Srikanta Patnaik, Lakhmi C. Jain, Spyros G. Tzafestas, Germano Resconi, Amit Konar (Eds.)

Innovations in Robot Mobility and Control

Srikanta Patnaik
Lakhmi C. Jain
Spyros G. Tzafestas
Germano Resconi
Amit Konar
(Eds.)

Innovations in Robot Mobility and Control

Springer

Professor Srikanta Patnaik
Department of Information
and Communication Technology
F. M. University
Vyasa Vihar
Balasore-756019
Orissa, India
E-mail: patnaik_srikanta@yahoo.co.in

Professor Germano Resconi
Department of Mathematics and Physics
Catholic University
Via Trieste 17, 25100 Brescia
Italy
E-mail: resconi@numerica.it

Professor Lakhmi C. Jain
School of Electrical & Info Engineering
University of South Australia
Knowledge-Based Intelligent Engineering
5095 Adelaide
Australia
E-mail: lakhmi.jain@unisa.edu.au

Professor Dr. Amit Konar
Department of Electronics and
Telecommunication Engineering
Artificial Intelligence Lab.
Jadavpur University
700032 Calcutta
India
E-mail: babu25@hotmail.com

Professor Dr. Spyros G. Tzafestas
Department of Electrical Engineering
Division of Computer Science
National Technical University
Zographou, 157 73 Athens
Greece
E-mail: tzafesta@softlab.ntua.gr

Library of Congress Control Number: 2005929886

ISSN print edition: 1860-949X
ISSN electronic edition: 1860-9503
ISBN-10 3-540-26892-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26892-5 Springer Berlin Heidelberg New York

Typesetting: by the authors and TechBooks using a Springer LaTeX macro package

Printed on acid-free paper SPIN: 10992388 89/TechBooks 5 4 3 2 1 0

Preface

A robot is a controlled manipulator capable of performing complex tasks and decision-making like the human beings. Mobility is an important consideration for modern robots. The book provides a clear exposition to the control and mobility aspects of modern robots.

There are good many books on mobile robots. Most of these books cover fundamental principles on motion control and path-planning using ultrasonic/ laser transducers. This book attempts to develop interesting models for vision-based map building in both indoor and outdoor environments, precise motion control, navigation in dynamic environment, and above all multi-agent cooperation of robots. The most important aspects of this book is that the principles and models introduced in the text are all field tested, and thus can readily be used in solving real world problems, such as factory automation, disposal of nuclear wastes, landmine clearing and computerized surgery.

The book consists of eight chapters. Chapter 1 provides a comprehensive presentation on multi-agent robotics. It begins with an introduction, emphasizing the importance of multi-agent robotics in autonomous sensor networks, building surveillance, transportation, underwater pollution monitoring and in rescue operation after large-scale disaster. Next the authors highlight some open-ended research problems in multi-agent robotics, including uncertainty management in distributed sensing, distributed reasoning, learning, task allocation and control, and communication overhead because of limited bandwidth of the communication channels. The design of multi-agent robotic system can be performed by both top-down and bottom-up approach. In this chapter, the authors employ the bottom-up approach that takes care of designing individual robots first, and then integrate the behavior of two or more robots to make the system amenable for real-world applications.

Chapter 1 encompasses functional architecture of the proposed multi-agent robots with special reference to information sharing, communication, synchronization and task sharing & execution by the agents. The fusion of multi-sensory data received by different agents to cooperatively use the fused information is then narrated in detail. The problems of cooperative navigation are then undertaken, and two possible approaches to solve this problem are presented.

The first approach is based on finite state automata, whereas the second approach attempts to formalize a biologically inspired model in a stochastic framework. In the latter model, the authors aim at optimizing the probability of a group of robots, starting at a given location and terminating at a given target region within a stipulated time.

The later part of the chapter presents several principles of cooperative decision-making. The principles include hybrid decision-making involving a logic-based planner and a reactive system that together can provide both short-term and long-term decisions. An alternative method concerning distributed path-planning and coordination in a multi-agent system is also presented. Examples of application in simulated rescue problem and game playing between two teams of robotic agents have also been undertaken.

The chapter ends with a discussion on emotion-based architectures of robotic agents with an ultimate aim to socialize the behavior of the agents.

Chapter 2 presents a scheme for vision-based autonomous navigation by a mobile robot. The central idea in this scheme is to recognize landmarks in the surrounding environment of the robot. Thus landmark serves as a navigational aid for the robot. After a landmark is successfully recognized, the robot approximates its current position, and derives an optimal path reaching the goal.

The chapter introduces a Selective Visual Attention Landmark Recognition (SVALR) architecture, which uses the concept of selective attention from physiological study as a means for 2-D shape landmarks recognition.

After giving a brief overview of monocular vision-based robots, the chapter emphasizes the need for two different neural networks, such as Adaptive Resonance Theory (ART) and Selective Attention Adaptive Resonance Theory (SAART) neural networks for shape recognition of objects in a given robot's world. Because of the dynamic nature of SAART, it involves massive computations for shape recognition. So, the main concept of SAART is re-engineered, and is re-named Memory Feedback Modulation (MFM) mechanism. The MFM system in association with standard image processing architecture leads to the development of SVALR architecture.

Given a topological map for self-localization, the laboratory model of the robot can autonomously navigate the environment through recognition of visual landmarks. It has also been observed that the 2-D landmark recognition scheme is free from variations in lighting conditions and background noise.

Chapter 3 presents vision-based techniques for solving some of the problems of micromanipulation. Manipulation and assembling at micro-scale is a critical issue in many engineering and biomedical applications. Unfortunately, many problems and uncertainty are encountered for design and manipulation at micro-scale. This chapter aims at characterizing the uncertainty that appears in the design of vision-based micromanipulators. In a micromanipulation system, the controlled movement of entities lies in the range of 1 micrometer to 1 millimeter.

To reduce the uncertainties in micromanipulation, the following methods are usually adopted. The environmental parameters such as humidity and temperature are to be controlled. Secondly, the precision mechanism for tools and fixtures that needs to be reconfigured for different applications should be increased. The important aspect in micromanipulation is the man-machine interface (MMI). The success of MMI depends on the understanding of the uncertainties in the complete system. The chapter addresses three major issues to reduce the scope of uncertainty in micromanipulation through appropriate visualization tools, automated visual servoing and automatic determination of system parameters.

The chapter introduces vision-based approaches to provide maximum assistance to human operators. To enhance resolution for precision, multiple views consisting of micro projective images and microscopic images together are used. These images together can provide global information about objects irrespective of limited field of view of the camera. A scheme for multiple view multiple scale visual servo is developed. The main emphasis in visual servo design is given on feature selection, correspondence finding and correction and motion estimation from images.

Chapter 4 provides an evolutionary approach to the well-known path-planning problem of mobile robots in a dynamic environment. It considers automatic sailing of a ship amidst static obstacles, such as lands and canals, and dynamic obstacles, such as other sailing ships. Like classical navigation problem, here too the authors consider a starting point and a given goal (destination) point of the ship, and the trajectory planning is performed on-line. The path-planning problem here has been formulated as a multi-criteria optimization problem that takes into account both safety of sailing (i.e. avoidance of collision) and economy of ship-motion. The overall path constructed is a sequence of linear paths, linked with each other at the turning points.

In the evolutionary planning algorithm introduced in this chapter, chromosomes are defined as a collection of genes representing the starting point, intermediate turning points and the destination point of the ship. The algorithm begins with a initialization of randomly selected paths (chromosomes), and then each path is evaluated to determine whether it is safe and economic for sailing, taking into consideration of both static and dynamic obstacles. The evaluation is done by a judiciously selected fitness function, which determines the total cost of the trajectory to maintain safe conditions and economic conditions (such as total length of sailing). Eight genetic operators have been used in the evolutionary algorithm for trajectory planning.

These are mutation (velocity selection), soft mutation (such as velocity HIGH or LOW), adding a gene, swapping gene locations, crossing, smoothing, deleting genes and individual repair. Simulation results presented at the end of the chapter demonstrate the correctness and elegance of the proposed technique.

Grippers are integral parts of a robot. Low cost robots too have grippers, but no sensors are attached to the grippers of these robots to prevent slippage. Chapter 5 provides a new direction in gripper design by attaching a slip sensor and a force sensor with the robotic gripper. A two-fingered gripper model and a simulation system is presented to demonstrate the design for complex grippers. The control of the end-effector in a two-fingered gripper system has been accomplished using a personal computer with a high-speed analogue input/output card. The simulation model for a complex gripper capable of handling load disturbances has been realized with a neuro-fuzzy controller. The main challenge of this work lies in augmentation of the neuro-fuzzy learning algorithm by reinforcement learning. It is indeed important to note that the reinforcement learning works on the basis of punishment/reward paradigm, and the employment of this algorithm has shown marked improvement in the overall performance of the gripping function. It is a well-known phenomenon that with large external (disturbing) forces acting on the object under consideration, the effector also produces high acceleration leading to slippage of the grasped object. The present work, however, has considerably eliminated the possibility of such slippage even under significant load variations.

Chapter 6 provides a new approach to model outdoor environment for navigation. While the robot is moving, the sensors attached with it acquire the information about its world. The information perceived by the sensors is subsequently used for localization, manipulation and path-planning. Sensors capable of obtaining depth information, such as scanner laser, sonars or digital cameras are generally employed for modeling *traversable regions*. Various techniques for modeling regions from outdoor scenes are prevalent. Some of these are digital elevation maps, geometric models, topological models and hybrid topo-geometric models. This chapter attempts to develop a topo-geometric type model, represented by a Voronoi diagram, based on the sensory information received from a 3-D scanner laser. The environment is thus divided into regions, clearly identifying which of these regions can be traversed by the robot.

The regions that can be traversed by the robot are defined as traversable regions. The “traversability characteristics” have been defined based on the robot and the terrain characteristics. Experimental results reveal that the proposed topo-geometric representation is good enough to model the outdoor environment in real time. A geographical positioning system (GPS), mounted on the robot can be used to integrate local models so as to augment the environmental database of a global map.

Chapter 7 addresses the problem of localization by a mobile robot in an indoor environment using only visual sensory information. Instead of attempting to build highly reliable geometric maps, emphasis is given on the construction of topological maps for their lack of sensitivity to poor odometry estimates and position errors. A method to incrementally build topological maps by a robot having a handheld panoramic camera to grab images has been developed. The robot takes snaps at various locations along its path, and augments the already developed map using the new features of the grabbed images. The methodology outlined in this chapter is very general, and does not impose any restriction on the environmental features for handling the localization problem. The feature-based localization strategies presented here are analyzed, and experimentally verified.

Precision engineering is steadily gaining momentum for increasing demands in high performance, high reliability, longer life, lower cost and miniaturization. This chapter takes into account precision motion system using Permanent Magnet Linear Motors (PMLM). The main advantage of PMLM lies in its high force density, low thermal losses, and high precision and accuracy of the system.

