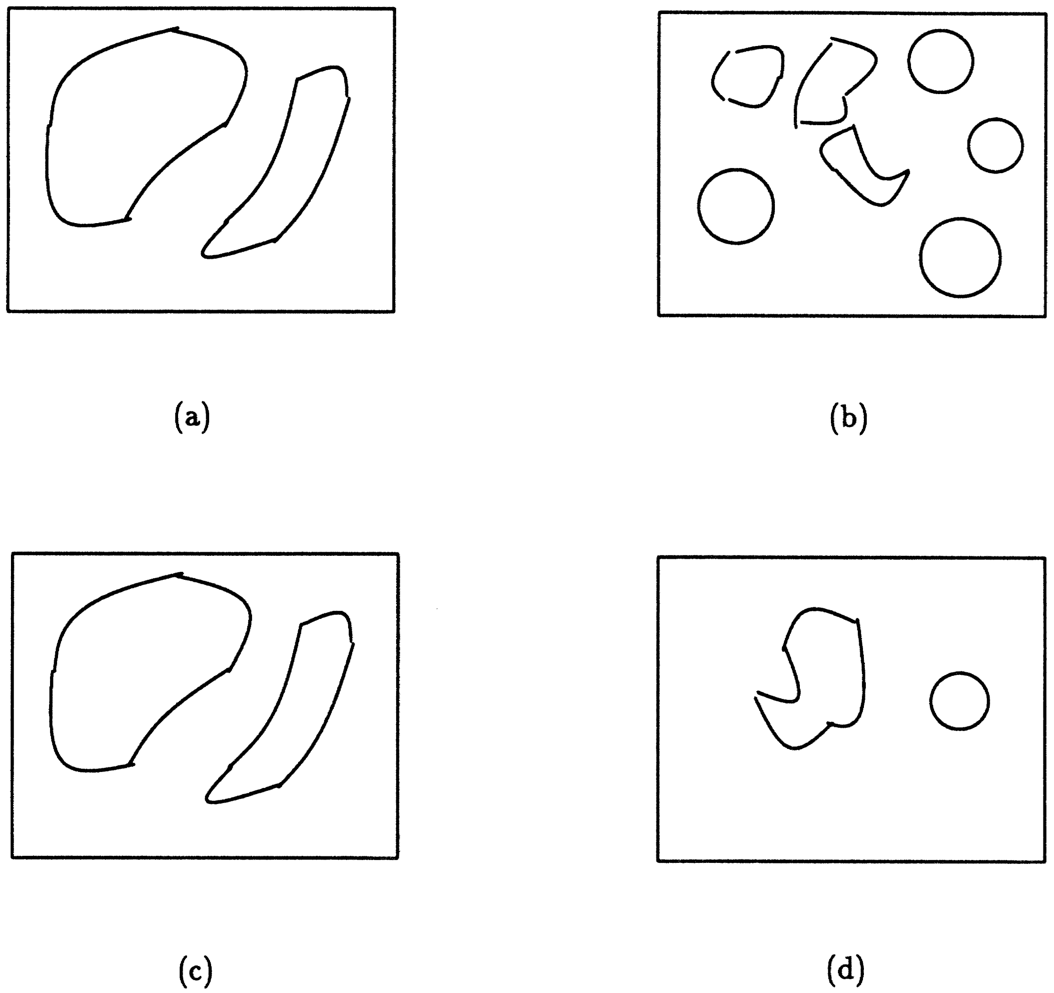


Figure 4.24: *Illustration of textures where attributes 2 and 3 are not redundant. (a) A binary map of a texture showing two holes. (b) A binary map of a texture that has the same ratio of hole to white region area as the texture in (a) but differs in the number of holes. (c) A binary map of a texture with two holes. (d) A texture binary map that differs from the map of (c) in the ratio of hole to white region area but contains the same number of holes.*

where t = number of white regions, $\alpha = \frac{6log10}{M-T}$, M_t = area of texture, and $T = M_t/64$. The rate of decay factor α was obtained assuming a value of 10^{-6} for the value of $f_1(t)$ at $t = M_t$[9]. Since the threshold T is rarely reached, $f_1(t)$ is linear for most textures.

2. Maximum number of holes in a white region: In cases when the textures being compared have about the same number of black regions inside all the white regions combined, then the maximum number of holes in a single white region can become a distinguishing factor. The weighting function for this factor, therefore, compares the maximum number of holes found in any white region in a given texture with the overall maximum number of holes in any white region of all the textured regions in any given image. The weighting function for this factor is chosen to be linear so that:

$$f_2(n) = n/n_{max} \tag{4.23}$$

where n = maximum number of holes in a white region in a given texture and n_{max} = the maximum number of holes in a white region across all textures in the image.

3. Area occupied by the holes within a white regions: The weighting function for this factor emphasizes those textures in which the holes are large in number and occupy a large area within a white region. The weighting function therefore is:

$$f_3(n) = n \tag{4.24}$$

where $n = \frac{FA_b}{FA_w}(1 - \frac{A_b}{FA_b})$, maximized over all white regions (attribute 3). Here FA_b = filled areas of all holes in a white region, FA_w = filled area of the white region, and A_b = the maximum area (pixel area) of a hole in the white region.

4. Shape of the holes: The weighting function for the shape of the holes initially emphasizes increasingly non-smooth shapes up to a threshold c_1. But for non-smoothness of shapes beyond this threshold (and before an upper threshold c_2) all

[9]The choice of the parameters such as α and the form of the weighting functions are described in Appendix B.

shapes are given equal importance. The upper threshold can be placed based on an analysis of randomly shaped bodies and noticing that except for thin curvy stripes, most region contours stay within this threshold. The form of the weighting function for non-smoothness as well as the thresholds v_1, c_1, c_2 was arrived at by performing some informal psychophysical experiments whose details are given in Appendix B. This weighting function is:

$$f_4(c) = \begin{cases} v_1 + c/c_1 & 0 \leq c \leq c_1 \\ 1 & c_1 \leq c \leq c_2 \\ e^{\alpha(c-c_2)} & c_2 \leq c \leq 1.0 \end{cases} \tag{4.25}$$

where $\alpha = \frac{-5log10}{1-c_2}$, and $c_1 = 0.2$ and $c_2 = 0.8$.

<u>5. Distribution of holes:</u> The weighting function for the distribution of distances between holes deemphasizes distributions with both very small and very large representative distance r_d. A very large value of r_d indicates that the majority of the holes in the map are sparsely distributed making the pattern less intricate in the given binary map. A very small value of r_d, on the other hand, indicates that the pattern is too dense. Between a lower and an upper threshold, the lower values of r_d are emphasized allowing for a more intricate pattern. The weighting function for this attribute is therefore:

$$f_5(n) = \begin{cases} -\frac{ln(1-n)}{c_1} & 0 \leq n \leq t_1 \\ 1 - e^{-c_2 n} & t_1 < n \leq t_2 \\ s_2 - c_3 ln(1 - n + t_2) & t_2 < n \leq t_3 \\ s_3 e^{-c_4(n-t_3)} & t_3 < n \leq t_4 \\ 0 & t_4 < n \leq 1.0 \end{cases} \tag{4.26}$$

where $n = r_d$, $t_1 = 0.1$, $t_2 = 0.4$, $t_3 = 0.5$, $t_4 = 0.75$, $s_1 = 0.8$, $s_2 = 1.0$, $s_3 = 0.7$, $s_4 = 10^{-3}$ and $c_1 = -\frac{ln(1-t_1)}{s_1}, c_2 = -\frac{ln(1-s_1)}{t_1}, c_3 = -\frac{(s_2-s_3)}{ln(1+t_2-t_3)}, c_4 = -\frac{ln\frac{s_4}{s_3}}{(t_4-t_3)}$.

<u>Combining attributes from binary maps to determine texture saliency</u>

In computing the attributes and their weighting functions, we have so far maintained the separation between the four binary maps that characterize the entire texture information. To determine the overall saliency of the texture region, the contributions from all the binary maps must be pooled together. In combining the contributions, we note that the maps highlighting the dark regions and the corresponding background reinforce each other, as do the pair of maps highlighting the bright regions and their corresponding background. This is because, in analyzing each of these maps, the attributes of the black regions alone are determined. The attributes of the white regions are obtained from the corresponding maps that highlight the background rather than the dark (or bright) regions. These attributes together constitute an analysis of the interplay between the black and white regions, i.e analyze the interaction of the bright (or dark) regions against their respective background. To allow for textures that become salient because of only the bright or the dark regions, their contributions are combined in a winner-take-all fashion. If we denote the weighting functions for the attributes i in binary map for the dark (bright) regions as f_{id} (f_{ib}) and in binary map for the corresponding background region as f_{ir_d} (f_{ir_b}), then the overall saliency of the texture region T is given by

$$Texture - Saliency(T) = \sum_{i=1}^{5} f_{si} \qquad (4.27)$$

where $f_{si} = \max\{f_{id} + f_{ir_d}, f_{ib} + f_{ir_b}\}$.

Results: We now illustrate the ranking of regions produced by the texture saliency measure derived above. Figure 4.25 shows a collage of 6 textured regions, and the ranking of these regions produced from the texture saliency measure is indicated in Table 4.2. Table 4.2 also lists the contributions from the various attributes i.e. values of f_{si}. The binary maps highlighting the dark and bright regions are also shown in the figure. Figure 4.26 shows another example of a collage of textures. These textures were taken from the Brodatz album [16]. Six texture regions are extracted from this collage of textures to illustrate the results of texture saliency computation. Since these textures have either the bright or dark regions, only one of

Region	f_{s1}	f_{s2}	f_{s3}	f_{s4}	f_{s5}	Saliency
1	0.09	0.60	0.0	0.70	0.0	1.40
2	0.04	0.21	0.16	0.64	0.15	1.21
3	0.12	0.19	0.08	0.53	0.01	0.94
4	0.16	1.0	0.39	0.26	0.29	2.12
5	0.20	0.19	.0	0.65	0.0	1.04
6	0.10	1.0	0.05	0.42	0.05	1.63

Table 4.2: *Contributions from various factors for determining the saliency of textures shown in Figure 4.25.*

Region	f_{s1}	f_{s2}	f_{s3}	f_{s4}	f_{s5}	Saliency
1	0.09	0.54	0.07	0.52	0.07	0.8
2	0.19	0.68	0.0	0.65	0.0	1.52
3	0.24	1.0	0.28	0.40	0.22	2.13
4	0.21	0.10	0.0	0.56	0.0	0.87
5	0.31	0.99	0.20	0.44	0.17	1.94
6	0.38	0.91	0.17	0.31	0.15	1.74

Table 4.3: *Contributions from various factors for determining the saliency of textures shown in Figure 4.26.*

the binary maps is shown in Figure 4.26b. Again the contributions from the various attributes is listed in Table 4.3 and the overall ranking produced by the measure is indicated in the last column of this table.

Use of salient texture regions for data-driven selection

Salient texture regions are primarily useful for data-driven selection in object recognition, when the object of interest has an interesting texture region that also appears among the salient texture regions in the given scene. In such cases, the search for data features that match model features can be restricted to the salient regions, thus

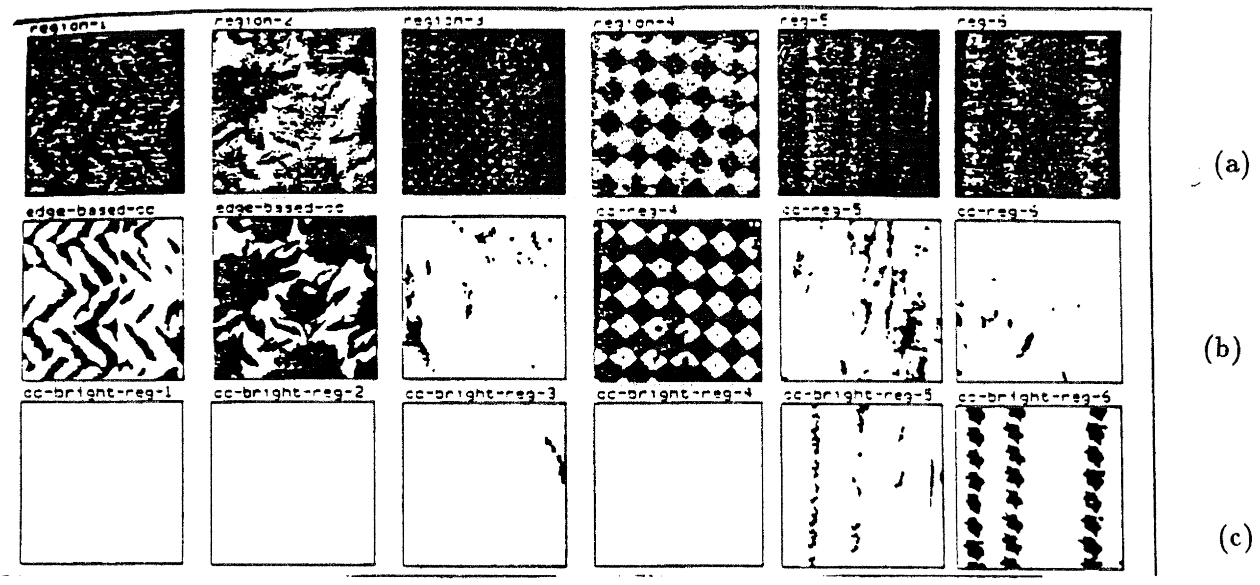


Figure 4.25: *Illustration of texture saliency computation for a collage of textures. (a) Six textures taken from various pieces of clothes. (b) Binary maps indicating the dark regions in the respective textures. (c) Binary maps highlighting the bright regions in the respective textures. The other two binary maps highlighting the background are not shown in the figure. The saliency values computed using the various attributes for the textures are shown in Table 4.2.*

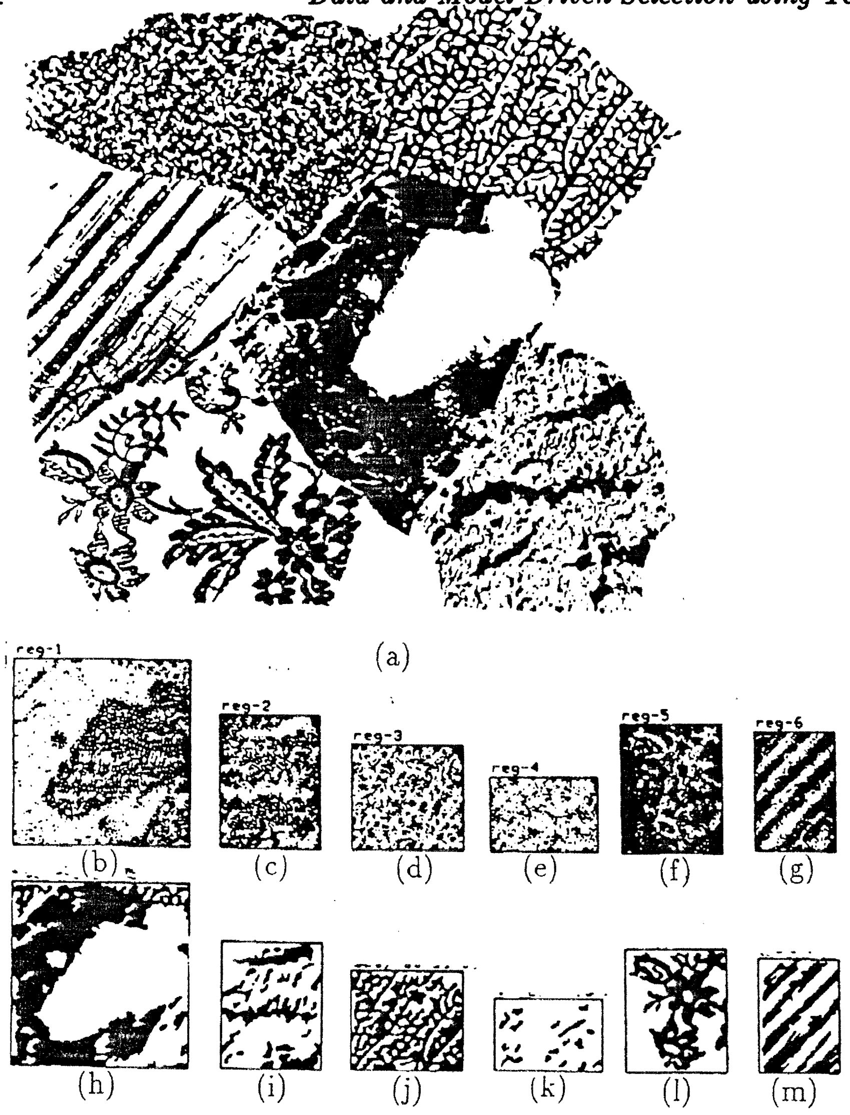


Figure 4.26: *Illustration of texture saliency – Another example. (a) A scene consisting of a collage of Brodatz textures. (b) - (g) Some regions isolated from an image of the scene depicted in (a). (h) - (m) Binary maps highlighting the dark and bright regions in the respective regions. The saliency values computed using the attributes for the textures are given in Table 4.3.*

avoiding needless search in other areas of the image. By selecting salient texture regions, we obtain a small number of groups (a region is itself a group), containing several features. By selecting features for recognition from these salient regions, search during recognition can be reduced, in a manner identical to that explained for color-based data-driven selection using the analysis given in Section 3.5.2 of Chapter 3 and will not be repeated here. We concentrate instead on illustrating data-driven selection using texture with a few examples. Figure 4.27a shows an image of three cloth textures. The result of texture region segmentation performed by the algorithm described in Section 4.2.4 is shown in Figure 4.27b. Figures 4.27c-e show the three most salient texture regions found by the saliency measure described above. As can be seen from the figure, interesting texture information in the scene is captured by the measure. Figure 4.28 shows another example of data-driven selection on an image that has both textured and non-textured regions in it. The results of texture region segmentation for the image of Figure 4.28a is shown in Figure 4.28b. As can be seen from the figure, the texture region segmentation is not perfect, although it does roughly indicate, the various texture regions. The binary maps produced for texture saliency computation are shown in Figure 4.28c-n. The three most salient regions produced by the texture saliency measure are indicated in Figure 4.28n,m,l. Again, as can be seen from this figure, the salient regions selected show the interesting texture information in the image. In the experiments done so far, the texture-saliency measure was found to select regions with texture information that showed good local contrast variation, and appeared significant.

Discussion: In deriving the measure of texture saliency above, it appears that are a number of free parameters. However, the brightness and darkness thresholds are the only free parameters here as they determine the nature of the four binary maps. The other parameters occur in the choice of the weighting functions for each of the factors. They were obtained by recording some statistics of the distinctive regions found by subjects in test scenes and approximating the resulting histograms as described in Appendix B. Other forms of approximating functions are also possible, but we chose the ones above as they seemed to highlight the salient texture regions on the scenes tested.

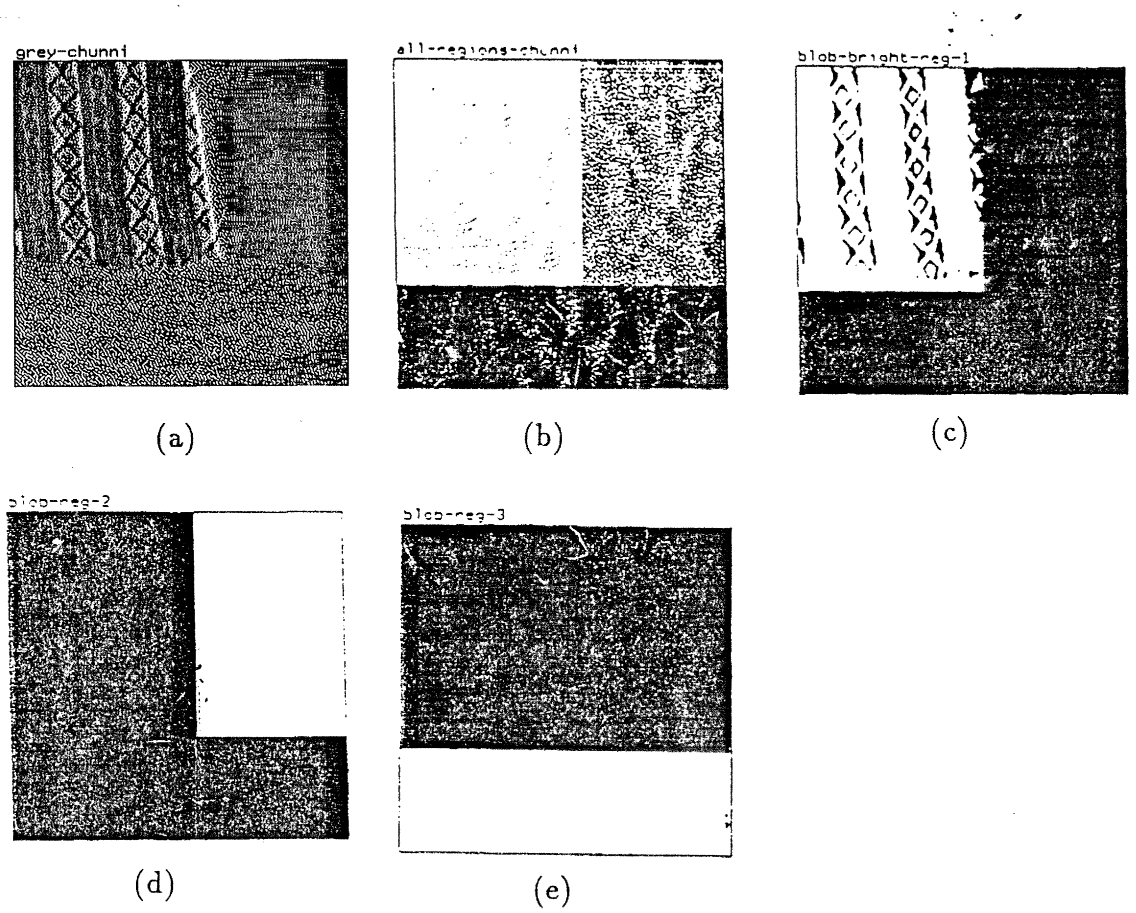


Figure 4.27: *Texture-based data-driven selection. (a) An image of a scene consisting of three textures. (b) Result of texture region segmentation. (c) - (e) The three most salient regions found in the segmented image of (b) using the texture saliency measure.*

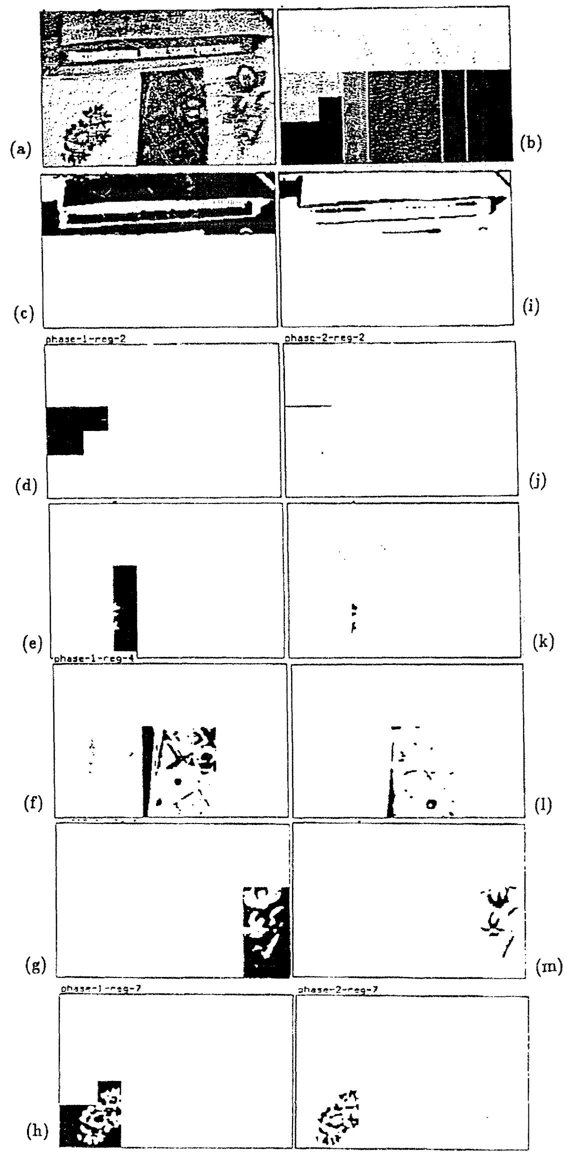


Figure 4.28: *Texture-based data-driven selection - Another example. (a) A scene consisting of textured and non-textured regions. (b) Result of texture region segmentation. (c) - (h) Binary maps highlighting the bright regions in the segmented image (b). (i) - (n) Binary maps highlighting the dark regions in the segmented image (b). Regions depicted in (n), (m), and (l) were considered the three most salient texture regions in the image.*

4.4 Texture-based Model-driven Selection

The previous section described a data-driven selection mechanism that was meant for an object possessing a salient texture. This will not be of much help when the object of interest is not salient in texture (but salient in some other domain, say color) or is not salient at all. In such cases, the information in the texture on the model object can be used to perform selection. The problem of texture-based model-driven selection, therefore, is to use a given description of a texture region on a model object to locate texture regions in an image that satisfy this description. Searching for an instance of a model object (or its texture) in an image is difficult as we mentioned in Chapter 3, because it may not appear the same in the scene as it does in the model description due to pose and illumination changes, occlusions, etc. Such changes can also affect the appearance of the texture on the object. For example, if we assume that the texture region on the model object is essentially a planar patch, then it is well-known that as the object undergoes a 3D linear transformation, the projection (orthographic) of the planar patch in the image undergoes an affine transformation [73]. A texture-based model-driven selection mechanism must be able to take this orientation change into account during selection. Similarly, to allow for occlusions or imperfect isolation of texture regions in an image, the selection mechanism must be able to tolerate changes in spatial extent between the model texture and its current instance in the image. Finally, to allow for illumination changes and other artifacts such as specularities and shadowing, a selection mechanism must be relatively insensitive to intensity changes across a texture region. Inability to tolerate such changes in a model texture during selection can cause unnecessary false negatives to be made later, by a recognition system. In addition, since selection is only a pre-processing stage that is meant to reduce the time spent in search during recognition, there is a need to keep the texture-based selection mechanism computationally simple.

4.4.1 Approaches to texture matching

From the above discussion it appears that the key to making a good model-driven selection mechanism using texture lies in the choice of an appropriate description of the model texture information, and a robust matching strategy that can account for the various changes described above. Previous approaches to texture analysis have considered one or more aspects of the texture-based model-driven selection problem under a variety of topics such as texture classification, template matching and texture recognition [137], [127], [83] where the goal was to classify a given texture as an instance of a library of known textures. Frequently, these have been aerial textures or textures from the Brodatz album [16]. The texture-based selection problem possesses some features that make it different from the texture classification problem in some important ways. First, the texture pattern is bound to a 3D model object so that more complex pose changes than simple in-plane rotations examined so far in texture classification methods will need to be considered. Secondly, a match to a specific texture pattern is required rather than to a class of patterns, so that fewer statistical variations within a texture pattern need to be tolerated. Finally, unlike in texture classification, the various candidate regions cannot be assumed to be perfectly isolated.

Nevertheless, some of the concepts used in existing texture analysis techniques can be useful in model-driven selection. Specifically, the texture descriptors employed in these approaches, such as the Fourier domain-based and parametric methods that we mentioned earlier, are as relevant for model-driven selection as they were found to be for data-driven selection. In particular, the linear prediction spectrum representation that combined the advantages of these texture descriptions is also useful for model-driven selection for the following reasons. First, the AR parameters give a compact description that has implications in storage conservation and fast indexing, factors which become important when the library of models for a recognition system becomes large. Secondly, since the linear prediction (LP) spectrum is not sensitive to exact spatial extent, it can be useful in recognizing textures even when portions of the object (and hence portions of the object's texture) are occluded in the given image. Because of the averaging affect in the calculation of AR parameters, some of the illumination changes can be tolerated (for more radical illumination changes, however, the LP spectrum can change significantly). Finally, the LP spectrum has an important property that makes it useful in any strategy of texture matching. As we will show next, the LP spectrum reflects the pose changes of the texture, thereby making it possible to recover such orientation changes by examining the LP spectrum domain. This has also significance in object recognition, since the partial pose of the object in the image can be established via texture which could ultimately lead to the determination of the complete pose of the object.

Because of the advantages of LP spectrum, we use it as the basis representation to perform texture-based model-driven selection. The basic approach is to describe the texture on a model object as well as the image texture by their linear prediction spectra and perform a match of the spectra taking into account the effect of orientation changes, illumination changes and occlusions to some extent. For this, the model texture region is assumed to be a planar patch on a 3D model object. The model object itself is assumed to be described by a set of 2D views so that the model texture we need to capture is what appears in one of the 2D views. If we characterize the texture region in a view by a discrete signal $s(m,n)$, then using the AR model and LP spectrum, the model texture information can be represented as a tuple $< s(m,n), a_{pq}, L(\omega_1,\omega_2) >$ where a_{pq} is a PxQ order AR model that is fit to the signal $s(m,n)$ and $L(\omega_1,\omega_2)$ is its LP spectrum. The order of the AR model can be arrived at using several methods including the Akaike's criterion [98]. Similarly, we can characterize any image texture region as a discrete signal and represent its information using a suitable order AR model and its LP spectrum.

The discussion of model-driven selection using the above texture representation will now proceed as follows. We first consider the problem of finding a match for the model texture with an already isolated texture region in the image. This problem is considered both in the case of perfect and imperfect isolation of texture regions. We then describe a way of analyzing a given image that allows the matching texture region to be found without prior texture region segmentation.

4.4.2 Model texture matching using isolated image texture regions

We now consider the problem of matching texture description on a model object with that of a given texture region isolated from the image. Using the LP spectrum representation, this involves determining if an image texture region description given by the tuple $< a_{Ipq}, L_I(\omega_1, \omega_2) >$ could match a model texture description $< a_{Mpq}, L_M(\omega_1, \omega_2) >$ where a_{Mpq}, a_{Ipq} and $L_M(\omega_1, \omega_2), L_I(\omega_1, \omega_2)$ are the AR parameters and the linear prediction spectra of the model texture and image texture respectively. The matching of model and image texture LP spectra must account for the fact that the pose of the model object in a given scene may not be as it appeared in its original description. Therefore we first examine the effect of pose changes on the LP spectrum representation of the model texture.

Since the surface texture on the model object is assumed to be planar, as the object undergoes a 3D linear transformation in space, the image of the surface texture undergoes an affine transformation [73]. It can be easily shown that the Fourier power spectrum and hence the LP spectrum are invariant to translation so that we need to consider those orientation changes of texture that represent a 2D linear transformation.

Effect of linear transformation of texture on its Fourier transform

To see the effect of linear transformation of texture on the LP spectrum we first examine its effect in the Fourier domain. Let the image texture region corresponding to the surface texture on the model object be denoted by a continuous 2d signal $s(t_1, t_2)$. Let M_t x N_t be its size in continuous coordinates. Let its discretized version be denoted by $s(m, n)$ where $s(m, n) = s(mT_1, nT_2)$ with the sampling rate being (T_1, T_2). Then its sampled size can be denoted by MxN where M*T$_1 \geq M_t$ and N*T$_2 \geq N_t$.

The Fourier transform of the texture is given by

$$F(\omega_1, \omega_2) = \int_0^{M_t} \int_0^{N_t} s(t_1, t_2) e^{-j\omega_1 t_1} e^{-j\omega_2 t_2} dt_1 dt_2$$

$$= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} s(m,n) e^{-j\omega_1 m} e^{-j\omega_2 n} \tag{4.28}$$

The discrete sample approximation of the Fourier transform in Equation 4.28 is valid under the assumption that $s(t_1, t_2)$ is adequately sampled according to the Nyquist rate.

Suppose the surface texture patch undergoes an orientation change. If a new image is created of the changed texture, and assuming the sampling rate remains (T_1, T_2), (which is true if both textures are imaged using the same camera with lens of the same focal length), then the new image texture representing the changed surface texture is denoted by $s'(m_r, n_r)$, and its Fourier transform denoted by $F'(\omega_{1r}, \omega_{2r})$ is

$$F'(\omega_{1r}, \omega_{2r}) = \sum_{m_r=0}^{M'-1} \sum_{n_r=0}^{N'-1} s'(m_r, n_r) e^{-j\omega_{1r} m_r} e^{-j\omega_{2r} n_r} \tag{4.29}$$

where M' x N' denote the changed size of the texture in the image.

Since the effect of a 3D linear transformation of surface texture is a 2d linear transformation (translation ignored) in image plane, the coordinates of the changed signal (image texture) given by $(m_r T_1, n_r T_2)$ correspond to some coordinates $(m_0 T_1, n_0 T_2)$[10] as follows:

$$\begin{pmatrix} m_r T_1 \\ n_r T_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} m_0 T_1 \\ n_0 T_2 \end{pmatrix} \tag{4.30}$$

Using the above relation, the transformed signal $s'(m_r, n_r)$ can be expressed in terms of the untransformed signal $s(m,n)$ as

$$s'(m_r, n_r) = s(m_0, n_0) \tag{4.31}$$

Expressing m_r, n_r in equation 4.29 terms of m_0, n_0 as given in equation 4.30, we get the Fourier transform of the transformed texture as

$$F'(\omega_{1r}, \omega_{2r}) = \sum_{m_0} \sum_{n_0} s(m_0, n_0) e^{-j\omega_{1r}(am_0 + bn_0)} e^{-j\omega_{2r}(cm_0 + dn_0)} \tag{4.32}$$

[10]The notation (m_0, n_0) is used to denote the fact that the corresponding points can be real numbers unlike (m,n).

$$= \sum_{m_0}\sum_{n_0} s(m_0,n_0)e^{-j(a\omega_{1r}+c\omega_{2r})m_0}e^{-j(b\omega_{1r}+d\omega_{2r})n_0} \tag{4.33}$$

$$= F(\omega_{10},\omega_{20}) \tag{4.34}$$

where

$$\begin{pmatrix}\omega_{10}\\ \omega_{20}\end{pmatrix} = \begin{pmatrix}a & c\\ b & d\end{pmatrix}\begin{pmatrix}\omega_{1r}\\ \omega_{2r}\end{pmatrix} \tag{4.35}$$

Note that the summation in equation 4.33 is not on a unit sample distance grid, since $s(m_0,n_0)$ may not fall at integer locations using the sampling rate (T_1,T_2). The equality however holds under the following interpretation. The definition of the Fourier transform in terms of the discrete samples is valid as long as the signal is sampled (regularly or otherwise) above the Nyquist rate. The summation in equation 4.33 corresponds to the continuous signal $s(t_1,t_2)$ resampled so that the samples are located at $(kdT_1 - lbT_2, -kcT_1 + laT_2)$, where $k = \frac{m_r}{ad-bc}$, $l = \frac{n_r}{ad-bc}$, and $0 \le m_r < M'$, $0 \le n_r < N'$. If the average sampling rate now is above the Nyquist rate, then this summation would still constitute a discrete approximation to the continuous Fourier transform, making the equality in equation 4.34 valid. A case where this breaks down is when large scale changes occur. A reduction in the size of the signal causes an expansion of the Fourier spectrum so that if the sampling rate is kept fixed (using the same camera with the same focal length, say) then the periodicity of the Fourier spectrum remains the same. This may eventually cause aliasing to occur so that the Fourier spectra can no longer be recognized as transformed versions of each other. For such large scale changes, it may not be possible to spot the resemblance between the two textures in the spatial domain either.

By rewriting equation 4.35 as

$$\begin{pmatrix}\omega_{1r}\\ \omega_{2r}\end{pmatrix} = \left(\begin{pmatrix}a & b\\ c & d\end{pmatrix}^T\right)^{-1}\begin{pmatrix}\omega_{10}\\ \omega_{20}\end{pmatrix} \tag{4.36}$$

and comparing with equation 4.30 we conclude that *as the image texture undergoes a linear transformation, its Fourier transform undergoes an inverse linear transfor-*

mation. This is a generalization of the previously known results on the effect of scaling and 2D (in-plane) rotational changes on the Fourier transform [57]. Thus for example, when b = 0, and c = 0, (pure scale changes), $m_r = am_0$, $n_r = dn_0$, $\omega_{1r} = \omega_{10}/a$ and $\omega_{2r} = \omega_{20}/d$ indicating that an expansion (compression) in space domain causes compression (expansion) of the Fourier transform. And for the case of 2D rotation, $\left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}^T\right)^{-1} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

indicating that the rotation of a 2D signal causes a rotation of its Fourier transform.

Effect of linear transformation of texture on the LP spectrum

We now examine the effect of a linear transformation of texture in the LP spectrum domain. Let the AR parameters for the texture $s(m,n)$ be denoted by a_{pq} and that of the transformed texture $s'(m_r,n_r)$ by $a'_{p_rq_r}$. We noted above that the samples of the transformed texture correspond to the samples of the original texture $s(t_1,t_2)$ located at $(kdT_1 - lbT_2, -kcT_1 + laT_2)$, where k and l are as defined in the previous section. That is, the transformed signal $s'(m_r,n_r)$ can be thought of as being obtained by taking the samples of the original signal $s(t_1,t_2)$ located at $(kdT_1 - lbT_2, -kcT_1 + laT_2)$, and placing them on a regular rectangular sampling grid. The parameters $a'_{p_rq_r}$ estimated from the transformed signal will, therefore, be the same as the ones estimated from the resampled version of the untransformed signal. That is,

$$a'_{p_rq_r} = a_{p_0q_0} \tag{4.37}$$

where

$$\begin{pmatrix} p_r \\ q_r \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p_0 \\ p_0 \end{pmatrix} \tag{4.38}$$

If we now call the Fourier transform of the AR parameters a_{pq} as $A(\omega_1, \omega_2)$, and of $a'_{p_r q_r}$ as $A'(\omega_{1r}, \omega_{2r})$ (here a_{00} is taken to be -1). Then

$$A(\omega_1, \omega_2) = \sum_p \sum_q a_{pq} e^{-j\omega_1 p} e^{-j\omega_2 q} \tag{4.39}$$

and

$$\begin{aligned} A'(\omega_{1r}, \omega_{2r}) &= \sum_{p_r} \sum_{q_r} a'_{p_r, q_r} e^{-j\omega_1 p_r} e^{-j\omega_2 q_r} && (4.40) \\ &= \sum_{p_0} \sum_{q_0} a_{p_0 q_0} e^{-j(a\omega_{1r} + c\omega_{2r})p_0} e^{-j(b\omega_{1r} + d\omega_{2r})q_0} && (4.41) \\ &= A(\omega_{10}, \omega_{20}) && (4.42) \end{aligned}$$

The last equality in the equation above holds using a similar resampling argument as the one given for the Fourier spectrum of the texture.