To improve reliability of PMLM control systems, the measurement system should yield a good resolution. Currently, laser interferometers are readily used to yield measurement resolution of 1 nanometer. The control electronics should have a high bandwidth to cope with high encoder count frequency at high speed of the motor. On the other hand, it should have a high sampling rate to circumvent anti-aliasing pits at low speed. Thirdly, the geometric imperfections of the mechanical system should be adequately accounted for in the control system to get high position accuracy. The chapter is concerned with the development of an integrated precision motion control system on an open-architecture and rapid prototyping platform. It attempts to take into account all the problems listed above.

Acknowledgments: Dr. Amit Konar, one of the Editors, gratefully acknowledges the academic support he received from the UGC-sponsored Project under *University with Potential for Excellence Program in Cognitive Scienc*e while working with this book. We are grateful to the authors and reviewers for their wonderful contribution.

Editors

Table of Contents

1 Multi-Robot Systems

Pedro U. Lima, Luis M. Custódio

Institute for Systems and Robotics, Instituto Superior Técnico,
Av. Rovisco Pais, 1,1049-001 Lisboa – Portugal
{pal, lmmc}@isr.ist.utl.pt

1.1 Introduction

Multi-robot systems (MRS) are becoming one of the most important areas of research in Robotics, due to the challenging nature of the involved research and to the multiple potential applications to areas such as autonomous sensor networks, building surveillance, transportation of large objects, air and underwater pollution monitoring, forest fire detection, transportation systems, or search and rescue after large-scale disasters. Even problems that can be handled by a single multi-skilled robot may benefit from the alternative usage of a robot team, since robustness and reliability can often be increased by combining several robots which are individually less robust and reliable [3]. One can find similar examples in human work: several people in line are able to move a bucket, from a water source to a fire, faster and with less individual effort. Also, if one or more of the individuals leaves the team, the task can still be accomplished by the remaining ones, even if slower than before. Another example is the surveillance of a large area by several people. If adequately coordinated, the team is able to perform the job faster and with reduced cost than a single person carrying out all the work, especially if the cost of moving over large distances is prohibitive. A larger rank of task domains, distributed sensing and action, and insight into social and life sciences are other advantages that can be brought by the study and use of MRS [22].

The relevance of MRS comes also from its inherent inter-disciplinarity. At the Intelligent Systems Lab of the Institute for Systems and Robotics at *Instituto Superior Técnico* (ISR/IST), we have been pursuing for several years now an approach to MRS that merges the contributions from two fields: Systems and Control Theory and Distributed Artificial Intelligence. Some of the current problems in the two areas are creating a natural trend towards joint research approaches to their solution. Distributed Artificial Intelligence focuses on multi-agent systems, either virtual (e.g., agents) or with a physical body (e.g., robots), with a special interest on organizational issues, distributed decision making and social relations. Systems and Control Theory faces the growing complexity of the actual systems to be modelled and controlled, as well as the challenges of integrating design, real-time and operation aspects of modern control systems, many of them distributed in nature (e.g., large plant process control, robots, communication networks).

Some of the most important, and specific to the area, scientific challenges one can identify in the research on MRS are, to name but the most relevant:

- The uncertainty in sensing and in the result of actions over the environment inherent to robots, posing serious challenges to the existing methodologies for Multi-Agent Systems (MAS), which rarely take uncertainty into account.
- The added complexity of the knowledge representation and reasoning, planning, task allocation, scheduling, execution control and learning problems when a distributed setup is considered, i.e., when there are multiple autonomous robots interacting in a common environment, and specially if they have to cooperate in order to achieve their common and individual goals.
- The noisy and limited bandwidth communications among teammates in a cooperative setting, a scenario which gets worse as the number of team members increase and/or whenever an opponent team using communications in the same range is present.
- The need to integrate several methodologies that handle the subsystems of each individual robot (extended to the robot team in a cooperative setting) in a consistent manner, such that the integration becomes the most important problem to be solved, ensuring a timely execution of planned tasks.

Our view of the *integration* problem for teams of cooperative robots, detailed in this chapter, is summarized in the sequel.

One of the key factors of success, for either a single robot or a robot team, lies on the capability to perceive correctly the surrounding environment, and to build models of the environment adequate for the task the robot (or the team) is in charge of, from the information provided by the sensors. Different sensors (e.g., vision, laser, sonar, encoders) can provide alternative or complementary information about the same object, or information about different objects. *Sensor fusion* is the usual designation for methods of different types to merge the data from the several sensors available and provide improved information about the environment (e.g., about the geometry, color, shape and relevance of its objects). When a team composed of several cooperating robots is concerned, the sensors are spread over the different robots, with the important advantage that the robots can move (thus moving its sensors) to actively improve the cooperative perception of the environment by the team. The information about the environment can be made available and regularly updated by different means (e.g., memory sharing, message passing, using wireless communications) to all the team robots, so as to be used by the other sub-systems.

Once the information about the world is available, one can think of using it to make the team behave *autonomously* and machine-wise *intelligently*. Three main questions arise for the team:

- Where and which *a priori* knowledge about the environment, team, tasks and goals, and perceptual information gathered from sensors, should be kept, updated and maintained? This involves the issue of *distributed knowledge representation* adequate to consistently handle different and even opposite views of the world.
- What must be done to achieve a given goal, given the constraints on time, available resources and distinct skills of the team robots? The answer to this should provide a team *plan*.
- How is the actual implementation of a plan handled, ensuring the consistency of individual and team (sub)-goals and the coordinated execution of the plan? This concerns the design of (functional, software) *architectures* suitable for the timely execution by the team of a planned task, and the introduction in such architectures of communication, information sharing and synchronization mechanisms.

Underlying the execution of a *plan* by an autonomous mobile robot is necessarily the *navigation system*. To navigate in an environment, possibly cluttered with obstacles, a mobile robot needs to know its posture (position plus orientation), either in an absolute or relative coordinate system, and when the plan establishes that it must move to a specific location, it must know how to do it (e.g., by planning an obstacle-free path or by moving towards the goal and keep avoiding the obstacles). In MRS, as will be noted below, several other challenging problems arise, related to formation control, region coverage and other issues.

The research on MRS at the Intelligent Systems Lab of ISR/IST concentrates on Cooperative Robots and follows a *bottom-up* approach to the *implementation* of a cooperative multi-robot team, starting from the development of single robot sub-systems (e.g., perception, navigation, decision-making) and moving towards behaviours involving more than one robot.

The system *design* has been following a *top-down* approach. The design phase establishes the specifications for the system:

- **qualitative specifications** concerning logical task design so as to avoid deadlocks, live-locks, unbounded resource usage and/or sharing non-sharable resources, as well as well as to execute subtasks in a sequence that does not violate the problem constraints (e.g., robot A cannot leave room B without first picking an object in that room);
- **quantitative properties** concerning performance features, such as accuracy (e.g., the spatial and temporal resolution, as well as the tolerance interval around the goal, at each abstraction level), reliability and/or minimization of task execution time given a maximum allowed cost.

Our past and current research in MRS includes topics related to the above issues, such as:

- single and multiple robot navigation;
- cooperative sensor fusion for world modelling, object recognition and tracking;
- multi-robot distributed task planning and coordination;
- cooperative reinforcement learning in cooperative and adversarial environments;
- behaviour-based architectures for real time task execution of cooperative robot tasks.

This research has been driven by applications to *soccer robots*, where the environment is fairly structured (well defined dimensions and coloured objects), and *rescue robots,* moving in an outdoors unstructured environment is considered, and requiring more complex task planning capabilities. Throughout the chapter, other examples of application to *toy problems* will also help illustrating the approaches.

The chapter organization reflects our approach to the problem and is as follows:

Section 1.2 covers architectures for MRS, both from a functional (i.e., how are behaviours and functions organized) and from a software (i.e., the mechanisms for information sharing, communications, synchronization and task execution) standpoints. The architecture developed for the SocRob project is described with some detail, as well as some recent extensions that aim at making it more general and consistently defined.

Section 1.3 concentrates on world modelling by cooperative sensor fusion. Even though most of the examples concern the cooperative localization of objects in soccer robots domain, the Bayesian approach followed is described in a general way, suitable for other applications, and taking into account some practical implementation issues.

Section 1.4 tackles different problems related to Cooperative Navigation. *Navigation controllability*, the problem of determining if a population of heterogeneous mobile robots is able to travel from an initial configuration to a target configuration in a topological map of the environment, is solved using controllability results for finite state automata. This results in a systematic way of, given a set of robots with different skills, and an environment that requires some of those skills, checking whether decisions on the distribution of the robots are feasible. *Formation feasibility* is also a methodology to check, given the kinematics of a set of heterogeneous robots and the geometric constraints imposed to the robots so that they move under a given formation, whether such a formation is feasible, further providing the feasible directions of motion for the formation. In both the above examples, a *static feasibility* problem is solved. The section ends with a biologically inspired formulation, in a stochastic framework, of the optimal control problem of moving a population of several robots from an initial region to a target region, at a given terminal time, with the goal of maximizing the probability of the robots ending in the target area, given the constraints on the robots dynamics and the environment uncertainty.

Section 1.5 describes several approaches to cooperative decision-making. One such approach is a hybrid decision system, where a logic-based planner and a reactive system concur to provide more elaborated decisions that can take into account a long-term horizon or to provide fast, short-term decisions, respectively. This way, the system can choose the best decisions, given time constraints. Another approach concerns distributed planning and coordinated task execution, where the problems to be tackled are distributed task planning and distributed task allocation in a multi-robot rescue system, assuming that teamwork (i.e., cooperative tasks) plays an important role on the overall planning system. Examples of application to a simulated rescue problem are given. Still following a logic-based approach, an implementation of a pass in robot soccer as an example of a method based on joint commitments formulation is also described. Finally, optimal decision making for a cooperative team playing against another team, based on dynamic programming applied to a stochastic discrete event model of the team behaviour, closes this section.

Section 1.6 refers to a topic where our group has been doing pioneer work: the use of the concept of Artificial Emotions as the building block for developing emotion-based agent architectures. The aim of this research is the study and development of methodologies and tools necessary to implement emotional robotic agents capable of dealing with unstructured, complex environments. Therefore, the goal is not to try optimizing some particular ability, but instead the interest is put on the general competence to learn, to adapt itself, and to survive. In order to practically test these ideas, many experimental works with simulated environments have been performed. Also tests were made with a small autonomous real robot in order to evaluate the usefulness of these ideas for robotics. Furthermore, as emotions play an important role in human social relationships, a relevant extension of this work is its application in multi-agent systems. Section 1.6 will also describe an application of the emotion-based architecture developed within our group in a multi-agent environment where interaction among the agents is vital for their survival.

We end the Chapter in **Section 1.7** with conclusions drawn from our research on MRS so far and several topics for future work that we are pursuing already or intend to pursue in the near term.

1.2 Architectures for Multi-Robot Systems

From the very beginning of our work on MRS, one main concern has been the development of behaviour coordination and modelling methods which support our integrated view to the design of a multi-robot population [50]. The literature is crowded with architectures for single and multi-robot systems, each of them with its own advantages concerning particular aspects. The original architecture considers three types of behaviours to be displayed by the team, following the concepts in [11]:

- **organizational**: those concerning the team organization, such as the roles of each player;
- **relational**: those concerning the display of relations among teammates (coordination and cooperation);
- **individual**: those concerning each robot as an individual.