Denoting the LP spectra of the textures $s(m, n)$ and $s'(m_r, n_r)$ as $L(\omega_1, \omega_2)$ and $L'(\omega_{1r}, \omega_{2r})$ respectively, and using equation 4.39 and 4.12, we have

$$\begin{aligned} L'(\omega_{1r}, \omega_{2r}) &= \frac{G^2}{\mid -A'(\omega_{1r}, \omega_{2r}) \mid^2} && (4.43) \\ &= \frac{G^2}{\mid -A(\omega_{10}, \omega_{20}) \mid^2} && (4.44) \\ &= L(\omega_{10}, \omega_{20}) && (4.45) \end{aligned}$$

Thus the LP spectrum follows the orientation changes in the texture in the same way as the Fourier transform, namely, by undergoing an inverse linear transformation.

Matching of image and model LP spectra

When the image texture region contains an instance of the model's texture (i.e. assuming perfect isolation), then by the above results, its LP spectrum reflects the orientation change. The parameters of such an orientation change given by

a, b, c, d in equation 4.30 can be recovered from the LP spectrum as follows. Using equation 4.35 and a pair of corresponding frequencies $\{(\omega_{110}, \omega_{120})(\omega_{11r}, \omega_{12r})\}$ and $\{(\omega_{210}, \omega_{220})(\omega_{21r}, \omega_{22r})\}$ between the model and image LP spectra respectively, we can solve for the transformation parameters a, b, c, d. The pairs of corresponding frequencies can be found as follows. From equations 4.34 and 4.45, we infer that as the texture undergoes a linear transformation, the location of the spectral peaks (in LP or Fourier spectra) changes according to an inverse (transpose) linear transformation, but their amplitude remains invariant. This means that the identity of peaks is retained (hence their relative ordering by strengths), so that in the ideal case, the two highest energy peaks of the LP spectra can be paired to solve for the transformation. In practice, however, since the image texture region may not have been perfectly isolated, or because of the illumination changes from model to image, the amplitude of the peaks in the LP (and also Fourier) spectra will be affected differentially, so that the highest energy peaks in the model's LP spectrum may no longer be of highest energy in the image texture's LP spectrum. Thus search may have to be done among the peaks in the LP spectra to find two corresponding pairs of peaks. This requires $O(K^2L^2)$ pairs to be examined where K = number of peaks in the model's LP spectrum, and L = number of peaks in the image region's LP spectrum. Since the LP spectra have only a few peaks, this is still computationally feasible.

Once the transformation parameters are obtained, the model's LP spectrum can be aligned with the image region's LP spectrum using these parameters. The correct set of parameters will ideally align all of the model texture's LP spectrum to the windowed region's LP spectrum indicating a match between the two LP spectra.

Effect of sampled LP spectra on the solution of transformation parameters

In the above discussion, we assumed that if the frequency correspondences were correct, the transformation parameters would also be correct and would align the LP spectra perfectly. But since the available LP spectra are sampled versions of the continuous LP spectra, picking pairs of integer frequency locations does not give the correct correspondence. This can be seen from equation 4.35 where $(\omega_{10}, \omega_{20})$ corresponding to the integer frequency $(\omega_{1r}, \omega_{2r})$ need not be at integer sampled locations themselves in the model's sampled LP spectrum. This could introduce significant errors in the solution of the transformation parameters and hence cause false positives and negatives to occur during texture matching. This problem can be overcome by pairing $m(m > 2)$ corresponding peaks (instead of two), to find a least squares solution of the transformation parameters, using the rationale that the errors get averaged out in this process. This can, however, drastically increase the complexity of the search from $O(K^2L^2)$ to $O(K^mL^m)$. Another solution is to increase the sampling rate in the Fourier domain, hoping that the locations $(\omega_{10}, \omega_{20})$ will fall at integer locations using the new sampling rate. But this may, in practice, require a much larger transform to be taken, thus offsetting the computational advantages offered by the LP spectrum. We now present a method of handling the errors due to sampled LP spectra that does not suffer from the above disadvantages. Given a pair of corresponding integer sampled frequencies in the model and image texture's LP spectra, $\{(\omega'_{10}, \omega'_{20})(\omega_{1r}, \omega_{2r})\}$, we first determine a neighborhood region such that the correct corresponding location in the model's LP spectrum lies within this neighborhood. Then to locate it more precisely, this neighborhood region is locally sampled and each such point is tried as a possible corresponding frequency location. The key observation that is used here is that both the size of the neighborhood and the local sampling can be derived from a specification of the error that can be tolerated in the estimation of the transformation parameters. This has implications in recognition as it indicates that the accuracy of pose solution of the texture (and hence the object) can be controlled and traded off against the chance of false positives and negatives during recognition. The method is described in detail below.

Let $(\omega_{11r}, \omega_{12r})$ and $(\omega_{21r}, \omega_{22r})$ be two peak locations in the sampled LP spectrum of the image region's texture. Then from equation 4.35, there exist frequencies $(\omega_{110}, \omega_{120})$ and $(\omega_{210}, \omega_{220})$ in the model texture's continuous LP spectrum such that they are related to $(\omega_{11r}, \omega_{12r})$ and $(\omega_{21r}\omega_{22r})$ by

$$\begin{pmatrix} \omega_{110} \\ \omega_{120} \end{pmatrix} = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} \omega_{11r} \\ \omega_{12r} \end{pmatrix} \tag{4.46}$$

and

$$\begin{pmatrix} \omega_{210} \\ \omega_{220} \end{pmatrix} = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} \omega_{21r} \\ \omega_{22r} \end{pmatrix}. \tag{4.47}$$

The parameters a, b, c, d can be expressed in terms of $(\omega_{110}, \omega_{120})$, $(\omega_{210}, \omega_{220})$, $(\omega_{11r}, \omega_{12r})$ and $(\omega_{21r}, \omega_{22r})$ as

$$\begin{pmatrix} \omega_{11r} & \omega_{12r} \\ \omega_{21r} & \omega_{22r} \end{pmatrix} \begin{pmatrix} a \\ c \end{pmatrix} = \begin{pmatrix} \omega_{110} \\ \omega_{210} \end{pmatrix} \tag{4.48}$$

and

$$\begin{pmatrix} \omega_{11r} & \omega_{12r} \\ \omega_{21r} & \omega_{22r} \end{pmatrix} \begin{pmatrix} b \\ d \end{pmatrix} = \begin{pmatrix} \omega_{120} \\ \omega_{220} \end{pmatrix}. \tag{4.49}$$

Suppose the correspondence between $[(\omega_{110}, \omega_{120}), (\omega_{11r}, \omega_{12r})]$, and $[(\omega_{210}, \omega_{220}), (\omega_{21r}, \omega_{22r})]$ is correct but while pairing peaks in the sampled LP spectra, the nearest integer locations $(\omega'_{110}, \omega'_{120})$ and $(\omega'_{210}, \omega'_{220})$ were chosen in the sampled model's LP spectrum. Let the errors in the frequency locations be denoted by $\epsilon_1 = (\omega'_{110} - \omega_{110})$, $\epsilon_2 = (\omega'_{210} - \omega_{210})$, $\epsilon_3 = (\omega'_{120} - \omega_{120})$, $\epsilon_4 = (\omega'_{220} - \omega_{220})$. By pairing such frequencies, we are actually solving for the parameters a', b', c', d' (instead of a, b, c, d) given by

$$\begin{pmatrix} \omega_{11r} & \omega_{12r} \\ \omega_{21r} & \omega_{22r} \end{pmatrix} \begin{pmatrix} a' \\ c' \end{pmatrix} = \begin{pmatrix} \omega'_{110} \\ \omega'_{210} \end{pmatrix} \tag{4.50}$$

and

$$\begin{pmatrix} \omega_{11r} & \omega_{12r} \\ \omega_{21r} & \omega_{22r} \end{pmatrix} \begin{pmatrix} b' \\ d' \end{pmatrix} = \begin{pmatrix} \omega'_{120} \\ \omega'_{220} \end{pmatrix}. \tag{4.51}$$

Using the above equations and denoting the errors in the estimation of the parameters a, b, c, d as $\delta_1 = a' - a$, $\delta_2 = c' - c$, $\delta_3 = b' - b$, $\delta_4 = d' - d$, we get

$$\begin{pmatrix} \omega_{11r} & \omega_{12r} \\ \omega_{21r} & \omega_{22r} \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_2 \end{pmatrix} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix} \tag{4.52}$$

and

$$\begin{pmatrix} \omega_{11r} & \omega_{12r} \\ \omega_{21r} & \omega_{22r} \end{pmatrix} \begin{pmatrix} \delta_3 \\ \delta_4 \end{pmatrix} = \begin{pmatrix} \epsilon_3 \\ \epsilon_4 \end{pmatrix}. \tag{4.53}$$

The above equations relate the error in the location of the frequency samples to the resulting error in the transformation parameters a, b, c, d. By placing a bound on $\delta_1, \delta_2, \delta_3, \delta_4$ to be $\leq \delta_{max}$, we can get a bound on the size of the neighborhood region that needs to be examined for a given integer frequency location in the sampled LP spectrum of the model texture. If we require δ_{max} to be $= 2^{-g}$ where $g > m+1$, and $m = logN_{fft}$, N_{fft} being the number of FFT points used to evaluate the spectrum (Fourier or LP spectrum as the case may be), then we have

$\epsilon_{max} = max(abs(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4)) = \max\left(((\omega_{11r} + \omega_{12r})\delta_{max}), ((\omega_{21r} + \omega_{22r})\delta_{max})\right) <$
$2\pi/N_{fft}$,

since $-\pi \leq \omega_{ijr} \leq \pi$, implying that for any choice of frequency locations $(\omega_{11r}, \omega_{12r})$ and $(\omega_{21r}, \omega_{22r})$, the size of the neighborhood will be at most one sample unit wide. That is, it ensures that the integer location ω'_{ij0} chosen for the actual frequency sample ω_{ij0} lies within one sampling frequency unit of the actual location. Once the size of the neighborhood region is determined, the local sampling rate can be derived in a similar way. There is an inherent tradeoff in the local sampling rate as it determines the number of locations within the neighborhood that need to be examined, as well as the resulting maximum error in the estimation of the transformation parameters. Its choice also depends on the frequency locations $(\omega_{11r}, \omega_{12r})$ and $(\omega_{21r}, \omega_{22r})$ chosen in the transformed texture's spectrum. For a given sampling

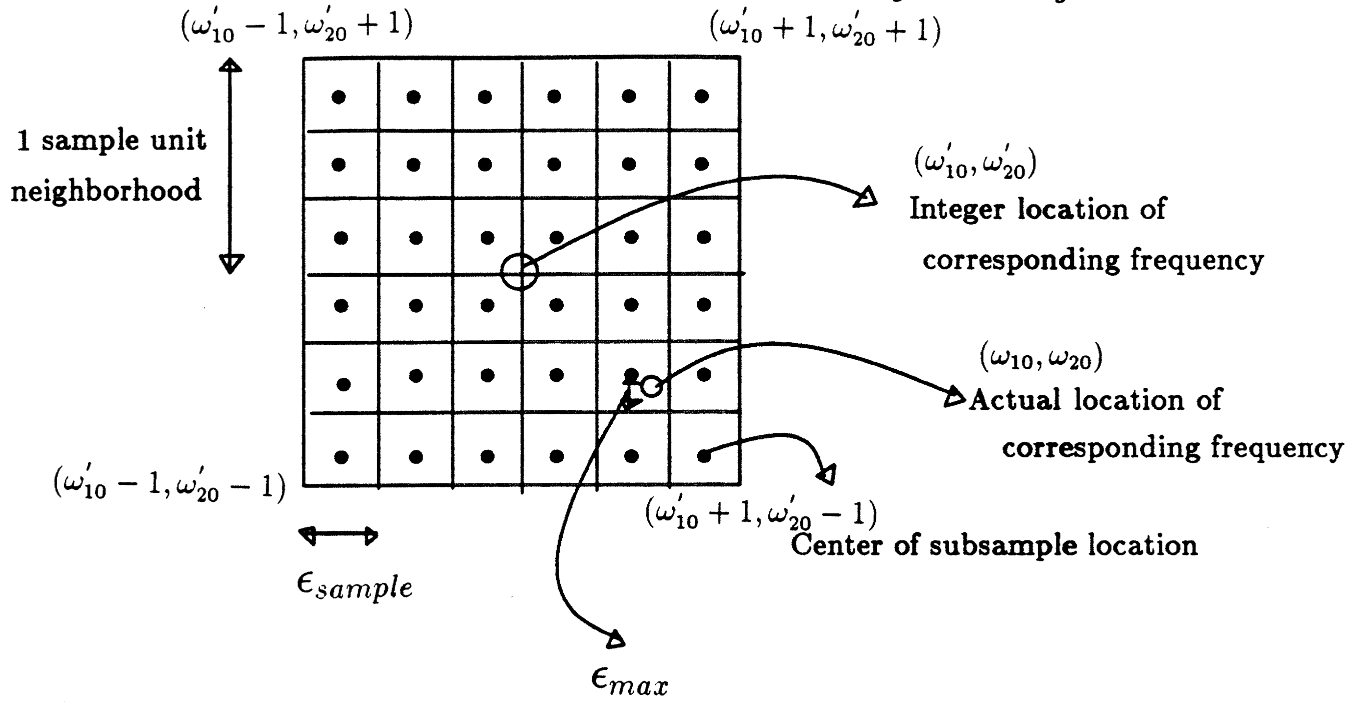

$(\omega_{10}' - 1, \omega_{20}' + 1)$
$(\omega_{10}' + 1, \omega_{20}' + 1)$
1 sample unit neighborhood
$(\omega_{10}', \omega_{20}')$
Integer location of corresponding frequency
$(\omega_{10}, \omega_{20})$
Actual location of corresponding frequency
$(\omega_{10}' - 1, \omega_{20}' - 1)$
$(\omega_{10}' + 1, \omega_{20}' - 1)$
Center of subsample location
ϵ_{sample}
ϵ_{max}

Figure 4.29: *Illustration of the method of local subsampling of sampled LP spectra for texture matching.*

rate ϵ_{sample}, the locations of corresponding frequencies can be chosen to be the centers of the bins each of which is ϵ_{sample} wide along each frequency dimension. In that case, the maximum error in $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ using the subsample location at the center of a bin is $= +/\text{-}\epsilon_{sample}/2$ when the correct bin is examined (see Figure 4.29 for an illustration of resampling). Thus for a given choice of $(\omega_{11r}, \omega_{12r})$ and $(\omega_{21r}, \omega_{22r})$, and a specified tolerable error in the estimation of parameters δ_{max}, the sampling rate ϵ_{sample} can be chosen such that

$$\delta_{max} = max\left\{abs(\frac{(\omega_{22r}\epsilon_1 - \omega_{12r}\epsilon_2)}{(\omega_{11r}\omega_{22r} - \omega_{21r}\omega_{12r})}), abs(\frac{(\omega_{11r}\epsilon_1 - \omega_{21r}\epsilon_2)}{(\omega_{11r}\omega_{22r} - \omega_{21r}\omega_{12r})})\right\} \leq 2^{-g}.$$

To get an estimate of the sampling rate that may have to be used in practice, let $(\omega_{11r}, \omega_{12r}) = (\frac{2\pi}{N_{fft}}.2, \frac{2\pi}{N_{fft}}.2)$, $(\omega_{21r}, \omega_{22r}) = (\frac{2\pi}{N_{fft}}.3, \frac{2\pi}{N_{fft}}.1)$, $N_{fft} = 64$, $\epsilon_1 = \epsilon_{sample}/2$, $\epsilon_2 = -\epsilon_{sample}/2$, and $\delta_{max} = 0.03067$ (or 3% error), then by sampling at a rate of $\epsilon_{sample} = \frac{4\pi}{N_{fft}}.2^{-2}$, that is by sampling only 16 locations, the error in the estimation of parameters can be restricted to within the desired limits[11]. Thus if the original correspondence was correct (except for the location inaccuracy), then by subsampling the unit sample interval surrounding the location ω'_{ij0} at locations ϵ_{sample} apart, and solving for the parameters for each such location, it is guaranteed that for at least one such sample location, the solution of the parameters lies within the specified error bound δ_{max}.

An example: We now illustrate the use of LP spectrum-based matching for the solution of the pose of a model texture by an example. Figure 4.30a shows a texture of size 128 x 190 whose LP spectrum using a 3x3 noncausal AR model (i.e. 48 AR parameters) and a 32 x32 point FFT to compute the LP spectrum is shown in Figure 4.30d. Figure 4.30b shows an instance of the same texture rotated in plane by about 7 degrees. The lighting conditions and camera geometry was fixed while taking the texture image shown in this figure. The LP spectrum of the transformed texture (again using a 3x3 noncausal AR model and a 32x32pt. FFT for computing the LP spectrum) is shown in Figure 4.30e. By pairing the two highest amplitude frequencies of Figure 4.30e with the two highest amplitude frequencies of Figure 4.30d, (here $(\omega_{11r}, \omega_{12r}) = (\frac{2\pi}{32}.1, \frac{2\pi}{32}.0)$, and $(\omega_{21r}, \omega_{22r}) = (\frac{2\pi}{32}.2, \frac{2\pi}{32}.1))$, and by locally sampling at 81 locations per sample frequency, the error was guaranteed to be within 8%, and the transformation parameters were obtained as $a = 1.0, b = 0.11111114, c = -0.11111114$, and $d = 0.9999999$, giving a rotation in plane of 6.38deg. The texture obtained by applying the transformation implied by the parameters a, b, c, d to the model texture is shown in Figure 4.30c. As can be seen from this figure, the orientation change in the texture shown in Figure 4.30b is well-captured in the recovered transformation parameters obtained by matching the respective LP spectra.

[11]The sampling rate derived above limits the maximum error. Depending on the location of the actual corresponding frequency location within the sampling interval, it is possible to obtain an actual error of less than the maximum error using even a lower sampling rate.

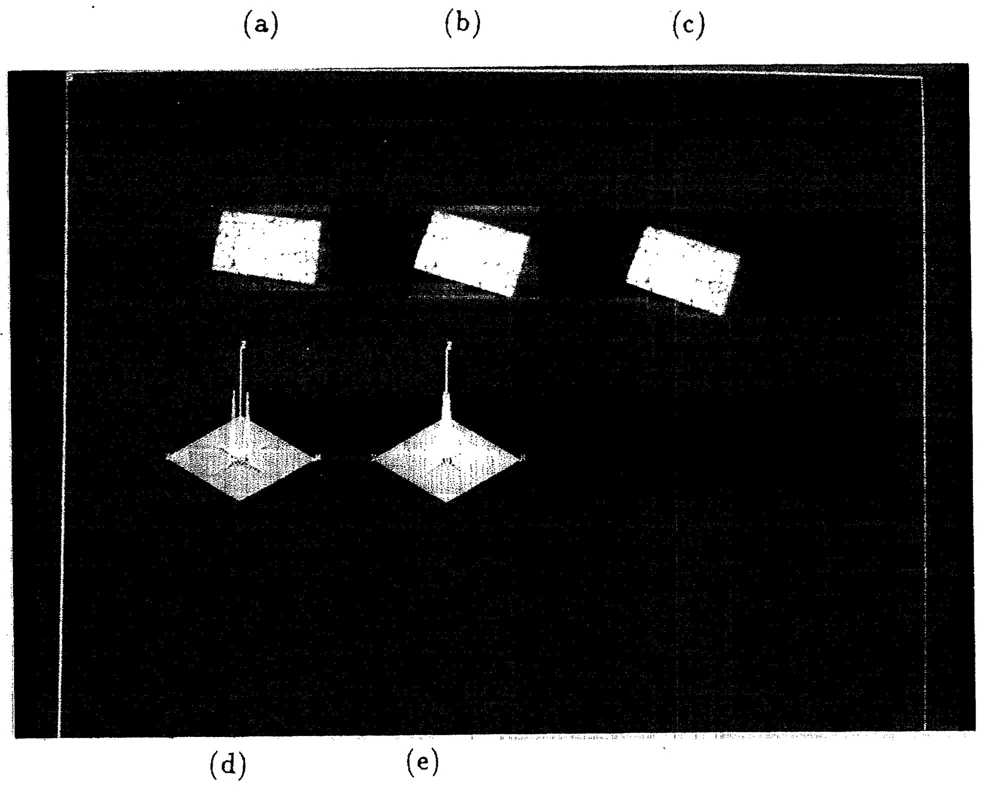


Figure 4.30: *Illustration of the solution of the pose of the model texture using LP spectrum. (a) Model texture. (b) An instance of the model texture rotated parallel to the image by 7 degrees. (d) LP spectrum of the model texture of (a) using a 3x3 non-causal AR model. (e) LP spectrum of the image texture of (b) using also a 3x3 non-causal AR model. (c) Recovered texture by solving for the transformation parameters using correspondence between two pairs of peaks in the LP spectra of model and image texture shown in (d) and (e).*

4.4.3 Texture matching under incomplete isolation of image texture regions

The method of matching LP spectra described in the above section considered the case where the image texture was perfectly isolated, i.e. contained all of and only the model texture. In practice, due to occlusions we can expect the image texture region to contain only a portion of the model texture. Also, due to illumination changes, the appearance of the image texture may be different. Although the LP spectrum is relatively stable to such changes in the model texture, it does show some missing as well as spurious peaks. For this reason, we developed a match measure that emphasizes those correspondences that show the best match under these conditions. Since the LP spectra of the model and the image region are registered as a result of alignment, they are matched by pairing each peak of the projected model's spectrum to the peak of the windowed region's LP spectrum that is nearest to it in magnitude (normalized w.r.t the highest peak) and lying within a neighborhood of certain radius $r (r \approx 2)$.

The match measure we describe below uses information derived from common (i.e matched) peaks as well as missing and spurious peaks in deciding to declare a match. It is very similar to the one presented in Section 4.2.3, but with the notion of matched peaks changed as given above.

LP spectrum-based match measure

Let P and P' denote the set of peaks in the LP spectra of the model and image texture regions respectively, obtained by taking a 2Px2Q order model for AR estimation in both textures. That is

$$P = \{p_1, p_2, ...p_m\}, P' = \{p'_1, p'_2, ...p'_n\} \tag{4.54}$$

Let the amplitude of peak p_i be denoted by $V(p_i)$ and its frequency location by $l(p_i)$. Let the set PP' denote the set of matched peaks (as defined above) after alignment as

$$PP' = \{(p_{i1}, p'_{j1}), ...(p_{ik}, p'_{jk})\}. \tag{4.55}$$

Let

$$PM = \{p_{i1}, ...p_{ik}\}, PM' = \{p'_{j1}, ...p'_{jk}\} \tag{4.56}$$

represent the first and the second of the pairs in the set PP'. Then the set P-PM denotes the set of missing peaks while the P$'$-PM$'$ denotes the set of spurious peaks using the model texture as reference.

The difference measure was obtained by recording the following information about each of the matching, the missing, and the spurious peaks : (i) their number, (ii) their relative amplitude, and (iii) their respective distributions of locations.

(a) Number of missing peaks: In general, the more the number of missing peaks, the less the certainty that the image texture is likely to contain the model texture. Further, the uncertainty is enhanced if the peaks that are missing are of high energy. These two factors are reflected in the following weighting function

$$Fact_1 = \frac{\sum_{p_i \in P-PM} V(p_i)}{M_p * V_{max}} \tag{4.57}$$

where $V_{max} = max\{V(p_i)\}$, and $M_p = \mid P - MP \mid$. The factor is normalized so that it lies between 0 and 1 with 1 indicating a greater difference between the two spectra.

(b) Number of spurious peaks: Once again, the more the number of spurious peaks, the less the likelihood that the model texture is present in the given image texture region. Further, this conclusion will be strengthened by the presence of strong spurious peaks. The weighting function below reflects this rationale:

$$Fact_2 = \frac{\sum_{p'_j \in P'-PM'} V(p'_j)}{N_{p'} * V'_{max}} \tag{4.58}$$

where $V'_{max} = max\{V(p'_j)\}$ and $N_{p'} = \mid P' - PM' \mid$.

(c) Number of matched peaks: The confidence in the match decreases as very few of the peaks in the two spectra are matched. This is expressed in the following factor:

$$Fact_3 = \frac{2 \mid PP' \mid}{\mid P \mid + \mid P' \mid} \tag{4.59}$$

LP spectrum-based match measure: Since each of the factors detect a certain type of difference that reinforce each other, the LP difference detection measure is formed by a simple linear combination as

$$Lp - diff = \sum_{i=1}^{3} Fact_i \tag{4.60}$$

The LP difference measure can now be used to rank the match for a given correspondence as explained in Section 4.4.2. The best match over all frequency correspondences considered is taken as representative of the match of the model's texture to the given region. Ultimate verification of the match may to be done in the space domain, however, to confirm the presence of the model texture. This can be done by projecting the model texture into the image using the transformation parameters and comparing the transformed model texture to the image texture region. The match in this case, however, requires the knowledge of translation.

4.4.4 Analyzing the texture image for model-driven selection

The method of matching LP spectra described in the earlier section required the isolation of an image texture region that contained only the model texture (even if not all of it is isolated). Such regions can be obtained, under some conditions, without an elaborate texture region segmentation of the image. This can be done by moving a window over the image and maintaining overlap between successive window positions as indicated in Figure 4.31. We now show that under some conditions, it is possible to find a window that spans only the model texture region regardless of the position and orientation of the model texture in the given image. For this, we first consider the case when the model texture in the image appears as a rectangle of size N_1 x N_2. Suppose we choose to analyze the image of dimensions I_1 x I_2 using rectangular windows of size M_1 x M_2 and an overlap of L_1 x L_2 as shown by the rectangles A and B in Figure 4.31. Suppose the rectangular patch of texture occurs at a location(x, y) in the image as shown by the rectangle C in that figure. Each analysis window starts at a location $(k.L_1, l.L_2)$ where k and l are integers such that $0 \leq k \leq \lfloor \frac{I_1-M_1}{L_1} \rfloor$ and $0 \leq l \leq \lfloor \frac{I_2-M_2}{L_2} \rfloor$. A window that spans only the model texture region must satisfy the constraints for its starting location as $k.L_1 \geq x$ and $k.L_1 \leq x+N_1$ and $k.L_1+M_1 \leq x+N_1$ for the x-coordinate and similarly $l.L_2 \geq y$ and $l.L_2 \leq y+N_2$ and $l.L_2+M_2 \leq y+N_2$ for the for the y-coordinate (by the rectangle containment rule). These constraints reduce to finding an integer value of k such that $\frac{x}{L_1} \leq k \leq \frac{N_1+x-M_1}{L_1}$. Such values of k and l exist provided $\frac{N_1+x-M_1-x}{L_1} \geq 1$, and $\frac{N_2+y-M_2-y}{L_2} \geq 1$, that is, $M_1+L_1 \leq N_1$ and $M_2+L_2 \leq N_2$.

Thus, as long as we choose a window size smaller than that of the texture and an overlap such that $L_1 \leq N_1 - M_1$ and $L_2 \leq N_2 - M_2$, we can scan the image using overlapping windows and find a window that spans a portion of the model texture region.

Next, consider the case of an actual instance of the model texture appearing in the image. We assume for simplicity that the texture on the model object is square-shaped of dimension N x N [12]. Then under linear transformation of the model object, the square texture region undergoes an affine transformation (specified by the parameters a, b, c, d, again ignoring translation) to form a parallelogram as shown by the shape marked E in Figure 4.31. Within this parallelogram, we can inscribe a rectangle of largest area by considering the closest opposite vertices as shown by the rectangle marked F in Figure 4.31. The dimensions of this rectangle indicated by N_3 x N_4 are a function of the transformation and are given by $N_3 = (a-b)N$ and $N_4 = (d-c)N$. Then by the previous analysis on rectangular texture regions in an image, if the overlapping windows satisfy the constraints $M_1+L_1 \leq N_3$ and $M_2+L_2 \leq N_4$, then a window containing only the model texture region will be

[12]Restricting to a square shape is not critical since a square can be inscribed in any arbitrary-shaped texture region on the model object.

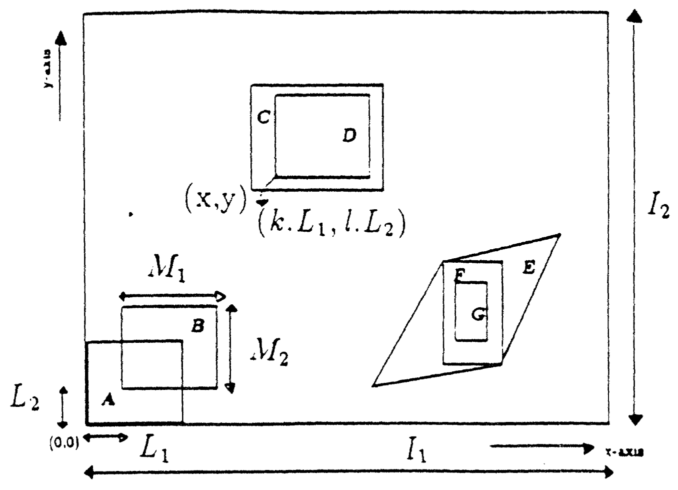


Figure 4.31: *Illustration of overlapping window analysis. The windows labeled A and B are two overlapping windows used to analyze the image. The extent of overlap is indicated by* (L_1, L_2). *A rectangle D that spans only the texture region in the rectangle C always exists under the conditions indicated in text. The parallelogram marked E represents the result of 2D affine transformation on an originally square texture region. The windowed region marked G spanning a portion of E can always be found under the conditions indicated in text.*

found. Although the dimensions N_3 and N_4 are unknown, we can place a bound on these values by restricting the allowable transformations on the model object (and hence the model texture) by requiring $(a - b) \geq s$ and $(d - c) \geq s$, where s is chosen to lie between 0 and 1. This bounds the dimensions N_3 and N_4 to be such that $s.N \leq N_3 \leq N$ and $s.N \leq N_4 \leq N$. With this bound on the dimensions, the windowing constraints can be satisfied by choosing the overlap and window dimensions such that

$$M_1 + L_1 \leq s.N \tag{4.61}$$

$$M_2 + L_2 \leq s.N \tag{4.62}$$

4.4.5 Texture-based model-driven selection

We now combine the concepts of moving window analysis, the LP spectrum representation, and the matching of LP spectra described in the earlier sections to develop a texture-based model-driven selection mechanism. First, the model texture is described by its LP spectrum using a suitable order AR model. Then the dimension N of the largest square region containing the model texture is noted. Next, a bound $''s''$ is chosen to limit the allowable transformations. The values of s and N are used to decide the window width (M_1, M_2) and the overlap (L_1, L_2) as specified in Equations 4.61 and 4.62. The image is analyzed by the overlapping windows, and the image region within each analysis window is described by its LP spectrum using the same order AR model as the model texture[13]. The LP spectrum of each window is then tried as a possible match to the model texture's LP spectrum as described in Section 4.4.2. The best few matches in the image are taken to indicate the possible places where the model texture (and hence the model object itself) could appear in the image.

[13]If the image region contains the model texture then for limited scale changes, the neighborhood of pixel dependence in the texture will remain the same, justifying the choice of the same order AR model for the image region

Results

We now illustrate texture-based model-driven selection by an example. Figure 4.32a shows a view of the model object (cup) and the texture patch on it that serves as model texture is shown in Figure 4.32b. The LP spectrum (using a 32x32 pt. FFT) of the model texture using a 2x2 non-causal AR model is shown in Figure 4.32c. Figure 4.32d shows a scene in which two instances of the model object appear, one of them reflecting a 3d affine transformation and appearing at a different imaging distance (small increase in the apparent size of the object). Since the model texture patch was of size 72 x 52 (was later zero-padded to form a square of size 72 x 72), the image was analyzed using windows of size 48 x 32 with an overlap of 24 x 16 (here s was chosen to be 1.0) to satisfy the constraints of equation 4.61 and 4.62. The various windows in which a match for the model texture was found are shown in Figure 4.32e. The LP spectra for some contiguous overlapping window regions that were found to match the model texture's LP spectra are shown in Figures 4.32f-i. As can be seen from the figure, although the correct matches are found where the model object exists, there are also a few spurious matches. In general, as the match measure developed in Section 4.4.3 is designed to tolerate some missing and spurious peaks, it can occasionally cause some false positives to occur even after the correct alignment is obtained using the method described in Section 4.4.2. These false positives can usually be eliminated when texture-based selection is used in combination with other cues such as color as we will see in later chapters.

Figure 4.33 shows another example of model-driven selection using texture being performed on a scene consisting of other objects beside the model. The model object is shown in Figure 4.33a, and a portion of its texture (a more or less planar patch) chosen as the model texture is shown in Figure 4.33b. Figure 4.33c shows a scene in which the model object occurs. This scene is taken under slightly different illumination conditions, and the model object in the scene occurs at a slightly different orientation, being rotated to the left about the vertical axis. The selected regions using the LP spectrum-based matching described above are shown in Figure 4.33d. As can be seen from this figure, most of the texture on the object has been iso-

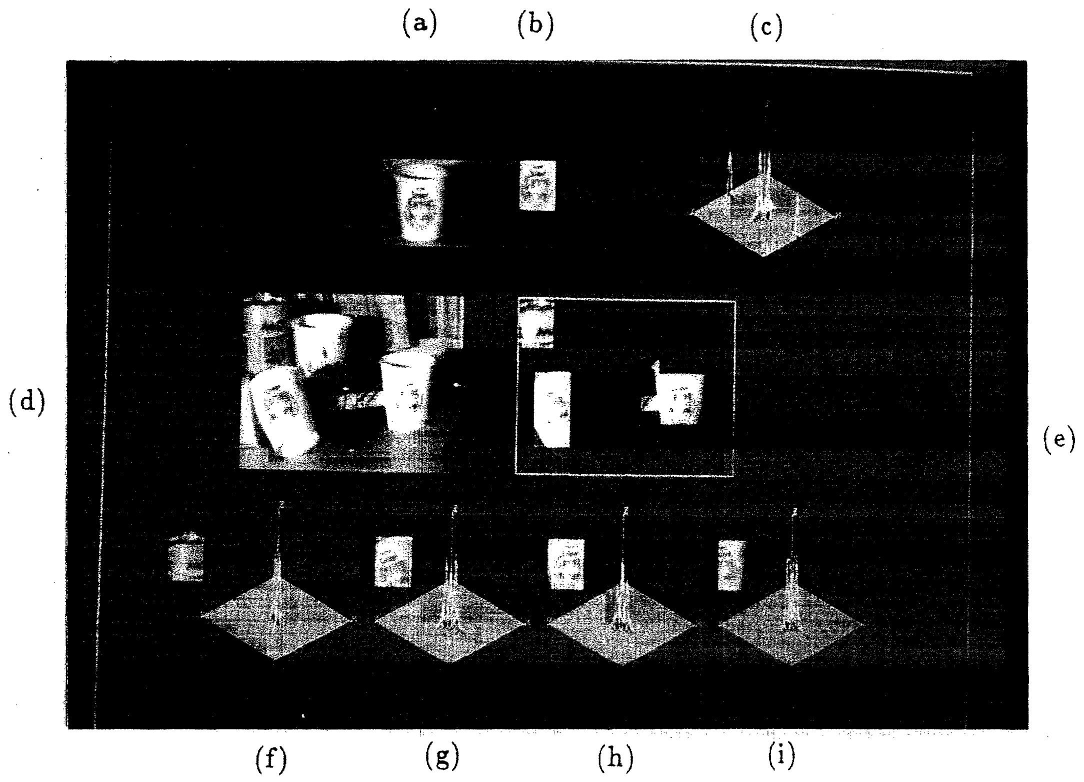


Figure 4.32: *Illustration of model-driven selection using texture. (a) A view of the model object. (b) An extract serving as model texture. (c) The LP spectrum of the model texture using a 2 x 2 non-causal AR model. (d) A scene in which instances of the model object occur. (e) Result of model texture-based selection. (f) -(i) The LP spectra of some of the windowed regions of (e) that were matched to the model texture by the match measure described in text.*

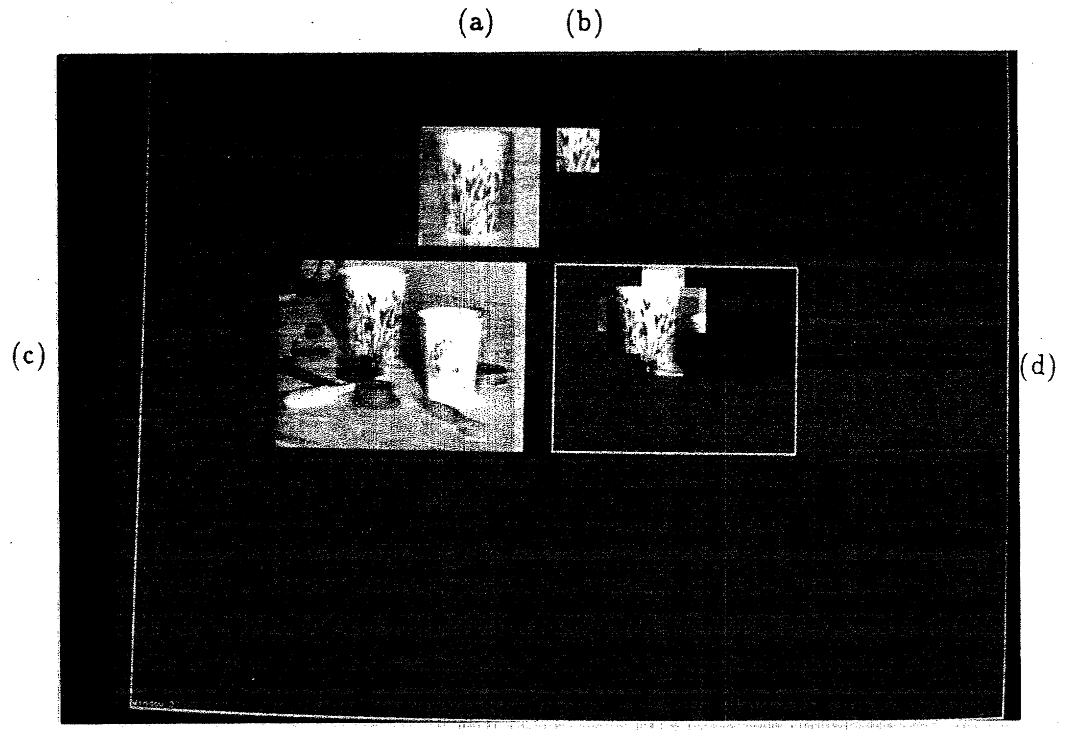


Figure 4.33: *Model-driven selection using texture. (a) A model of a paper cup. (b) A nearly planar extract of the cup serving as model texture. (c) A scene in which the cup occurs. (d) Result of texture-based model-driven selection.*

lated using the selection mechanism, although there are again a few false positive selections.

4.5 Use of Texture-based Model-driven Selection in Recognition

So far we have discussed how a texture pattern on a model object can be used to locate areas in an image that are likely to contain the model object. A recognition system can use this information by choosing features such as points or edges from within these regions for finding the corresponding features to solve for the pose of the model object in a given image. Specifically, if a recognition method such as the linear combination of 2D views is used, it is known that at least four non-coplanar corresponding features are required for finding an alignment transform and hence for recognizing the object [157]. Since we assumed the texture region on the model object to be planar, texture-based selection alone is not sufficient for recognition and at least one of the features needed for an alignment transform (for the object) must come from outside the texture region. But having a much smaller region to search for the rest of the corresponding features can greatly reduce the search involved, particularly, because more than 4 features are actually needed in practice to find a reasonable alignment transform. To get an estimate of the search reduction, let (M,N) = the number of features such as corners or edges present in the model object and image respectively. Let (M_t, N_t) = the number of feature left in the texture region of the model and image respectively. Then using the alignment method that requires at least 7 corresponding features in practice for good alignment $O(M^7N^7)$ matches may have to be tried in the worst case, without any selection. When texture-based selection is used, most of the features (up to six) can be chosen from within the selected texture region so that the number of matches reduces to $O(M.N.M_t^6N_t^6)$. To see the search reduction using these estimates, we isolated corner features from both the model and image and recorded the number of such features within the regions of model texture-based selection. The results are recorded in Table 4.4. As can be seen from this table, there is a fairly large reduction in the number of matches when texture-based selection is used as compared to the case when no prior selection is done. However, a recognition system that actually explores the number of matches indicated in the table is far from practical. These estimates can often be improved when texture-based selection is combined with selection based on other cues such as color as described in the previous chapter. We will describe the results of such experiments using an actual recognition system in Chapter 8.

Although the search is reduced, texture-based matching does not eliminate the chance of false negatives in selection. This happens mostly in instances when the model texture and the image of the scene do not satisfy the assumptions made in

S.No	M	N	M_t	N_t	Color selected corners in		Estimated search	
					model	image	No selection	With selection
1.	114	484	35	38	69	45	1.55×10^{33}	3.05×10^{23}
2.	96	580	84	74	12	34	1.66×10^{33}	3.2×10^{27}
3.	96	1401	84	76	12	25	7.96×10^{35}	9.1×10^{27}
4.	64	256	52	47	12	31	3.17×10^{29}	3.49×10^{24}
5.	138	392	58	145	80	67	1.35×10^{33}	1.91×10^{28}

Table 4.4: *Estimated search reduction during recognition using texture-based model-driven selection.*

the approach as well as in cases where the texture match measure misses some likely matches.

4.6 Conclusions

In this chapter we have presented an approach to data and model-driven selection using texture information. As in color, we have once again used the intended task to design a representation of texture as well as ways of analyzing textured images. Unlike color though, we took here a signal processing approach to address data and model-driven selection. Thus autoregressive modeling of textures and the linear prediction spectrum was advocated as being appropriate texture representations for both data and model-driven selection. Modeling the texture image as a short-space stationary process led to a texture region segmentation that satisfied the requirement of data-driven selection. A theory of texture saliency was then presented that led to a way of identifying salient texture regions in images. Lastly, the LP spectrum representation was shown to be an appropriate representation of model texture information that not only allowed its isolation in scenes without requiring detailed segmentation, but also led to a solution of the pose of the texture region on the model object. This points to a new use of selection in recognition (apart from the search reduction), namely, to provide a partial solution of the pose of the model object. In later chapters, we shall evaluate this approach to texture-based selection by integrating with a recognition system and estimating its performance using texture-based selection.

Chapter 5

Selection using Parallel-line Groups

In the previous chapters, we saw how color and texture information present in images could be used to select relevant regions in the scene for recognizing an object. This reduced the number of data features to be considered for recognition by allowing large portions of the image to be ignored. It also allowed the data features to be grouped based on their enclosing color or texture regions. By providing correspondence between such regions in the model and image, the possible matches to features on a model object was restricted to lie within the corresponding enclosing regions. But since the regions isolated by such selection mechanisms were rather large, and contained a large number of features, the number of matches between model and image features was still considerably large. If features within such regions could be grouped further such that only a small number of features fall into a group, then by finding a correspondence between such small region groups on the model and image, the total number of combinations of model and image features can be greatly reduced. In this chapter, we shall explore the use of a property called closely-spaced parallelism that is often exhibited by lines on objects, to create small-sized groups of lines on the model object and in the image. Since the end result here is a grouping of lines based on a constraint, this falls into the class of grouping methods that have been tried earlier for recognition.

The chapter is organized as follows. We first discuss the need for grouping data features into small-sized groups for the purposes of recognition. This gives us a set of requirements that must be met by any scheme for grouping data features. We then briefly review the existing grouping methods in the light of these requirements. Next, we present a method for grouping line features that exploits the property of closely-spaced parallelism among lines. We show how data and model-driven selection can be performed using such line groups. Once again, in keeping with our general strategy of using the paradigm of attentional selection, data-driven selection is achieved by selecting some salient line groups, while model-driven selection is performed by utilizing the description of line groups on the model object.

5.1 Role of Grouping in Model-based Recognition

In the earlier chapters, we saw how region selection using color and texture reduced the search involved in recognition by removing a large number of data features from consideration. Even so, once a set of regions is selected, a large number of matches between features in corresponding model and image regions may have to be tried. Using the alignment method for recognition [1] (in particular, the linear combination of views version of this method [156]) we know that at least four matching features must be found for alignment (and hence recognition). If there are M features (say, points) in a model region and N features in the corresponding image region, then $O(M^4N^4)$ matches per pair of corresponding regions may have to be tried, in the worst case, with such region selection. For the typical number of features ($M \approx 100, N \approx 300$) found in color or texture regions, this is still a very large number of matches to be tried ($\approx 10^{18}$) . If the data features within such regions can be further grouped into some meaningful structures or groups consisting of a small number of features each, then the search can be reduced by pairing such groups, and trying

[1] Recognition methods are described in Chapter 7.

combinations of features within matching groups, as before. Previous research has explored the role of grouping in recognition for reducing the search in precisely this fashion [76, 25, 95, 59]. To see how grouping of features can reduce the combinatorics of search drastically, we reproduce here the analysis of grouping given in earlier work [76, 25].

Let us consider the case of grouping being performed both on the model and image features. Let M_g and N_g be the number of model and image groups respectively, and let m_i and n_j be the number of features in the model group i and image group j. If the size of the model and image groups are identical, and each group contains features coming from a single object, then the number of matches that need to be tried are $O(\sum_{i=1}^{M_g} \sum_{j=1}^{N_g} m_i!)$

since all pairs of model and image groups may have to be tried and $m_i!$ accounts for all permutations of feature matches within a pair of matching groups. Further, if the features in a group can be linearly ordered, the number of matches reduces to $O(\sum_{i=1}^{M_g} \sum_{j=1}^{N_g} m_i)$. If the number of features in the model and image groups are not identical or if not all the features in groups come from a single object, then assuming at least one image group contains at least 4 features of a model group, a solution for the pose of the model object can be obtained by trying, in the worst case, all matches of four features within each pair of image and model groups. The number of matches that need to be tried in such case becomes $O(\sum_{i=1}^{M_g} \sum_{j=1}^{N_g} m_i^4 n_j^4 4!)$.

For small-sized groups (say, about 5 features each), this is essentially $O(M_g N_g)$ or linear in the number of groups [2].

From the above analysis, we see that in order to reduce the search involved in recognition, a grouping scheme must possess some desirable properties. Ideally, a grouping method must produce highly reliable (that is, groups coming from a single object) equal-sized groups in the model and image. If this is not possible, it must contain at least a sufficient number of features (four being the minimum) coming from a single object to make recognition possible. When groups satisfy this

[2]This ignores the effort required for verifying a match assuming it is the same for recognition with or without grouping.

"minimum reliability", the number of extraneous features in a group must be as small as possible. In other words, it is desirable to have small-sized groups, so that the complexity of search remains linear in the number of groups. Another requirement to keep the number of matches small is to lower the number of possible groups (to a low-order polynomial). The number of groups cannot, however, be reduced by arbitrarily discarding groups as this could create unnecessary false negatives during recognition. That is, it may cause an object to be not recognized because groups corresponding to the model groups were discarded. Finally, since grouping is a pre-processing step to recognition, the group generation process (i.e., the algorithm for assembling the groups) itself must be fast and simple.

The above discussion suggests that one of the keys to making grouping useful for recognition is to group features in an image based on constraints that capture some salient and easily detectable structures that point in turn to meaningful structures on objects in scenes. In this way, the number and size of groups can be kept small, since not all tuples of features will be meaningful, and being easily detectable, the groups can be generated by a fast and efficient algorithm. Finally, since such groups point to meaningful structures on objects in scenes, they tend to be more reliable.

5.1.1 Approaches to grouping

We now examine some of the previous work on grouping in vision in the light of the above requirements for their use in recognition. We will focus here on grouping of edge features, remarking on grouping methods for other data features only briefly. More extensive reviews of grouping are available elsewhere in literature [76], [95].

Grouping was initially studied in psychology, mainly as a perceptual phenomenon. There it was noticed that when we look at an edge image of a scene, we often pick up any structural information present in a collection of edges or lines. Figure 5.1 illustrates this with examples of line arrangements in which we can identify some perceptual structure. Early Gestalt psychologists demonstrated through a variety of examples that humans use cues such as simplicity, proximity, similarity, symmetry and familiarity for grouping features [162]. Their explanation for the perception of

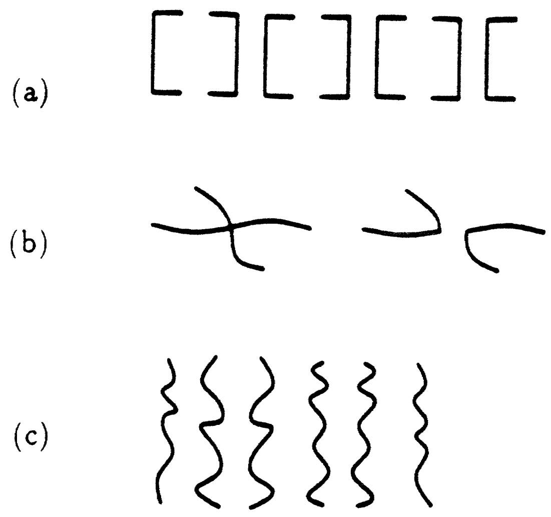


Figure 5.1: *Illustration of perceptual structure apparent in line arrangements. (a) A group of lines perceived as a collection of squares due to closure (or continuation). (b) Lines seen as crossing due to good continuation. (c) Bilateral symmetry evident from the group of lines shown. (Adapted from Figure 2-1 of [95])*

groups based on such cues seemed plausible but lacked a quantitative basis due to the difficulty in precisely defining concepts such as simplicity and familiarity. Later studies tried to make terms such as simplicity a little more concrete by using the concept of minimum entropy from information theory [65]. The explanations put forward by psychologists about this ability to group a collection of features based on constraints all seem to imply that it reflects an underlying knowledge of what makes a collection of edges come from a single object. In other words, the grouping process reflects an inherent bias towards collecting those edges that are likely to belong to a single object.

While the work on grouping in psychology had concentrated on observing it as a phenomenon and developing explanations for it, the work on grouping in computer vision has focused more on ways of making it useful for computer vision. Towards this end, several roles of grouping have been envisaged. In the early work of Marr, for example, grouping was suggested as a way of abstracting information in the raw primal sketch derived from the image [102]. He suggested grouping based on constraints such as curvilinearity, parallelism, and collinear displacements. Later work developed techniques to perform grouping such as the use of the Hough transform to capture collinearity information in points [40]. Grouping was also suggested as a useful step in both geometric and symbolic methods of recognition. Lowe proposed grouping as a way of establishing good primitives for recognition [95]. Jacobs and Clemens showed the extent of search reduction possible using grouping as a preprocessing step in recognition [27]. The role of grouping in geometric methods of recognition was merely to organize the data features, while the actual recognition was done by using the data features from the groups. The role of grouping in symbolic recognition methods, on the other hand, was to provide the groups themselves as high-level match primitives to be used directly for recognition using symbolic reasoning techniques. Early vision systems used grouping in this sense, such as ACRONYM in which edges were grouped into ribbons and the recognition of objects proceeded based on the ribbons and their topology [17]. More recently, grouping has been used for purposes of indexing into a library of objects in geometric methods of recognition [26], [90]. It has also been used for this purpose in symbolic methods of recognition to extract meaningful structures in scenes based on constraints of parallel and skew symmetry [138], [105] and proximity[38].

The role of grouping in extracting meaningful structures in scenes has also been emphasized in grouping schemes based on region and contour features (rather than edge or point features). This can be seen in the work of Shashua and Ullman on the grouping of image contours to capture salient curves [136], of Dolan and Weiss on the grouping of curved lines using proximity [38], and of LeClerc on hierarchical grouping of regions based on the minimum description length principle [92].

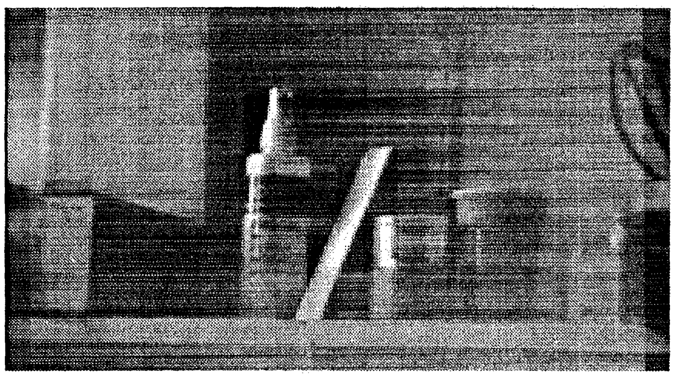
A class of approaches in computer vision have attributed the tendency to group features to the ability of humans to recognize the non-accidental occurrence of the relation underlying the groups. That is, the degree to which a relation is unlikely to have arisen by an accident of viewpoint, rather than the knowledge that they belong to a single object, is the motivation behind grouping features based on that relation. This was concluded by Witkin and Tenenbaum [165] after observing that such non-accidentalness was used to interpret groups of features even when the ultimate interpretation of the groups was not known. They also pointed out that since such a relation was expected to remain stable over a large number of viewpoints, it must reflect some meaningful structure in the scene. Lowe extended the non-accidentalness argument behind grouping to identify the set of image relations that are unlikely to occur by an accident of viewpoint [95]. Using the assumption that the viewpoint of the camera is independent of the objects in the scene, he showed that only certain image relations, such as convexity and parallelism, are likely to remain stable over a large range of viewpoints. He also concluded that because of this viewpoint invariance, the detection of such relations in an image implied that they were likely to be the projection of a meaningful and specific 3d structure. Lowe showed that this property can make such groups useful for the recognition of three dimensional objects. For example, using the non-accidentalness of viewpoint, he showed that parallel lines in the image are most likely to come from parallel lines in space. So if the model object contained parallel lines, then this justifies the matching of (groups of) parallel lines in the image to (groups of) parallel lines on the model, thus making such groups useful primitives for recognition.

Another class of approaches in computer vision explored the same argument for grouping that was put forward by psychologists based on the likelihood of the grouped features coming from a single object. For example, in the early work of Roberts, vertices connected by straight edges were grouped using the rationale that connected vertices were likely to be part of the same object [128]. This argument was given a more quantitative basis by Jacobs who proposed that grouping of data features in an image should be based on a relation that points to the likelihood of such features coming from a single object [76]. Specifically, he used distance and orientation constraints to explore the convexity relation between a group of edges. By doing a statistical analysis of occlusions and merging of edges with background in images, he showed that it is unlikely for a randomly selected group of edges to form a convex polygon, thereby implying that the detection of such a relation between edges pointed to their likelihood of coming from single objects. Other researchers that have used a similar argument for grouping are Bolles and Cain who used the proximity relation to groups features in their local feature focus method [8], Brooks who grouped edges forming ribbons or trapezoids [17], and Clemens who grouped edges enclosing open regions [25].

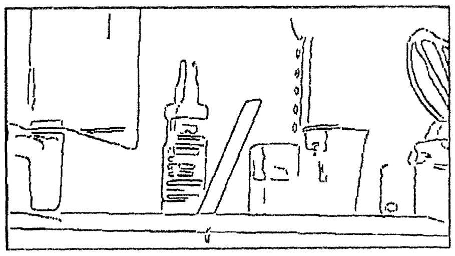
Let us now evaluate some of the existing schemes for grouping from the point of recognition. Grouping schemes, such as that of Shashua and Ullman [136] for grouping image contours, that attempt to capture meaningful structures in scenes without reasoning about scene geometry, often produce groups that span wide areas of the image, making them unreliable for recognition. Other grouping schemes based on viewpoint invariance that use constraints or relations that are likely to hold over a wide range of viewpoints such as parallelism [95], or convexity[76], [72] also produce groups that attempt to capture meaningful structures in the scene. However, the ease with which such relations are detected in images decides the number of groups generated as well as their size. Since several nearby edges could satisfy relations such as parallelism and convexity, different combination of edges have to be explored by grouping algorithms leading to a very large number of groups and taking time of equal complexity. For example, Huttenlocher grouped edges based on connectivity by considering all possible sequences of edges of length three (leading to $O(N^3)$ groups for N edges)[73]. Later work on grouping tried to generate a smaller number of groups by filtering some of the groups. In Huttenlocher and Wayner [72] for example, a grouping algorithm was presented that works in $O(n\ logn)$ time and generates a linear number of convex edge groups. The filtering was done by using a cost function to rank neighbors of an edge and allowing only the least-cost neighbor to participate in a convexity relation. Although the number of groups are restricted by this method, it is not clear whether such decisions can be made on a purely local basis. Also, since there is no analysis of the kind of groups that will be missed, it is not clear that such groups do not cause a recognition system to make false negative identifications. Another grouping scheme explored by Clemens also restricts the number of groups to be linear in the number of edges [25]. Here the groups are designated by open regions that are enclosed by a group of edges. Such open regions of the image were considered likely to come from a single object because a transition from one object to another almost always caused a change in intensity sufficient to produce an edge that splits a region. The grouping algorithm used assigns an edge to at most four regions thus ensuring that the number of groups remains linear in the number of edges. Due to feature instabilities, imaging artifacts, etc. an open region is rarely bounded by edges forming a complete connected closure, causing such assignments of edges to 4 neighbors using purely local judgment to group together edges that do not necessarily come from the single object. Thus in the existing approaches to grouping, it appears that restricting the number of groups may either cause some relevant groups to be missed or may make the grouping scheme unreliable, causing it to group features that don't necessarily come from a single object.

5.2 Grouping Based on Closely-Spaced Parallelism

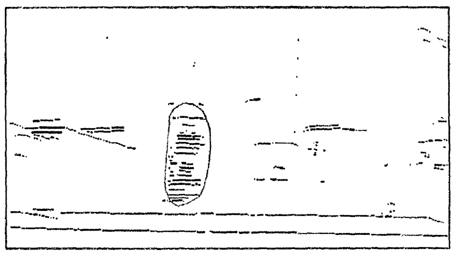
We now present a grouping method that exploits the relation of closely-spaced parallelism commonly occurring between lines on objects, to produce groups that possess many of the desirable properties for purposes of recognition. Many commonly occurring objects in indoor scenes such as books, cups or tables possess some pattern-like structures that often attract our attention. Such structures usually contain groups of closely-spaced parallel lines of a few orientations. For example, printed letters on the surface of an object such as a book, or a bottle, and wooden texture on pieces of furniture such as a table contain groups of closely spaced parallel lines. Sometimes such parallel lines form texture-like patterns as on the bottle in Figure 5.2a, while in other cases they capture some interesting structures from parts of objects such as the parallel contours in the triangular block of Figure 5.2a. Even when they can be treated as textures we consider them as a separate cue for the following reasons. First, they capture the property of parallelism which as we will see, is of direct use in recognition as a grouping method. Secondly, while color and texture have been primary features being extracted directly from the intensity image, parallel-line groups serve as a secondary feature being extracted from the edges or line features. So treating them as a separate cue illustrates, as we will see in the next chapter, an implementation of the model of attentional selection possessing a feature hierarchy.



(a)



(b)



(c)

Figure 5.2: *Illustration of implicit and explicit closely-spaced parallelism on objects. (a) An image of a scene containing objects showing explicit and implicit parallelism. (b) Line segment image of (a). Note the parallelism explicit in the contour of the triangular block. (c) An image showing only the nearly horizontal lines in the image of (b). Note that the parallelism implicit in the letter texture on the bottle in (b) becomes explicit in this image.*

The groups of parallel lines we want to capture include cases of both explicit and implicit parallelism. Figure 5.2a shows a scene containing objects showing instances of both types of parallelism. The contour of the triangular block has two explicitly parallel lines as can be seen from Figure 5.2b, while the letter texture on the bottle has implicit parallelism as can be seen from the group of parallel lines in Figure 5.2c where only the nearly horizontal lines of Figure 5.2b are highlighted. As we will see later, the projection of such patterns in images continue to show closely-spaced parallelism among the projected lines over a wide range of viewpoints. This makes it possible to capture closely-spaced parallelism on objects by examining such a relation between the edges (or lines) in their projections. Since it is rare that adjacent object regions in an image possess similarly-spaced parallel lines of similar orientation, the detection of closely-spaced parallel lines in images is also likely to point to single objects. Further, since not all edges in the image are likely to show closely-spaced parallelism, this could lead automatically to fewer groups. And for objects showing characteristic textural information that contains such closely-spaced parallel-lines, the groups can be useful clues that point to the identity of such objects. Also, when the spacing allowed between parallel lines is small, such groups capture compact areas in both the image and the object and contain fewer features in a group. Finally, as we will see later, such groups can be easily found in images using a simple algorithm. Thus groups of closely-spaced parallel lines in images capture not only meaningful structures on objects in scenes but also possess the desirable properties required of a grouping scheme for recognition.

Grouping based on such parallelism, however, has the disadvantage that unlike in conventional grouping, a single pair of matching groups is not sufficient for recognition. This is because recognition methods such as the linear combination of views-based alignment method require at least 4 non-coplanar points for alignment. Since a group of parallel lines in space span a plane, the features such as points or lines derived from them are coplanar, needing at least two matching pairs of groups to be found. However, since more than four corresponding features are needed in practice, other grouping schemes have also found the need for finding more than a pair of matching groups [76].

5.2.1 Closely-spaced parallelism constraint

So far we have only loosely specified the property of closely-spaced parallelism and have given intuitive arguments about the advantages of grouping based on this relation. We now make the definition more precise to allow the generation of groups from line segments in an image based on this relation. Ideally, the structure in space we want to capture using the closely-spaced parallelism constraint is a set of (3D) parallel line segments on an object with a given inter-line spacing. To see how such a structure appears in an image (i.e. in a projection), we exploit some well-known results in descriptive geometry [77]. These results indicate that under orthographic projection and scale (often used to approximate perspective projection), a parallel-line group in 3D always projects to a group of parallel lines in the image under any view. In practice, because of the noise in the imaging process, and depending on the method used to obtain line segments from edges, such a group appears as a set of closely-spaced lines with slight skew between the lines but with the overall orientation of the group remaining more or less uniform. When perspective effects are dominant, however, parallel lines in 3D appear as a set of converging lines. For most imaging distances, this convergence is slight, so that such lines have only a small amount of inter-line (as well as overall) skew. Thus 3D line segments on objects showing strict closely-spaced parallelism between them actually appear as groups of closely-spaced lines in image that are almost parallel (i.e. with slight inter-line as well as overall skew). However, we will refer to such groups in both the image and object as closely-spaced parallel-line groups (or in short as line groups) with the implication of strict parallelism between lines in 3D and approximate parallelism between their projections.

To precisely define such groups in an image, we begin with some terminology relating to 2d non-intersecting line segments.

Terminology

1. Overlapping lines: Two line segments are said to overlap if the projection of at least one end point of one of the lines lies inside the other line segment. Figure 5.3a shows examples of overlapping and non-overlapping line segments.

2. Across-the-line-distance: The across-the-line distance between two lines L_1, L_2 whose end points are designated by $p_{l11}, p_{l12}, p_{l21}, p_{l22}$ respectively, is defined as follows. Let S denote the set of pairs $\{(p_{l11}, p_{l21}), (p_{l11}, p_{l22}), (p_{l12}, p_{l21}), (p_{l12}, p_{l22})\}$. Let d_{min} = min $\{d(p_i, p_j) | \forall (p_i, p_j) \in S\}$, where $d(p_i, p_j)$ is the euclidean distance between the points of the pair (p_i, p_j). Let (p_r, p_s) be the pair in S that has this minimum distance d_{min}. Let $L(p_r)$ and $L(p_s)$ be the lengths of the projection of points p_r and p_s onto lines L_2 and L_1 respectively. Then the across-the-line distance d_{across} between lines L_1 and L_2 is defined as

$$d_{across} = \begin{cases} min\{L(p_r), L(p_s)\} & \text{if } L_1 \text{ and } L_2 \text{ are overlapping} \\ d_{min} & \text{otherwise} \end{cases} \tag{5.1}$$

Figure 5.3b shows examples of some non-intersecting line segments and the across-the-line distance between them.

3. Along-the-line-distance: The along-the-line distance d_{along} between two lines L_1 and L_2 is defined as:

$$d_{along} = \begin{cases} \min\{\sqrt{(d_{min}^2 - L(p_r)^2)}, \sqrt{(d_{min}^2 - L(p_s)^2)}\} & \text{if } L_1 \text{ and } L_2 \text{ are non-overlapping} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where the terms d_{min}, $L(p_r)$, and $L(p_s)$ are as given in Definition 2. Figure 5.3c shows some non-intersecting line segments and the along-the-line distance between them.

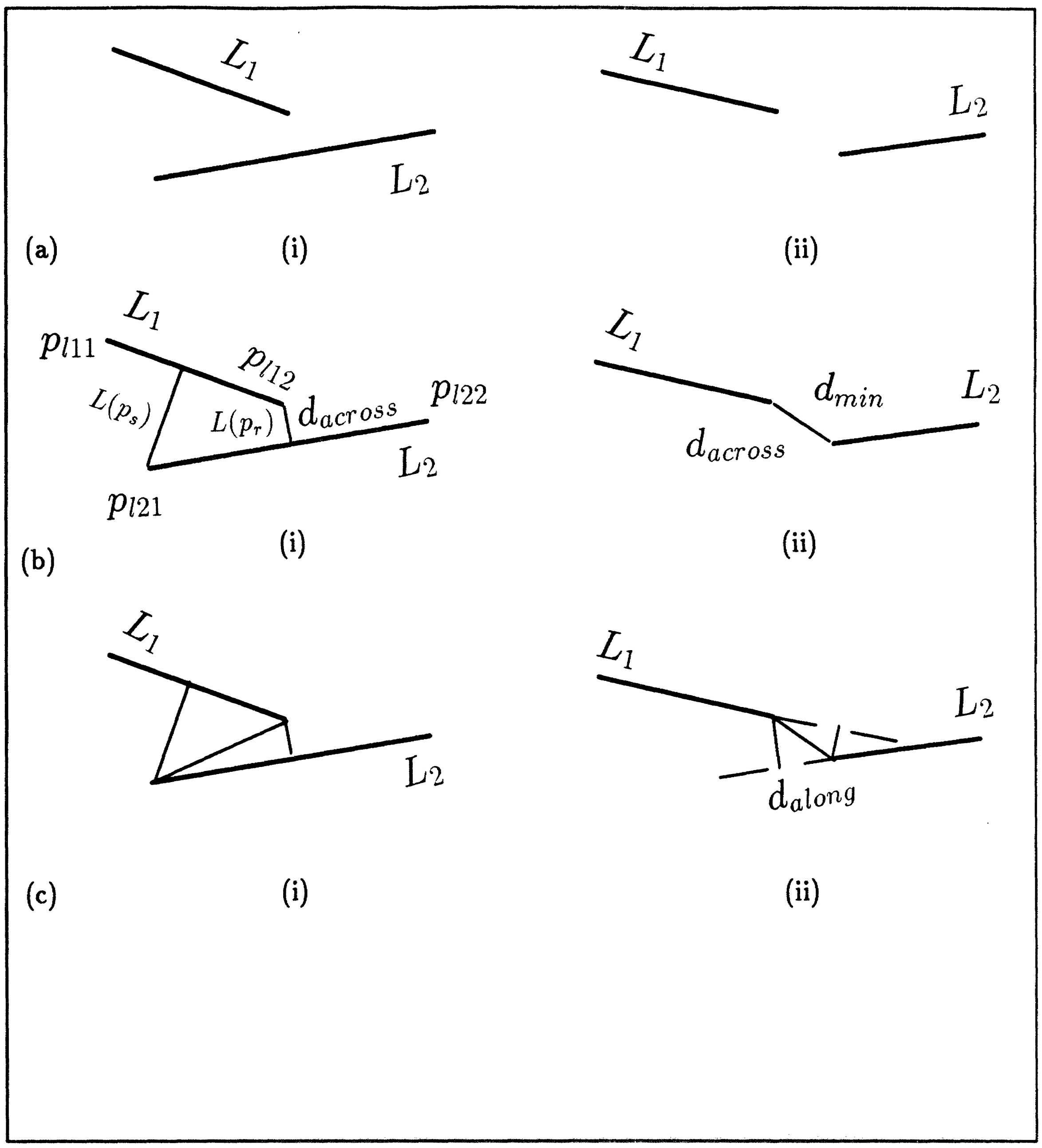


Figure 5.3: *Illustration of some of the terminology relating to 2D non-intersecting line segments. (a) The difference between overlapping (i) and non-overlapping (ii) line segments according to Definition-1 in the text. (b) Across-the-line distance shown for both overlapping (i) and non-overlapping (ii) line segments. (c) Along-the-line distance shown for both overlapping (i) and non-overlapping (ii) line segments.*

A closely-spaced parallel-line group

A closely-spaced parallel-line group in the image, specified by the tuple
$< t_{across}, t_{along}, t_{local-orient}, t_{global-orient} >$,

is the largest group of non-intersecting line segments such that for each line in the group, there exists another line in the group obeying all of the following constraints:

1. The across-the-line distance d_{across} between the lines is no more than the threshold t_{across}.
2. The along-the-line distance d_{along} between the lines is no more than the threshold t_{along}.
3. The orientation difference between the lines is no more than a threshold $t_{local-orient}$.

Moreover the entire group must satisfy the condition that the maximum orientation change between any two lines in the group is no more than a threshold $t_{global-orient}$.

The above definition of closely-spaced parallelism allows for almost parallel lines to be grouped, which as we said above, is a more useful structure to capture in the image. Further, by allowing non-overlapping lines to be grouped, it can not only capture groups of non-overlapping parallel lines in space, but also allows some occlusions that cover portions of line segments, to be handled. The constraint on global orientation change, $t_{global-orient}$, is imposed to keep the entire group almost parallel since otherwise successive deviations in orientation between lines could lead to a group of fairly skewed lines. This also makes the above grouping constraint different from the one used earlier for assembling groups of parallel lines in a data-driven fashion [126]. Finally, the choice of the thresholds $t_{across}, t_{along}, t_{local-orient}, t_{global-orient}$ dictates the kind of groups that will be generated. We will discuss their choice when using the groups to perform data and model-driven selection.

5.2.2 Algorithm to generate the line groups

We now present an algorithm to generate closely-spaced parallel line groups in an image for a given choice of thresholds $t_{across}, t_{along}, t_{local-orient}$, and $t_{global-orient}$. It works by first extracting line segments from edges in an edge image using one of the standard algorithms for line-segment approximation [112]. The resulting line segments are used to generate the groups as follows:

1. Each line segment is initially kept in a separate group.

2. For each line segment L, the following operations are done:

   (a) A rectangular neighborhood about L that is $2t_{across}$ in breadth and $2(t_{along}+l)$ in length, where l is the length of the line, is scanned, and all lines that either pass through this neighborhood or have an end point in it are retained.

   (b) Among the lines obtained in step-2a, those that satisfy the local orientation change constraint $t_{local-orient}$ with L are retained.

   (c) A new group is formed by successively merging the enclosing groups of lines obtained after step 2b with the enclosing group of L taking care to see that no enclosing group being added contains a line violating the $t_{global-orient}$ constraint with the currently created group.

Analysis

The grouping algorithm performs steps 1-2 using the union-find data structure to record and update information about line groups [29]. In this data structure, information is organized as a forest of trees. The essential information within a tree is summarized in its root. The basic operations that can be performed on this data structure are **make-set (a)** that creates a single node tree with element **a**, **find (a)** that finds the root of the tree containing **a**, and **union(a,b)** that merges the trees containing elements **a** and **b**. Using a technique called merging by rank with path compression [29], it is known that m operations of make-set take time $O(m)$, of find take time $O(m)$ while m union operations take time $O(mA(m,n))$ where n is the number of elements in the data structure, and $A(m,n)$ is the Ackerman's function. For most values of m and n, the function $A(m,n)$ is almost constant so that a single one of these operations can be done in constant amortized time.

Using the union-find data structure, Step-1 requires n make-set operations for n line segments. For each line L, Step 2a requires all lines to be scanned requiring $O(n)$ time. Similarly Step 2b requires $O(n)$ time, in the worst case, to examine all the retained lines. If the least orientation in a group is stored as part of the information in the roots of trees, then the constraint checking in Step 2c can be done by a simple find operation per line. Finally, the merging in Step 2c can be done by a union operation. Thus the entire step 2 can be done in time $O(n)$ per line with the result that the grouping algorithm itself runs in $O(n^2)$ worst-case time.

Results

We now illustrate the grouping algorithm with a few examples. Figure 5.4a shows the line segments obtained by doing a line segment approximation to the edges in the image of Figure 5.2a. The closely-spaced parallel line groups obtained using the grouping algorithm with a constraint specification of $< 10, 5, 6, 10 >$ are shown in Figures 5.4f-i. These groups are shown along four major orientations (vertical, horizontal, obtuse, and acute) for clarity. The individual groups are highlighted by drawing the convex hull of the end points of line segments. The line segments that are grouped can be seen in the corresponding Figures 5.4b-e. Similarly, Figure 5.5 shows another example of grouping performed by the algorithm. By using the algorithm on a number of edge images, the number of groups, their average size (number of constituent lines) and the average area spanned by the groups were recorded. The results are shown in Table 5.1. From the table it can be seen that the number of groups is linear in the number of line segments, and the size of the line groups tends to be small.