Behaviours are externally displayed and emerge from the application of certain *operators*. This separation between *operators* and their resulting *behaviours* is one of the key points of our architecture. Operators implement actions that lead the robot team to display certain behaviours. In order to design operators systematically, it is sometimes relevant to distinguish what kind of behaviour they are supposed to display. A typical example are individual *vs.* relational behaviours: both are implemented by operators at the individual robot level, but relational behaviours imply the establishment of commitments among the involved robots, which in turn require implicit or explicit communication among the operators of each robot. Popular behaviour-based architectures (e.g., ALLIANCE [32]) do not make this distinction, and assume a hierarchy of operators designated there as behaviours (e.g., motivational behaviours and behaviour sets). From an *operator* standpoint, our architecture considers three levels:

- **Team Organization Level**, where, based on the current world model, a *strategy* (i.e., *what to do*) is established, including a goal for the team. This level considers issues such as modelling the opponent behaviour to plan a new strategy. Strategies may simply consist of enabling a given subset of the behaviours at each robot.
- **Behaviour or Task Coordination Level**, where switching among behaviours, both individual and relational, occurs so as to coordinate behaviour/task execution at each robot towards achieving the team goal, effectively establishing the team *tactics* (i.e., *how to do it*). Either a finite state automaton or a rule-based system can currently implement this level, but other alternative representations are possible, such as Petri nets.
- **Behaviour Execution Level**, where primitive tasks run and where they interface the sensors, through the blackboard, and the actuators, through the navigation functions at each robot. Primitive tasks are linked to each other to implement a behaviour. Currently, each behaviour is implemented as a finite state automaton whose states are the primitive tasks and transitions are associated to logical conditions on events that are detected by the system. Behaviours can be individual, if they run in one robot only, or relational, if two or more robots are running behaviours that are coordinated through commitments and synchronisation messages to achieve a common goal.

Fig. 1.1. shows the functional architecture from an operator standpoint. In a knowledge representation framework, the blackboard module is a knowledge base with all the robot's current beliefs (processed data organized in a convenient structure), goals (intentions) and commitments, represented by first order formulas. **Fig. 1.2**. zooms the Behaviour Execution Level. From the figures, it is noticeable that the organization level distributes roles (i.e., sets of allowed behaviours) per team members. The coordination level dynamically switches between behaviours, enabling one behaviour per robot at a time (similarly to [32]), but considering also relational behaviours where some sort of synchronization among the involved robots is necessary. The execution level implements behaviours by finite state machines, whose states correspond to calls to primitive tasks (i.e., actions such as kicking the ball, navigation functions and algorithms, e.g., plan a trajectory).

The functional architecture main concepts (operators/behaviours, primitive tasks, blackboard) are not much different from those present in other available architectures [32][51]. However, the whole architecture provides a complete framework able to support the design of autonomous multi-robot systems from (logical and/or quantitative) specifications at the task level. Similar concepts can be found in [18], but the emphasis there is more on the design from specifications, rather than on the architecture itself. Our architecture may not be adequate to ensure specifications concerning tightly coupled coordinated control (e.g., as those required for some types of robot formations, such as when transporting objects by a robot team), even though this class of problems can be loosely addressed by designing adequate relational behaviours.

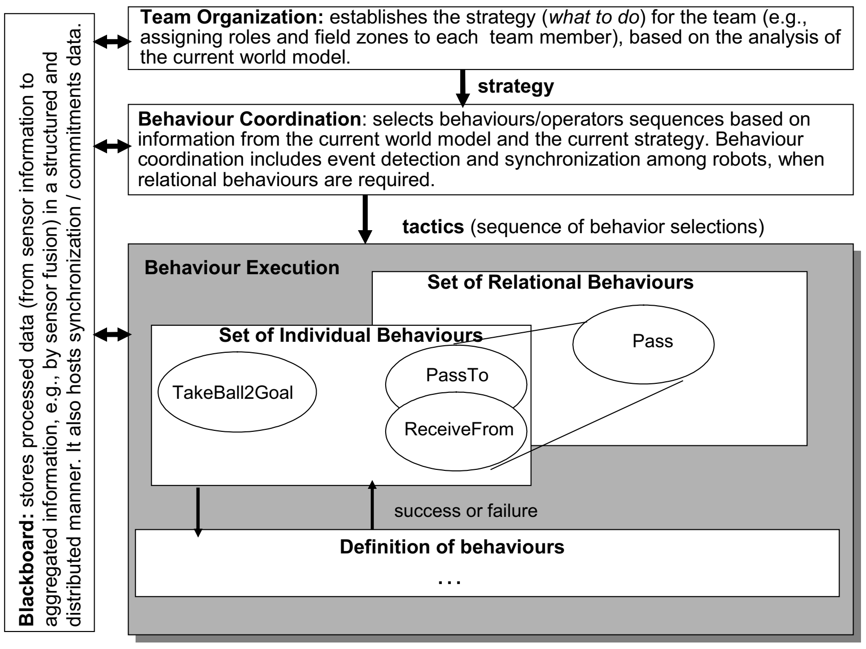

Team Organization: establishes the strategy (what to do) for the team (e.g., assigning roles and field zones to each team member), based on the analysis of the current world model.
strategy
Behaviour Coordination: selects behaviours/operators sequences based on information from the current world model and the current strategy. Behaviour coordination includes event detection and synchronization among robots, when relational behaviours are required.
tactics (sequence of behavior selections)
Behaviour Execution
Set of Relational Behaviours
Pass
Set of Individual Behaviours
TakeBall2Goal
PassTo
ReceiveFrom
success or failure
Definition of behaviours
...
Blackboard: stores processed data (from sensor information to aggregated information, e.g., by sensor fusion) in a structured and distributed manner. It also hosts synchronization / commitments data.

Fig. 1.1. Functional architecture from an operator standpoint

The software architecture developed for the soccer robots project has been defined so as to support the development of the described behavioural and functional architecture, and is based on three essential concepts: *micro-agents*, *blackboard* and *plugins*.

Each module of the software architecture was implemented by a separate process, using the parallel programming technology of threads. In this context, a module is named *micro-agent* [50]. Information sharing is accomplished by a distributed blackboard concept, a memory space shared by several threads where the information is distributed among all team members and communicated when needed.

The software architecture distinguishes also between the displayed behaviour and its corresponding implementation through an operator. Operators can be easily added, removed and replaced using the concept of *plugin*, in the sense that each new operator is added to the software architecture as a *plugin*, and therefore the *micro-agent* `control`, the one responsible for running the intended operator, can be seen as a multiplexer of *plugins*. Examples of already implemented operators are: `dribble`, `score`, or `go`, to name but a few. Each virtual vision sensor is also implemented as a *plugin*. The software architecture is supported on the Linux Operating System.

1.2.1 Micro-Agents and Plugins

A *micro-agent* is a Linux thread continuously running to provide services required for the implementation of the reference functional architecture, such as reading and pre-processing sensor data, depositing the resulting information in the blackboard, controlling the flow of behaviour execution or handling the communications with other robots and the external monitoring computer. Each *micro-agent* can be seen as a *plugin* for the code. The different *plugins* are implemented as shared objects. In the sequel, the different micro-agents are briefly described (see also **Fig. 1.3**.).

Micro-agent VISION: This micro-agent reads images from one of two devices. Examples of such devices are USB web cams whose images can be acquired simultaneously. However, the bandwidth is shared between the two cameras. Actually, one micro-agent per camera is implemented. Each of them has several modes available. A mode has specific goal(s), such as to detect the ball, the goals, to perform self-localization or to determine the region around the robot with the largest amount of free space, in the robotic soccer domain. Each mode is implemented as a plugin for the code.

Micro-agent SENSORFUSION: This *micro-agent* uses a Bayesian approach to the integration of the information from the sensors of each robot and from all the team robots. Section 1.3 provides details on sensor fusion for world modelling.

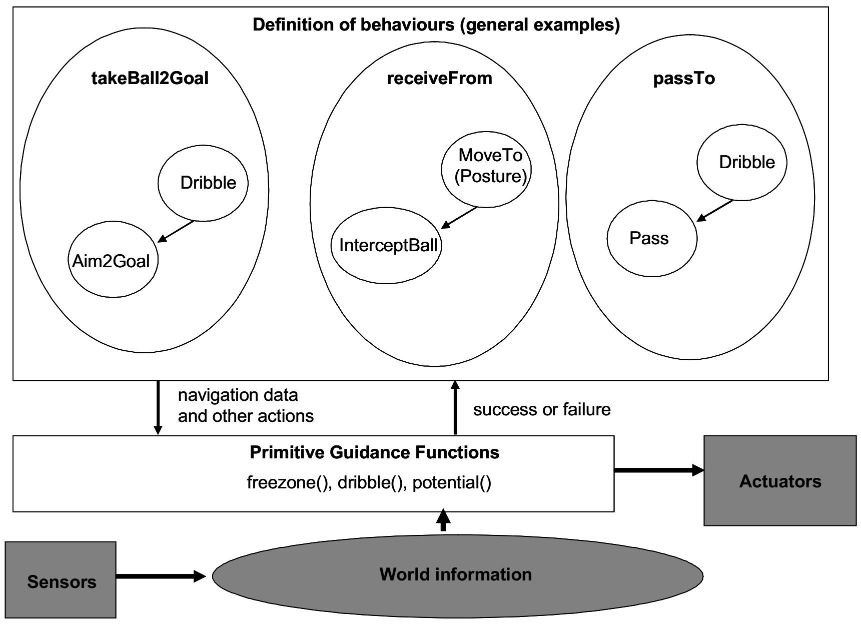

Definition of behaviours (general examples)
takeBall2Goal
Dribble
Aim2Goal
receiveFrom
MoveTo (Posture)
InterceptBall
passTo
Dribble
Pass
navigation data and other actions
success or failure
Primitive Guidance Functions
freezone(), dribble(), potential()
Actuators
Sensors
World information

Fig. 1.2. Functional architecture from an operator standpoint (detail of the Behaviour Execution Level)

Micro-agent CONTROL: This *micro-agent* receives the operator/behaviour selection message from the `machine` *micro-agent* and runs the selected operator/behaviour, by executing the appropriate *plugin*. Currently, each micro-agent is structured as a finite state machine where the states correspond to primitive tasks and the transitions to logical conditions on events detected through information put in the blackboard by the `sensorfusion` *micro-agent*. This *micro-agent* can also select the vision modes by communicating this information to the `vision` *micro-agent*. Different control *plugins* correspond to the available behaviours.

Micro-agent MACHINE: This *micro-agent* coordinates the different available operators/behaviours (`control` *micro-agents*) by selecting one of them at a time. The operator/behaviour chosen is communicated to the `control` *micro-agent*. Currently, behaviours can be coordinated by:

- a finite state machine, where each state corresponds to a behaviour and each transition corresponds to a logical condition on events detected through information put in the blackboard by the vision (e.g., found ball, front near ball) and control (e.g., behaviour success, behaviour failure) *micro-agents*.

- a rule-based decision-making system, where the rules left-hand side test the current world state and the rules right-hand side select the most appropriate behaviour.

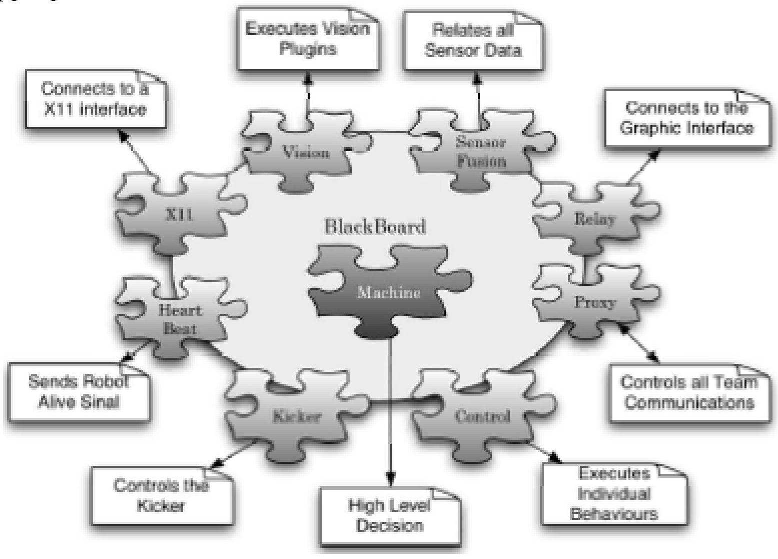


Fig. 1.3. Software architecture showing micro-agents and the blackboard

Micro-agent PROXY: This *micro-agent* handles the communications of a robot with its teammates using TCP/IP sockets. It is typically used to broadcast through wireless Ethernet the blackboard shared variables (see below).

Micro-agent RELAY: This *micro-agent* relays the BB information on the state of each robot to a “telemetry” interface running in an external computer, using TCP/IP sockets. Typically, the information is sent through wireless Ethernet, but for debug purposes a wired network is also supported.

Micro-agent X11: This *micro-agent* handles the X11-specific information sent by each robot to the external computer, using TCP/IP sockets. It is typically used to send through wireless Ethernet the blackboard shared variables for text display in an X-window.

Micro-agent HEARTBEAT: This *micro-agent* sends periodically a message from each robot to its teammates to signal that the sender is alive. This is useful for dynamic role changes when one or more robots “die".

1.2.2 Distributed Blackboard

The *distributed blackboard* extends the concept of blackboard, i.e., a data pool accessible to several agents, used to share data and exchange communication among them. Traditional blackboards are implemented by shared memories and daemons that awake in response to events such as the update of some particular data slot, so as to inform agents that require that data updated. Our *distributed blackboard* consists, within each individual robot, of a memory shared among the different *micro-agents*, organised in data slots corresponding to relevant information (e.g., ball position, robot$_1$ posture, own goal), accessible through data-keys. Whenever the value of a blackboard variable is updated, a time stamp is associated to it, so that is validity (based on recency) can be checked later. Some of the blackboard variables are *local*, meaning that the associated information is only relevant for the robot where the corresponding data was acquired and processed, but others are *global*, and so their updates must be broadcasted to the other teammates (e.g., the ball position).

Ultimately, the blackboard stores a model of the surrounding environment of the robot team, plus variables that allow the sharing of information among team members. **Fig. 1.4**. shows the blackboard and its relation with the sensors (through sensor fusion) and the decision/control units (corresponding to the `machine` and `control` *micro-agents*) of our team of (four) soccer robots. We will be back to the *world model* issue in Section 1.3.

1.2.3 Hardware Abstraction Layer (HAL)

The Hardware Abstraction Layer is a collection of device-specific functions, providing services such as the access to vision devices, kicker (through the parallel port), robot motors, sonars and odometry, created to encapsulate the access to those devices by the remaining software. Hardware-independent code can be developed on the top of HAL, thus enabling simpler portability to new robots.

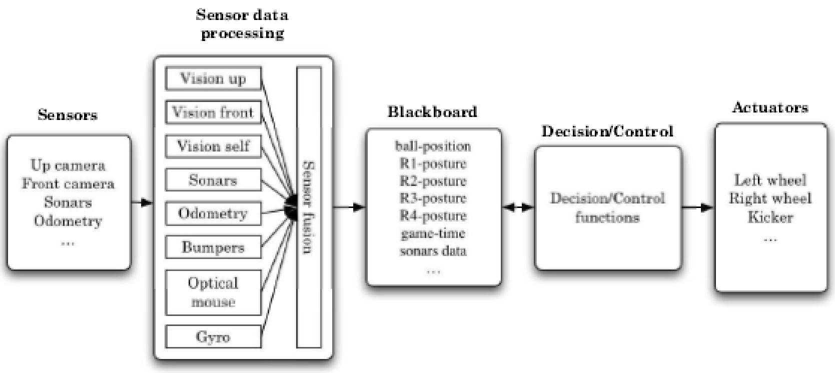


Fig. 1.4. The blackboard and its interface with other relevant units

1.2.4 Software Architecture Extension

More recently, we have developed a software architecture that extends the original concepts previously described and intends to close the gap between hybrid systems [13] and software agent architectures [1, 2], providing support for task design, task planning, task execution, task coordination and task analysis in a multi-robot system [15].

The elements of the architecture are the *Agents*, the *Blackboard*, and the Control/Communication *Ports.*

An *Agent* is an entity with its own execution context, its own state and memory and mechanisms to sense and take actions over the environment. They have a *control interface* used to control their execution. The control interface can be accessed remotely by other agents or by a human operator. Agents share data by a *data interface*. Through this interface, the agents can sense and act over the world. There are *Composite Agents*, encapsulating two or more interacting agents and *Simple Agents*, which do not control other agents and typically represent hardware devices, data fusion and control loops. Several agent types are supported, corresponding to templates for agent development. We refer to the mission as the top-level task that the system should execute. In the same robotic system, we can have different missions. The possible combinations among these agent types provide the flexibility required to build a Mission for a cooperative robotics project. The mission is a particular agent instantiation. The agents' implementation is made to promote the reusability of the same agent in different missions.

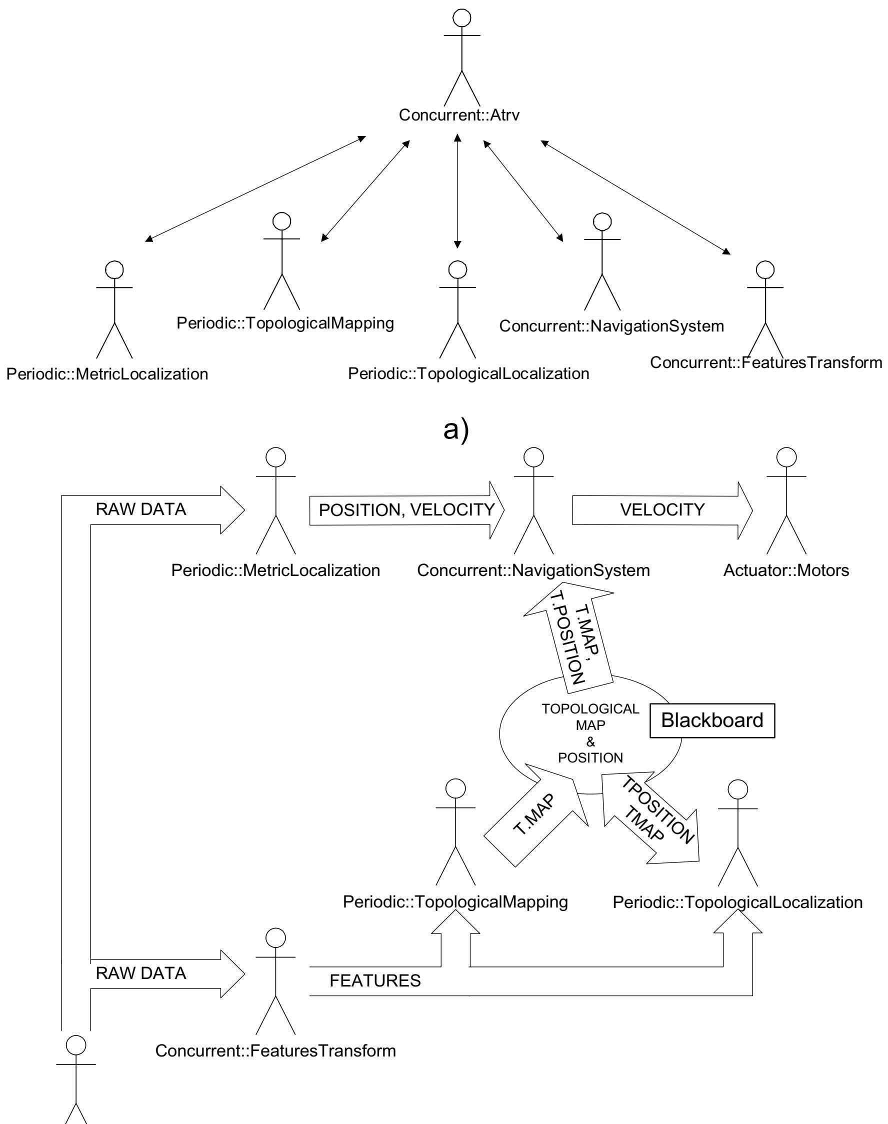

Concurrent::Atrv
Periodic::MetricLocalization
Periodic::TopologicalMapping
Periodic::TopologicalLocalization
Concurrent::NavigationSystem
Concurrent::FeaturesTransform
a)
RAW DATA
POSITION, VELOCITY
VELOCITY
Periodic::MetricLocalization
Concurrent::NavigationSystem
Actuator::Motors
T.MAP, T.POSITION
TOPOLOGICAL MAP & POSITION
Blackboard
T.MAP
TPOSITION TMAP
Periodic::TopologicalMapping
Periodic::TopologicalLocalization
RAW DATA
FEATURES
Concurrent::FeaturesTransform
Sensors::Raw
b)

Fig. 1.5. Example of application of the software architecture (extended version) to the modelling of (**a**) control flow and (**b**) data flow within the land robot of the rescue project

Ports are an abstraction to keep the agents decoupled from other agents. When an agent is defined, his ports are kept unconnected. This approach enables using the same agent definition in different places and in different ways.

There are two types of ports: *control ports* and *data ports*. *Control ports* are used within the agent hierarchy to control agent execution. Any *simple agent* is endowed with one upper control interface. The upper interface has two defined control ports. One of the ports is the input control port, which can be seen as the request port from where the agent receives notifications of actions to perform from higher-level agents. The other port is the output control port through which the agent reports progress to the high level agent. *Composite agents* also have a lower level control interface from where they can control and sense the agents beneath him. The lower level control interface is customized in accordance to the type of agent.

Data ports are used to connect the agents to the blackboard data entries, enabling agents to share data. More than one port can be connected to the same data entry. The data ports are linked together through the blackboard.

Under this architecture, a different *execution mode* exists for each development view of a multi-robot system. Five *execution modes* are defined:

- **Control mode**, which refers mostly to the run-time interactions between the elements and is distributed through the telemetry/command station and the robots. Through the control interface, an agent can be enabled, disabled and calibrated.
- **Design mode**, where a mission can be graphically designed.
- **Calibration mode**, under which the calibration procedure for behaviour, controller, sensor and different hardware parameters that must be configured or calibrated is executed.
- **Supervisory Control Mode**, which enables remote control by a human operator, whenever required.
- **Logging and Data Mode**, which enables the storage of relevant mission data as mission execution proceeds, both at the robot and telemetry/command station.