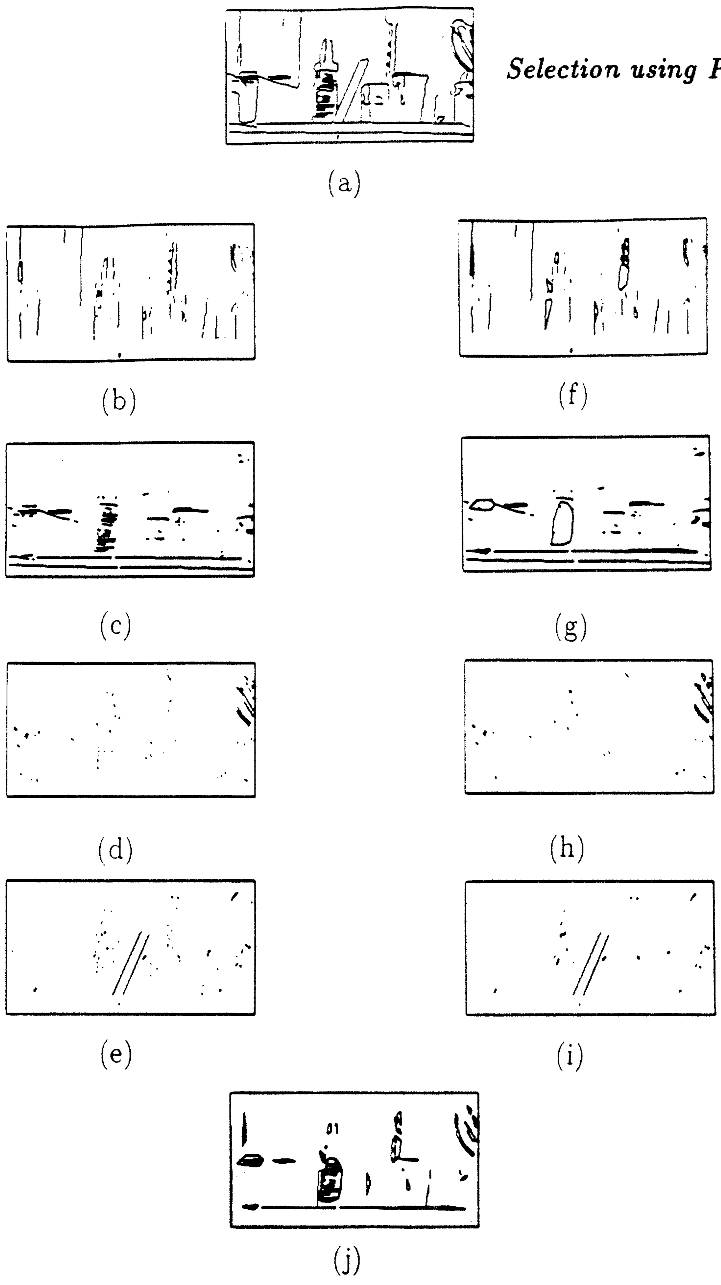


Figure 5.4: *Illustration of grouping based on closely-spaced parallelism and salient group detection. (a) Line segments to be grouped based on closely-spaced parallelism. (b)-(e) Line segments shown along four major orientations, namely, vertical, horizontal, obtuse, and acute orientations. (f)- (i) The line groups formed using the algorithm shown also along the respective major orientations for clarity. The thresholds used were* $t_{across} = 10, t_{along} = 5, t_{local-orient} = 6^{\circ}, t_{global-orient} = 10^{\circ}$. *(j) The 40 most salient groups among the line groups of (f) - (i) found using the saliency measure. Note that none of the salient groups span more than one object.*

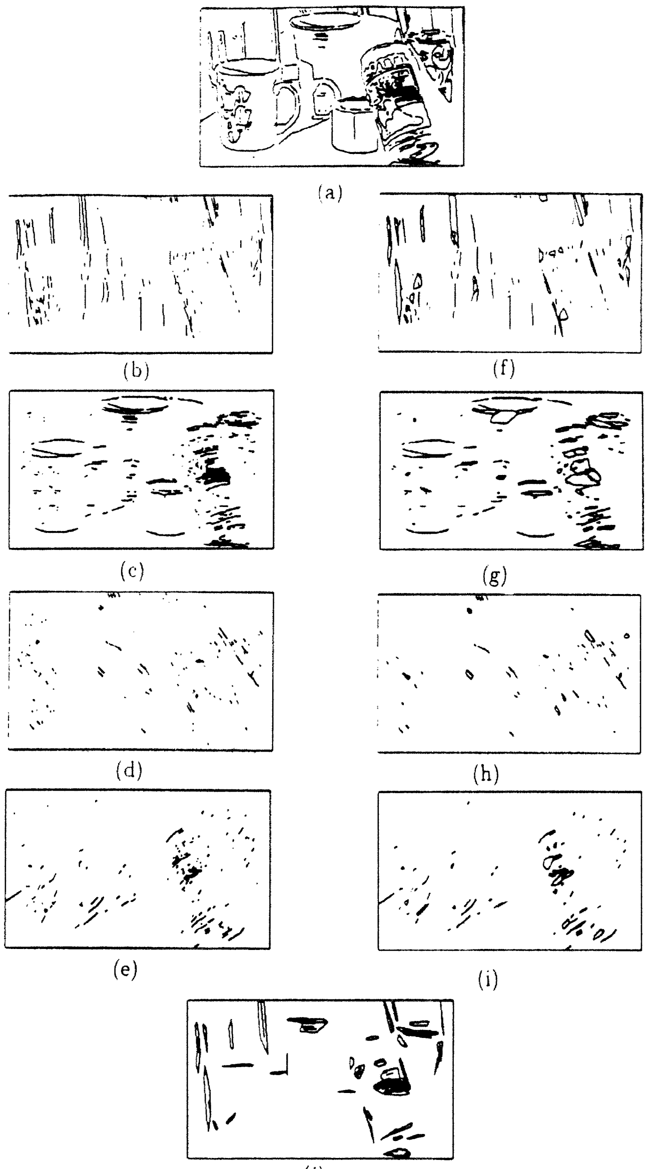


Figure 5.5: ***Illustration of grouping based on closely-spaced parallelism and salient group detection — Another example. (a) Line segments to be grouped based on closely-spaced parallelism. (b)-(e) Line segments shown along four major orientations, namely, vertical, horizontal, obtuse, and acute orientations. (f)- (i) The line groups formed using the algorithm shown also along the respective major orientations for clarity. The thresholds used were*** $t_{across} = 6, t_{along} = 5, t_{local-orient} = 6^o, t_{global-orient} = 10^o$. ***(j) The 40 most salient groups among the line groups of (f) - (i) found using the saliency measure. Note that only two of the salient groups span more than one object.***

S.No.	Image Size	Num. Lines	Group Constraints	Num. Groups	Avg. Group Size	Max. Group Size	Avg. Group Area
1.	320 x 576	395	$< 10, 5, 6, 10 >$	34	4	21	0.002
2.	256 x 416	756	$< 6, 5, 6, 10 >$	91	3.0	11	0.001
3.	240 x 240	233	$< 5, 0, 6, 10 >$	22	3.1	6	0.0009
4.	232 x 576	884	$< 5, 5, 6, 10 >$	119	3.9	7	0.0008
5.	200 x 492	1232	$< 5, 10, 9, 12 >$	243	3.77	13	0.0028
6.	224 x 416	316	$< 5, 5, 6, 10 >$	75	3	17	0.003

Table 5.1: *Characteristics of closely-spaced parallel line groups generated by the grouping algorithm. The average group area is normalized with respect to the image size. Only groups containing more than one line are considered here.*

Discussion

The number of groups generated by the grouping algorithm is in fact linear in the number of lines, since each line belongs to at most one group at the end of Step 2. If the constraints did not involve $t_{global-orient}$, it is clear that only a linear number of groups would have been possible (recall that we are considering only the largest such groups). With the fourth constraint $t_{global-orient}$ added, the starting line as well as the order in which lines are examined determines the lines that ultimately belong to a group as well as its size. In such cases, more groups than are generated by the algorithm are possible. A case where this happens is shown in Figure 5.6. Figure 5.6a shows an arrangement of closely-spaced parallel lines in the image and Figure 5.6b shows the groups that will be generated by the algorithm. Finally, Figure 5.6c shows some other groups that are possible from the arrangement in Figure 5.6a but are not generated by the algorithm. The groups generated by the algorithm correspond to a left to right, bottom to top scan of the line segments in the image. Such a scan often produces groups that resemble the groups we perceive using a frame of reference with the origin at the left hand bottom corner of the image.

In general, if a large number of lines fall within the specified neighborhood of a

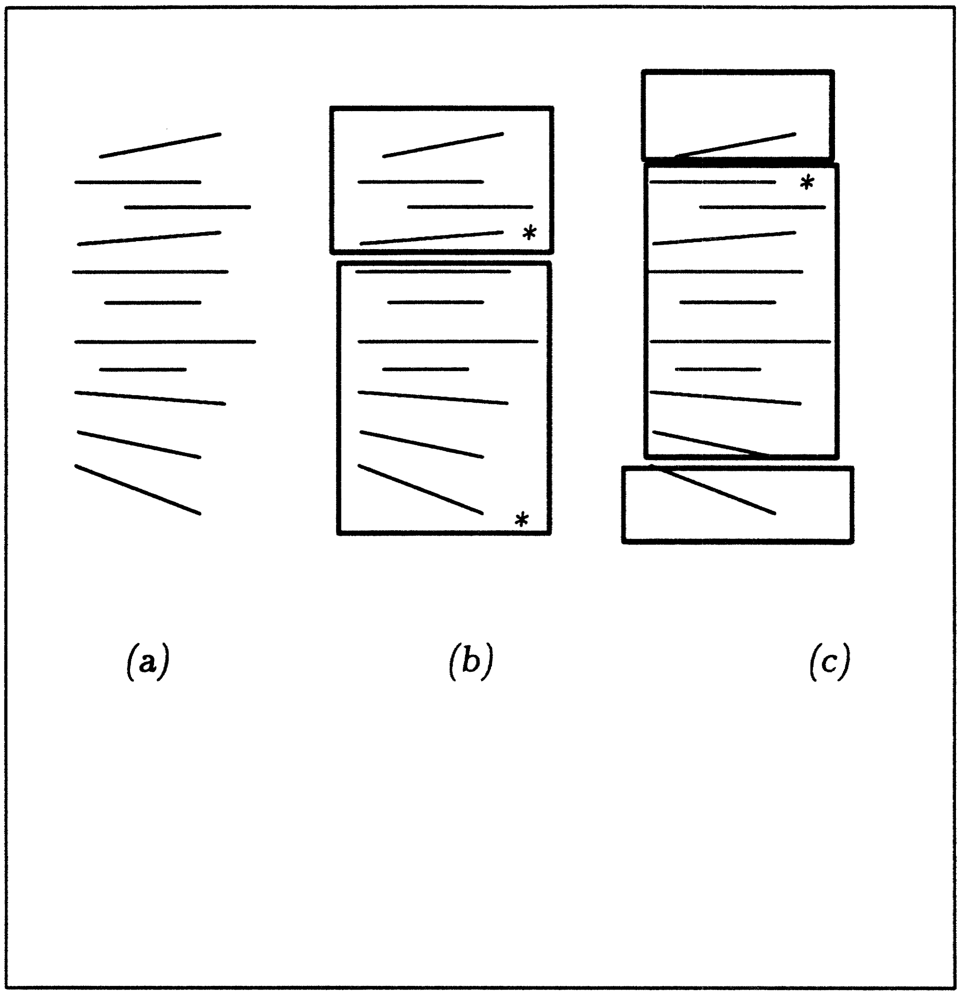


Figure 5.6: *Example to illustrate some of the line groups that are not generated by the grouping algorithm. (a) An arrangement of closely-spaced parallel lines. (b) The groups generated by the algorithm shown within the two rectangular boxes. The asterisk mark indicates the starting line segment used to assemble the line groups. (c) Another set of groups possible from the arrangement of (a) that also obey all the four grouping constraints. Note the overlap between these groups and those generated by the grouping algorithm.*

line (in Step 2a), the possible combinations of lines obeying all 4 constraints could become very large. The grouping algorithm described above generates only a subset of such groups, and in some sense, therefore, does a filtering operation. We mentioned earlier in Section 5.1.1 that grouping approaches that filtered groups to keep them to a small number could cause a recognition system that subsequently uses these groups to make unnecessary false negatives. We now show that this does not happen with the above grouping algorithm. For this, we notice that the closely-spaced line groups satisfying all 4 constraints of $\{t_{across}, t_{along}, t_{local-orient}, t_{global-orient}\}$ are subsets of groups satisfying the first 3 constraints $\{t_{across}, t_{along}, t_{local-orient}\}$ (called main groups here). The grouping algorithm generates only some of the possible subsets, but such groups (called aggressive groups, henceforth) generate a cover of the main group. That is, every line of a main group belongs to some aggressive group. Suppose that the groups are fed to a recognition system. Assuming an alignment style of recognition (such as the linear combination of views method [157]), we know that at least 4 matching features must be found to solve for the pose of the object. Since parallel line groups in the image that come from parallel lines in space represent coplanar points, we may need two such groups to derive these features. Let us assume that each group provides two features and that the features are derivable from a single line in each of the groups (the end points of a line are the features, say). If there existed a pair of closely-spaced parallel-line groups in the image obeying all 4 constraints that were the correct pair of groups (i.e., they contained sufficient number of features to recognize the object) but were not generated by the grouping algorithm, then a recognition system using the groups given by the algorithm could make false negatives. But since the aggressive groups form a cover, each correct group has partial overlap with at least one aggressive group suggesting that those pairs of aggressive groups would also be the correct groups containing sufficient features to recognize the object thus preventing a false negative identification.

Thus the above grouping algorithm keeps the number of groups small by filtering possible groups, but at the same time, prevents unnecessary false negatives during recognition due to insufficient number of groups being produced.

5.3 Data-driven Selection using Line Groups

We now discuss the use of closely-spaced parallel-line groups to perform data-driven selection. The goal of data-driven selection, as we outlined in earlier chapters, is to isolate regions in an image that are likely to come from a single object based on information available in the image and some a priori knowledge about scenes. For a given choice of thresholds, not all the groups generated by the above algorithm represent useful structures in the scene as can be seen from the examples in Figures 5.4 and 5.5. Some of the groups may span more than one object, while others come from spurious line segments, or scene clutter rather than objects of interest in the scene. For the purposes of recognition, it would be useful to order and consider only some of the more reliable ones from these groups. In keeping with our general paradigm of data-driven selection, we order the groups using a saliency measure and select a few of the salient groups. In this section, therefore, we describe a measure of saliency for the line groups and then discuss the utility of salient group-based selection in recognition.

5.3.1 Saliency of parallel-line groups

As in the development of color and texture region saliency, the focus in designing a measure of saliency of parallel-line groups will be on capturing the sensory component of distinctiveness. Thus those properties of lines that are commonly perceived and fairly general will be considered. The strategy for assembling the saliency measure is, as before, to record the factors affecting saliency and to combine them appropriately in a way that reflects their importance. Unlike in the case of color and texture saliency, however, the saliency measure for line groups is designed to emphasize the inherent reliability of groups more than the match with our perceptual judgment of their importance.

Factors affecting saliency of line groups

Among the properties of line-groups that are often observed are the length of the constituent lines, the region spanned by them, the number of constituent lines, and their overall orientation span. These properties of line groups were chosen as the factors affecting saliency using the following rationale:

1. Length of constituent lines (L): The length of the constituent lines is chosen as a factor because groups with both very long or very short lines are undesirable from the point of recognition. Very long lines in an image are more likely to span multiple objects, while very short lines often tend to be due to spurious line segments resulting from scene clutter or from a very fine line-segment approximation. Since a group has lines of varying length, the average length of lines in the group is taken as a measure of the length of the group.

2. Region span of a group (R): The region spanned by a line group can give indications of its reliability. A large region span often indicates a group spanning more than one object. We measure the region spanned by a closely-spaced parallel-line group by the area of the convex hull of the end points of the constituent lines.

3. Number of constituent lines (N): Groups with a very large number of lines may not be entirely desirable from the point of recognition as they could contain a large number of features. Also, such groups could potentially span multiple objects. A sparse group (with one or two lines), on the other hand, may not indicate any meaningful structure. Either way, the number of constituent lines in a group can affect its saliency.

4. Orientation span (Θ): A low orientation span indicates a greater amount of par-

allelism in a group. Since the aim in grouping lines here is to capture instances of closely-spaced parallelism on objects in a scene, groups exhibiting a greater degree of parallelism are more likely to indicate a meaningful structure. This also follows from the viewpoint invariance argument of Lowe that we mentioned in Section 5.1.1, namely, that a group of parallel lines is unlikely to have arisen from an accident of viewpoint, thereby making them more likely to come from either a single object or a single structure. The orientation span is measured by recording the maximum difference in orientation between lines in a group.

Weighting functions for factors affecting saliency

To develop a measure of saliency for the line groups, each of the factors must be weighted to appropriately reflect their individual contributions in deciding the saliency of a group. The form of the weighting function, in most cases, was derived using three criteria: It should (a) reflect the likelihood of a group coming from a single object, (b) it should be a smooth function so that discontinuities do not indicate an abrupt change in judgment, (c) it should be verifiable from statistical experiments. The weighting functions chosen for the factors are as follows:

1. Length of constituent lines (L): Using the rationale given earlier, the weighting function is chosen to de-emphasize both very long and very short lines, while giving about equal importance to intermediate length lines. The decision of very short lines is made on an absolute basis, i.e., lines shorter than 5 pixels are considered very short, while long lines are decided relative to the size of the image (by choosing the diagonal length in the image as the normalizing factor). The weighting function is as follows:

$$f_1(L) = \begin{cases} -\frac{ln(1-L_n)}{c_1} & 0 \le L \le l_1 \\ 1 - e^{-L_n c_2} & l_1 < L \text{ and } L_n \le l_2 \\ 1.0e^{-c_3(L_n - l_2)} & l_2 < L_n \le 1.0 \end{cases} \tag{5.3}$$

where $L_n = \frac{L}{L_{max}}$, L_{max} = diagonal length of the image, and the various thresholds are $l_1 = 5$ (pixels), $l_2 = 0.4$, $c_1 = -\frac{ln(1-\frac{l_1}{L_{max}})}{l_2}$, $c_2 = \frac{6ln10}{l_2}$, $c_3 = \frac{3ln10}{1-l_2}$.

The form of the weighting function was derived by performing the following experiments. Groups were formed from several edge images using the grouping algorithm and the average line lengths of groups were recorded. The scenes of the images varied in complexity having different amounts of scene clutter, contained several objects, and showed illumination artifacts such as specularities, interreflections, etc. Figures 5.2a and 5.5a shows examples of some typical images tried. A histogram of the number of groups with a given normalized length L_n (using 200 bins) was plotted. From this, the number groups that came from a single object and had normalized line length falling in a given bin were noted. The ratio of the number of groups of a given L_n coming from a single object to the total number of groups of that line length was taken to represent the weighting function $f_1(L)$. This ratio was plotted against L_n and smooth functions were fit to the resulting curve. These functions were described by the parameters c_1, c_2, c_3. Finally, the thresholds l_1, l_2 were found from the breakpoints in this ratio curve.

2. Region span of a group (R): The weighting function for the region span was chosen to emphasize small and compact groups. The form of the weighting function was derived by performing studies similar to the one described above. Here, the ratio of the number of groups with a given normalized region span that came from a single object to the total number of groups with the given span was taken to represent the weighting function. The weighting function derived from this ratio was:

$$f_2(R) = \begin{cases} -\frac{ln(1-R_n)}{c_4} & 0 \leq R_n \leq r_1 \\ 1 - e^{-c_5 n} & r_1 < R_n \leq r_2 \\ s_2 - c_6 ln(1 - R_n + r_2) & r_2 < R_n \leq r_3 \\ s_3 e^{-c_7(R_n - r_3)} & r_3 < R_n \leq r_4 \\ 0 & r_4 < R_n \leq 1.0 \end{cases} \qquad (5.4)$$

where $R_n = \frac{R}{R_{max}}$, and R_{max} = image size, $r_1 = 0.1$, $r_2 = 0.4$, $r_3 = 0.5$, $r_4 = 0.75$, $s_1 = 0.8$, $s_2 = 1.0$, $s_3 = 0.7$, $s_4 = 10^{-3}$ and $c_4 = -\frac{ln(1-r_1)}{s_1}, c_5 = -\frac{ln(1-s_1)}{r_1}, c_6 = -\frac{(s_2-s_3)}{ln(1+r_2-r_3)}, c_7 = -\frac{ln\frac{s_4}{s_3}}{(r_4-r_3)}$. Here again the thresholds are chosen in a manner similar to the one described for the weighting function for the length of constituent lines.

3. Number of constituent lines (N:) Since the average size of a group is small in the groups generated by the grouping algorithm, the weighting function is chosen to emphasize densely packed groups, i.e. groups with a larger number of constituent lines, as they often indicate some textural information. If such groups span a large area then they will be de-emphasized in the weighting function $f_2(R)$ for the region-span. The weighting function chosen was:

$$f_3(N) = \frac{N}{N_{max}} \tag{5.5}$$

where N = number of constituent lines in a group, and N_{max} = maximum number of lines in any line group in the given edge image.

4. Orientation span (Θ): Using the rationale given earlier, the weighting function here is designed to emphasize groups showing a greater degree of parallelism, i.e., a smaller orientation span. To avoid unfair bias towards groups of single lines (which will have an orientation span of zero), we assign a small penalty toward single line groups. The resulting choice of function is:

$$f_4(\Theta) = \begin{cases} 0.1 & \theta = 0 \text{ and } N = 1 \\ (1 + \frac{\theta(1.0-ct_1)}{t_{global-orient}}) & \text{otherwise} \end{cases} \tag{5.6}$$

where Θ is the orientation span, and ct_1 =0.4.

Saliency measure for a closely-spaced parallel-line group

The saliency measure for a closely-spaced group of line segments is obtained by combining the weighting functions reflecting the contributions from the various factors. Since the factors record independent properties of line groups, we chose to combine them linearly to give the following saliency measure:

$$\text{Saliency of a line group} = f_1(L) + f_2(R) + f_3(N) + f_4(\Theta) \tag{5.7}$$

Results

We now illustrate the use of the saliency measure to judge the reliability of closely-spaced parallel-line groups. Figures 5.4f-i show the line groups found by the grouping algorithm in the image of Figure 5.4a. Among these, the 40 most salient groups found using the above saliency measure are shown in Figure 5.4j. As can be seen from the figure, all of the 40 salient groups come from single objects. Figure 5.5 shows another example in which the line groups generated are shown in Figures 5.5f-i and the top 40 salient groups among them are shown in Figure 5.5j. Here only two of the salient groups did not come from single objects. Table 5.2 shows the results of performing saliency experiments on a number of images whose average complexity is indicated by the number of constituent line segments listed in the table. Here, the last column lists the percentage of unreliable groups in the top 100 salient groups found using the saliency measure. From these studies we conclude that the saliency measure captures reliable groups and can, therefore, be useful in a data-driven selection mechanism for recognition.

In the discussion so far, we have not analyzed the extent to which the groups selected by the saliency measure match our perceptual judgment of the importance of such groups. We chose not to emphasize this aspect for several reasons. First, in an edge image, other relations in addition to closely-spaced parallelism, may exist between lines. For example, long smooth curves may be more salient than parallel-line groups in an edge image. For comparing the performance of the saliency measure,

S.No.	Image Size	Num. Lines	Group Constraints	Num. Groups	Avg. Salient Group Size	Avg. Salient Group Area	% Unreliable Groups
1.	320 x 576	395	$< 10,5,6,10 >$	219	6	0.009	1
2.	256 x 416	756	$< 6,5,6,10 >$	382	8	0.006	2
3.	224 x 416	316	$< 5,5,6,10 >$	207	6	0.008	7
4.	200 x 492	1232	$< 5,10,9,12 >$	552	4.25	0.003	3
4.	232 x 576	884	$< 5,5,6,10 >$	454	5.3	0.004	6

Table 5.2: *Characteristics of salient closely-spaced line groups ranked by the saliency measure described in text. In each case, the top 100 salient groups are considered. The number of groups listed here include single line groups.*

the subjects should be made to look at only closely-spaced line groups and ignore other cues for grouping, a task difficult to achieve in practice. Even if scenes showing only instances of closely-spaced parallelism were examined, there is the additional problem due to the grouping algorithm generating only a subset of the possible groups. Thus not all the groups perceived by a subject may be generated and this affects the groups selected by the saliency measure. Finally, perceptual judgments may be based on a collection of groups of different orientation (this could indicate groups of curves, for example), and this is not considered by the saliency measure.

5.3.2 Use of salient line groups-based selection in recognition

Data driven selection based on salient line groups is primarily useful when the object of interest has at least one parallel-line group that appears among the selected salient line groups. In such cases, the search for data features that match model features can be restricted to salient groups thus avoiding needless search in other areas of the image. In order for a model group to be found among the salient line groups, however, it should first be generated by the grouping algorithm. That is, the choice of the four constraints characterizing an image group should be such that a model group, if it exists in the image, can be captured by these constraints. Since no specific knowledge of model objects can be used in data-driven selection, the thresholds can be set based on some rough a priori knowledge about expected objects in scenes, the distances at which they are imaged, and some general knowledge about the parameters of the 3D structures that are meant to be captured in line groups in the images of such scenes. Among the thresholds, the local and global orientation thresholds, $t_{local-orient}$ and $t_{global-orient}$, are essentially independent of the objects in the library, and can be chosen based on an analysis of the imaging noise and the noise in line-segment approximation. Since a group captures almost parallel lines, there is not much leeway in their choice, in that they can be only low values. We chose to model this noise by allowing 5-7 degree skew in between lines ($t_{local-orient}$) and an overall skew ($t_{global-orient}$) of 10 degrees. The values of t_{across} and t_{along} however, have a major effect on the lines that are ultimately grouped, and cannot in general, be chosen independent of objects in the library. Larger values of these thresholds allow somewhat widely-spaced parallelism to be captured, such as the parallelism inherent in the contour of the triangular block in Figure 5.2b. However, this would also decrease the reliability of the groups, as it allows lines belonging to separate objects to be grouped. Since our aim (at least in data-driven selection) is to capture letter and wooden texture occurring on objects in indoor scenes, we found that for the distances at which the objects are typically imaged, an interline separation (t_{across}, t_{along}) of 5 to 10 pixels is sufficient for capturing most such parallel-line groups in images. Better methods for choosing these thresholds, could be devised, however.

Search reduction using salient line groups

We now estimate the search reduction that can be achieved by using salient-line groups-based data-driven selection. Following the analysis of the number of matches using grouping given in Section 5.1, if only S salient groups are retained for M_i image groups, then the number of matches to be tried using the 4-point alignment scheme of recognition is $O(\sum_{i=1}^{M_i}\sum_{j=1}^{S} m_i^4 n_j^4 4!)$ where m_i and n_j are the number of features in the model and the selected salient image groups, respectively. To estimate the number of matches using such salient line groups, we chose a few model

S.No	M	N	M_g	Avg. m_i	Group Constraints	Avg. n_j	Num. Matches	
							No Selection	Data-driven Selection
1.	466	1768	9	6	$< 5, 5, 6, 10 >$	12	1.88x10^{22}	5.1x10^{11}
2.	140	1768	10	4	$< 10, 5, 6, 10 >$	12	1.49x10^{20}	1.0x10^{11}
3.	140	2464	10	4	$< 10, 5, 6, 10 >$	10	5.6x10^{20}	4.66x10^{10}
4.	358	1453	8	16	$< 2, 2, 6, 10 >$	18	2.98x10^{21}	1.38x10^{14}
5.	130	790	5	23	$< 10, 5, 6, 10 >$	12	4.38x10^{18}	4.46x10^{13}

Table 5.3: *Estimated search reduction using salient line groups-based selection. The terms M, N, M_g, m_i, n_j are as explained in text. Here $N_g = 40$, i.e., the top 40 salient groups are retained for selection.*

objects exhibiting closely-spaced parallelism between lines on the object, generated line groups using the grouping algorithm and retained a few of the groups. By placing these objects in various scenes, and using the grouping algorithm and the saliency measure, we retained a few (about 40) salient line groups in the images of these scenes. The number of features (i.e. the end points of line segments) in both model and data groups were recorded. The number of matches with salient groups-based selection was found using the above formula. For purposes of comparison, the number of matches without any grouping was also computed using the formula O(M^4N^4) where M and N are the total number of features found in all the model and data line groups. The results are summarized in Table 5.3. As can be seen from the table, the search is always considerably lower when salient groups-based selection is done prior to recognition. The number of matches with this type of selection scheme can still be large, however, as all pairings of model and salient image groups are tried. Also, some of the salient groups are large in size thus increasing the number of matches that need to be tried within a pair of groups.

5.4 Model-driven Selection using Line Groups

So far we have considered the use of closely-spaced parallel line groups for data-driven selection which required the object of interest to have a salient line group. Further, it was assumed that such a group could be detected by the use of some default threshold values for the constraints characterizing closely-spaced line groups. This will not be of much help when the object of interest has closely-spaced parallel line groups but they are either not salient or cannot be captured using the default thresholds. In such cases, the description of the line groups present on the model object can be used to perform selection. We now describe one such line groups-based model-driven selection mechanism. The approach adopted here is to selectively generate line groups in the image based on the description of closely-spaced parallel line groups on the model object. Thus the group generation process is constrained here so that only the likely matching groups are generated. Once the candidate matching line groups in the image are obtained, recognition can proceed by examining pairs of model and image groups as usual.

The criteria developed for a model-driven selection mechanism in the earlier chapters are also relevant to line groups-based model-driven selection. That is, it must be sufficiently selective to avoid considering obviously impossible matches, but at the same time, be sufficiently flexible to take into account the various problems in imaging that may cause a model line group in an image to appear different from its original description. As we mentioned in the earlier chapters, the object may appear different in a scene because it has undergone pose changes, or because it is occluded, or because illumination changes as well as artifacts in the scene such as specularities, interreflections have altered the appearance of the object. These changes (called observation conditions, henceforth) can also alter the appearance of the line groups on the model object. We first examine, therefore, the effects of these changes on the model line groups. This will then be used to design a description of model line groups as well as a strategy for generating matching line groups in the image.

5.4.1 Effect of observation conditions on a model line group

Consider a closely-spaced parallel (3D) line group on a model object. It can be specified by a tuple $< t_{3d-across}, t_{3d-along} >$ with the following interpretation: Between any two consecutive parallel lines in the model 3D group, the across-the-line distance $d_{3d-across}$ is no more than the threshold $t_{3d-across}$ and the along-the-line distance $d_{3d-along}$ is no more than the threshold $t_{3d-along}$. The distances $d_{3d-across}$ and $d_{3d-along}$ for parallel lines in 3D are defined in a way analogous to that of d_{across} and d_{along} given in Section 5.2.1 [3]. We now analyze the effect of observation conditions on the appearance of such 3D line groups on model objects when they placed in a scene.

Pose changes: If the allowed transformation of the model object is restricted to a 3D affine transformation, then under orthographic projection and scale (to approximate perspective projection) and assuming no imaging noise and no errors in line segment approximation, it is known that closely-spaced parallel 3D line groups project to a set of closely-spaced parallel lines in the image [152][4]. Further, the order of the lines in the group is preserved, although the resulting orientation of the parallel lines in the image can be arbitrary. The inter-line spacing d_{across} and d_{along} between the projected lines will, however, be different from the inter-line spacing $d_{3d-across}$ and $d_{3d-along}$ between the corresponding 3D lines. If no scale change has occurred during the transformation, then the spacing between the projected parallel lines in the image can only decrease. This is because d_{across} forms a side of a right triangle whose hypotenuse is the orthographic projection of the inter-line spacing $d_{3d-across}$ (as shown in Figure 5.7), and will always be less than or equal to $d_{3d-across}$ [5]. Using

[3]The distance $d_{3d-across}$ is simply the distance between two consecutive parallel lines defined as the length of projection of the end point of one 3D line on the other. The along-the-line distance $d_{3d-along}$ is as defined for 2d-lines except that the distance d_{min}, $L(p_r)$, and $L(p_s)$ are distances between points in 3D.

[4]If the transformation takes some of the object out of view, then this can be treated as the case when some projected parallel lines coincide.

[5]This is also true when perspective projection is approximated by orthographic projection and scale.

a similar argument, we can show that the along-the-line spacing in the image d_{along} is less than or equal to $d_{3d-along}$. If the transformation includes scale changes, then the inter-line spacing between parallel lines varies proportional to the scale. That is, for a scale change s (s > 1.0 or < 1.0) we have $d_{across} \leq s * d_{3d-across}$ and similarly $d_{along} \leq s * d_{3d-across}$. Thus the effect of pose changes is to vary the inter-line spacing between lines while still maintaining the property of parallelism.

Occlusions: The most common effect of occlusions is to corrupt the projected model line group. That is, depending on the geometry of the occlusion, some lines in the group may either partially or totally disappear. But unless a group is completely occluded, the visible lines of the group maintain the same relative ordering and the inter-line spacing is dictated by the pose changes undergone by the model. In rare cases, when the occluding object has similar closely-spaced parallel line groups, it may cause the two groups to be merged, and may even affect the inter-line spacing of the model line group. Thus the effect of occlusions on a model line group is mainly to change the number of constituent lines in the model groups, and in rare cases to even alter the inter-line spacing in the groups.

Illumination changes and other imaging artifacts: When the wavelength characteristics of the light source illuminating the scene is different from the one illuminating the model object, it may cause a change in the apparent color of the object's surface. But since we are looking at an edge image to generate the groups, the edges tend to remain more or less stable. When the light source location is changed, however, then depending on the position of the 3D parallel line groups relative to the light source, lines in the projected group may be either partially or totally hidden in shadow. Further, if the surface of the object is specular, then specularities occurring in the region of the line group may cause the group to be corrupted by the partial or total masking of the lines in the group. Finally, interreflections can cause spurious line segments to appear as part of the model line group depending on the pattern on the object that is being reflected from the model's surface. Thus the effect of illumination changes is similar to occlusions in that both the number of constituent lines and

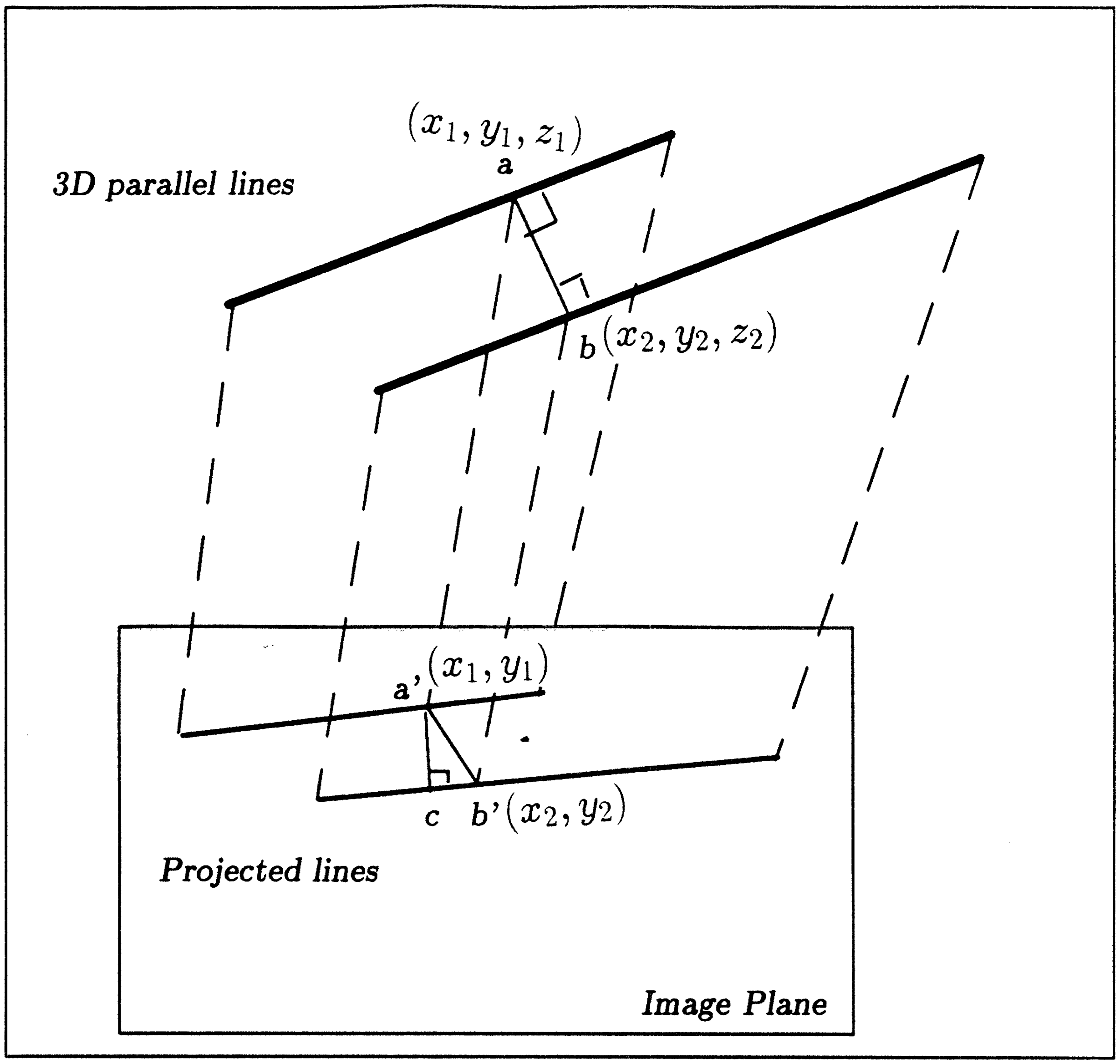


Figure 5.7: *Illustration to show that the distance between the projections of 3D parallel lines is less than or equal to the 3D distance between the lines. The projection of the 3d distance $d_{3d-across}$ (line ab) is given by the line $a'b'$. The length of $a'b'$ is $\leq$ length of ab since $ab = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$ and $a'b' = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. The distance between the projected lines given by $a'c$, is $\leq a'b'$ by the hypotenuse of a right triangle rule.*

the inter-line spacing in the model groups may be changed.

5.4.2 Model line group description

We now develop a description of the model line groups taking into account the effect of the observation conditions. From the above analysis, we know that a closely-spaced strictly parallel line group on the model specified by $< t_{3d-across}, t_{3d-along} >$ also occurs as a group of parallel lines in an image with an inter-line spacing that is decided by the scale change involved in the undergone transformation. By restricting the scale changes to lie between $< s_1, s_2 >$ (with $s_1 <= 1.0$ and $s_2 > 1.0$), the variation in the inter-line spacing d_{across} that can be handled for any model line group appearing in any image can be restricted to lie in the range $[s_1 * t_{3d-across}, s_2 * t_{3d-across}]$ and similarly, the spacing d_{along} to lie between $[s_1 * t_{3d-along}, s_2 * t_{3d-along}]$. So far, the effect of imaging noise and errors in line segment approximation were not taken into account. As we remarked earlier, their effect is to cause the lines in the model group to be slightly skewed in the image. The thresholds $t_{m-local-orient}$ and $t_{m-global-orient}$ can be used to specify the tolerable inter-line skew and the overall skew in the line group. The resulting description of a model line group as it appears in any image can be given by the tuple $< t_{3d-across}, t_{3d-along}, t_{m-local-orient}, t_{m-global-orient}, s_1, s_2 >$. Different line groups on the model will differ in the first two terms (as the thresholds $t_{local-orient}$ and $t_{global-orient}$ are independent of the line group), and the scale changes are specified with respect to the model object.

To generate candidate line groups in a given image using the above model description, however, the inter-line spacing and the tolerable scale changes must be expressed in terms of pixel spacing. For this, the distance at which the model is imaged for building the general model description can be used as the reference distance. That is, the pixel spacing corresponding to $t_{3d-across}$ and $t_{3d-along}$ can be taken to be the one existing between the projected lines in the image when the model is placed at the reference distance and oriented in such a way that the model line group is parallel to the image plane. By moving the object closer or farther than the reference distance by the scale factors s_1 and s_2 respectively, the corresponding change in the pixel spacing can be recorded. If $t_{m-2d-across}$ and $t_{m-2d-along}$ represent the pixel spacing corresponding to $t_{3d-across}$ and $t_{3d-along}$ respectively, and p_1 and p_2 represent the change in pixel spacing corresponding to the scale changes s_1 and s_2, then the description of the model line group that can be used to generate the line groups in the image becomes $< t_{m-2d-across}, t_{m-2d-along}, t_{m-local-orient}, t_{m-global-orient}, p_1, p_2 >$.

5.4.3 Selective generation of matching line groups

We now present an algorithm for selectively generating line groups in the image that match a given model line group description. Since the model description places a bound on the tolerable scale and pose changes, the basic strategy is to generate line groups with successively increasing inter-line (both d_{across} and d_{along}) spacing until the upper bound specified in the model description is reached. That is, given a model line group description $< t_{m-2d-across}, t_{m-2d-along}, t_{m-local-orient}, t_{m-global-orient}, p_1, p_2 >$, the maximum across-the-line spacing allowed between the lines varies from $t_{m-2d-across} - p_1$ to $t_{m-2d-across} + p_2$ (and similarly for the along-the-line spacing). The matching groups are generated by hypothesizing a value of spacing between the lines lying in the above range and generating all groups with inter-line separation specified by that value. In particular, successive integer pixel spacing from $t_{m-2d-across} - p_1$ to $t_{m-2d-across} + p_2$ are used to generate the groups. Such groups, called augmented closely-spaced parallel line groups, can be specified by the tuple $< t_{across-low}, t_{across}, t_{along-low}, t_{along}, t_{local-orient}, t_{global-orient} >$ with the following interpretation: It is the largest group of non-intersecting line segments such that for each line in the group, there exists no line in the group such that $d_{across} < t_{across-low}$ and $d_{along} < t_{along-low}$ and there exists at least one line in the group obeying the following constraints:

1. The across-the-line distance d_{across} between the lines is such that $t_{across-low} < d_{across} \leq t_{across}$.

2. The along-the-line distance d_{along} between the lines is such that $t_{along-low} < d_{along} \leq t_{along}$.

3. The orientation difference between the lines is no more than a threshold $t_{local-orient}$.

and the entire group satisfies the assumption that the maximum orientation change between any two lines in the group is no more than a threshold $t_{global-orient}$.

If the observation conditions include only pose changes, and if all the lines in a model group are equi-spaced, then the model group appearing in the image (the visible part of it, that is) is bound to be present in one such augmented groups because its consecutive lines would exhibit an inter-line spacing within the specified range. In the presence of occlusions and other observation conditions, and when the inter-line spacing in the model group is not uniform, the model group can still be captured in the augmented groups, albeit in a fragmented form. That is, the model group may be partitioned (and possibly merged with adjacent lines) into several augmented groups. But as long as two adjacent lines in the model group are visible in the image, they can still be captured in one of the augmented groups and this should, in theory, be sufficient for recognition (as the two line end points can provide four features).

Algorithm for generating augmented line groups

The algorithm for selectively generating the groups satisfying a model description $< t_{m-2d-across}, t_{m-2d-along}, t_{m-local-orient}, t_{m-global-orient}, p_1, p_2 >$ proceeds by first generating the line group $< t_{m-2d-across}-p_1, t_{m-2d-along}-p_1, t_{m-local-orient}, t_{m-global-orient} >$ using the grouping algorithm of Section 5.2.2. This can capture model groups that have unequal inter-line separation in the case of the model object undergoing pose change specified by the lower limit of $t_{m-2d-across} - p_1$. Then augmented closely-spaced line groups specified by $< t^{i-1}_{across}, t^{i}_{across}, t^{i-1}_{along}, t^{i}_{along}, t_{local-orient}, t_{global-orient} >$ are successively generated using a modified version of the grouping algorithm of Section 5.2.2 as follows:

1. Let $t^0_{across} = t_{m-2d-across} - p_1$ and $t^0_{along} = t_{m-2d-along} - p_1$.

2. For i = 1 to p2 - p1 do

- Let $t^{i}_{across} = 1 + t^{i-1}_{across}$. and $t^{i}_{along} = 1 + t^{i-1}_{along}$.
- The grouping algorithm of Section 5.2.2 is applied to the line segments with the following modifications to Steps 2a and 2c:
  - In Step 2a, an annulus of neighborhood lying between the rectangles $2t^{i-1}_{across}$ x $2(t^{i-1}_{along} + l)$ and $2t^{i}_{across}$ x $2(t^{i}_{along} + l)$, where l is the length of the line, is scanned and all lines passing through this annulus but not through the inner rectangle are retained.
  - In Step 2c, in addition to checking for $t_{m-global-orient}$, each enclosing group being merged with the current group is checked for violations against the annulus neighborhood constraint mentioned above.

Analysis

The above algorithm makes $(p_2 - p_1)$ passes over the line segments in generating the matching groups for each model line group. However, the total number of matching groups generated is still linear in the number of line segments since each line can belong to at most $(p_2 - p_1)$ groups, one for each line lying in the annulus of neighborhood between the outer and inner rectangles of dimensions $2t^{i-1}_{across}$ x $2(t^{i-1}_{along} + l)$ and $2t^{i}_{across}$ x $2(t^{i}_{along} + l)$. The above procedure can be repeated for generating matching groups for each model group separately. Alternatively, the allowable inter-line spacings for all model line groups can be pooled together to form ranges of pixel spacing for all the model groups, and the search for matching groups can be done for each such range using the above algorithm. The time to generate the augmented line groups for an iteration i is still O(n^2) (n is the number of line segments) since the grouping algorithm is the same as before, and Step 2c examines each pair of line segments at most once. For the allowed scale changes, the range $(p_2 - p_1)$ is small enough so that the entire operation of selective group generation can be done in $O(kn^2)$ time, where $k \ll n$ is a constant representing the number of passes over the line segments.

An Example

We now illustrate model-driven selection using parallel-line groups with an example. Figure 5.8 shows model line groups being used to perform selection. Here the parallel-lines on the ladder part of the toy fire truck serve as the model line groups and are specified by the constraints $< 5, 0, 6, 10, 3, 0 >$ implying that the allowable scale changes are from a maximum across-the-line spacing of (5-3 =) 2 pixels up to 5 pixels (in other words, allowing the object to be imaged farther than it is in the model description). Here no variation is allowed in the along-the-line spacing as the model lines all overlap (i.e. have $t_{m-2d-along} = 0$). The model-line groups with the given specification are shown in Figure 5.8b. Figure 5.8c shows (an edge-image of) a scene in which the model object appears at a different orientation and has a portion of it occluded. Figures 5.8d-g show the matching augmented closely-spaced line groups obtained using successively increasing line-spacing as specified by the range $< 0, 2, 0, 0, 6, 10 >$, $< 2, 3, 0, 0, 6, 10 >$, $< 3, 4, 0, 0, 6, 10 >$, and $< 4, 5, 0, 0, 6, 10 >$, respectively. These matches to the indicated model line groups are shown collectively in Figure 5.8h. As can be seen from the figure most of the model line groups are captured in the matching groups, although there is evidence of fragmentation in two of the groups marked 1 and 2 as shown in Figure 5.8b.

Discussion

Model-driven selection using the above algorithm generates enough candidate match groups for a model line group to avoid false negatives under most observation conditions. Further, by requiring the groups to have a minimum inter-line spacing, it avoids generating unnecessary false positive matches to model line groups. This can easily happen if a simpler strategy for group generation were used such as generating ordinary (rather than augmented) closely-spaced parallel line groups by successively increasing the line spacing from $t_{2d-across} - p_1$ to $t_{2d-across} + p_2$. Such a scheme would create successively bigger groups (since a group with small line spacing would always satisfy the constraint of a bigger line spacing) that are often more unreliable and unlikely matches to model groups.

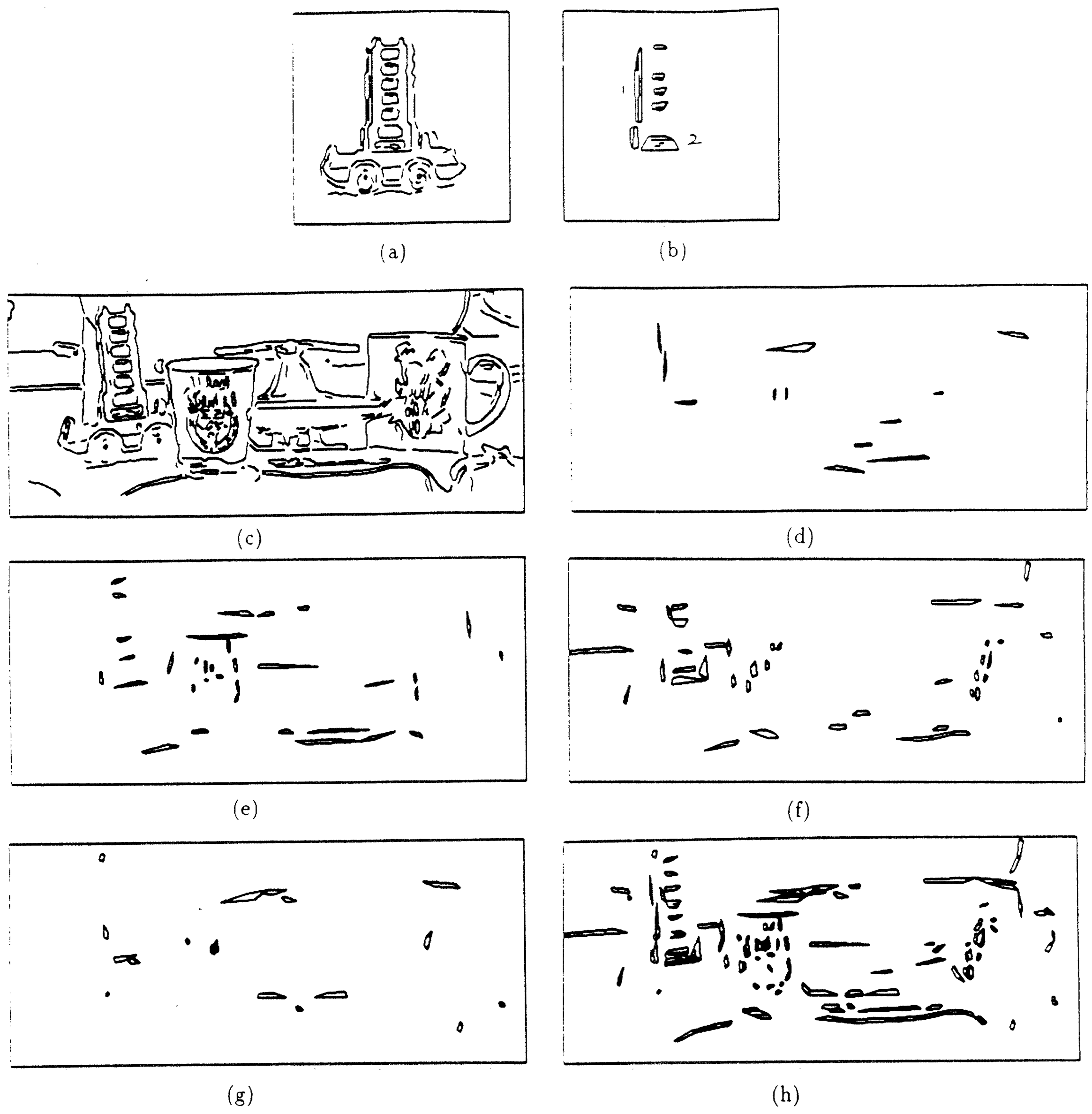


Figure 5.8: *Illustration of line-groups-based model-driven selection. (a) Edge image of a model object showing instances of closely-spaced parallelism between lines. (b) Some of the line groups extracted using the grouping algorithm using the constraints of* $t_{across} = 5, t_{along} = 0, t_{local-orient} = 6, t_{global-orient} = 10$. *(c) An edge image of a scene in which the model object appears. (d) - (g) Augmented closely-spaced line groups generated using the description of the model line groups shown in (b). (h) The line groups in the image that are possible matches to the model line groups of (b) under the allowed scale and pose changes.*

5.4.4 Search reduction using model-driven line grouping

The model-driven selection mechanism using line groups described above identifies candidate groups in the image that could be potential matches for model line groups under some allowable transformation and taking into account the effect of occlusions, illumination changes, etc. These matching model and image line groups can then be given to a recognition system that will isolate features from the line groups to actually solve for the pose of the model object. To see the search reduction possible with model-driven selection, let M_g model line groups be used to perform model-driven grouping. Let the total number of matching groups be N_g with k_i image groups matching a model group i. Letting (i, j) represent a match between model group i and image group j, and letting (m_i, n_j) stand for the number of features in the matching groups, the number of matches that may have to be tried to align the model object with the image is $O(\sum_{i=1}^{M_g} \sum_{j=1}^{k_i} m_i^4 n_j^4 4!)$. To estimate the search reduction due to model-driven grouping, we selected some model objects (views of them, that is) possessing closely-spaced parallel lines on the surface, generated and retained some line groups, and recorded their specifications as $\{t_{2d-across}, t_{2d-along}, t_{local-orient}, t_{global-orient}\}$. We then used scale bounds of $s_1 = 0.5$ and $s_2 = 2.0$ (this allows objects in the scenes to be imaged at half to twice the distance at which their model descriptions were recorded) to complete the model line group descriptions. By placing these objects in scenes containing clutter, and allowing partial occlusions and illumination changes, we ran the selective group generation algorithm of Section 5.4.3 to record the matching image line groups for model line groups. The end points of line segments were considered as features to be used for recognition, and the number of features in both the model and image line groups were recorded. These experiments gave the values for M_g, N_g, m_i, n_j, k_i in the above formula. Table 5.4 shows the results of these studies, with column 10 showing the number of matches using model-driven grouping evaluated using the above formula. The number of matches that would be required without grouping is also shown in the table for comparison. As can be seen, the number of matches is far less with model-driven grouping. To get an estimate of the actual search re-

S.No.	M	N	M_g	Avg. k_i	Avg. m_i	Avg. n_j	Group constraints	Num. Matches	
								No Selection	Model-driven Selection
1.	466	1768	9	150	6	4	$< 5,0,6,10,3,0 >$	$1.88\text{x}10^{22}$	$5.9\text{x}10^{10}$
2.	140	1768	10	111	4	4	$< 7,0,6,10,5,2 >$	$1.49\text{x}10^{20}$	$6.38\text{x}10^{9}$
3.	140	2464	10	94	4	6	$< 7,0,6,10,5,1 >$	$5.6\text{x}10^{20}$	$2.86\text{x}10^{10}$
4.	358	1453	8	48	16	8	$< 3,1,6,10,1,2 >$	$2.98\text{x}10^{21}$	$6.65\text{x}10^{12}$
5.	130	790	5	25	23	6	$< 6,0,6,10,2,1 >$	$4.39\text{x}10^{18}$	$9.00\text{x}10^{11}$

Table 5.4: *Estimated search reduction using line groups-based model-driven selection. The terms M,N, M_g, N_g, k_i, m_i, n_j are as explained in text. The allowed scale and pose changes in each case are indicated in the augmented group constraints.*

duction as well as the number of false positives and negatives due to model-driven grouping, however, the grouping mechanism should be integrated with an actual recognition system and its performance evaluated. The results of such experiments will be discussed in later chapters.

Even with model-driven grouping, the number of matches when considered on an absolute basis, is still very large. This can again be attributed to the large number of matches k_i for model line groups, and to the sometimes large size of the matching groups. To handle scale changes, large inter-line spacing values have to be examined for group generation, unlike in the case of data-driven selection. This may cause some of the groups to be unreliable or large-sized because of merging across objects. Both these problems can be alleviated if model-driven grouping is used in conjunction with prior region selection done using more reliable cues such as color and texture. One such method of combining grouping with prior selection is discussed in the next section.

5.5 Line Grouping in Conjunction with Prior Region Selection

So far we have examined grouping of line segments based on the constraint of closely-spaced parallelism as an independent selection mechanism. But our original motivation for grouping lines was to organize the features within prior selected color or texture regions into small groups, primarily for reducing the search in recognition. We now explore the use of line grouping within regions that are selected a priori based on cues such as color and texture.

Data or model-driven selection using line groups can be easily achieved within previously selected regions by modifying the grouping algorithms of Sections 5.2.2 and 5.4.3 to assemble lines obeying an additional constraint of all lying within the selected regions. This not only restricts the number of groups generated in the image but also their size, by preventing the lines belonging to adjacent objects from being merged, thus making such groups more reliable. Moreover, when model-driven grouping is done within prior selected regions, an additional constraint is provided by the enclosing regions and restricts the possible matches to model line groups even further. For example, when the regions are prior selected based on model color regions, then we know from Chapter 3 that a correspondence between model and selected image color regions is also established. The search for image line groups matching a model line group, therefore, can be restricted to color regions in the image that correspond to the model color regions spanned by the model line group.

To estimate the search reduction using line grouping in conjunction with prior region selection, we performed experiments in which model-driven line groups were generated within model color regions. The information about color regions spanned by a model line group was used as an additional constraint in finding matching image line groups. An example of such restricted model-driven selection using line groups is indicated in Figure 5.9. Here, the model object, its groups and the scene are the same as in Figure 5.8. The model group specification again is also the same as before, namely, $< 5, 0, 6, 10, 3, 0 >$. Figure 5.9d shows the regions isolated in

S.No	M	N	M_g	Avg. k_i	Avg. m_i	Avg. n_j	Group constraints	Num. Matches	
								No Selection	With Prior Region Selection
1.	466	1768	9	26	6	4	$< 5, 0, 6, 10, 3, 0 >$	1.88x10^{22}	1.77x10^{9}
2.	140	1768	10	20	4	4	$< 7, 0, 6, 10, 5, 2 >$	1.49x10^{20}	2.07x10^{8}
3.	140	2464	10	19	4	4	$< 7, 0, 6, 10, 5, 1 >$	5.6x10^{20}	1.87x10^{8}
4.	358	1453	8	17	16	6	$< 3, 1, 6, 10, 1, 2 >$	2.98x10^{21}	2.39x10^{11}
5.	130	790	5	13	23	4	$< 6, 0, 6, 10, 2, 1 >$	4.39x10^{18}	3.89x10^{10}

Table 5.5: ***Estimated search reduction using restricted line groups-based model-driven selection within prior selected color regions.***

the image using color-based model-driven selection. Figure 5.9e-h show the matching line groups generated using the augmented closely-spaced grouping algorithm within the selected regions of Figure 5.9d. Finally, Figure 5.9i shows all the matching groups using the allowable transformations for the line groups shown in Figure 5.9b. By performing similar experiments on a number of model object and scenes, we recorded the resulting values of $M_g, N_g, m_i, n_j, k_i, n_j$ (these terms were defined in Section 5.4.4) and these are shown in Table 5.5. The number of matches using model-driven grouping with prior region selection was calculated using the same formula that was given in Section 5.4.4 and is shown in Column 10 in Table 5.5. This can be compared with the number of matches using model-driven line grouping without prior region selection given in Table 5.4. As can be seen from the tables, combining line grouping with prior region selection based on cues such as color can greatly reduce the search involved in recognition. This is also corroborated, as we will see in later chapters, by experiments done with an actual recognition system. Restricting line grouping within prior selected regions has the disadvantage though that it relies on the correctness of the prior selection mechanism. This, as we saw in earlier chapters, is not always the case. Color-based selection for example, does identify a good portion of the regions containing the object. But the region isolation is not often very precise so that some spurious line segment-containing groups may still be formed in such a grouping process.

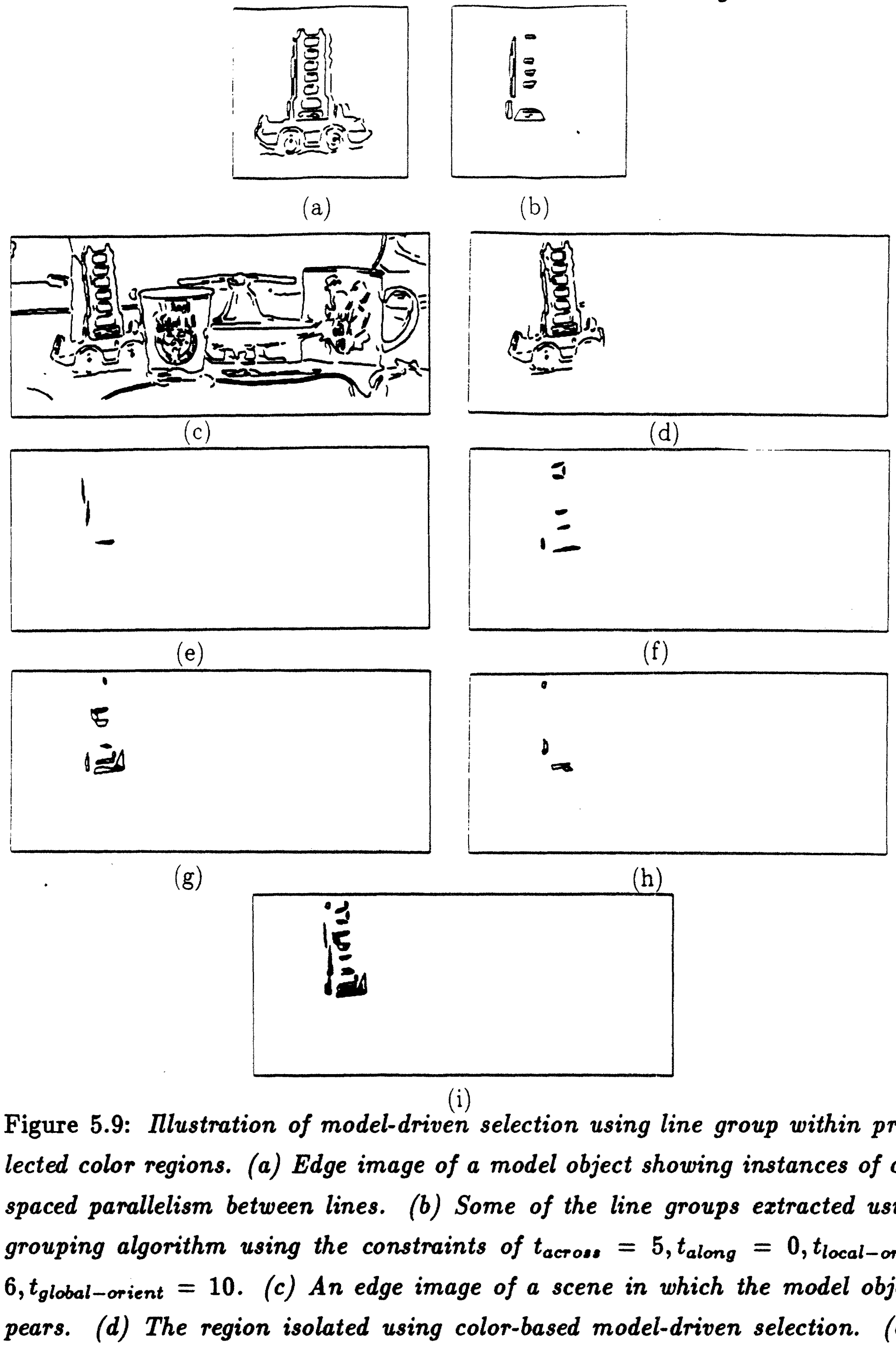


Figure 5.9: *Illustration of model-driven selection using line group within prior selected color regions. (a) Edge image of a model object showing instances of closely-spaced parallelism between lines. (b) Some of the line groups extracted using the grouping algorithm using the constraints of* $t_{across} = 5, t_{along} = 0, t_{local-orient} = 6, t_{global-orient} = 10$. *(c) An edge image of a scene in which the model object appears. (d) The region isolated using color-based model-driven selection. (e)- (h) Matching line groups using successively increasing inter-line spacing as described in text. (i) The line groups in the image that are possible matches to the model line groups of (b) under the allowed scale and pose changes.*

5.6 Conclusions

In this chapter we have examined the use of the property of closely-spaced parallelism between lines in performing data and model-driven selection. Towards this end, a scheme for grouping line segments was presented that possessed several features that make it compare favorably with other existing schemes of grouping of edges. First, closely-spaced parallelism occurs commonly within letter textures and contours of geometrical objects. Secondly, the groups generated tend to be compact and more likely to come from single objects (particularly in the data-driven mode). Also, the number of such groups is linear in the number of lines, and can be generated by a fast algorithm. Lastly, the size of the groups tends to be mostly small, except when merging across objects occurs (which can be reduced when grouping is restricted to prior selected regions). Thus grouping based on closely-spaced parallelism presented here satisfies most of the desirable requirements of grouping for recognition. Finally, unlike in existing approaches to grouping, we have also examined the use of grouping in the model-driven mode. In doing so, an analysis of the changes that can occur to model line groups due to observation conditions was done, and this was found useful in performing model-driven selection using such line groups.

Although closely-spaced line groups are useful as a mechanism of selection, such groups by themselves, may not be as useful in actually solving for the pose of the object, as for example, groups assembled using other constraints such as convexity. Closely-spaced parallel line groups tend to span a small portion of an object (as compared to convex groups) and even though this makes them good for achieving reliability, it makes solution of the pose difficult with features derived from within a group requiring often more than two groups.

Chapter 6

Selection using a Combination of Cues

The previous chapters discussed data and model-driven selection individually using the cues of color, texture, and parallel-line groups. In this chapter we discuss how these cues can be combined to achieve a final selection of the scene. The resulting selection mechanism will then be recast as an implementation of the model of attentional selection proposed in Chapter 2. This will complete the description of the implementation of the attentional selection model for the task of object recognition.

6.1 Combining cues for data and model-driven selection

The problem of combining various cues such as color, texture, depth, etc. has been addressed in both previous models of attention [89, 153, 154, 24] as well as in general scene understanding systems [116, 54, 115]. The purpose behind integration of cues in scene understanding systems is to describe the information in all of the scene better by exploiting the constraints provided in the various feature maps. For example, in the Vision Machine Project, integration of information from several different cues such a color, texture, and depth was attempted with the aim of achieving high performance in unstructured environments for the task of recognition and navigation [116, 54, 115]. Here the brightness edges were treated as the primary cue for integration and were used to verify the presence of discontinuities in other feature maps such as that of color or depth using a Markov random field model [116]. Parallel coupling of different modules was suggested in Marr's primal sketch [102] and in the intrinsic images of Barrow and Tenenbaum [6]. In the intrinsic images approach, the intrinsic images (i.e. versions of the image processed for elementary features) are spatially registered, and the integration proceeds by interpreting first the intensity edges and then creating the appropriate edges in the intrinsic images. In addition, parallel local operations modify the values in each intrinsic image to make them consistent with some intra-image constraints while another set of processes acting across the images makes them meet some inter-image constraints. Finally, a third set of processes operates to insert and delete edge elements in each of the intrinsic image maps. Thus the purpose in both the intrinsic images approach and the Vision Machine Project was to remove spurious discontinuities via the integration of cues so that the remaining discontinuities correspond to physical surfaces and serve as suitable features for later processing stages such as in object recognition.

Other methods of combining cues were proposed in previous computational models of attentional selection mentioned in Chapter 2. Here the aim was to combine the cues to perform a selection of the scene to focus attention. In Koch and Ullman's model [89], the combining of cues was done to create a single saliency map representing a global measure of conspicuity in the image. No specific scheme of combination was mentioned, however. In Ferrier's model [24], the combination of cues was done by assigning weights to each feature value at every pixel location and linearly combining them. The weights can vary over time and were meant to implement the shift of attention focus as well as serve as a medium for higher level cognitive knowledge to decide the importance of feature and hence the overall saliency. The linear combination method, however has the disadvantage that it may cause a region of the image that is only moderately salient in all the features to be declared the most salient because of the summation of saliency values. Also, it assumes that the feature maps are spatially registered.

We now present our approach to combining cues in both data and model-driven modes.

6.1.1 Combining cues in data-driven mode

Using attract-attentional selection as a paradigm for data-driven selection, we combine the cues such that the final selection among the salient regions obtained using the various cues will be towards the most salient location based on any of the cues. The individual approaches to color, texture and line groups-based data-driven selection processed these features in a uniform way by attaching a numerical measure of saliency to each of the feature regions where the value of the measure for a region indicates the conspicuity of that region, i.e. by how much it differs from other regions based on this property. Therefore, the overall most salient region selected is such that it is the most salient region in an individual saliency map and also differs the most from other regions in that saliency map. That is, if we denote the normalized saliency of a unit [1] j in feature type i by s_{ij}[2] and let $M_i = max\{s_{ij}, \forall j\}$ and $SM_i = max\{s_{ij}, \forall j\} - \{M_i\}$ (i.e. the second most salient region), then the most salient region is chosen to be the one which has the largest value of $M_i - SM_i$. In fact, we can also rank the different individually salient feature regions to arrive at a global saliency map as follows. Let the regions in the individual saliency maps be ordered by their saliency values, i.e., $s_{i,j} \geq s_{i,j+1} \forall i, j, 1 \leq i \leq p, 1 \leq j \leq n_i$, where p is the number of features and n_i are the number of units in the saliency map for feature i. Let $\delta_{i,j}$ be the extent of difference between the saliency value of unit j in feature type i with the next lower saliency-valued unit. That is, $\delta_{i,j} = s_{i,j} - s_{i,j+1}$. Since the saliency values are ordered, $\delta_{i,1}$ represents the difference between the most salient unit in the individual saliency map i and the next most salient unit in that

[1] A unit in each of the saliency map refers to either a region such as a color or texture region, or to a group of low-level features such as parallel line groups.

[2] The constant for normalization is the maximum possible value of the saliency measure for the respective feature.

map. Let $L_j = \{\delta_{ij}, i = 1, ..., p_j\}$, where p_j = number of feature types that have at least j units. Thus L_j represents the set of all jth ranked units in all features. Let r_{ij} be the rank of δ_{ij} in L_j. Finally, let $N = n_1 + n_2 + ... + n_p$ be the total number of units across all the maps. Then the overall saliency computation assigns a value to the jth unit in feature type i (i.e. to each of the N units) in the final representation as follows:

$$S_{ij} = N - (j-1) * p + r_{ij} \tag{6.1}$$

The most salient unit is then chosen to be the one with the highest value of S_{ij}.

The above method maintains the order of saliency within each map intact by sorting individually among the equal-ranked units in individual saliency maps as represented by the sets L_j. In doing so, it suggests that the dominant region in each of the cues only will play a role in determining the overall dominant region, unlike other combination methods that can sometimes emphasize a region that is mediocre in more than one map because of the method of combination such as in the linear combination proposed in [24]. Since units in different feature saliency maps could span spatially different regions (i.e. they are not spatially registered), the saliency values could not be linearly combined anyway. This method of combination also seems to suggest that given a region that is most salient in, say, color, and another that is the most salient in, say, texture, the region finally selected will either be the color region or the texture region but not both unless they span the same spatial region.

Finally, although only the most salient region may be used in the attracted attention mode, we chose to rank all the selected region so as to cause more than the salient region to be selected as input to recognition, thus reducing the chance of false negatives. Also, note that the individual saliency measure ranks spatially overlapping region so that if more than one region is finally selected and used in recognition, they could span a common portion of the image. Lastly, since each of the saliency maps were designed to highlight the reliability of the regions for recognition, the overall saliency measure described above highlights the reliable regions better for recognition using such a combination of cues.

This method does have some limitations. It regards each cue as equally important and attaches no weighting function to the cues. Also, it ignores any arrangement of regions of different cues that may be distinctive.

6.1.2 Combining cues in model-driven mode

The problem of combining cues in model-driven mode is fundamentally different from its counterpart in data-driven mode because the selected regions have different interpretations in the two modes. Using the attracted-attention paradigm for data-driven selection, the individual salient regions indicated a greater likelihood of coming from a single object but were not necessarily bound to the same object. In the case of model-driven selection, however, the selected regions in various feature maps indicated the likelihood of the desired object being present in that region. Thus the selected regions are meant to be bound to the same object. When such regions show spatial locality, i.e., are either spatially overlapping or contiguous, the likelihood of the object being found increases further. Therefore, the approach we adopt to combine cues for model-driven selection looks for overlapping or contiguous occurrence of the selected regions as reinforcing evidence for the presence of the desired object at that spatial location. In looking for such evidence, we treat some cues as primary while others are treated as secondary cues. Specifically, we treat color and texture as primary cues while the parallel-line groups are used as a secondary cue. This is because for locating regions likely to come from the model object, color and texture are more informative cues than line groups since a number of parallel-line groups could exist on several objects in a scene while a particular combination of, say, adjacent color regions as specified in the model region adjacency graph is not likely to be found on many objects in a scene. These three cues are, therefore, combined for model-driven selection as follows. First, selection using the individual cues is performed in the model-driven mode as described in Chapters 3, 4, 5 to get a set of selected regions in the image. Then spatially overlapping or contiguous occurrences of such selected regions in the primary cues, i.e., color and texture, are found. Next, the occurrences of secondary cues, i.e., parallel-line groups, within this collection of regions is noted [3]. All the regions are then ranked by the amount of evidence shown for the model object. Thus when the object has color, texture, and parallel-line groups information, image regions showing the combined occurrences of all three cues-based selected regions are ranked above image regions showing partial occurrences (such as, say, color and line groups but no texture-selected region).

This method of combining cues does not attach numerical values to each of the selected regions nor does it weigh all the cues equally as was done in combining cues in data-driven mode. By only looking at the conjunction of spatially overlapping or contiguous regions, it ignores the exact positional relationship between the object's color, texture and line group regions which may cause some false positive selections. Even so, this way of combining the cues reduces the overall number of false positive and negative selections when compared to using the individual cues. False negatives are reduced because any missing information in one of the cues can be reinforced by the presence of information in another cue. That is, if there is no evidence for a particular model feature due to the imaging conditions, say a texture region not being found because of occlusions, the presence of other model features such as, say, a few color regions still visible in the unoccluded portion can reinforce the presence of the object when the cues are combined. The false positives are reduced because it is unlikely for a combination of features such as those present in the model to occur accidentally as spatially contiguous locations in a given scene. In this method of combining cues, while the presence of spatially contiguous or overlapping regions based on individual selection is added evidence, the lack of such contiguity, however, cannot resolve between a false positive selection due to one feature and a false negative selection due to another. So, for example, if a region in the image showed evidence for the object by being a texture-selected region but no color-selected region was found, then this could either be a case of false negative in the

[3]When the model object itself does not possess one or more of these features, say, color but no texture, then these cues are not involved and the combination proceeds with the rest of the available cues.

color map or a case of false positive in the texture map. In practice though, the selection mechanism developed using color is more general than that using texture (which restricts the set of allowed transformations more and is also more sensitive to illumination changes and occlusions), the chance of false negatives with texture is more than the chance of false positives with color. Finally, this method of combining feature in model-driven mode seems to agree with psychophysical evidence from tasks requiring search for a conjunction of features where it was found that illusory conjunctions are possible when features do not span the same location [149].

6.2 Interpreting the Processing of Cues in the Framework of Attentional Selection

The discussion on selection based on individual cues as well as their combination has focused so far on the problem of selection in recognition, i.e., on the problem of isolation of regions likely to come from a single object. But in the approach adopted to address the problem, we were actually developing an implementation of the attentional selection model proposed in Chapter 2. By recasting this work as such an implementation we obtain an instantiation of the model depicted in Figure 6.1. The individual color, texture and line group generation algorithms can be thought of as instantiations of feature detectors, with the color regions, texture regions and line groups being interpreted as the feature maps. The data and model-driven selection mechanism for each of the cues can be interpreted as two different strategies for the selection filters, with the data-driven mechanism serving as a default strategy in each of the selection filters. The strategies for the model-driven case can be fed into the selection filters via an attentional module whose simple instantiation would be to feed the different algorithms into the various modules when the task is object recognition. Finally, the methods of combining cues can be interpreted as two different strategies for the arbiter module.

Let us see how this implementation exhibits some of the features advocated in the model of attentional selection. First, at the level of feature maps, the model allowed

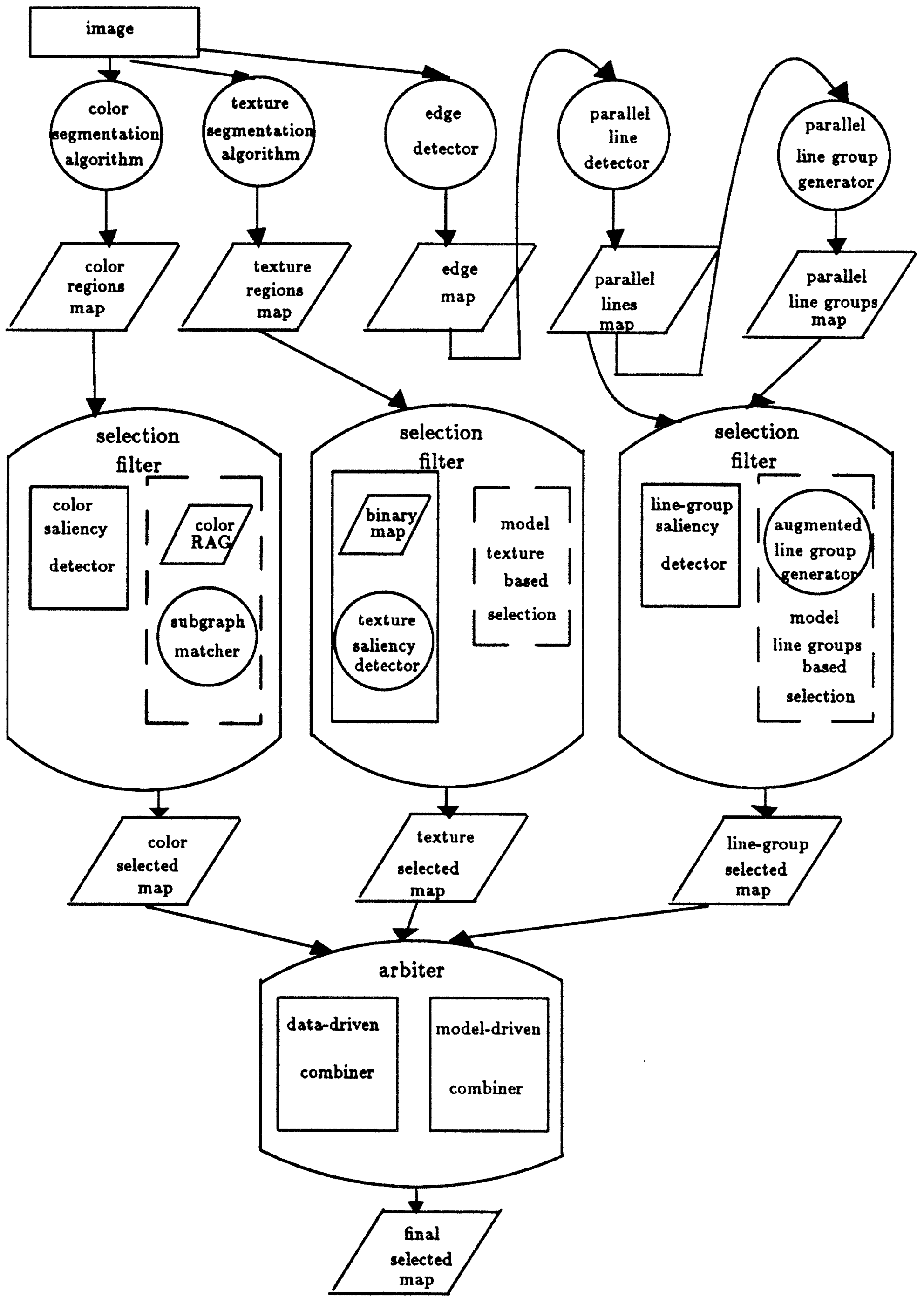


Figure 6.1: ***An instantiation of the proposed computational model of attentional selection. The various data and model-driven selection mechanism using the individual cues of color, texture, and line groups, and their combination can be treated as part of a restricted implementation of the attentional selection model proposed in Chapter 2.***

a hierarchy which the implementation does too in the case of line group detection where parallel lines map forms a hierarchy with the edge map. Next, the model allowed feature detector inactivation through the task. This is again borne out in the instantiation when the texture segmentation stage is bypassed in model-driven selection using texture. At the level of selection filters, the model allowed the selected maps to be non-retinotopic and this is seen in the case of color-based model-driven selection where the solution subgraphs of the image region adjacency graph could be spatially overlapping. Also, the model did not require the saliency computation on all feature maps. The absence of such a computation on the edge map exhibits this in the implementation of the model. Finally, the model suggested that the selection filters could be fairly complex and employ intermediate image representations. This was seen throughout the design of data and model-driven selection mechanisms for color, texture, and parallel-line groups, such as the generation of region adjacency graph for color and binary maps for texture.