An example of application of this agent-based architecture to the modelling of control and data flow within the land robot of the RESCUE project [21], where the Intelligent Systems Lab at ISR/IST participates, is depicted in **Fig. 1.5**. More details on the RESCUE project are given in Section 1.5.

1.3 World Modelling by Multi-Robot Systems

In dynamic and large dimension environments, considerably extensive portions of the environment are often unobservable for a single robot. Individual robots typically obtain partial and noisy data from the surrounding environment. This data is often erroneous, leading to miscalculations and wrong behaviours, and to the conclusion that there are fundamental limitations on the reconstruction of environment descriptions using only a single source of sensor information. Sharing information among robots increases the effective instantaneous visibility of the environment, allowing for more accurate modelling and more appropriate response. Information collected from multiple points of view can provide reduced uncertainty, improved accuracy and increased tolerance to single point failures in estimating the location of observed objects. By combining information from many different sources, it would be possible to reduce the uncertainty and ambiguity inherent in making decisions based only in a single information source.

In several applications of MRS, the availability of a world model is essential, namely for decision-making purposes. **Fig. 1.4**. depicts the block diagram of the functional units, including the world model (coinciding, in the figure, with the blackboard) for our team of (four) soccer robots, and its interface with sensors and actuators, through the sensor fusion and control/decision units. Sensor data is processed and integrated with the information from other sensors, so as to fill slots in the world model (e.g., the ball position, or the robot self-posture). The decision/control unit uses this to take decisions and output orders for the actuators (a kicker, in this application) and the navigation system, which eventually provides the references for the robot wheels.

Fig. 1.6. shows a more detailed view of the sensor fusion process followed in our soccer robots application. The dependence on the application comes from the sensors used and the world model slots they contribute to update, but another application would follow the same principles.In this case, each robot has two cameras (up and front), 16 sonars and odometry sensors. The front camera is used to update the information on the ball and goal positions with respect to the robot. The up camera is actually combined with an omnidirectional mirror, resulting into a catadioptric system that provides the same information plus the relative position of other robots (teammates or opponents), as well as information on the current posture of the robot, obtained from a single image [25] and on the surrounding obstacles. The sonars provide information on surrounding obstacles as well. Therefore, several local (at the individual robot level) and global (at the team level) sensor fusion operations can be made. Some examples are:

- the ball position can be locally obtained from the front and up camera, and this information is fused to obtain the *local* estimate of the ball position (in world coordinates);
- the *local* ball position estimates of the 4 robots are fused into a *global* estimate;
- the *local* opponent robot position estimates obtained by one robot are fused with the its teammates estimates of the opponent position estimates, so as to update the world model with a *global* estimate of all the opponent robot positions;
- the *local* robot self-posture estimate from the up camera is fused with odometry to obtain the *local* estimate of the robot posture;
- the *local* estimates of obstacles surrounding the robot are obtained from the fusion between sonar and up camera data on obstacles.

1.3.1 Sensor Fusion Method

There are several approaches to sensor fusion in the literature. In our work, we chose to follow a Bayesian approach closely inspired in Durrant-Whyte's method [12] for the determination of geometric features observed by a network of autonomous sensors. This way, the obtained world model associates uncertainty to the description of each of the relevant objects it contains.

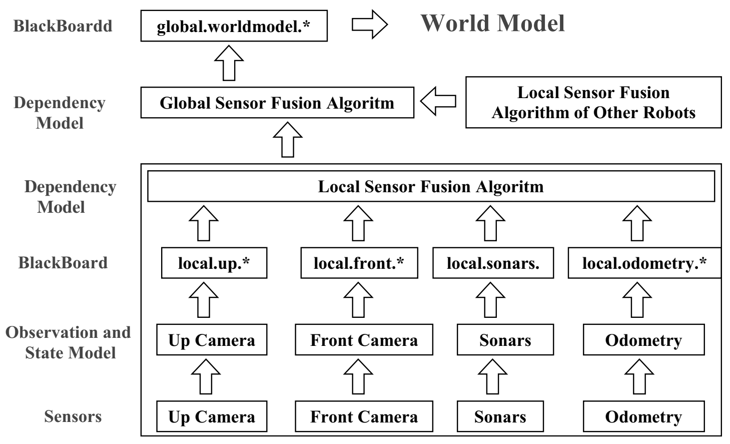


Fig. 1.6. Detailed diagram of the sensor fusion process for the soccer robots application and its contribution to the world model

In order to cooperatively use sensor fusion, team members must exchange sensor information. This information exchange provides a basis through which individual sensors can cooperate with each other, resolve conflicts or disagreements, and/or complement each other's view of the environment. Uncertainties in the sensor state and observation are modeled by Gaussian distributions. This approach takes into account the last known position of the object and tests if the readings obtained from several sensors are close enough, using the Mahalanobis distance, in order to fuse them. When this test fails, no fusion is made and the sensor reading which has less variance (more confidence) is chosen.

A sequence of observations $z^P = \{z_1, ..., z_n\}$, of an environment feature $p \in P$ (e.g., the ball position in robotic soccer, or a victim in robotic rescue), which are assumed to derive from a sensor modeled by a contaminated Gaussian probability density function, is considered, so that the i^{th} observation is given by:

$$f_i(z_i \mid p) = \left[(1-\varepsilon)N\left(p, \Lambda_i^1\right) + \varepsilon N\left(p, \Lambda_i^2\right)\right] \tag{1}$$

where $0.05 < \varepsilon < 0.1$ and $\Lambda_i^2 >> \Lambda_i^1$.

It is well known that if the prior distribution $\pi(p)$ and the conditional observation distribution $f(z \mid p)$ are modeled as independent Gaussian random vectors $p \sim N(\overline{p}, \Lambda_0)$ and $z_i \mid p \sim N(\overline{p}, \Lambda_i)$ respectively, then the posterior distribution $\pi(p \mid z_1)$ after taking a single observation z_1 can be derived using Bayes law and is also jointly Gaussian with mean vector

$$\overline{p}' = \left[\Lambda_0^{-1} + \Lambda_1^{-1}\right]^{-1} \left[\Lambda_1^{-1} z_1 + \Lambda_0^{-1} \overline{p}\right] \tag{2}$$

and covariance matrix

$$\Lambda' = \left[\Lambda_0^{-1} + \Lambda_1^{-1}\right]^{-1} \tag{3}$$

This method can be extended to n independent observations.

In a multi-Bayesian system, each team member individual utility function is given by the posterior likelihood for each observation z_i:

$$u_i(\hat{p} = \delta_i(z_i), p) = \pi(p \mid z_i) \approx N(\overline{p}, \sigma_i),\ i = 1,2 \tag{4}$$

A sensor or team member will be considered rational if, for each observation z_i of some prior feature $\delta_i(z_i) \in P$, it chooses the estimate that maximizes its individual utility $u_i(\delta_i(z_i), p) \in \Re$. In this sense, utility is just a metric for constructing a complete lattice of decisions, allowing any two decisions to be compared in a common framework. For a two-member team, the team utility function is given by the joint posterior likelihood:

$$U(\hat{p}(z_1, z_2), p) = F(p \mid z_1, z_2) = f_1(p \mid z_1) f_2(p \mid z_2) \tag{5}$$

The advantage of considering the team problem in this framework is that both individual and team utilities are normalized so that comparisons can be performed easily, supplying a simple and transparent interpretation to the group rationality problem. The team itself will be considered group rational if together the team members choose to estimate $\hat{p} \in P$ (environment feature), which maximizes the joint posterior density.

$$\hat{p} = \arg\max F(p \mid z_1, z_2) = \arg\max f_1(p \mid z_1) f_2(p \mid z_2) \tag{6}$$

There are two possible results for (6)

- $F(p \mid z_1, z_2)$ has a unique mode equal to the estimate $\hat{p}$;
- $F(p \mid z_1, z_2)$ is bimodal and no unique group rational consensus estimate exists.

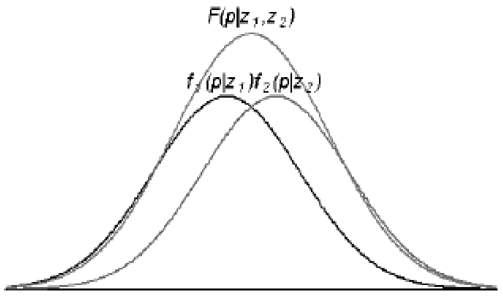


Fig. 1.7. Two Bayesian observers with joint posterior likelihood indicating agreement

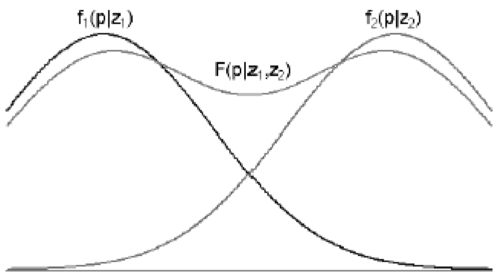


Fig. 1.8. Two Bayesian observers with joint posterior likelihood indicating disagreement

If $F(p \mid z_1, z_2)$ has a unique mode, as displayed in **Fig. 1.7.**, it will satisfy:

$$\max F(\mathrm{p} \mid \mathrm{z}_1, z_2) \geq \max f_i(\mathrm{p} \mid \mathrm{z}_\mathrm{i}) \qquad i = 1,2 \tag{7}$$

Conversely, if $F(p \mid z_1, z_2)$ is bimodal, as displayed in **Fig. 1.8.**, then:

$$\max f_i(\mathrm{p} \mid \mathrm{z}_\mathrm{i}) \geq \max F(\mathrm{p} \mid \mathrm{z}_1, z_2), \quad i = 1,2 \tag{8}$$

A rational team member will maximize utility by choosing to either agree or disagree with the team consensus. If a team member satisfies (8), then it will not cooperate with the team estimate. Thus the decision made by a team member based of its observations z_i is:

$$\hat{p} = \delta(z_i) = \arg\max\{f_i(p \mid z_i), F(p \mid z_1, z_2)\}, \quad i = 1,2 \tag{9}$$

Whether or not the individual team members will arrive at a consensus, the team estimate will depend on some measure of how much they disagree $|z_1 - z_2|$. If z_1 and z_2 are close enough, then the posterior density $F(p \mid z_1, z_2)$ will be unimodal and satisfy (7), with the consensus estimate given by (6). As $|z_1 - z_2|$ increases, $F(p \mid z_1, z_2)$ becomes flatter and eventually bimodal. At this point, the joint density will satisfy (8), and no consensus team decision will be reached. To find the point at which this space is no longer convex and disagreement occurs, one must ensure that the second derivative of the function $F(p \mid z_1, z_2)$ is positive. Differentiating leads to:

$$\begin{aligned}\frac{\partial^2 F}{\partial p^2} &= \frac{1}{f_1}\frac{d^2 f_1}{dp^2}+\frac{1}{f_2}\frac{d^2 f_2}{dp^2}+\frac{2}{f_1 f_2}\frac{df_1}{dp}\frac{df_2}{dp} \\ &= \left(\sigma_1^{-2}+\sigma_2^{-2}\right)-\left[\sigma_1^{-2}(p-z_1)+\sigma_2^{-2}(p-z_2)\right]\end{aligned} \tag{10}$$

For this to be positive and hence $F(p \mid z_1, z_2)$ to be convex, we are required to find a consensus over the feature of the environment p which satisfies

$$\left[\sigma_1^{-2}(p-z_1)+\sigma_2^{-2}(p-z_2)\right]^2\left(\sigma_1^{-2}+\sigma_2^{-2}\right)^{-1} \le 1 \tag{11}$$

Notice that (11) is a normalized weighted sum, a scalar equivalent to the Kalman gain matrix. The consensus $\hat{p}$ that maximizes F is therefore given by

$$\hat{p} = \frac{\left(\sigma_1^{-2} z_1 + \sigma_2^{-2} z_2\right)}{\left(\sigma_1^{-2}+\sigma_2^{-2}\right)} \tag{12}$$

Replacing (12) into (11), we obtain

$$\frac{(z_1-z_2)(z_1-z_2)}{\left(\sigma_1^{-2}+\sigma_2^{-2}\right)} = D_{12}(z_1, z_2) \tag{13}$$

where $D_{12} \le 1$. The disagreement measure $D_{12}(z_1, z_2)$ is called the Mahalanobis distance.