6.3 Conclusions

In this chapter we have presented ways of combining cues in data and model-riven modes. Also, while we kept the discussion in the earlier chapters self-contained by focusing on the problem of selection in recognition, we recast the resulting mechanism of selection using the individual cues and their combinations as implementation of the model of attentional selection proposed in Chapter 2. The instantiation of the model thus obtained exhibited many of the features proposed in the attentional selection model. It differed from the model in one respect, namely, that it retained more than one region as the overall selected region. This was done to avoid false negatives during recognition.

Chapter 7

A Recognition System with Attentional Selection

The previous chapters saw the development of an attentional selection mechanism using the cues of color, texture and parallel-line groups. It was meant to serve as a data or model-driven selection mechanism to improve the performance of a recognition system. We now consider the problem of evaluation of this mechanism. Following the reasons given in Chapter 2, we consider only the evaluation of pay-attentional selection. Towards this end, we discuss here the integration of attentional selection with a recognition system. This is done by first describing a bare recognition system and then presenting a way to embellish it using attentional selection. The performance of such an integrated recognition system is described in the next chapter.

7.1 Need for Building a Recognition System

Building a recognition system to test the performance of attentional selection is not only natural but also necessary to correctly assess the performance of selection. Recall that in the earlier chapters examining selection using the individual cues of color, texture and line groups, we obtained estimates of the search reduction using such selection in recognition. Even though the search estimates showed a great reduction in search using selection (when compared to the case of no selection), they were based on a worst-case analysis of the matching situation. It is possible that the search reduction is perhaps not as dramatic in practice either because there was not much information to work with (no color or texture information available, say) or because the selection mechanism gave a lot of spurious regions containing many features. Also, as we will see later in this chapter, a selection mechanism besides reducing the number of features to be examined also provides some additional information that can be useful during the various stages of recognition which can lead to a more accurate identification of objects. Finally, a selection mechanism, however, carefully designed, can be prone to errors. In such cases, it may be necessary to examine the effect of errors in the selection process on recognition in terms of the additional false positives and negatives.

7.2 Choosing a Recognition System

In choosing a recognition system to test the performance of attentional selection, we restrict to recognition systems that recognize rigid objects using stored geometric descriptions of objects as models in a library. Even so, we have a considerable number of recognition systems to choose from the available literature [2], [8], [73], [60]. These recognition systems use a geometric description of the model object in terms of features such as points and lines. They then extract similar types of features from a given image, and match enough data features to model features to obtain a transformation that projects the model into image coordinates. Finally, the correctness of the transformation is verified by finding ample evidence for the projected model in the available image data. The basic difference between the various recognition approaches lies in the way they control the combinatorial explosion that can result from examining all matches between data and model features to obtain the correct transformation. They can be classified as correspondence space-based [61, 60, 8], pose space-based [2, 143, 37] or alignment-based methods [71, 156, 155, 73, 157, 95]. In correspondence space-based methods, the space of all possible matches between data and model features, called correspondence space, is explored by a search mechanism such as the depth-first backtracking search of a tree of interpretations done in the RAF system [61]. The pruning of the search space is done by exploiting unary and binary geometric constraints on the data and model features. Other variations of the interpretation tree approach differ mainly in the manner of exploration of the interpretation tree and the additional information used to order the nodes for exploration [2]. In the local feature-focus (LFF) method, for example, certain salient or focal features on a model object are used to guide the search through relevant portions of the correspondence space [8]. While the search for a consistent hypothesis is done explicitly through the correspondence space in these approaches, the pose space-based methods use clustering techniques such as the generalized Hough transform to [5, 32] examine instead the pose space, i.e. the space of possible transformations to find a pose that is consistent with the most number of data-model feature matches [2, 143]. Here, all possible matches between model and data features are considered, and for each such match, the set of transformations that could give rise to it are predicted. Each transformation is represented by a point in the pose space, and for each transform consistent with a data-model match, we accumulate a vote supporting the transform. Points in the space that acquire a large number of votes are likely candidates for an object's pose [37, 143]. Finally, alignment-based methods [71, 73, 155, 156, 157, 95] explore only a portion of the tree of interpretations by examining matches between a certain number of data and model features that are sufficient to find an alignment transform that can align the chosen model features with the data features. The correctness of the alignment is verified by projecting all of model features into the image using the transform and finding evidence for their presence in the nearby data features.

Although selection was shown to be crucial to correspondence space-based methods for reducing the search complexity [58, 59], we chose an alignment-based recognition system as a test bed for the following reason. The alignment-based methods require a minimal set of distinguished features to produce a close-to-correct transformation to keep them from verifying every aligned match. Such methods tend to perform poorly in cluttered environments where features often get masked by the occlusions [59]. Attentional selection using cues such as color and texture can help resolve such masked features. Thus an alignment-based recognition system can be a suitable test bed to highlight the advantages of attentional selection.

7.3 A Linear Combination of Views-based Recognition System

A recognition system was developed to test attentional selection using a particular alignment-based method called recognition by linear combination of 2D views [156, 157]. The design of the recognition system involved choosing features, building model object and image descriptions based on these features, and choosing strategies for generating feature matches and for verification. Since the recognition method influences the design of each of these components of the recognition system, it is described first.

7.3.1 The Linear Combination of Views method

In the linear combination of views method, the model object is represented by a small set of 2D views (two or three) with full correspondence provided between views [156, 7, 157]. Then recognition proceeds as in other alignment-based methods, i.e. by computing the alignment transform using a small set of matching features, and verifying the model presence in the image by projecting the model using the alignment transform. In building the recognition system, a version of this method was used that was meant for objects with sharp edges and for transformations that include the 3D affine transform. Object with smooth contours can still be recognized by working with line segment approximation of the contours. The method is described below.

Let O be a rigid object. Let P be a 2-D image of O and P_1 be the image of O following a rotation by R (a 3x3 matrix). The two views P and P_1 will be called the *reference views* with P being the *primary view*. Let O' represent O following a 3D affine transformation and let P' be the new view corresponding to O' under orthographic projection, so that

$$O' = AO + T \tag{7.1}$$

where A is a linear transformation matrix, and T is the translation vector and (A, T) specifies the 3D affine transformation. Letting u_1, u_2 and t_x, t_y stand for the first two rows of A and T respectively, we can express the coordinates of a point (x', y') in the new view as

$$(x', y') = (u_1.p + t_x, u_2.p + t_y) \tag{7.2}$$

Let r_1 stand for the first row of R, and let e_1 and e_2 stand for the first two rows of an identity matrix. Then if e_1, e_2, r_1 are three linearly independent vectors [1], then they span R^3 so that any vector such as u_1 can be expressed as a linear combination of these three basis vectors as

$$u_1 = a_1e_1 + a_2e_2 + a_3r_1 \tag{7.3}$$

and similarly,

$$u_2 = b_1e_1 + b_2e_2 + b_3r_1. \tag{7.4}$$

Using Equations 7.3 and 7.4 in 7.2 we can express (x', y') as

$$(x', y') = (a_1x + a_2y + a_3x_1 + a_4, b_1x + b_2y + b_3x_1 + b_4) \tag{7.5}$$

where $a_4 = t_x$ and $b_4 = t_y$, and (x_1, y_1) are the coordinates of the point (x, y) in the second view designated by P_1.

If the correspondence between four such points $(x, y), (x_1, y_1), (x', y')$ in views 1,2, and the new view were known, the coefficients $(a_i, b_i), i = 1, 2, 3, 4$ can be solved. When correspondence between all the visible points in the two reference views is known, these linear combination coefficients can be used to align all the other points

[1]For the three vectors to be linearly independent, there must be some rotation about the y-axis, a condition that can be met fairly easily in generating the model views.

of the primary view with the new view. Detailed verification can then be used to test the correctness of the alignment transform. This is the essence of the linear combination method.

When the new object view is not isolated as described above but is part of a scene, the method can still be applied in principle, as long as four corresponding points can be found between the new view (or the portion of the image containing the new model view) and the primary model view. This will still define an alignment transform that can align points on the model with the given instance in the image.

7.3.2 Building model descriptions

The linear combination of views method requires correspondence between points visible in all the 2D reference views of the 3D model object. Building such a model description by manually picking the corresponding points between reference model views can not only be tedious, but is also likely to be error prone as we cannot accurately identify the corresponding positions. We therefore adopted the method described in [134] for finding the full correspondence between any two views of the model object. This method provides dense point to point correspondence between views from which sparser model descriptions can be generated for matching. The method is described here briefly. A more detailed description is available in [134].

Obtaining full correspondence between model views

The method of generating full correspondence between views assumes that the two views differ only in the transformation undergone by the object with the illumination conditions remaining the same, and the viewer direction fixed, so that the change from one view to another can be ascribed only to the motion undergone by the model object [134]. The basic idea behind finding the corresponding points is to combine information from both the geometry or shape of the surface and from the photometry or intensity variations along the surface. Specifically, an affine coordinate system is set up using four non-coplanar points o, p_1, p_2, p_3 in a view so that every other point p can be described as

$$op = \sum_{j=1}^{3} b_j(op_j) \tag{7.6}$$

using o as the origin and treating the points as vectors. The key observation exploited in the method is that the coefficients b_j are invariant to 3D affine transformations of the object so that the point p' corresponding to p in the second view can also be described by the same coefficients as

$$o'p' = \sum_{j=1}^{3} b_j(o'p'_j). \tag{7.7}$$

Thus if the corresponding points to the four non-coplanar points are known in the second view, the above equations provide two constraints to solve for the three coefficients b_j. The remaining constraint is provided by using information from photometry or intensity variations along the surface of the object. Specifically, the optical flow constraint equation [66] is used to completely solve for the coefficients for every point in the second view, thus obtaining full correspondence between the two views.

7.3.3 Choice of features

The model description obtained using the method described above provides a dense point to point correspondence between points visible in the two model views. If the points were used as features, there would be a large number of them. Further, with no additional information to distinguish between these points, a large number of matches need to be tried. To benefit from the alignment method, we need a few distinguishing features on the object that can be easily detected in an image, are relatively stable, i.e. retain their identity over a range of model views, and are sufficient for performing the alignment [59]. For this reason, we chose corners as features for finding the alignment transform. Since the objects in a scene are likely to have only a few corners, this provides a sparse set of features. Also, a corner retains its identity over all the view changes over which the incoming edges into a corner are still visible, and hence is a relatively stable feature. Further, the corner features tend to be spread out over an object, so that a few such features can give a better alignment transform than closely-spaced features.

Although corner features possess several advantages as features for performing alignment, their accurate localization is a tricky problem. Several corner detectors have been developed in the past [130], [85], [34]. Some of the techniques detect corners as points where the rate of change of gradient direction is maximum [85], or as points where the direction of an object's boundary changes rapidly [130]. Here we used a simple approach to corner detection in which line segment approximations to curves in the edge image was done and each junction point where two lines meet was initially considered a corner. Then these corners were thresholded based on the angle of corner so that very slowly changing angle of turn corners (wide-angled corners) were discarded. Although the accurate localization of the corner was still a problem here, most of the corners in the image were identified. Figures 7.1 and 7.2 show examples of images and the corners localized using this technique. Better methods could have been used, however.

A corner was described in our recognition system by its location, the angle included in the corner, and the orientation of the angular bisector of the corner. Although the identity of the corner remains invariant under pose changes of the model object, its angle as well as the orientation of the bisector can change considerably. By restricting the allowable transformations, (which we have to anyway since the linear combination of views method of recognition tolerates only a limited range of views) we can bound the changes on these values. This can provide additional constraints during matching.

While corner features provided sparse and distinctive features for alignment, final verification of a match requires more detailed features. For this purpose, we chose line segments as the features for verification. The line segments used are those resulting from the line segment approximation done during the extraction of corners described above. These line segments are described in our recognition system by

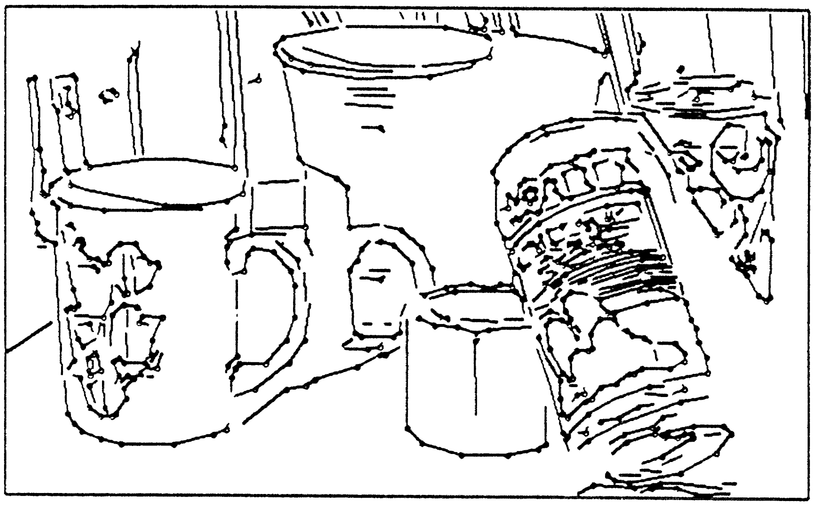


Figure 7.1: *Illustration of a scene whose corners are well-captured by the corner detector described in text. The corners detected are shown in circles superimposed on the image shown.*

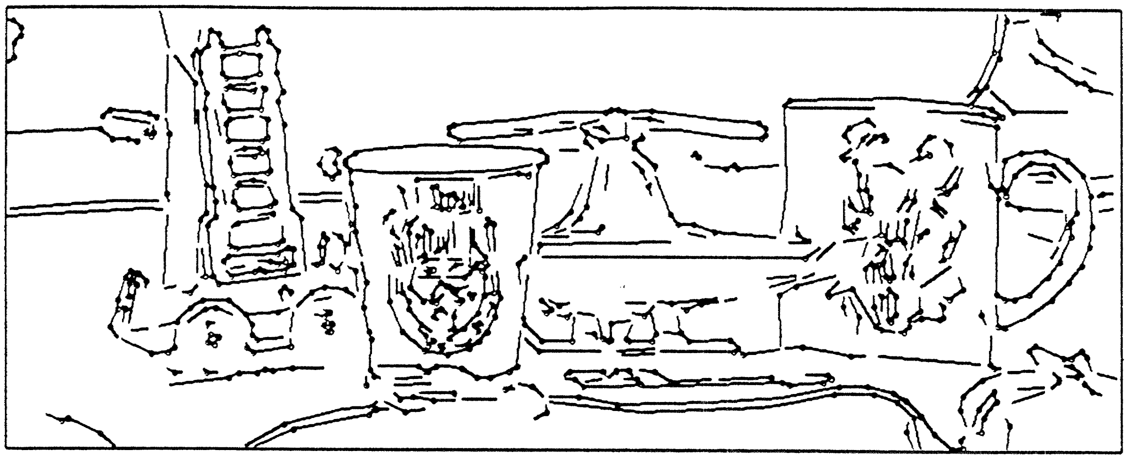


Figure 7.2: *Another Illustration of corner detection using the simple corner detector described in text. The corners detected in the scene shown in the figure are indicated in circles.*

their length and orientation. A threshold on the length was also chosen to limit the amount of occlusion that can be tolerated.

Matching features for alignment

Once the features from the model and image are extracted, the linear combination method requires that at least 4 sets of matching model and image features to be found to find an alignment transform. In the absence of any further information, all 4-tuples may have to be tried leading to $O(M^4N^4)$ matches to be tried where M and N are the number of model and data features respectively. In practice, more matches are actually needed (about 7 to 10 features) to get an accurate estimate of the linear combination coefficients. Because the features are often not precisely localized (this is true of corners), taking more such features for correspondence and obtaining a least squares solution can average out some of the errors. This has a potential of blowing up the number of matches from $O(M^4N^4)$ to $O(M^7N^7)$ (assuming 7 features are sufficient).

In the recognition system developed, the tolerable bounds on the changes in the angle of the corner and the orientation of the bisector were used as constraints to prune some impossible matches. The search for the matching features is done in the framework of the interpretation tree [59, 60, 61] as follows. Denoting the number of features needed to obtain a good alignment transform by "s", a tuple of s image features is generated. For each such choice, the matching model features are examined by using an s-level deep interpretation tree. The choice of model features that can potentially match an image feature at each level of the interpretation tree are pruned based on two constraints. The bound on the angle of corner is used as a unary constraint, while a bound on the variation of the angle between the bisectors of pairs of model and image corners is used as a binary constraint. The binary constraints are checked by pairing the image feature at a given level with all the ancestor image features and similarly for model features. Using image rather than model features as reference in the interpretation tree search allows for missing model features to be tolerated without using the wild card branch at every level of the search tree [59, 60, 61]. But this can still cause needless search when the tuple of image features do not come from a single object (or the model object). Once the constraints are satisfied at each level of the tree and the maximum depth is reached, the matched features are used to solve for the alignment transform. Verification is then carried out using this transform.

7.3.4 Solving for the alignment transform

At this point, the four (or more) sets of matching model and image alignment features have been obtained as a result of interpretation tree search for a given combination of image features. Since each model feature has all the corresponding view information associated with it, the alignment transform can be solved using the system of linear equations indicated in equation 7.5.

7.3.5 Verification

Once an alignment transform is obtained, it can be verified by aligning the rest of the model features with the image features. The line segment features are used for verification as follows. The model line segments are projected into the image using the alignment transform. For each projected model line, all image line segments lying within an ϵ neighborhood ($\epsilon \approx 4$) are considered as potential matches. Then an orientation threshold (≈ 2 change in orientation) and length thresholds specified in the model feature description are used to filter the image line segments. The remaining image lines are ranked by the following measure:

$$\frac{distance - diff}{\epsilon} + \frac{orient - diff}{180} + \frac{length - diff}{model - line - length} \tag{7.8}$$

where $distance - diff$ is the distance between the projected model line and the candidate image line, and $length - diff$ and $orient - diff$ are the difference in the length and orientation of the matched pair of lines. The denominators in the above equation are normalizing factors. The lowest valued image line using the above measure is taken as the matching line. Once a matching line has been found it is removed from the image line segments to be considered for the next model line segment to be verified. A threshold on the total number of long model lines (i.e. those that are within 60% of the longest model line) matched decides whether the alignment transform can be considered to be correct and adequately recognizing an object.

7.4 Integration of Attentional Selection with Recognition

We now investigate the integration of the attentional selection mechanism with the object recognition system developed above. The selection mechanism provides a ranked list of selected regions that are likely to come from an object (either a single object or the model object as the case may be). A simple way to integrate such a mechanism into the recognition system is to consider the image corner features from only the selected regions for matching. However, the information derived from the processing of cues during attentional selection can be of use in several other stages of recognition. We now explore the use of this information in the recognition system to obtain a better integration of attentional selection.

7.4.1 Model description

The information in the color, texture, and line groups description of the model object can be associated with the corner and line segment features. Specifically, the corner descriptions were augmented by the following information: (1) the color region number in the model color RAG along with the color, (2) the texture region number in the model texture description, and (3) the line group information as characterized by model line group $< t_{across}, t_{along}, t_{local-orient}, t_{global-orient} >$ described in Chapter 5. Here, the corners that occur as end points of some line in the line group are considered. The line segment features used for verification similarly were augmented with the color and texture regions in which they occur (if they span more than one region then this is also indicated here) and the line groups to which they belong are characterized as given above.

Similar information can be associated with data features in a given image using the results of processing with the cues in the selection mechanism.

7.4.2 Matching image and model features

The data features needed for matching are taken from the selected regions provided by the attentional selection mechanism. Next the extra information in the augmented corner features is used as an additional unary constraint to prune the possible matches to model features. Thus, for example, image corners are paired with only those model corners that occur in the matching enclosing color regions. A set of binary constraints is implicitly provided in the color-based selection when the solution subgraphs of the image RAG satisfy region adjacencies. This ensures that the unary constraint-based pruned list of matching model features that are associated with an image corner automatically satisfy the adjacency constraints. Thus corners belonging to adjacent color regions will be matched to corresponding adjacent model color regions. The number of matches are, therefore, reduced both by reducing the number of features to be considered as well as by the additional constraints during matching.

7.4.3 Verification

The additional information provided in the line segment features can be used for developing a better verification strategy. We modified the verification algorithm described in Section 7.3.5 to check for the additional constraints provided in the line segment descriptions. Thus matching line segments are constrained to lie within the matching enclosing selected regions. This can rule out several false positive identifications more quickly without requiring detailed verification of all the projected model features. Since most of the time spent in alignment-based scheme is during verification, having a better verification strategy that can rule out false positives quickly will greatly improve the performance of the recognition system.

7.5 Summary

In this chapter we have presented a recognition system to use as a test bed for evaluating attentional selection. We found that there is a potential for both reducing the search and the chance of false positive and negative identifications using the recognition system integrated with the selection mechanism. The next chapter will bear this out in studies using this system to test the recognition of objects in scenes.

Chapter 8

Results

In this chapter, we report results of experiments done using the recognition system integrated with attentional selection. We had already mentioned earlier the reason behind testing only the pay-attentional selection. We first describe the objects and scenes used to test the recognition system. We then report results that test the performance of attentional selection for a model object possessing one or more of the three cues, namely, color, texture, and parallel-line groups. These experiments mainly indicate the search reduction that is possible using attentional selection. Lastly, we discuss the chance of false positives and negatives made by the recognition system that can be attributed to attentional selection.

8.1 Description of Test Models and Images

The model objects we considered were often colored, and sometimes had texture patterns on them. In addition, some of the models possessed structural information that showed either explicitly or implicitly as a group of parallel lines of some orientation. Figure 8.1 shows an example of a model tried that illustrate one or more of these features. The scenes for testing the recognition of such models were generated by placing the model object among other distractors that often had features resembling some of the model features. Figure 8.2 shows an example of a typical scene tried. Here the distractors have similar color regions to the model object of Figure 8.1b. The scenes were imaged using a CCD color camera at a distance that was more or less the same as for assembling the model object description so that not much variation in scale was allowed. The orientation of the object in the scene was changed with the transformation being mostly rotation about the vertical (y-axis) and horizontal (x-axis). Such changes caused portions of model to go out of view or come into view and provided some difficult test cases for the selection mechanism. To test selection under illumination changes, the direction of light source was generally different in the model description and the scenes. For example, in the scene depicted in Figure 8.2, the model object of Figure 8.1b is taken using a different illumination direction. Also, the same scene was imaged using two different light sources, namely, sun light (using a quartz lamp) and overhead multiple tube lights as in a room. To test for occlusions, objects were placed in front of or behind the model object. Some of the objects used were made of plastic materials so that the scene displayed artifacts of illumination such as specularities and inter-reflections. Thus the scenes generated depicted fairly complex imaging conditions that varied considerably from the model description.

8.2 Experiments with the Integrated Recognition System

We now describe the experiments used to test the recognition system integrated with attentional selection. The image features for matching are chosen so that they are distributed across the regions provided by the selection mechanism. Thus when color is used in selection, the features are distributed such that at most 3 features are chosen from a color region when enough color region matches are found. When line groups are also used in selection, the features for matching are chosen from far apart line groups in the selected regions.

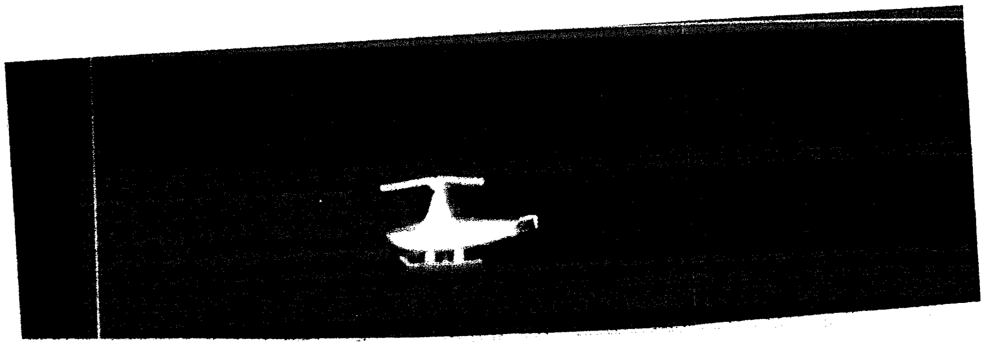


Figure 8.1: *An example of a model used to test the recognition system.*

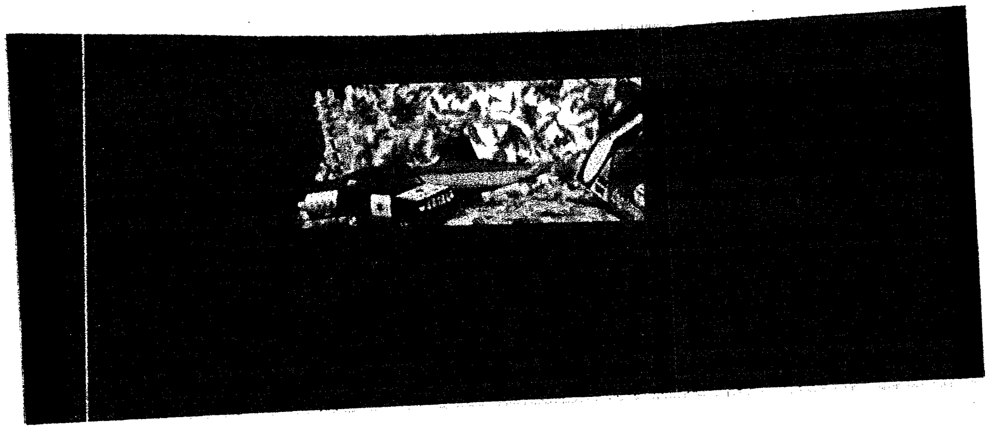


Figure 8.2: *An example scene on which the recognition system was tested.*

8.2.1 Order of feature pairings

In the search for the matching features using the interpretation tree in the recognition system, the order in which the tuples of image or model features are tried can affect the actual number of matches explored by the system before a solution is found. To avoid a bias towards a favorable ordering, the tuples of features were randomly generated in a restricted sense in that they had to be widely spaced but could be chosen in any order. About 600 random orderings were tried for each recognition experiment and the number of matches tried before a correct alignment transform is obtained is noted for each such ordering [1]. In this way the number of matches are fairly representative of the search involved in recognition.

8.2.2 Recognition with selection using color

We first present results of recognition when selection can be performed using only one of the cues. Since texture alone cannot be used (being a planar patch), and line groups were meant to be used as a secondary cue, we consider here the case when only color information is available about the model object to perform selection. Figure 8.3 and 8.4 show an example of recognition using attentional selection when only color information is available. Figures 8.3a,b,c show three views of a model object, the first two of which were used in the final model description[2]. A color region segmentation of the first view using the algorithm of Section 3.4.2 is shown in Figure 8.3d. The regions depicted there constitute the model color description as indicated by the RAG in Figure 8.3e. Figure 8.3f shows a scene in which the model object appears partly occluded. This image is also taken under different illumination conditions that slightly changes the appearance of some of the colors on the model object. The color regions in the scene obtained using the color segmentation algorithm is shown in Figure 8.3g. The segmentation is not perfect due to the merging of the colors

[1]The cases tested always had the model object present in the scene.

[2]The third view shown here was used in an earlier implementation of the recognition system that used three rather than two views. Recognition using a linear of combination of three views was also described in the work of Basri and Ullman [157, 7]

of the object with portions of the background, as seen for example in the yellow blades portion and the blue portion of the object. The result of color-based model-driven selection is shown in Figure 8.3h. Although the image represents a scene in which the color-based selection does not isolate the object perfectly, the isolated region, nevertheless, contains a portion of the object sufficient to initiate recognition. Recognition using features from the color-selected region is shown in Figure 8.4. Figure 8.4a shows the primary model view with corners(in circles) and edge features. There are about 50 corners in the model object. Figure 8.4b shows the corner and edge features of the image of Figure 8.3f. There are about 600 lines and 400 corners in this image. Figure 8.4c shows the corner features within the color-selected region shown in Figure 8.3h. The selected region contains about 40 corners. Because of the errors made in color-based selection, we can see that there are spurious features within the selected region. Even so, there has been a large reduction in the number of features. After searching for the matching features in recognition, the set of matched features found are shown in Figures 8.4d and e. These features gave an alignment transform that was verified to be correct by the recognition system. As can be seen, the corner features for a good alignment transform are distributed across the various color regions. Finally, Figure 8.4f shows the model object projected into the image of Figure 8.4b using the alignment transform given by the correspondence shown in Figure 8.4d and e. By considering several scenes such as the one in Figure 8.4, and several (around 600) random orderings of the list of corner features in the color-selected regions as described in the previous section, we recorded the average number of matches that needed to be tried before successful verification by the recognition system. These results are shown in Table 8.1. The number of corners indicated in Column 3 of the table is indicative of the complexity of the scenes tried. The complexity of the model color descriptions is indicated by the size of the model region adjacency graph indicated in Column 5 in this table. Finally, the number of matches using features from the color-selected regions during recognition is shown in Column 8. The number of matches actually explored by the recognition system for finding seven corresponding corner features from the color-selected region are listed. This can be compared with the estimates using no selection that was given in Chapter 3

S.No	Model corners	Image corners	Retained corners	Model regions	Retained regions	No selection Num. matches	Color-based Selection Num. matches
1.	23	120	17	2	2	1.46×10^{16}	2.48×10^{5}
2.	22	380	19	1	1	3.66×10^{19}	1.49×10^{6}
3.	23	317	23	2	2	1.46×10^{19}	4.62×10^{5}
4.	75	244	58	5	3	1.86×10^{22}	1.79×10^{7}
5.	52	391	52	5	5	3.51×10^{22}	1.32×10^{6}

Table 8.1: *Results of recognition using attentional selection using only color.*

for the same objects and scenes and is shown again in column 7 of Table 8.1. As can be seen from the table, the number of matches that are explored in practice are far fewer compared to the case of recognition without any prior selection. Even so, the number of matches considered is rather high. This can be attributed to the fact that using color alone some false positives are likely. That is, more than one subgraphs of the image RAG may have to be explored before a matching subgraph is found. This is likely to happen, however, in scenes with an object having similar colors as the model object but showing a different spatial relationship. Recall that the color subgraph matching algorithm of Section 3.6.3 is not very constrained in that, apart from adjacency and relative size information, the precise spatial layout of the regions of the model object is not exploited in searching for the matching subgraph. This was done to avoid false negatives in the case of occlusions and other imaging condition changes that may cause such relationships to be violated, with the result that there is a slight increase in the number of false positives in region identification.

8.2.3 Recognition with selection using texture and color

We now describe experiments when at least two cues are available for selection. We first consider the case when the model objects have both color and texture information that can be used to perform selection. Since the model color region description spans an entire view of the object, it includes the texture region. The

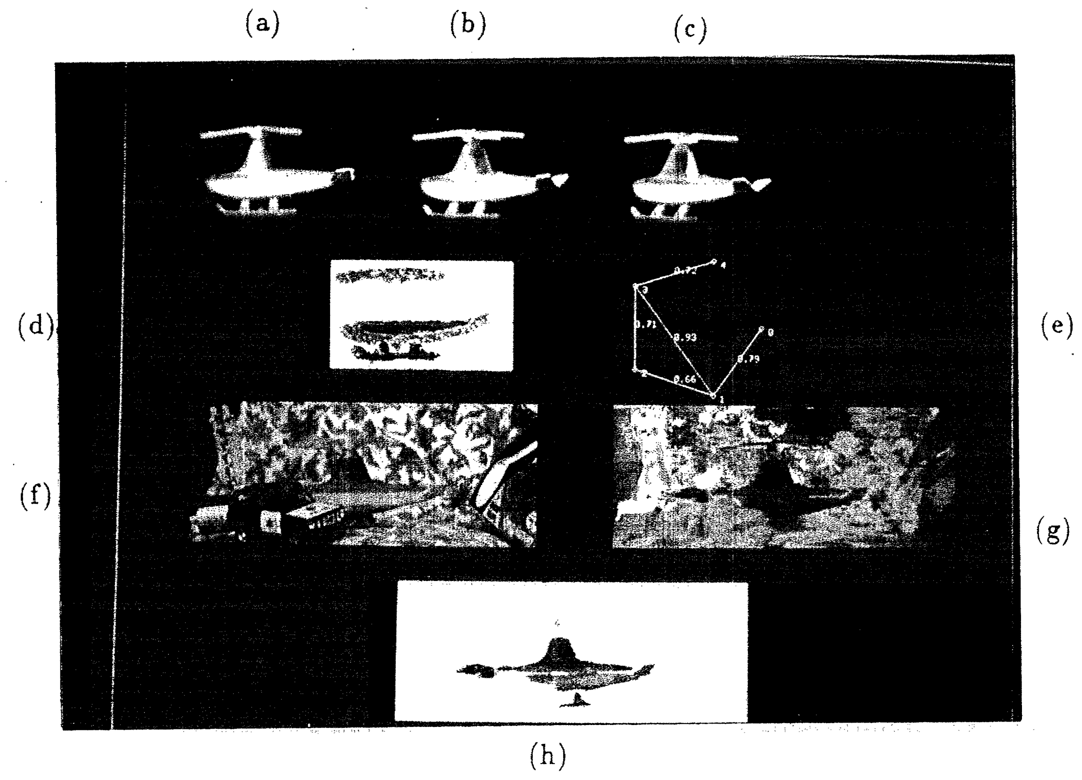


Figure 8.3: *Illustration of attentional selection using only color information. (a)-(c) Thee views of an object serving as the model. (d) The color regions in the model extracted using the color region segmentation algorithm of Section 3.4.2. (e) The color region adjacency graph description of the model color regions. (f) A scene in which the object occurs. (g) A color region segmentation of the scene generated using the same segmentation algorithm. (h) The regions selected using the model-driven color-based selection described in Section 3.6.3.*

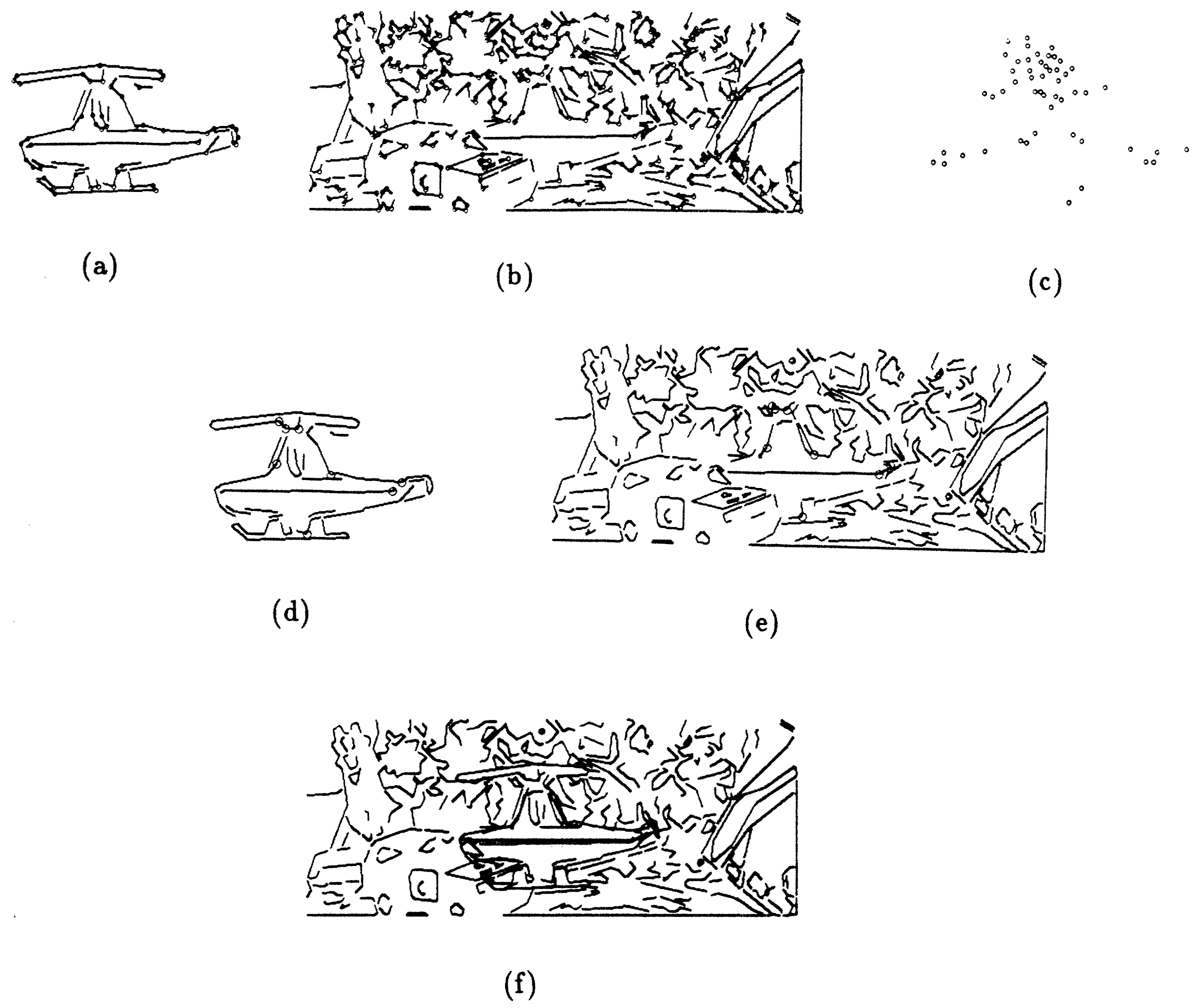


Figure 8.4: *Illustration of recognition within the regions selected by the attention mechanism using only color. (a) The primary view of the model object showing corner (in circles) and line features. (b) An edge image of a scene showing corner (again in circles) and line segments. The model object and the scene are those depicted in Figure 8.3. (c) The corner features in the region selected by the attentional selection mechanism that was indicated in Figure 8.3h. (d) - (e) A set of corner features in the model and image that were declared to be a match using interpretation tree search in the matching stage of the recognition system. (f) The model object projected into the image using the alignment transform computed from the set of matched features shown in (d) and (e). The projected view was verified to be correct by the verification stage of the recognition system.*

retained regions, therefore, are contiguous or spatially overlapping occurrences of color and texture-selected regions. Figure 8.5 illustrates the combined use of color and texture-based selection. Figure 8.5a shows a model object possessing both color and texture information. The color description used for color-based selection is shown in Figure 8.5b while the texture patch used for texture-based selection is shown in Figure 8.5c. Figure 8.5d shows a scene in which the model object occurs. The result of color-based selection is shown in Figure 8.5f while the result of texture-based selection is shown in Figure 8.5e. The result of the combined use of color and texture-based selection is shown in Figure 8.5g. As can be seen from this figure, the false match shown in Figure 8.5e is removed by the combined use of color and texture information.

The features for recognition were then chosen from both the color and texture regions isolated in the image. Figure 8.6 shows recognition being performed using the (color and texture) selected regions depicted in Figure 8.5g fro the model shown in Figure 8.5a. Here one of the views used to build the cup model for recognition is shown in Figure 8.6a with corners (in circles) and edge features. Figure 8.6b shows the corner and edge features of the scene. Figure 8.6c shows the corner features that are retained in the model texture patch of Figure 8.5c. The edge features in the texture and color-selected region are shown in Figure 8.6e. The matching corner features in the model and image that gave an alignment transform that was verified by the recognition system are shown (in circles) in Figures 8.6d and e. Finally, Figure 8.6f shows the model object projected into the image of Figure 8.6b using the alignment transform given by the correspondence shown in Figures 8.6d and e. As can be seen from this figure, the alignment of the texture patch is better than the rim of the cup. This is to be expected since a frontal view of the model object was used in the model description while its instance in the scene is imaged from above (Thus showing the entire rim). By considering several scenes and several (around 600) random orderings of the list of corner features in the texture and color-selected regions in such scenes, we recorded the average number of matches that needed to be tried before successful verification by the recognition system. These results are shown in the last column of Table 8.2. Here the number of matches actually explored

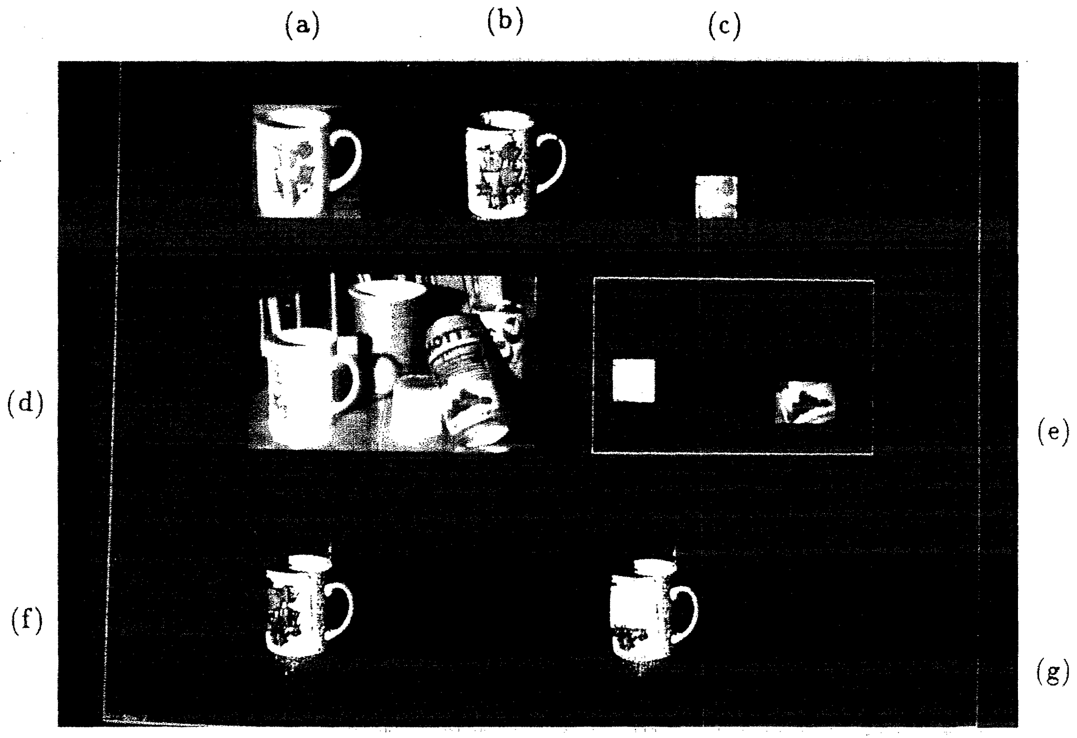


Figure 8.5: *Illustration of attentional selection using color and texture. (a) A view of an object used as a model. (b) The color regions extracted to constitute the color description of the model of (a). (c) An extract of the model view shown in (a) used for building model texture description. (d) A scene in which the model object occurs. (e) The result of texture-based model-driven selection. (f) The result of color-based model-driven selection. (g) The result of combining selection using the two cues as described in Section 6.1.2.*

S.No	M	N	M_t	N_t	Color selected corners in		Estimated search		Actual search
					model	image	No selection	With selection	
1.	114	484	35	38	69	45	1.55 x 10^{33}	3.05 x 10^{23}	2.37 x 10^{6}
2.	96	580	84	74	12	34	1.66 x 10^{33}	3.2 x 10^{27}	3.67 x 10^{6}
3.	96	1401	84	76	12	25	7.96 x 10^{35}	9.1 x 10^{27}	7.8 x 10^{6}
4.	64	256	52	47	12	31	3.17 x 10^{29}	3.49 x 10^{24}	1.67 x 10^{5}
5.	138	392	58	145	80	67	1.35 x 10^{33}	1.91 x 10^{28}	2.38 x 10^{7}

Table 8.2: *Estimated and actual search reduction during recognition with attentional selection using color and texture. The estimated search is shown using texture-based selection. The actual search indicated in the last column is obtained by the combined use of texture and color-based selection as described in text.*

by the recognition system for finding seven corresponding corner features are listed. This can be compared with the search estimates using no selection that were given in Chapter 4 and is shown again in column 8 of Table 8.2. As can be seen from the table, the number of matches that are explored in practice are far fewer compared to the case of recognition without any prior selection. In fact, compared to the number of matches explored, detailed verification was done for only a few (about a 1000) of the matches. From this we conclude that the combined used of texture and color-based selection can lead to a vast improvement in the performance of a recognition system.

8.2.4 Recognition with selection using color and line groups

We now describe experiments in which color and line group information is available on the model object to perform selection. Since line groups are treated as secondary cues, color-based selection is performed followed by line grouping in the retained regions. Two pairs of matching line groups were searched and features within the matching pairs were tried for finding the alignment transform. Sometimes three pairs of matching line groups had to be tried to obtain sufficient features for good alignment. Figure 8.7 shows an example of recognition being performed with selec-

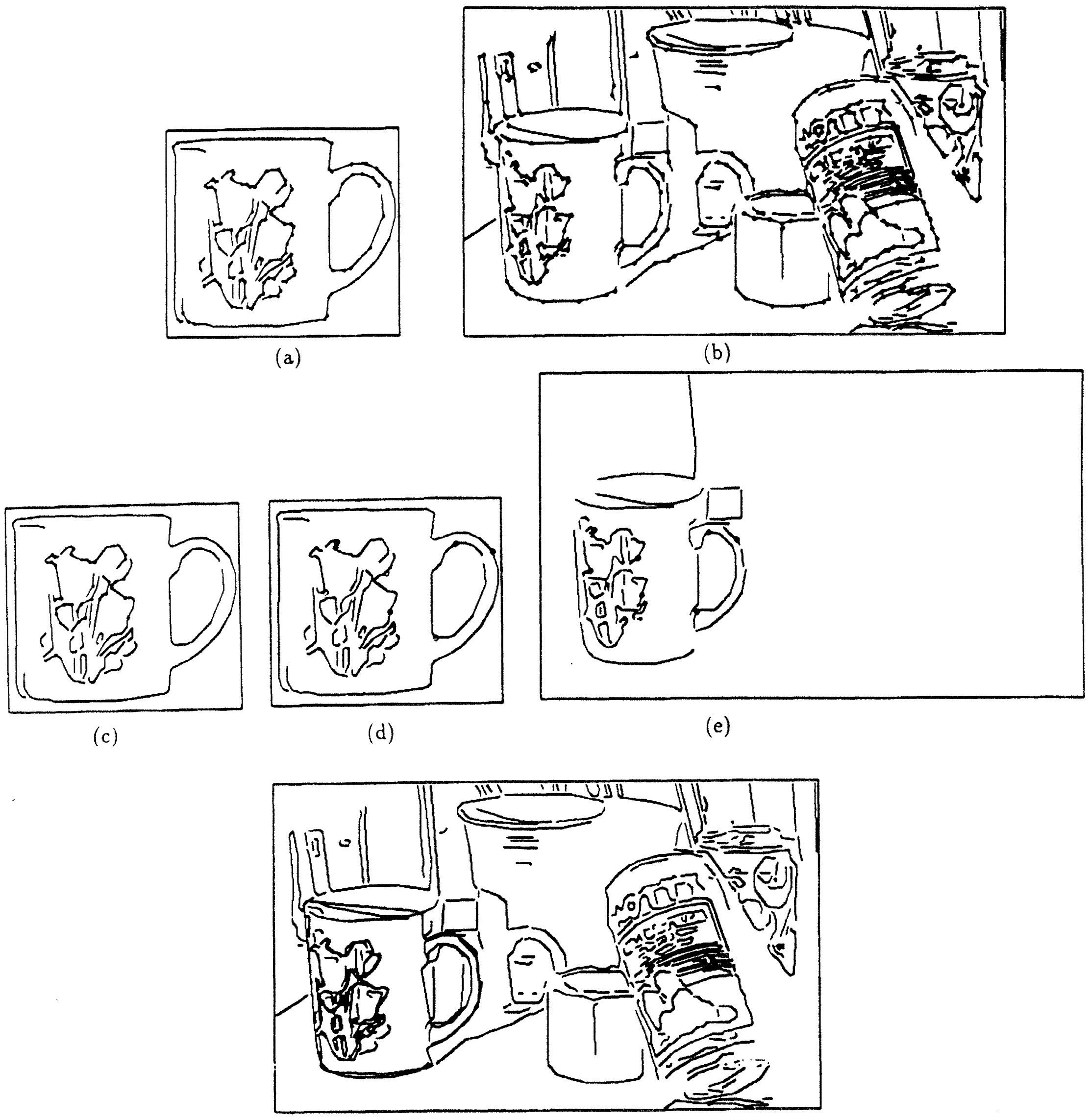


Figure 8.6: *Illustration of recognition in the regions selected by the attentional selection mechanism using color and texture information. (a) The primary view of the model object showing corners (in circles) and line features. (b) An edge image of a scene showing corner (again in circles) and line segments. The model object and the scene are those depicted in Figure 8.3. (c) Corners in the model texture patch of Figure 8.5c. (d) - (e) A set of corner features in the model and image that were declared to be a match using interpretation tree search in the matching stage of the recognition system. (f) The model object projected into the image using the alignment transform computed from the set of matched features shown in (d) and (e). The projected view was verified to be correct by the verification stage of the recognition system.*

tion based on color and line grouping. A view of the model object that exhibits closely-spaced parallelism between lines is shown in Figure 8.7a. Some of the line groups extracted using the grouping algorithm of Section 5.2.2 using the constraints of $t_{across} = 5, t_{along} = 0, t_{local-orient} = 6, t_{global-orient} = 10$ are shown in Figure 8.7b. Figure 8.7c shows an edge image of a scene in which the model object appears. The model and the scene shown here are the same used in Figure 5.9, described in Section 5.5. After performing color-based selection and then model-driven line grouping as described in Section 5.4.3, the line groups that are possible matches to the model line groups of Figure 8.7b are shown in Figure 8.7. A set of matching line groups in the model and selected region and the corresponding corner features within them that yield a transformation that is verified by the recognition system to be correct are shown in Figure 8.7e and f. The projected model overlayed on the original image is shown in Figure 8.7g. By considering again several (around 600) random orderings of the list of groups and features within groups in a number of different scenes, we recorded the average number of matches that needed to be tried before successful verification. Some of these results are shown in Table 8.3. The models and scenes are the same as those used in Tables 5.4 and 5.5 of Chapter 5, but the features here are corners instead of the end points of line segments that were used to develop the search estimates in that chapter. The number of matches actually explored by the recognition system for finding seven corresponding corner features is indicated in Column 10 of Table 8.3. The rather larger number of matches for a smaller model object in entry 5 of the table is due to the larger size of groups (the maximum size was 23) even though the number of model groups is small. Compared to the number of matches explored, detailed verification was done for only a few (about a 1000) of the matches. The estimated number of matches that would be explored without selection for seven corresponding features is shown in Column 9 for comparison. From these results, we see that line grouping-based selection when performed within prior selected regions can lead to a great improvement in the performance of the recognition system.

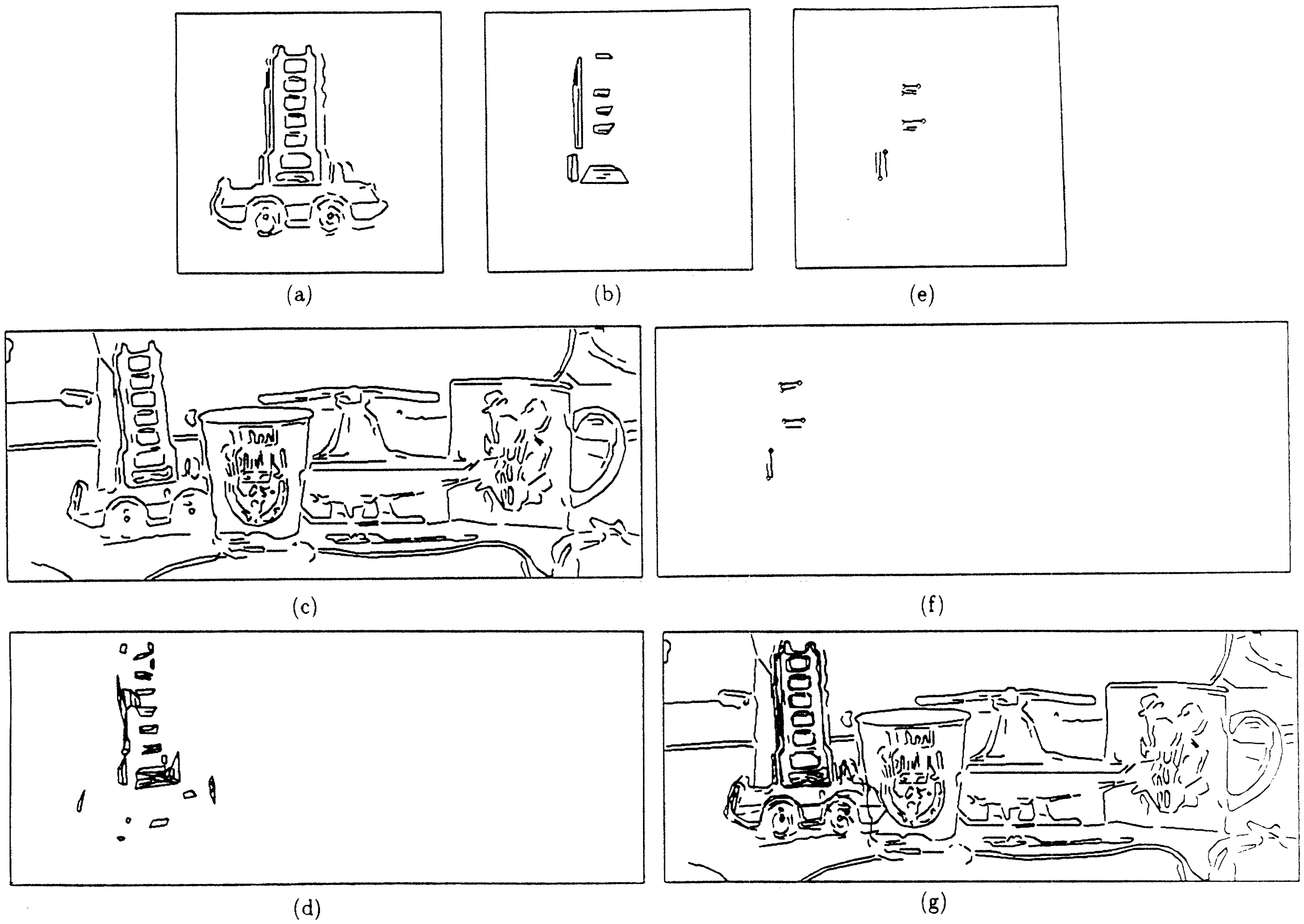


Figure 8.7: *Illustration of recognition in the regions selected by the attentional selection mechanism using color and line groups. (a) Edge image of a model object showing instances of closely-spaced parallelism between lines. (b) Some of the line groups extracted using the grouping algorithm using the constraints of* $t_{across} = 5, t_{along} = 0, t_{local-orient} = 6, t_{global-orient} = 10$. *(c) An edge image of a scene in which the model object appears. (d) The line groups in the image region selected by color that are possible matches to the model line groups of (b) under the allowed scale and pose changes. (e)- (f) The pairs of matching model and image line groups that found a good alignment transform. The circles here show the matching corner features within the line groups. (g) The model object projected into the image of (c) using the alignment transform given by the correspondence shown in (e) and (f).*

S.No	Model corners	Image corners	M_g	Avg. k_i per scale	Avg. m_i	Avg. n_j	Group constraints	Num. Matches	
								No Selection	Explored in recognition
1.	160	580	7	8	4	3	$< 5, 0, 6, 10, 3, 0 >$	9.93×10^{30}	3.9×10^{4}
2.	54	580	10	7	3	3	$< 7, 0, 6, 10, 5, 2 >$	3.77×10^{27}	8.4×10^{4}
3.	54	391	10	9	3	3	$< 7, 0, 6, 10, 5, 1 >$	2.34×10^{26}	1.5×10^{4}
4.	114	484	8	11	4	4	$< 3, 1, 6, 10, 1, 2 >$	2.45×10^{29}	9.2×10^{4}
5.	30	95	5	13	6	3	$< 6, 0, 6, 10, 2, 1 >$	1.13×10^{20}	1.35×10^{5}

Table 8.3: *Actual search reduction using restricted line groups-based model-driven selection within prior selected color regions. Here m_i and n_j refer to number of corner features (rather than the end points of line segments) within a line group. Seven corresponding corner features were used for recognition.*

8.2.5 Recognition with selection using color, texture, and line groups

We now explore the case when the model object has all three features of color, texture and line groups available on them. In this case, the attentional selection mechanism selects regions in the image that are contiguous or overlapping occurrences of the individual selection using the three cues as explained in Chapter 6. Because of the three cues being used, the region selected first by the attentional mechanism is more likely to contain the object than if only one or more of the cues were used. Since groups are formed, the features for recognition are chosen from at least two image line groups (or as many as needed to get 7 features). The matching model features are generated from model groups and have the additional information of the enclosing color and texture regions associated with them as explained in Chapter 7. Figure 8.8 and 8.2.5 illustrate recognition using a combination of all cues. Figure 8.8a and b show the two views used in generating a model of an object possessing color, texture and parallel-line group information. The color regions of the model are shown re-colored in Figure 8.8c. The color RAG description of the model that is used for color-based selection is shown in Figure 8.8d. The texture patch (of size 88 x 56) on the object used for texture-based selection is shown in Figure 8.8e. A scene in which the object is recognized is shown in Figure 8.8f. The results of color and texture-based selections are shown in Figures 8.8g-h. The texture-based selection was done using a 3x3 AR model and a 32 x 32 point FFT for the LP spectrum and a 64 x 50 window analysis with an overlap of 24 x 6. The result of combining selection based on color and texture is shown in Figure 8.8i. Figure 8.2.5 shows the result of selection based on color, texture and line grouping and also illustrates recognition within the selected region. Figure 8.8i had indicated the result of combining the primary cues of color and texture. Since line-groups are treated as secondary cues, the occurrences of matching line groups are noted within the region selected by the primary cues. This is illustrated in Figure 8.2.5. The model line groups used for matching are shown in Figure 8.2.5c. The group constraints for the model line groups chosen are $< 10, 0, 8, 12 >$, $< 0, 4, 0, 1, 6, 10 >$,$< 4, 12, 0, 1, 6, 10 >$. The line groups in the prior selected region of Figure 8.8i that match one or more of the model line groups are shown in Figure 8.2.5e. A set of matching image and model line groups and the corresponding corner features within them that yield a transformation that is verified by the recognition system to be correct are shown in Figures 8.2.5f and g. The projected model overlayed on the original image using the computed transformation is shown in Figure 8.2.5h.

The recognition system using the combined cues-based selection was tried on a few scenes, and the average number of matches tried before successful verification were recorded after trying several random orderings of the list of groups and features within groups. These results are shown in Table 8.4. The number of matches explored by the recognition system for finding seven corresponding corner features using line grouping-based selection within color-selected regions are listed in Column 11 of this table. The estimated number of matches without selection are also shown in Column 9 of this table for comparison. From this we see that using a combination of all three cues reduces the search even further. In interpreting the number of matches with selection using a combination of cues in this table, we must note that this represents the average search over a set of data and model feature orderings some of which are unfavorable. A better indication would perhaps have been given by the average number of detailed verification done which is typically low ($\approx$ 300-600)

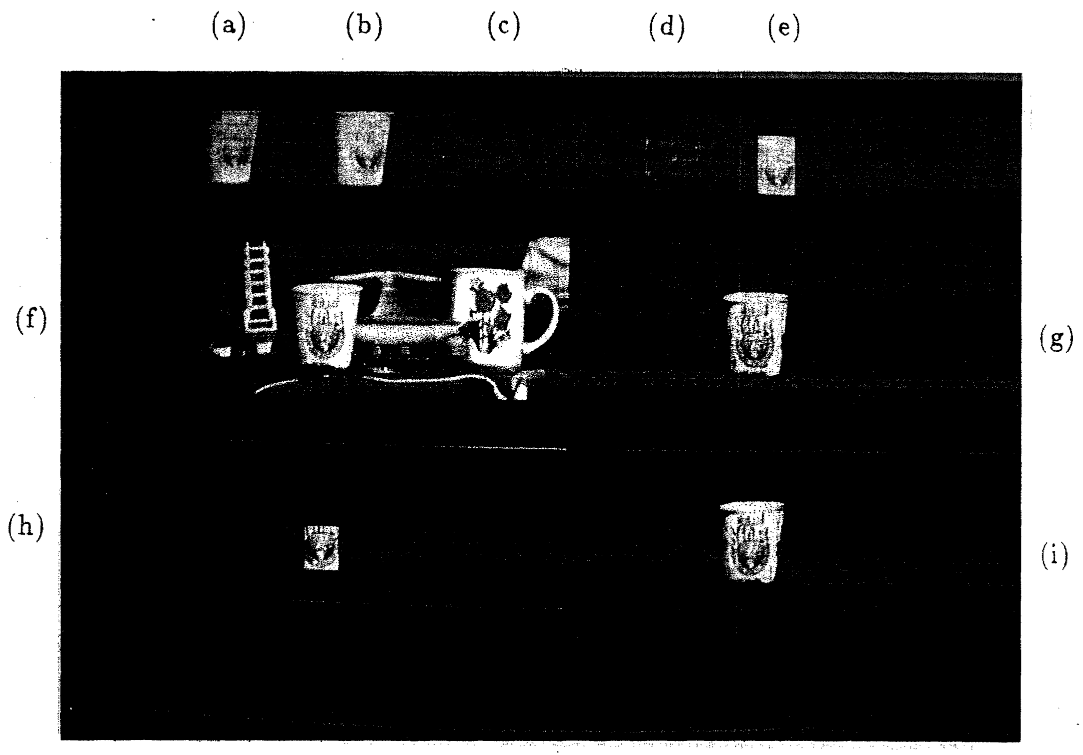


Figure 8.8: *Illustration of attentional selection using all three cues of color, texture, and line groups. (a)-(b) Two views of an object used as a model. (c) The color regions extracted to constitute the color description of the model of (a). (d) The color RAG of the model used for color-based selection. (e) An extract of the model view shown in (a) used for building model texture description. (f) A scene in which the model object occurs. (g)- (h) The result of selection using the individual cues of color and texture. (i) Spatially overlapping regions selected based on color and texture.*

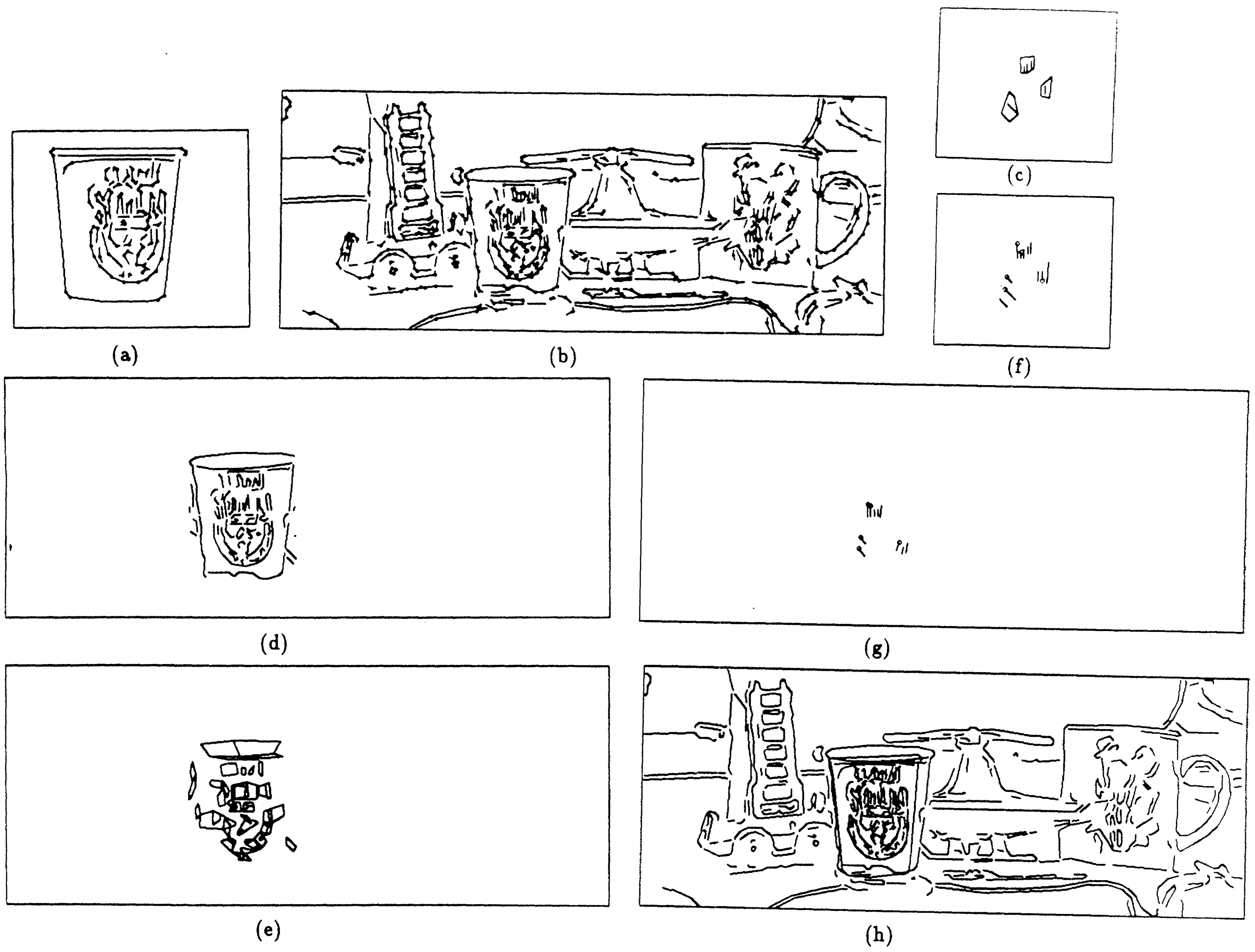


Figure 8.9: *Illustration of recognition within the regions selected by the attention mechanism using all the three cues of color, texture and line groups. (a) The primary view of the model object showing corner (in circles) and line features. (b) An edge image of a scene showing corner (again in circles) and line segments. The model object and the scene are those depicted in Figure 8.8 (c) Parallel-line groups retained in the model description.(d) The edges in the selected region of Figure 8.8i within which the occurrence of matching line groups was noted. (e) The line groups that matched to one or more of the model line groups of (c). (f) - (g) A set of matching image and model line groups . The corresponding corner features that were declared to be a match using interpretation tree search in the matching stage of the recognition system are shown in circles. (h) The model object projected into the image using the alignment transform computed from the set of matched features shown in (f) and (g). The projected view was verified to be correct by the verification stage of the recognition system.*

S.No	M	N	M_s	M_g	Avg. k_i	Avg.	Avg.	Num. Matches		
					per group	m_i	n_j	No Selection	Estimated	Explored
1.	114	484	83	8	9	2	3	1.55 x 10^{12}	1.04 x 10^{11}	1.39 x 10^{3}
2.	114	580	112	8	10	2	3	5.52 x 10^{33}	1.43 x 10^{11}	2.23 x 10^{2}
3.	114	1401	105	8	6	2	2	2.65 x 10^{36}	1.81 x 10^{9}	4.32 x 10^{3}
4.	96	580	108	3	10	3	2	1.66 x 10^{33}	7.55 x 10^{9}	3.31 x 10^{2}
5.	96	1401	101	3	13	3	2	7.96 x 10^{35}	1.66 x 10^{10}	3.45 x 10^{3}
6.	96	256	71	3	8	3	3	5.41 x 10^{30}	6.61 x 10^{10}	2.43 x 10^{3}
7.	138	392	67	5	10	3	4	1.35 x 10^{33}	4.47 x 10^{12}	1.45 x 10^{4}

Table 8.4: *Estimated and actual search reduction during recognition with attentional selection using all three cues of color, texture and parallel-line groups. Here M,N are the corner features in model and image, and M_s is the number of features in the region selected using all the three cues.*

8.2.6 Nature of search reduction using a combination of cues

Using progressively more information about the object, we should expect to do better at localizing the object and hence avoid considering large portions of the scene. This is borne out to some extent in the experiments described above. Using one of the cues alone such as color, we saw that the search reduction was comparatively smaller. This was mainly due to the fact that even though the model object is likely to be among the selected regions, it may not be discovered in the highest-ranked selected region. Results from texture-based selection, on the other hand, indicated that the search was still considerable because texture regions tend to have more edge features. When groups are formed from these features as was done in line grouping, we see that the performance improves considerably. Using all three cues thus improved the search performance even more. But is the search necessarily smaller when a combination of all three cues are used? If the best-ranked region using a subset of the cues contains the object (or a portion of it) and is also present in the best-ranked region using a combination of all three cues, then there may be an increase in the number of matches (when compared to using a subset of cues) due to the additional features in the region selected using the combination of cues.

8.2.7 False positives and negatives using attentional selection

So far, we saw that attentional selection using one or more cues does improve the performance of the recognition system in terms of the number of matches to be explored. We now investigate the chance of additional false positives or negatives made by the recognition system that can be attributed to the selection mechanism. That is, we would like to know if the recognition system using selection actually made more mistakes in terms of declaring an object present when it is not, and declaring an object absent when it is in fact present in the given scene. One way to do this would be to run the recognition system on sample scenes, and record the number of cases of false positives or negatives that occur using the selection mechanism that would not have occurred when the recognition system was run without this mechanism. For the complex scenes on which the integrated recognition system was tested, it was not possible to run it without selection in any reasonable amount of time to record such false positives and negatives. We analyze here instead, the possible cause of such occurrences and argue that the chance of making a false positive or negative due to selection is rare in practice.

A recognition system that makes false positives using selection would also have done so without it, because features in the selected region would also have been examined for finding matches [3]. If the selection mechanism merely ranked all regions rather than selecting only a few of them, then the recognition system would not make additional false negatives either. If, however, only a few best matching regions are given as a result of selection, then false negatives could occur in two cases: (1) when the selection mechanism either gives no region at all or the given region is completely off the mark, i.e. indicates the object at a wrong location, and (2) when the selected region does contain the model object, but not in the way implied by the selection

[3]This assumes that the recognition system looks for all instances of an object in a scene and does not stop when once an instance of the object is found.

mechanism, i.e., the correspondence using color regions or line groups implied by the selection mechanism is incorrect.

The selection mechanism fails to locate the object in cases of severe occlusions or pose and scale changes that are beyond the bounds tolerated in the individual selection based on color, texture, and line groups. For example, when the current view of the object in the scene cannot be captured as part of the given model color view, or in cases of severe occlusions, the subgraph matching algorithm in color-based selection may fail to identify the region containing the object. Similarly, for objects whose texture information cannot be described by an AR model, then texture-based selection using the method of LP spectrum may not yield a correct match. Also, for pose and scale changes that are beyond the bounds tolerated in the model description, the overlapping window analysis will fail to capture a region containing only the model texture and hence fail to locate the object correctly. Finally, when the changes undergone by the object in its current instance in the image are such that its parallel-line groups are no longer spaced within the bounds allowed, or when excessive splitting of the groups occurs during model-driven grouping, then again selection may fail to locate the object correctly. The latter case can happen when the nearby objects present distracting features that have the same orientation as the model line groups.

The second case of false negatives occur when the object is correctly localized but the internal correspondences implied by the selection mechanism are incorrect. This can happen in color-based selection, for example, when the color description of the object has several identical colored-regions, and the spatial adjacency constraints are not sufficient to resolve between two matching subgraphs that assign an image color region to different model regions of identical color. Figure 8.10a shows a hypothetical model color region arrangement and the associated color RAG is shown in Figure 8.10b. A collection of regions in a hypothetical scene that contained the model regions but in a different pose is indicated in Figure 8.10c. Two subgraphs that match the two regions differently and yet satisfy the same RAG are shown in Figure 8.10d and e. A recognition system that uses the wrong assignment such as the correspondence implied by Figure 8.10e may again make a false negative identification.

8.3 Conclusions

In this chapter we have shown the use of attentional selection in object recognition. The combining of cues had led to a powerful way of ordering the regions in an image in terms of their likelihood of coming from the model object. Further, using the information derived from such a selection in generating the possible matches during recognition and during verification, we saw that the number of matches to be explored in practice, reduces considerably. Since the worst-case complexity is not affected by the selection mechanism, there could still be some cases of data orderings that may cause the recognition system to take a very long time to recognize an object, but these results say that the average performance is improved a great deal using the selection mechanism. Finally, we argued that this search reduction using selection can still be obtained without much increase in the chance of false positives and negatives.