This measure of disagreement represents an advantage of our approach (based on Durrant-Whyte's method) to other probabilistic approaches to object localization, such as [39], which uses multiple hypothesis tracking to track multiple opponent robots in robot soccer, and the likelihood of hypotheses to discard some of them, or [51], where a Markov process is used as an observation filter for a Kalman filter which tracks the ball (also in robot soccer), assuming that motion is equally possible in all directions, with Gaussian distributed velocities. The advantage of having an expression to compute (dis)agreement comes at the expense of requiring Gaussian distributions, while the referred approaches assume no distribution, iteratively updating a probability distribution over a discretization grid [44].

1.3.2 Experimental Results for Soccer Robots

Using the fusion algorithm described in the previous section, we have been built a partial world model of the environment where our soccer robots evolve: a 12x8m green field, with white line markings, one yellow and one blue goal, an orange ball and robots which have different color markings around themselves, at a specified height.

The decision process to determine the ball position is made by first determining if both observed ball positions from the two cameras can be merged locally through the Mahalanobis distance. This is accomplished by putting a time stamp in each camera observation, and using the time difference between stamps to modify the variance of each observation, in order to synchronize the fusion. When this synchronization is possible, the ball position will be the result of the fusion; otherwise, the observation with the smallest variance is chosen, meaning that the observation with the highest confidence is used to determine the ball position. After the local ball position estimate has been determined, the estimation of the global ball position is attempted, by fusing all local estimates of each robot, to get a global sensor fusion, as explained before. Each player acts as a sensor, taking observations from its two cameras, modifying the variance based on the difference of the observation time stamps, fusing and reporting them to the other team members.

To test the global sensor fusion, three robots were placed on the field. We then ran the algorithm in each robot with only local sensor fusion working (**Fig. 1.9**.a) and then with both local and global sensor fusion working (**Fig. 1.9**.b). Each robot has a measure of quality (local fusion variance) of its local sensor fusion, using it to decide who has priority in the global sensor fusion. The robot with the best measure of quality has priority over the others.

As seen in **Fig. 1.9**., the global sensor fusion improved the ball estimate.

In **Fig. 1.10**., although one of the robots cannot see the ball with its own cameras, because it is too far away, it knows where the ball is, since all robots share the same world information. This is the result of communicating all the features that each robot extracts from the environment to all the other teammates, and then using sensor fusion to validate those observations. Testing the agreement among all the team sensors eliminates spurious and erroneous observations. In **Fig. 1.10**.b), the robot in the bottom part of the field cannot see the ball, so it gets the ball position from the global fusion of the other robot observations. Since the other two robots disagree with each other, the global fusion becomes equal to the local fusion of the robot with the best variance among the two.

In **Fig. 1.11**.a) two robots showing disagreement are depicted. This happened in this case because there were two balls in the field and each robot was detecting a ball in different positions. Although each robot has its own local sensor fusion estimate, they cannot reach an agreement about the global sensor fusion. When this happens, the robot makes its global sensor fusion estimate equal to its local sensor fusion estimate. In **Fig. 1.11**.b) we see the same two robots showing agreement.

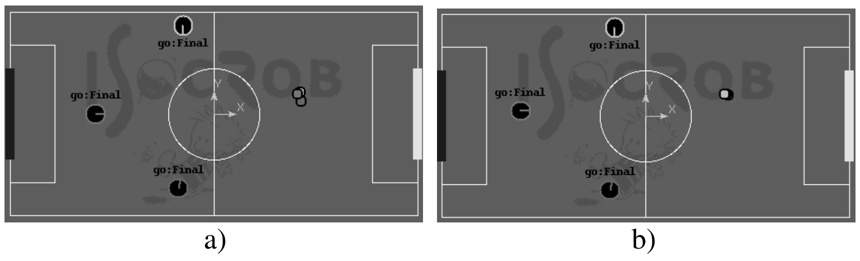


Fig. 1.9. (**a**) Local sensor fusion enabled and sensor fusion disabled; (**b**) Both local sensor fusion and global sensor fusion enabled. The larger circles with a mark denoting orientation represent the robots, while the small circles represent the balls as seen by each of the robots (denoted by the corresponding colors)

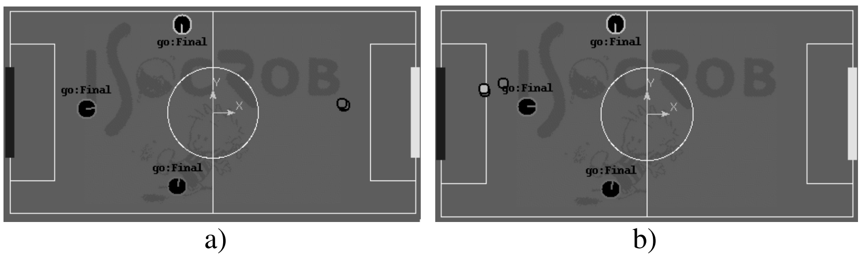


Fig. 1.10. (**a**) Leftmost robot receives ball position information of the other two; (**b**) Bottom robot receives ball position from top robot, while top and leftmost robot disagree

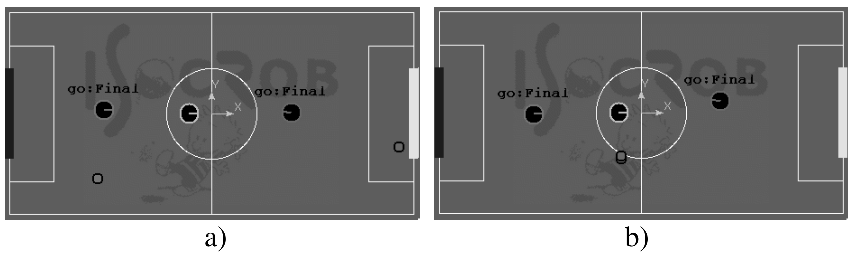
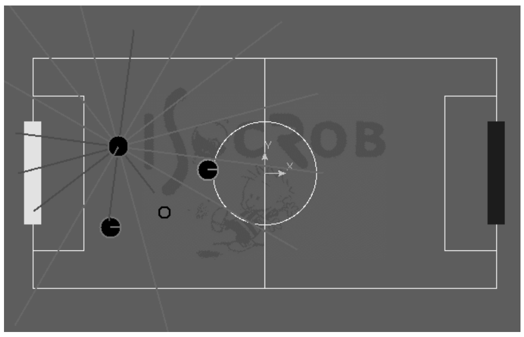
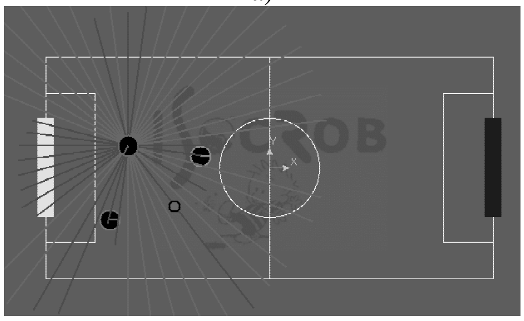


Fig. 1.11. (**a**) Two robots showing disagreement; (**b**) Two robots showing agreement


a)


b)

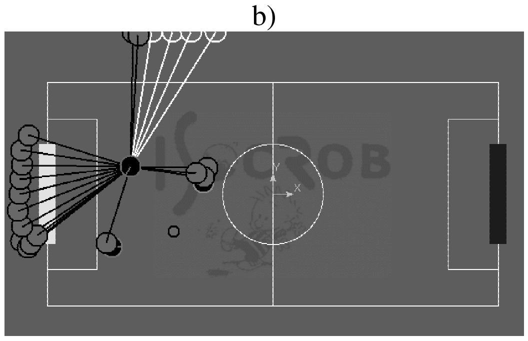

c)

Fig. 1.12. Obstacle detection by virtual and real sonar sensor fusion for a single robot: (**a**) 16 real sonar readings; (**b**) 48 virtual sonar readings; (**c**) results of fusing the readings in (a) and (b)

Although they have slightly different local sensor fusion estimates, they have the same global sensor fusion estimate of the ball, which is a result of the fusion of their local estimates.

Before each local fusion is made, each sensor observation and the local sensor estimate at the previous step are fused, with an increase in the variance of the latter, to reflect the time that has passed since the fusion was made. This helps to validate the new observation, because if fusion is successful then the new observation is a valid one and we are predicting the same feature as in the previous fusion operation. Otherwise, this means that the latest observation was probably a bad one and that we could not predict the feature evolution.

Another application of this method concerns the detection of obstacles in the soccer scenario, specifically other robots and the goals. In the example shown in **Fig. 1.12**., the algorithm was applied to the fusion of the information from a sonar ring around the robot, composed by 16 sonars, separated of 22.5º, and from 48 virtual sonars, separated of 7.5º, resulting from splitting the image of the up omnidirectional camera into 48 sectors.

Fig. 1.12.a) shows the readings of the real sonars, **Fig. 1.12**.b) depicts the virtual sonar readings and the final fused result is represented in **Fig. 1.12**.c). In a) and b), red rays mean that an obstacle was detected, e.g., the yellow goal, two robots, a wall located on the image top left and the ball (ignored in the final fusion). In c), all relevant obstacles are represented by circles connected to the robot by black and white rays.

1.4 Cooperative Navigation

The navigation sub-system provides a mobile robot with the capabilities of determining its location in the world and of moving from one location to another, avoiding obstacles. Whenever a multi-robot team is involved, the concept of navigation is extended to several new problems, basically concerning how to take the team from one region to another, while avoiding obstacles.