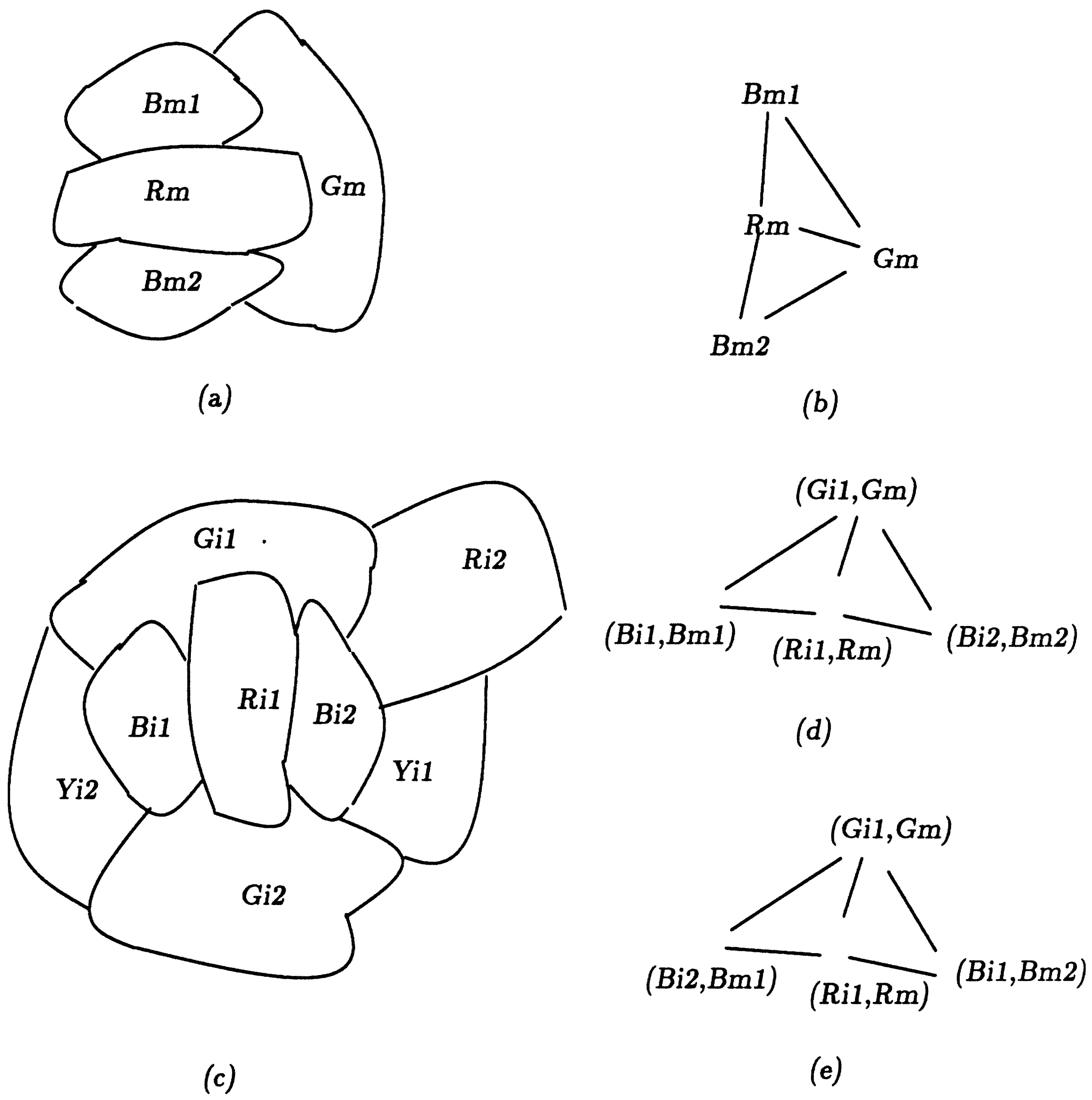


Figure 8.10: *(a) A hypothetical set of color regions on a model object. (b) Their color RAG description. (c) A hypothetical arrangement of color regions in an image. (d)-(e) Two subgraphs showing different pairings of model and image color regions that satisfy the model RAG of (a).*

Chapter 9

Conclusions

In this thesis we have attempted to provide a framework for addressing selection, one of the key problems in object recognition, using the concept of visual attention. Specifically, two modes of attentional behavior were identified as appropriate paradigms for data and model-driven selection. In this framework, data-driven selection is achieved by selecting regions in a scene that are salient in some feature, i.e., attract our attention, while model-driven selection is achieved by paying attention to the features on the model object. To indicate how this can be achieved, a computational model of attentional selection was proposed. A restricted implementation of the model led to new ways of performing data and model-driven selection in recognition using the cues of color, texture, and parallel-line groups and their combination. The thesis also evaluated the attentional selection model by integrating with a recognition system and conducted studies to record any improvement in performance. These studies showed that using attentional selection, the search for matching features can be reduced considerably without causing additional false positives and negatives. Also, by combining cues as done in the selection mechanism, it was found that objects can be recognized even under very cluttered conditions such as in the scene of Figure 8.4. This demonstrated the usefulness of such a selection mechanism for object recognition.

Some of the new ideas developed in this thesis include the concept of perceptual categorization of color spaces to provide a relatively stable specification of color for the extraction of color regions and for locating colored objects, and the interpretation of geometric changes to textures on objects in the domain of signal processing. Some of the specific results obtained during the processing of cues include the result that a linear transformation of texture manifests as an inverse linear transformation of the linear prediction spectrum, the derivation of a set of conditions under which overlapping window analysis can capture a region containing only the model texture irrespective of its location in a scene, the result that grouping using the closely-spaced parallelism constraint generates a linear number of small-sized and reliable groups without causing any additional false negatives, and the result that the spacing between parallel lines can only decrease under orthographic projection and orientation changes that don't involve a change in scale.

The above results formed the basis of data and model-driven selection using each of the three cues. Data-driven selection was achieved using all these cues by developing saliency measures for capturing the distinctiveness and reliability of regions extracted using the respective cues. Model-driven selection was performed differently for each cue but again had a common theme of having a model description based on a cue and a strategy for locating the object based on this description. In each of these approaches, however, selection was achieved at the cost of making some assumption about the imaging conditions, especially, regarding the scale changes allowed. An extended implementation of the attentional selection model using other cues could deal, perhaps, with the issue of scale.

Since selection was meant to be a pre-processing step for recognition, there was an effort throughout to keep the algorithms for performing selection simple as seen in the linear time algorithms for color region segmentation and line group generation.

The integration of attentional selection with a recognition system showed the importance of having a richer description of the model object. That is, describing the model object in terms of its color, texture regions and line groups can not only help in roughly locating the object but can also be useful information to be associated with the actual data features used for recognition.

The integrated recognition system developed in the thesis used the regions selected in the pay-attention mode of the attentional selection model. The regions selected in the attract-attention mode of the model can also be useful. As they are fairly large regions containing many features, they could be useful for indexing into a library of models [27]. Thus saliency-driven selection presented in this thesis could lead to new ways of organizing information about the objects in a model library for indexing, making it an appropriate topic for future investigation. Also, the attentional selection model suggested here is fairly generic so that it could possibly be used for other tasks that do not necessarily involve recognition. Thus using this selection mechanism, a robot navigating in an unfamiliar environment can be designed to quickly notice some distinguishing feature in a scene and react, another topic for future consideration. Finally, the individual modules of the selection mechanism themselves such as the color and texture segmentation algorithms can be utilized in other visual tasks that require the extraction of such regions.

Although models of attention have been proposed before, the computational model of attentional selection proposed in this thesis differs from these in being backed by an implementation that can actually demonstrate such attentional selection, and in addressing the question of evaluation of such a model. The evaluation was, however, done only for the pay-attention mode. Although the implementation of the attract-attention mode tried to capture some of the often-noticed features of regions, the weighting functions used for recording their saliency seem rather ad hoc. A less ad hoc method would have been to develop such functions using carefully designed psychophysical experiments. Perhaps, the realization of the role of various attributes of regions in determining their saliency is what is important in this work. Better saliency measures if designed in future can still be consistent with the architecture for information flow suggested in the model by serving as different strategies for driving the selection filters of the model. Finally, it will be interesting to see if the three-stage processing namely, feature detection, individual saliency computation, and overall saliency detection suggested in this and other computational model of attentional selection will bear out in physiological studies of the future.

Appendix A

Color Categorization Experiments

In this appendix we describe the psychophysical experiments done to derive the color categories. The aim of these experiments was to record the perceptual judgements of colors in different regions of the color space by a systematic exploration of the color space. For this, the hue-saturation-value representation of color space was used. As shown in Figure A.1, the entire spectrum of computer recordable colors (2^{24} colors) was quantized into 7200 bins corresponding to a 5 degree resolution in hue, and 10 levels of quantization of saturation and intensity values. In order to scan the color space systematically, the colors in bins were observed starting with the bins of red hue and going around the color space back to the red hue again. The display set up involved a 24-bit high resolution monitor with appropriate monitor calibration to observe the colors in dark room conditions with a minimum viewing distance of 2 feet. Uniform color samples (mondrians) of size 64 x 64, corresponding to the hue-saturation-and brightness value in each bin were displayed on the screen. The set of mondrians displayed on the screen varied in purity vertically, and intensity horizontally, while the hue was kept constant. For each hue the colors initially displayed had a resolution of 0.2 in brightness and saturation. Four subjects were

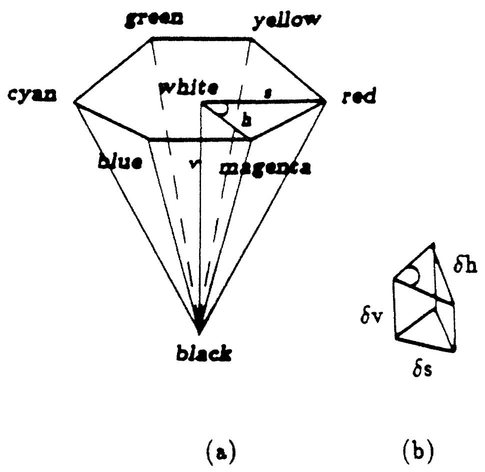


Figure A.1: *Illustration of the quantization of the hsv-color-space. (a) hsv-color model. (b) a cell of the quantized color space using the categorization data shown in Table 3.1.*

tested individually and were supplied with a chart that showed the gradations in brightness and purity varying in a manner that corresponded to the color spectrum shown on the display. Each subject was then asked to group the color samples displayed on the screen into perceptually uniform color groups and mark the result on the chart provided, so that the end result was a segmentation of the chart into perceptually uniform colored groups. The presence of a boundary was taken to mark a change in color category. To precisely locate this boundary, the color samples around the boundary were redisplayed with a finer resolution (of 0.1) in brightness and saturation. Before assigning a new category label each group is compared with groups of previous hue by displaying the colors in the previous group along with a given group and asking the subject to judge if this group could be merged with the previous hue groups. The observation of successive mondrians was done with a 10 minute intervals in between to remove after-effects of the previous display. The mondrians displayed were sufficiently apart on the screen to keep the effects of simultaneous contrast small. By averaging out the differences in the responses between subjects, we found about 220 different color categories were sufficient to describe the color space. The color category information was then summarized in a color-look-up table.

Appendix B

Experiments for Choosing the Weighting Functions

B.1 Choosing Weighting Functions for Color Saliency

We describe here experiments done to arrive at the choice of weighting functions for absolute and relative size factors that contribute to the saliency of color regions. The general strategy was to record the distribution of regions declared distinctive by observers as a function of the absolute and relative sizes. The exact form of the weighting functions was then derived by approximating the resulting curves by smooth logarithmic functions. The rationale behind this strategy was that weighting functions derived from such curves can help emphasize regions in natural scenes that would also be perhaps noticed by observers looking at an image of the same scene. The experiments performed to arrive at such distributions are described below.

Mondrians of colored regions of random shapes were displayed on a screen in a darkened room and observers seated at a distance of two feet from the screen were asked to point to a region they found most distinctive in the display. The dark room conditions removed the effects of illumination artifacts (such as specularities and inter-reflections). The colors and areas of the regions in the mondrians were varied to generate a set of test mondrians. Figure 3.5 of Chapter 3 shows sample mondrians. The regions in a mondrian image of dimensions M x N were generated by choosing a horizontal extent M_i for each row i, that varied randomly between 0 and N. As many regions as could fit in the chosen dimensions were generated using the above method. The number of regions, therefore, varied among mondrians. But this method ensured a good distribution of regions of different sizes among the mondrians. By randomly varying the colors of the regions of a given mondrian, a different arrangement of color regions could be easily obtained.

Each mondrian was observed by five subjects for a duration of 30 seconds. Successive displays for a subject were spaced at least 5 minutes apart. Each subject was tested on about 600 mondrians over a period of three months. The distinctive regions chosen by subjects varied sometimes in a given display. The distinctiveness of a region was indicated by simply pointing to that region. No control of the fixation point was done, nor were the eye movements recorded. Thus we could not be sure whether the regions were pre-attentively perceived. The aim of these informal experiments was to record some general properties of such distinctive regions. The areas of the regions were divided into 20 bins of range 0.05% of the image size. The number of distinctive regions chosen by subjects across the displays that had an absolute size in each range was noted and a histogram plotted. Logarithmic functions were fit to the histograms over the smoothly varying portions to obtain the weighting function for absolute size mentioned in equation 3.11 of Chapter 3 and repeated below:

$$f_3(n) = \begin{cases} -\frac{ln(1-n)}{c_1} & 0 \le n \le t_1 \\ 1 - e^{-c_2 n} & t_1 < n \le t_2 \\ s_2 - c_3 ln(1 - n + t_2) & t_2 < n \le t_3 \\ s_3 e^{-c_4(n-t_3)} & t_3 < n \le t_4 \\ 0 & t_4 < n \le 1.0 \end{cases} \tag{B.1}$$

where $t_1 = 0.1$, $t_2 = 0.4$, $t_3 = 0.5$, $t_4 = 0.75$, $s_1 = 0.8$, $s_2 = 1.0$, $s_3 = 0.7$, $s_4 = 10^{-3}$ and $c_1 = -\frac{ln(1-t_1)}{s_1}, c_2 = -\frac{ln(1-s_1)}{t_1}, c_3 = -\frac{(s_2-s_3)}{ln(1+t_2-t_3)}, c_4 = -\frac{ln\frac{s_4}{s_3}}{(t_4-t_3)}$ and n = size of region R = r(R).

Studies on the effect of the size of a region on its conspicuity were done earlier

[47], typically by displaying a bright circle of increasing diameter against a dark background of randomly distributed bright circles, and performing some careful psychophysical experiments for estimating the visual conspicuity area. Our experiments, though informal, were performed on displays that reflected regions in images of natural scenes as they were allowed to be of arbitrary shapes and were contiguously placed as in a natural scene.

A way of obtaining the best neighbor for each color region in an image was indicated in Chapter 3. This was used to obtain the best neighbor for each distinctive region chosen by subjects. A histogram of the number of distinctive regions with a given relative size was similarly plotted. Smoothly varying logarithmic function was again chosen to approximate this histogram to given the weighting function for relative size mentioned in Chapter 3 as $f_5(n) = 1 - e^{-12n}$.

B.2 Experiments for Choosing the Weighting Functions for Texture Saliency

The experiments performed to obtain the weighting functions for texture saliency were similar to those for color saliency. Thus here again, properties of distinctive regions chosen by subjects were recorded. The difference lay in the displays generated and the properties studied. Mosaics of textured regions consisting of several cloth textures (Figure 4.25), and textures from the Brodatz album [16](Figure 4.26)were generated. The experimental set up was similar to the one for recording color saliency in mondrians and here again subjects were asked to point to distinctive regions. Then the analysis of the various texture regions was done as described in Section 4.3.1. Since we generated mosaics of textures with well-defined boundaries, no texture segmentation was attempted. Instead the four binary maps were generated for each of the mosaics and the dark and bright regions inside each texture region were separately analyzed. The number, area, shape and distribution of holes in the distinctive texture regions chosen by subjects were noted. Histograms of their values were plotted and smooth logarithmic functions were again fit to the resulting curves to obtain the weighting functions for shape and distribution of holes mentioned in Chapter 4.

Bibliography

[1] D.D. Astous and M.E. Jernighan. Texture discrimination based on detailed measures of the power spectrum. In *Proceedings Int. Conf. on Pattern Recognition*, pages 83–86, 1984.

[2] N. Ayache and O.D. Faugeras. Hyper: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):44–54, 1986.

[3] R. Bajcsy. Computer description of textured surfaces. In *Proceedings Int. Joint. Conf. Artif. Intell.*, 1973.

[4] R. Bajcsy and L. Lieberman. Texture gradient as a depth cue. *Computer Graphics and Image Processing*, 5:52–67, 1976.

[5] D.H. Ballard. Generalizing the Hough transform to detect arbitrary patterns. *Pattern Recognition*, 13(2):111–122, 1981.

[6] H. Barrow and J.M. Tenenbaum. Recovering intrinsic scale characteristics from images. In A. Hanson and E. Riseman, editors, *Computer Vision Systems.* New York: Academic Press, 1978.

[7] R. Basri. *The recognition of 3-D solid objects from 2-D images.* PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1990.

[8] R.C. Bolles and R.A. Cain. Recognizing and locating partially visible objects: The local feature-focus method. *International Journal of Robotics Research*, 1(3):57–82, 1982.

[9] R.C. Bolles and P. Horaud. 3DPO: A three dimensional part orientation system. *International Journal of Robotics Research*, 5(3):3–86, 1986.

[10] M.H. Bornstein, W. Kessen, and S. Weiskopf. Color vision and hue categorization in young human infants. *Jl. of Exper. Psych. : Human Percep. and Performance*, 2(1):115–129, 1976.

[11] C. Bouman and B. Liu. Segmentation of textured images using a multiple resolution approach. In *Proc. Int. Conf. Acoust. Speech and Signal Process.*, pages 1124–1127, 1990.

[12] C. Bouman and B. Liu. Unsupervised estimation of image textures using an autoregressive model. In *Proc. Int. Conf. Acoust. Speech and Signal Process.*, 1990.

[13] A.C. Bovik et al. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–72, 1990.

[14] D. Broadbent. The role of auditory localization and attention in memory span. *Jl. of Exper. Psych.*, 47:191–196, 1954.

[15] D. Broadbent. *Perception and Communication.* London: Pergamon Press, 1958.

[16] P. Brodatz. *Textures: A Photographic Album for Artists and Designers.* New York: Reinhold, 1968.

[17] R. Brooks. Symbolic reasoning among 3d models and 2d images. *Artificial Intelligence*, 17:285–348, 1981.

[18] C. Bushnell et al. Behavioral enhancement of visual responses in monkey cerebral cortex i. modulation in posterior parietal cortex related to selective visual attention. *J. Neurophysiology*, 46(4):755–772, 1981.

[19] E. Carterette and M. Friedman. *Perceptual Coding.* New York: Academic Press, 1978.

[20] S. Chatterjee and R. Chellappa. Maximum likelihood texture segmentation using Gaussian markov random field models. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 1985.

[21] R. Chellappa and R. Kashyap. Texture synthesis using 2d non-causal autoregressive models. *IEEE Transactions on Acoustics Speech and Signal Processing*, 33(1):194–203, 1985.

[22] P.C. Chen and T. Pavlidis. Segmentation by texture using correlation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):64–69, 1983.

[23] A.P. Choo et al. Image segmentation for complex natural scenes. *Image and Vision Computing*, 8(2):155–163, 1990.

[24] J.C. Clark and N. Ferrier. Modal control of an attentive vision system. In *Proceedings of the International Conference on Computer Vision*, pages 514–523, 1988.

[25] D.T. Clemens. *Region-based feature interpretation for recognizing 3D models in 2D images.* PhD thesis, Artificial Intelligence Lab, M.I.T., AI-TR-1307, 1991.

[26] D.T. Clemens and D. Jacobs. Model-group indexing for recognition. In *DARPA IU Workshop*, pages 604–613, 1990.

[27] D.T. Clemens and D.W. Jacobs. Space and time bounds on indexing 3d models from 2d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:1007–1017, 1991.

[28] G.B. Coleman and H.C. Andrews. Image segmentation by clustering. *Proceedings IEEE*, 67(5):773–785, 1979.

[29] T.H. Cormen, C.E. Lieserson, and R.L. Rivest. *Introduction to Algorithms.* New York: McGraw Hill, Cambridge: MIT Press, 1990.

[30] G.R. Cross and A.K. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):25–39, 1983.

[31] S.R. Curtis, A.V. Oppenheim, and J.S. Lim. Signal reconstruction from Fourier transform sign information. *IEEE Transactions on Acoustics Speech and Signal Processing*, 33:643–657, 1985.

[32] L.S. Davis. Hierarchical generalized Hough transform and line segment-based generalized Hough transforms. *Pattern Recognition*, 15:277, 1982.

[33] K. Deguchi and I. Morishita. Texture characterization and texture-based image partitioning using two-dimensional linear estimation techniques. *IEEE Trans. Comput.*, 27:739–745, 1978.

[34] R. Deriche and G. Giraudon. Accurate corner detection : An analytical study. In *Proceedings of the Int. Conf. on Computer Vision*, pages 66–70, 1990.

[35] H. Derin and W.S. Cole. Segmentation of texture images using Gibbs random fields. *Computer Vision, Graphics, and Image Processing*, 35:72–98, 1986.

[36] J. Deutsch and D. Deutsch. Attention: Some theoretical considerations. *Psychological Review*, 70:80–90, 1963.

[37] M. Dhome and T. Kasvand. Polyhedra recognition by hypothesis accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):429–438, 1987.

[38] J. Dolan and R. Weiss. Perceptual grouping of curved lines. In *Proc. DARPA IU Workshop*, pages 1135–1145, 1989.

[39] D.Sagi and B. Julesz. "Where" and "What" in vision. *Science*, 228:1217–1219, 1985.

[40] R.O. Duda and P.E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of ACM*, 15(1):11–15, 1972.

[41] D.E. Dudgeon and R.M. Mersereau. *Multidimensional signal processing.* Englewood Cliffs, NJ: Prentice Hall, 1984.

[42] J. Duncan. Directing attention in visual field. *Perception and Psychophysics*, 30:90–93, 1981.

[43] C.R. Dyer and A. Rosenfield. Fourier texture features: Suppression of aperture effects. *IEEE Transactions on Systems, Man and Cybernetics*, 6:703–705, 1976.

[44] E. Egeth. Selective attention. *Psychological Bulletin*, 67:41–57, 1967.

[45] J.O. Eklundh. On the use of Fourier phase features for texture discrimination. *Computer Graphics and Image Processing*, 9:199–201, 1979.

[46] E.Land and J.J. McCann. Lightness and retinex theory. *Journal of the Optical Society of America*, 61:1–11, 1971.

[47] F.L. Engel. *Visual Conspicuity as an External Determinant of Eye Movements and Selective Attention.* T.H. Eindhoven, 1976.

[48] M.G. Engel. Visual conspicuity and selective background interference in eccentric vision. *Vision Research*, 14:459–471, 1974.

[49] R.M. Evans. *The Perception of Color.* New York: John Wiley and Sons, 1974.

[50] O.D. Faugeras and W.K. Pratt. Decorrelation methods for texture feature extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):323–332, 1980.

[51] B. Fischer and B. Breitmeyer. Mechanisms of visual attention revealed by saccadic eye movements. *Neuropsychologia*, 25:73–83, 1987.

[52] J.D. Foley and A. Van Dam. *Fundamentals of Interactive Computer Graphics.* Reading: Addison Wesley, 1984.

[53] B. Funt and J. Ho. Color from black and white. In *Proceedings of the International Conference on Computer Vision*, pages 2–8, 1988.

[54] E. Gamble and T. Poggio. Integration of intensity edges with stereo and motion. Technical report, Artificial Intelligence Lab, M.I.T., AI-Memo-970, 1987.

[55] R. Gershon, A. D. Jepson, and J.K.Tsotsos. From R,G,B to surface reflectance: Computing color constancy descriptors in images. In *Proceedings Int. Joint. Conf. Artif. Intell.*, pages 755–758, 1987.

[56] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Baltimore, MD: The Johns Hopkins University Press, 1983.

[57] R.C. Gonzalez and P. Wintz. *Digital Image Processing.* Reading: Addison-Wesley, 1987.

[58] W.E.L. Grimson. The combinatorics of object recognition in cluttered environments using constrained search. In *Proceedings of the International Conference on Computer Vision*, 1988.

[59] W.E.L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* Cambridge: MIT Press, 1990.

[60] W.E.L. Grimson and T. Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3(3):3–35, 1984.

[61] W.E.L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, July 1987.

[62] P.E. Haenny et al. Cells in prenulate cortex alter response to visual stimuli of different behavioural significance. *Perception*, 13:A12, 1984.

[63] R.M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[64] Heitz et al. Application of autoregressive models to fine arts painting analysis. *Signal Processing*, 13:1–14, 1987.

[65] J.E. Hochberg. Effects of the gestalt revolution: The Cornell symposium on perception. *Psychological Review*, 64(2):73–84, 1957.

[66] B.K.P. Horn. *Robot Vision.* Cambridge: MIT Press, 1986.

[67] G.W. Humphreys and V. Bruce. *Visual Cognition: Computational, Experimental and Neuropsychological Perspectives.* Hillsdale: Lawrence Erlbaum Assoc., 1989.

[68] A. Hurlbert. *The computation of color.* PhD thesis, Artificial Intelligence Lab, M.I.T., TR-1154, 1989.

[69] A. Hurlbert and T. Poggio. Spotlight on attention. Technical report, Artificial Intelligence Lab, M.I.T., AI-Memo-817, April 1985.

[70] A. Hurlbert and T. Poggio. Visual attention in brains and computers. Technical report, Artificial Intelligence Lab, M.I.T., AI-Memo-915, June 1986.

[71] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proceedings of the International Conference on Computer Vision*, pages 102–111, 1987.

[72] D. P. Huttenlocher and P.C. Wayner. Finding convex edge groupings in an image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 406–412, 1991.

[73] D.P. Huttenlocher. *Three-dimensional recognition of solid objects from a two-dimensional image.* PhD thesis, Artificial Intelligence Lab, M.I.T., AI-TR-1045, 1988.

[74] R.J. Marks II. Restoring lost samples from an oversampled bandlimited signal. *IEEE Transactions on Acoustics Speech and Signal Processing*, 31:752–755, June 1983.

[75] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, 44:87–116, 1988.

[76] D. Jacobs. The use of grouping in visual object recognition. Master's thesis, Artificial Intelligence Lab, M.I.T., AI-TR-1023, 1988.

[77] H.H. Jordan and F.M. Porter. *Descriptive Geometry.* Boston: Ginn and Company, 1929.

[78] B. Julesz. Experiments in the visual perception of texture. *Scientific American*, 232:34–43, 1975.

[79] B. Julesz and J.R. Bergen. Textons, the fundamental elements in preattentive vision and perception of textures. *The Bell Systems Technical Journal*, 62(6):1619–1645, 1983.

[80] E. Kandel and J. Schwartz. *Principles of Neuroscience.* New York: Elsevier, 1985.

[81] H. Kaneko and E. Yodogawa. A markov random field application to texture classification. In *Proc. Pattern Recognition and Image Process.*, pages 221–225, 1982.

[82] B. Kartikeyan and A. Sarkar. An identification approach for 2d autoregressive models in describing textures. *CVGIP: Graphical Models and Image Processing*, 53(2):121–131, 1991.

[83] L.R. Kashyap and A. Khotanzad. A model-based method for rotation invariant texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):472–481, July 1986.

[84] R.A. Kinchla et al. Attending to different levels of structure in a visual image. *Perception and Psychophysics*, 33(1):1–10, 1983.

[85] L. Kitchen and A. Rosenfield. Gray-level corner detection. *Pattern Recognition Letters*, pages 95–102, December 1982.

[86] G.J. Klinker, S.A. Shafer, and T. Kanade. Using a color reflection model to separate highlights from object color. In *Proceedings of the International Conference on Computer Vision*, 1987.

[87] G.J. Klinker, S.A. Shafer, and T. Kanade. Using color reflection model to separate highlights from object color. In *Proceedings of the International Conference on Computer Vision*, pages 614–619, 1987.

[88] G.J. Klinker, S.A. Shafer, and T. Kanade. A physical approach to color image understanding. *International Journal of Computer Vision*, 4(1):7–38, 1990.

[89] C. Koch and S. Ullman. Selecting one among the many: A simple network implementing shifts in selective visual attention. Technical report, Artificial Intelligence Lab, M.I.T., AI-Memo-770, January 1984.

[90] Y. Lamdan and H.J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Proceedings of the International Conference on Computer Vision*, pages 218–249, 1988.

[91] E. Land. Recent advances in retinex theory. In T. Ottoson and S. Zeki, editors, *Central and Peripheral Mechanisms of Color Vision*, pages 5–17. London: McMillan, 1985.

[92] Y.G. LeClerc. Region grouping using the minimum description length principle. In *Proc. DARPA IU Workshop*, pages 473–481, 1990.

[93] J.S. Lim. Image restoration by short space spectral subtraction. *IEEE Transactions on Acoustics Speech and Signal Processing*, 28:191–197, April 1980.

[94] S-S. Liu and M.E. Jernighan. Texture analysis and discrimination in additive noise. *Computer Vision, Graphics, and Image Processing*, 49:52–67, 1990.

[95] D. G. Lowe. *Perceptual Organization and Visual Recognition.* Boston: Kluwer Academic, 1985.

[96] S.A. Johns L.S. Davis and J.K. Aggrawal. Texture analysis using generalized co-occurrence matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(3):251–259, 1979.

[97] J. Mahoney. Image chunking: Defining spatial building blocks for scene analysis. Technical report, Artificial Intelligence Lab, M.I.T., TR-980, 1986.

[98] J. Makhoul. Linear prediction : A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

[99] J. Malik and P. Perona. A computational model of texture segmentation. In *Proceedings of the International Conference on Computer Vision*, pages 326–332, 1989.

[100] S.G. Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Transactions on Acoustics Speech and Signal Processing*, 37(12):2091–2110, 1989.

[101] L.T. Maloney and B. Wandel. Color constancy: A method for recovering surface spectral reflectance. *Journal of the Optical Society of America*, 3:29–33, 1986.

[102] D. Marr. *Vision*. San Francisco: W.H. Freeman and Co., 1982.

[103] McCormick and N. Jayaramamurthy. Time series model for texture synthesis. *Int. Jl. Comput. Inform. Sci.*, 3(4):329–343, 1974.

[104] G.W. Meyer and D.P. Greenberg. Perceptual color spaces for computer graphics. *Computer Graphics*, 14(3):254–261, 1980.

[105] R. Mohan and R. Nevatia. Perceptual organization for segmentation and description. In *Proc. DARPA IU Workshop*, pages 415–424, 1989.

[106] J. Moran and R. Desimone. Selective attention gates visual processing in the extra striate cortex. *Science*, 229:782–785, August 1985.

[107] K. Nakayama and G. Silverman. Serial and parallel processing of visual feature conjunctions. *Nature*, 320:264–265, 1986.

[108] W.T. Neill. Decision processes in selective attention: Response priming in the stroop colour-word task. *Perception and Psychophysics*, 23:80–84, 1978.

[109] Y. Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. *Computer Graphics and Image Processing*, 13:221–231, 1980.

[110] E. Oja et al. Regularity measurement, classification, and segmentation of textures. In J.C. Simon, editor, *From Pixels to Features*, pages 207–218. Elsevier Science Publishers, North Holland, 1989.

[111] A.V. Oppenheim and R.W. Shafer. *Digital Signal Processing.* Englewood Cliffs, N.J.: Prentice Hall, 1975.

[112] T. Pavlidis. *Algorithms for Graphics and Image Processing.* Rockville: Computer Science Press, 1982.

[113] T. Pavlidis and S.L. Tanimoto. Texture identification by a directed split and merge procedure. In *Proceedings Conference Comput. Graphics, Pattern Recognition and Data Structure*, 1975.

[114] A. Perry and D. Lowe. Segmentation of textured images. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 319–325, 1989.

[115] T. Poggio et al. Mit Vision Machine. In *DARPA IU Workshop*, pages 177–198, 1988.

[116] T. Poggio, E. Gamble, and J. Little. Parallel integration of vision modules. *Science*, 242:436–440, 1988.

[117] M.J. Posner and S.J. Boies. Components of attention. *Psychological Review*, 78(5):391–408, 1972.

[118] F.P. Preparata and M.I. Shamos. *Computational Geometry.* New York: Springer Verlag, 1985.

[119] H.M. Raafat and A.K.C. Wong. A texture information-directed region growing algorithm for image segmentation and region classification. *Computer Vision, Graphics, and Image Processing*, 43:1–21, 1988.

[120] L.R. Rabiner and R.W. Shafer. *Digital Processing of Speech Signals.* Englewood Cliffs, N.J.: Prentice Hall, 1978.

[121] S. Ranganath and A. Jain. Two-dimensional linear prediction models: Part I, spectral factorization and realization. *IEEE Transactions on Acoustics Speech and Signal Processing*, 33(1):280–299, 1985.

[122] T.R. Reed and H. Wechsler. Texture analysis and clustering using the Wigner distribution. In *Proceedings Int. Conf. on Pattern Recognition*, pages 770–772, 1988.

[123] T.R. Reed and H. Wechsler. Texture segmentation and organization using the Wigner distribution. In J.L. Lacoume et al., editors, *Signal Processing IV : Theories and Applications*, pages 263–266, 1988.

[124] T.R. Reed and H. Wechsler. Segmentation of textured images and gestalt organization using spatial/spatial-frequency representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):1–12, 1990.

[125] D. Reynolds. Effects of double stimulation: Temporary inhibition of response. *Psychological Bulletin*, 62:333–347, 1964.

[126] G. Reynolds and J.R. Beveridge. Searching for geometric structure in images of natural scenes. In *DARPA IU Workshop*, pages 257–271, 1987.

[127] K. Shanmugam R.M. Haralick and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3(6), 1973.

[128] L.G. Roberts. Machine perception of three-dimensional objects. In Tippet et al., editors, *Optical and Electro-Optical Information Processing*, pages 159–197. Cambridge: MIT Press, 1966.

[129] E. Rosch. The nature of mental codes for color categories. *Jl. of Exper. Psych. : Human Percep. and Performance*, 1(4):303–322, 1975.

[130] W.S. Rutkowski and A. Rosenfield. A comparison of corner detection techniques for chain coded curves. Technical report, University of Maryland, Tech. Report No. 263, 1977.

[131] T. Sato. Effects of attention and stimulus interaction on visual responses of inferior temporal neurons in macaque. *J. Neurophysiology*, 60:344–364, 1988.

[132] B.J. Schachter et al. Some experiments in image segmentation by clustering of local feature values. *Pattern Recognition*, 11:19–28, 1979.

[133] S. Shafer. Using color to separate reflection components. *Color Research and Applications*, 10(4):210–218, 1985.

[134] A. Shashua. Correspondence and affine shape from two orthographic views: Motion and recognition. Technical report, Artificial Intelligence Lab, M.I.T., AI-Memo-1327, December 1991.

[135] A. Shashua and S. Ullman. Structural saliency : The detection of globally salient structures using a locally connected network. In *Proceedings of the International Conference on Computer Vision*, pages 321–327, 1988.

[136] A. Shashua and S. Ullman. Grouping contours by iterated pairing network. In R. Lippman, J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 335–341. San Mateo: Morgan Kaufmann Inc., 1991.

[137] P. De Souza. Texture recognition via autoregression. *Pattern Recognition*, 15(6):471–475, 1982.

[138] F. Stein and G. Medioni. Recognition of 3d objects from 2d groupings. In *Proc. DARPA IU Workshop*, pages 667–674, 1992.

[139] E. Sternheim and R. Boynton. Uniqueness of perceived hues investigated with a contiguous judgmental technique. *Jl. of Exper. Psych.*, 72:770–776, 1966.

[140] M.J. Swain and D. Ballard. Indexing via color histograms. In *Proceedings of the International Conference on Computer Vision*, pages 390–393, 1990.

[141] M.J. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[142] C.W. Therrien. An estimation-theoretic approach to terrain image segmentation. *Computer Vision, Graphics, and Image Processing*, 22:313–326, 1983.

[143] D.W. Thompson and J.L. Mundy. Three-dimensional model matching from an unconstrained viewpoint. In *Int. Conf. on Robotics and Automation*, pages 208–220, 1987.

[144] F. Tomita and S. Tsuji. *Computer Analysis of Visual Textures.* Boston: Kluwer Academic, 1990.

[145] A. Treisman. Selective attention in man. *Brit. Med. Bull.*, 20:12–16, 1964.

[146] A. Treisman. The role of attention in objet perception. In O.J. Braddick and A.C. Sleigh, editors, *Physical and Biological Processing of Images*, pages 316–325. Berlin: Springer-Verlag, 1983.

[147] A. Treisman. Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing*, 31:156–177, 1985.

[148] A. Treisman et al. Form perception and attention. In L. Spillmann and J.S. Werner, editors, *Visual Perception: The Neurophysiological Foundations*, pages 273–316. San Diego: Academic Press, 1990.

[149] A. Treisman and G. Gelade. A feature integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.

[150] A. Treisman and J. Souther. Search asymmetry : A diagnostic for preattentive processing of separable features. *Jl. of Exper. Psych. : General*, 114:285–310, 1985.

[151] M. Tsukuda and Y. Ohta. An approach to color constancy using multiple images. In *Proceedings of the International Conference on Computer Vision*, pages 385–389, 1990.

[152] A. Tuller. *A Modern Introduction to Geometries.* Princeton: D. Von Nostrand Co., 1967.

[153] S. Ullman. Visual routines. Technical report, Artificial Intelligence Lab, M.I.T., AI-Memo-723, June 1983.

[154] S. Ullman. Visual routines. *Cognition*, 18:97–159, 1984.

[155] S. Ullman. Aligning pictorial descriptions: an approach to object recognition. *Cognition*, 32:193–254, 1989. Also: in MIT AI Memo 931, Dec. 1986.

[156] S. Ullman and R. Basri. Recognition by linear combination of models. Technical report, Artificial Intelligence Lab, M.I.T., AI-Memo-1152, August 1989.

[157] S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 992–1006, October 1991.

[158] H. Voorhees. Finding texture boundaries in images. Master's thesis, Artificial Intelligence Lab, M.I.T., AI-TR-968, 1987.

[159] H. Voorhees and T. Poggio. Computing texture boundaries from images. *Nature*, 333(6171):364–367, 1988.

[160] B. Wandel. The synthesis and analysis of color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:1–13, 1987.

[161] L. Wang and D-C. He. Texture classification using texture spectrum. *Pattern Recognition*, 23(8):905–910, 1990.

[162] M. Wertheimer. Principles of perceptual organization. In D. Beardslee and M. Wertheimer, editors, *Readings in Perception*, pages 115–135. Princeton: Princeton University Press, 1958.

[163] J.S. Weszka. A comparative texture classification experiment. In *Map Data Processing*, pages 265–278. New York: Academic Press, 1980.

[164] L.E. Wickson and D. Ballard. Real-time detection of multi-colored objects. In *SPIE Sensor II: Human and Machine Strategies*, volume 1198, 1989.

[165] A.P. Witkin and J. M. Tenenbaum. On the role of structure in vision. In Beck, Hope, and A. Rosenfield, editors, *Human and Machine Vision*, pages 481–543. New York: Academic Press, 1983.

[166] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae.* New York: John Wiley and Sons, 1982.