In this section we will present results for three such problems:

- A **navigation controllability problem**: given *N* (in general heterogeneous) robots distributed by *M* sites of a topological map, determine under which conditions one can drive the robots from an initial configuration (i.e., a specific distribution of the *N* robots by the *M* sites) to a final or target configuration.
- A **formation feasibility problem**: given the kinematics of several robots along with inter-robot constraints, determine whether there exist robot trajectories that keep the constraints.
- A **population optimal distribution control problem**: given a population of robots whose motion can be modelled by a stochastic hybrid automaton, determine the optimal command sequence that brings the population from an initial spatial probability distribution at time $t = 0$ to the closest possible distribution from a target spatial probability distribution at time *T*.

1.4.1 Navigation Controllability

Whenever the navigation of MRS is considered, the first question one might want to ask before start moving the robots from their initial locations to some target locations is whether it will ever be possible to reach the latter from the former, and, if possible, under what conditions. For several reasons, some due to the environment (e.g., one-way doors or roads), others due to the robot capabilities (e.g., some can push doors, others can push and pull them, some can climb stairs, others can not) and some others due to incorrect robot paths, this may never be possible or be only possible if we establish the appropriate path for the robots, given their and the environment constraints. Therefore, *checking whether a robot team can run into a non-recoverable configuration on its way to the target configuration and, in case it can, determining whether it is possible to supervise the team robot paths in such a way that this will never happen* is an important first step whenever MRS navigation is concerned.

The approach we followed to solve this problem, in joint work with the Mobile Robotics Laboratory at ISR/IST, is described in [26, 27], and assumes, so as to reduce the complexity of the problem, that the navigation environment is described by a topological map, where nodes represent locations of interest and directed edges between 2 nodes represent the existence of an oriented path between the locations represented by the nodes. Notice that the edges may capture the constraints on the robots or on the environment, e.g., if there is a descending stair from the room represented by node 1 to the room represented by node 2, a bidirectional edge will link nodes 1 and 2 whenever the environment is represented, but only a node from 1 to 2 will exist in the model of a robot that can not climb stairs moving in that environment.

The robot population is modelled as a finite-state automaton [6] whose blocking and controllability properties are checked so as to answer the above questions. Each automaton state corresponds to a given configuration, and the edges between states are labelled by actions corresponding to moving one robot from one location to another. Therefore, a blocking automaton state corresponds to a distribution of the robots from which the desired target configuration is not achievable, e.g., because one of the robots has reached a location from where it cannot exit. Finite state automaton controllability means that such blocking states are avoidable: it is possible to disable some actions (i.e., some robot displacements) to prevent the robots from ever reaching blocking configurations.

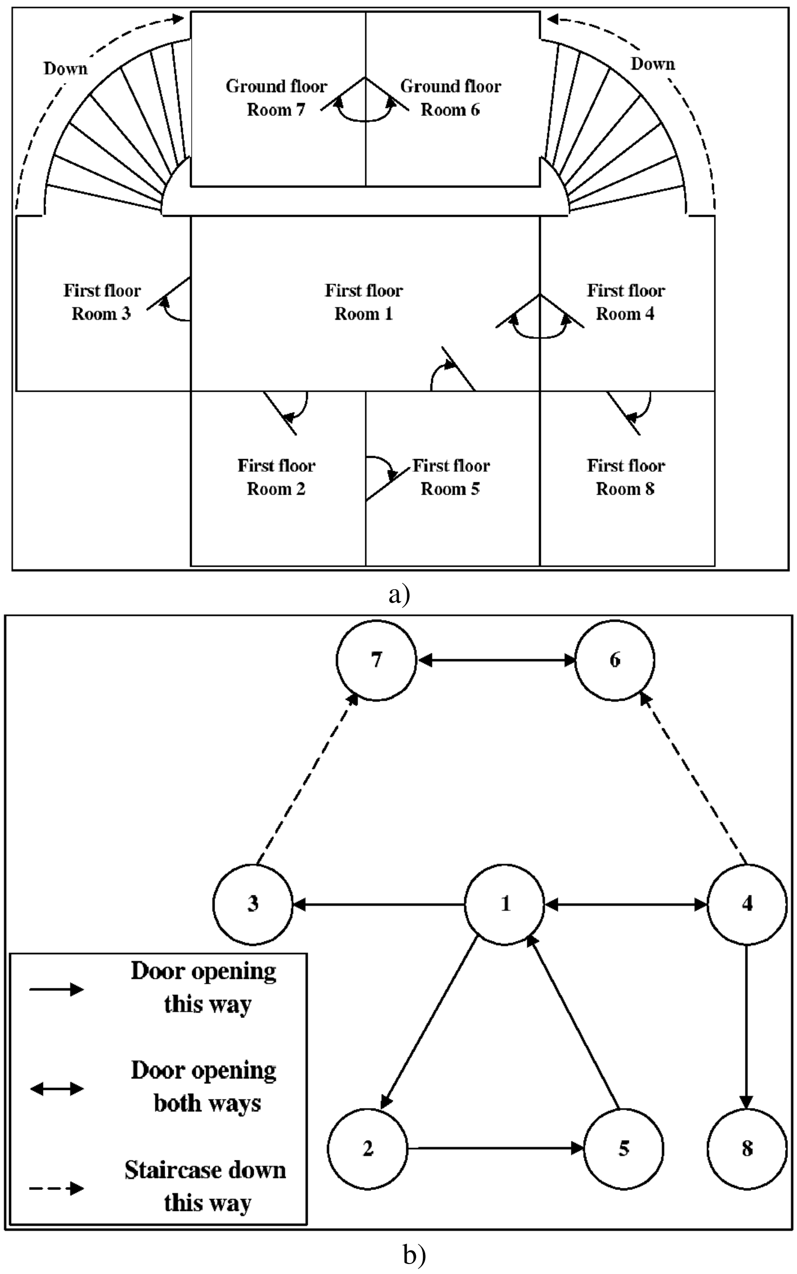


Fig. 1.13. Indoors rescue scenario: (**a**) Physical map; (**b**) Topological map

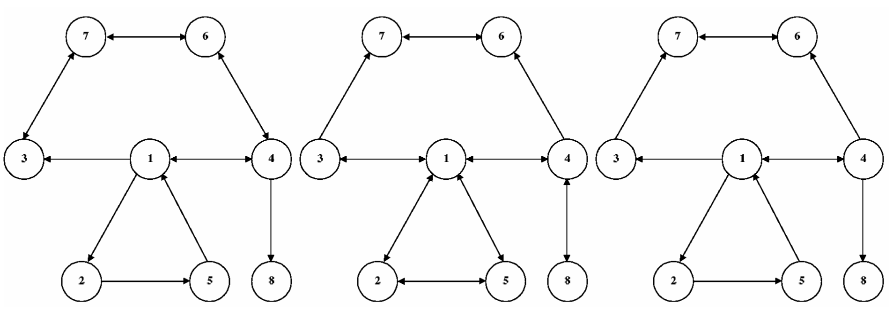


Fig. 1.14. Navigation automata for the *Crawler*, *Puller* and *Pusher* robots (from left to right)

One problem with such an approach is that the size of the finite state automaton modelling an *M*-robots-*N*-sites scenario will grow quickly with the number of robots and/or sites. Our results allow checking the blocking and controllability properties of the (potentially large) automaton modelling the multi-robot system with the blocking and controllability properties of smaller automata, designated as *navigation automata*, which model the navigation of each individual robot in the population. In navigation automata for a given robot, each state corresponds to the site where the robot is.

Due to the available space, we will only refer here the blocking result for heterogeneous robots (homogeneous robots are a particular case with specific results). For the controllability results and other details consult [28].

Theorem: In a generic *N*-robots-*M*-sites system, for its modelling automaton G to be non-blocking, all the navigation automata G_i, $i=1,\ldots,N$ must be non-blocking. Each G_i is a marked automaton with a marked state per site where there is at least one robot *i* in the target configuration. Similarly, if G is blocking, there is at least one *i* and one marked state of G_i such that G_i is blocking.

One modelling example following this approach concerns a team of three heterogeneous robots, with the following individual skills:

- the *Crawler* has tracker wheels and is capable of climbing and descending stairs. It is able to open doors only by pushing;
- the *Puller* is a wheeled mobile manipulator, able to open doors either by pushing or pulling. However, it is not able to climb stairs;
- the *Pusher* is a wheeled robot, able to open doors only by pushing. It cannot climb stairs.

The rescue operation takes place in the indoor environment depicted in **Fig. 1.13**. (an indoor rescue scenario). **Fig. 1.13**.a) represents the physical map, and **Fig. 1.13**.b) the corresponding topological map. Each of the robots is described by a different navigation automaton, as represented in **Fig. 1.14**. The robots will leave room 1 to assist three different victims, somewhere in the building.

The doors open as shown in **Fig. 1.13**., thus limiting the robots access to the different rooms. Inside rooms 6 or 7, only the Crawler can go upstairs. In rooms 3 and 4, all the robots may fall downstairs, i.e., events Go(6) and Go(7) are uncontrollable for all robots. Blocking and controllability results concerning this result are presented in [27].

1.4.2 Formation Feasibility

Formation control is a relatively recent area of research, e.g., [8], where many fundamental questions remain unanswered. The control of a formation requires individual robots to satisfy their kinematics while constantly satisfying inter-agent constraints. In typical leader-follower formations, the leader has the responsibility of guiding the group, while the followers have the responsibility of maintaining the inter-robot formation. Distributing the group control tasks to individual robots must be compatible with the control and sensing capabilities of the individual robots. As the inter-robot dependencies get more complicated, a systematic framework for controlling formations is vital.

In a joint work with the GRASP Lab, at the University of Pennsylvania, we have proposed a framework to determine motion feasibility of multi-robot formations [43]. Formations are modelled using *formation graphs*, i.e., graphs whose nodes capture the individual robot kinematics, and whose edges represent inter-robot constraints that must be satisfied.

We assume kinematic models for each robot, described by drift free control systems. This class of systems is rich enough to capture holonomic, nonholonomic, or underactuated vehicles.

Two distinct types of formations are considered: *undirected formations* and *directed formations*. In *undirected formations* each robot is equally responsible for maintaining the formation. For each formation constraint between two robots, cooperation is assumed to satisfy the constraint. *Undirected formations* therefore present a more centralized (in the sense of the required information) approach to the formation control problem, as communication between all the robots is, in general, necessary. In *directed formations*, for each inter-robot constraint, only one of the robots (the follower)is responsible for maintaining the constraint. *Directed formations*, therefore, represent a more decentralized solution to the formation control problem. Not only the required information flow is restricted to the pairs of robots linked by an edge but also the synthesis of feedback control laws enforcing the constraints is also simpler.

Two problems are tackled in this work, for which we just summarize the main results here, as a detailed explanation would require a mathematical background that is out of the context of this book:

- **feasibility problem**: given the kinematics of several robots along with inter-robot constraints, determine whether there exist robot trajectories that maintain those constrains. For both directed and undirected (not necessarily rigid) formations we obtain algebraic conditions that determine formation motion feasibility.
- When a formation has feasible motions, the **formation control abstraction problem** is then considered: given a formation with feasible motions, obtain a lower dimensional control system that maintains formation along its trajectories. Such control system allows controlling the formation as a single entity, therefore being well suited for higher levels of control. The directions in which a feasible formation can be controlled are determined, providing an abstraction of the formation that can be controlled as a single entity.

1.4.3 Population Optimal Distribution Control

One of the most relevant (and hard) topics in MRS is the modelling of large-size robot population behaviour. Under the current state-of-the-art, it seems that results for small-sized populations do not scale necessarily well for large-scale ones. Therefore, the mathematical modelling of large-size agent populations should be useful to predict the evolution of a population and subsequently design controllers or supervisors capable of changing the population behaviour by the suitable adjustment of appropriate parameters. One approach with large potential for this purpose is based on recent results on the mathematical modelling of biological systems [33]. In fact, our work in this direction has been originally developed for biological experiments modelling, jointly with biologists [30].

The work concerns a large size population of robots that navigate in an environment known with some associated uncertainty [29]. The motion of the robots in this environment is modelled by a stochastic hybrid automaton with discrete states representing a set of motion commands (e.g., representing a set of directions that the robots should follow while navigating), and a continuous state space representing the robot motion state (e.g., its posture and velocity). This hybrid automaton is stochastic because the transitions between discrete states are governed by transition probabilities or, more precisely, under a Markov assumption, by transition rates corresponding to the rates of exponential distributions that model the times between events that cause the transitions. The transition rates represent both the uncertainty about the environment, which causes the robots to fail executing some commands (e.g., due to terrain irregularities or lack of communication visibility), and the control signals (in the form of control rates) that can modify that uncertainty, thus controlling the population spatial distribution over time, as the robots move.

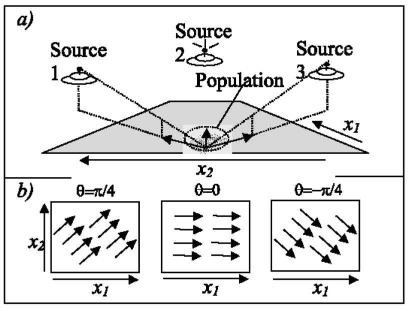
An important result of this work is the following

Theorem: The continuous time Markov Chain hybrid automaton endowed with one input (a stochastic event sequence) and having as output a function of the continuous part of the state, with N discrete states and state probability given by $\dot{P}(t) = L^T P(t)$ where $P(t) = \left[P_1(t) \quad \ldots \quad P_N(t)\right]$ is the probability of the discrete state and $L = \left[\lambda_{ij}\right]^T$ is a transition rate matrix and λ_{ij} is the rate of transition from discrete state i to discrete state j. The vector of probability density functions $\rho(x,t) = \left[\rho_1(x,t) \quad \ldots \quad \rho_N(x,t)\right]$, where $\rho_i(x,t)$ is the probability density function of state (x,i) at time t, satisfies

$$\frac{\partial \rho(x,t)}{\partial t} = L^T \rho(x,t) - \begin{bmatrix} \nabla.(f_1(x,t)\rho_1(x,t)) \\ \vdots \\ \nabla.(f_N(x,t)\rho_N(x,t)) \end{bmatrix} \tag{14}$$

where $f(x,i)$ is the vector of vector field values at state (x,i).

If (x,i) represents the hybrid state of a large population of robots, the result in this theorem can be used to predict the evolution of the probability density function (pdf) of the population spatial distribution. **Fig. 1.15**. represents a 2D example where a large number of land (e.g., rescue) robots is left in a given region and are afterwards commanded by 3 aerial robots which can order them to move in pre-defined directions. In the same figure, the corresponding stochastic hybrid automaton modelling the population spatial distribution over time is also represented. Using (14), the predicted evolution of the population spatial distribution for a given set of transition rates and at several time instants is represented in **Fig. 1.16**. by the contours of the pdfs for each discrete state and for the summation of the discrete state pdfs.



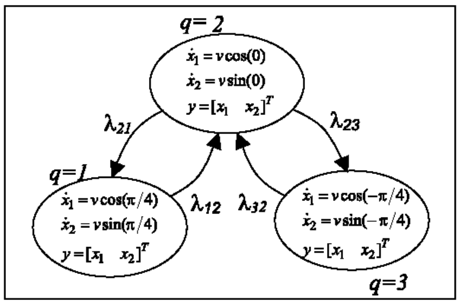


Fig. 1.15. On the left: (**a**) A robotic population controlled by three aerial robots (sources) and (**b**) the vector fields created by control signal sources. On the right: the stochastic hybrid automaton modelling the population spatial distribution over time

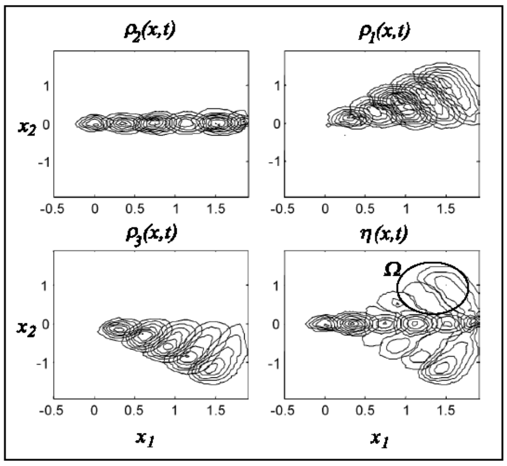


Fig. 1.16. The pdf contours of the robot population states $_i(x,t)$, and the pdf of the robots position $\eta(x, t)$, for a given set of transition rates, given the model of Fig. 1.15. Plots shown at 6 time instants (from left to the right in the pictures), starting at time $t = 0$. Ω is a region of interest for the mission

If a given region, such as the one denoted by Ω in **Fig. 1.16.**, is of some particular interest for our robotic mission and we want most of the robots at time instant T in that region, an optimal control problem can be formulated where the control signal u is a vector composed by the transition rates between discrete states of the stochastic hybrid automaton and the performance function to be maximized is given by

$$J(u) = \int_X w^T(x)\rho(x,T)$$

where the dependence in u comes from the dependence of ρ with the transition rates, and w is a window function that spatially weights the state pdf, e.g., to confine it to a sub-region Ω of the state space X.

The solution of this optimal control problem is not trivial, since (14) is a partial differential equation. However, for certain cases, it is possible to compute an open loop control solution. The derivation of such solution is out of the scope of this book, but we provide an example for a one-dimensional version of the problem depicted in **Fig. 1.15**., shown in **Fig. 1.17**. **Fig. 1.18**. shows a pdf at time $T = 3$ h for this system very close to the desired one. This resemblance depends in general of the control amplitude, the system model and the time T.

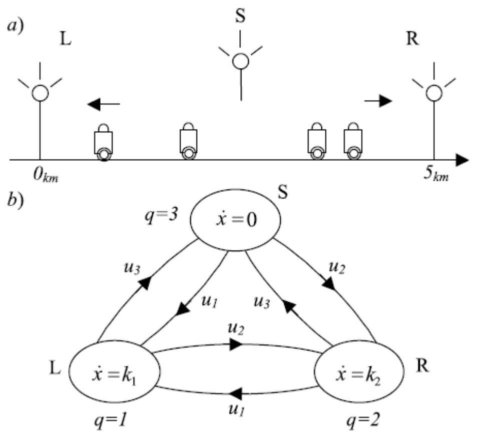


Fig. 1.17. One-dimensional version of the robotic population example in Fig. 1.15. In this example the robots can move left, right or stop

1.5 Cooperative Decision-Making

Previously in the chapter, we have already described solutions for MRS architectures, cooperative perception and cooperative navigation. But to act autonomously and machine-wise intelligently, a MRS team must be able to plan action sequences and to take its own decisions autonomously.

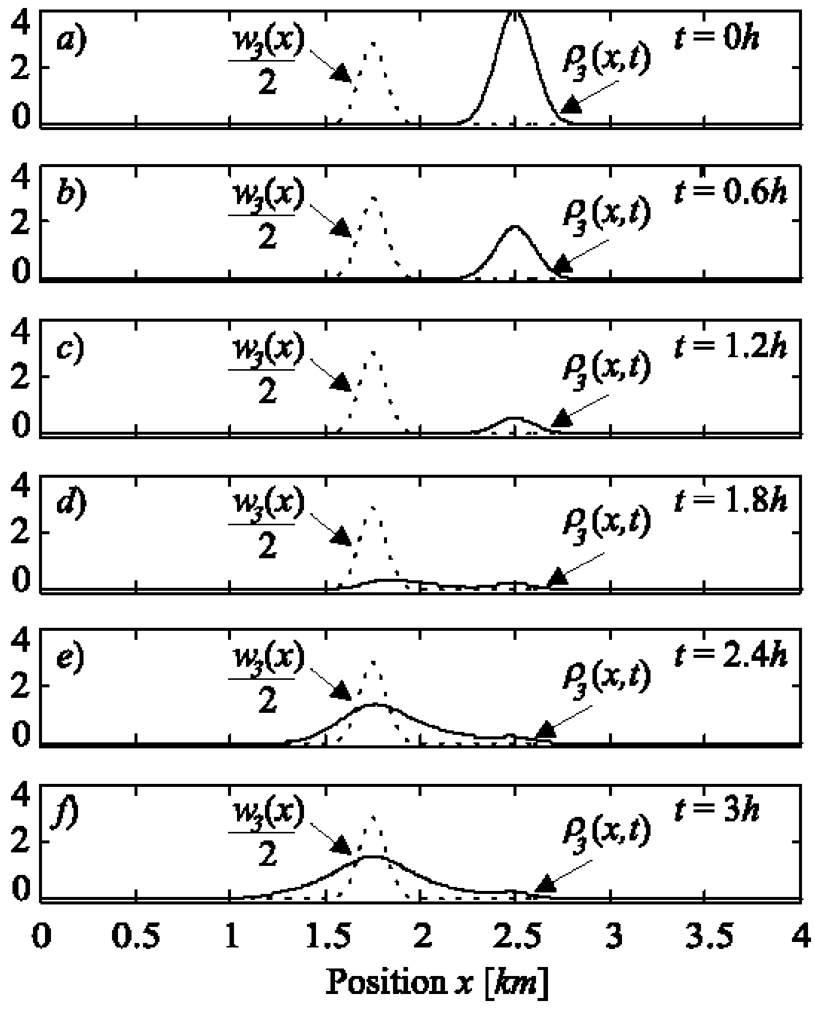


Fig. 1.18. Pdfs for discrete state 3 (stopped robots) concerning the example in Fig. 1.17., for $k_1 = -0.5$, $k_2 = 0.25$, $w^T(x) = [0 \quad 0 \quad w_3(x)]$, $u \in [0, 2]$, at several time instants, including the terminal time $T = 3$ h

Of course, the decisions depend on the perception the robot has of its surrounding environment, and most actions require navigating from one point to another. The literature is rich in planning solutions for single agents, but multi-agent task planning, and especially multi-robot task planning are relatively recent research subjects. Relevant issues for our group in this research is the use of logic-based approaches to ensure the application of formal verification methods, the inclusion of uncertainty in the task, plan and action models and the reduction of the search space whenever optimal stochastic solutions are sought. Some of the work done in those directions is described in the next sections.

1.5.1 Hybrid Logic-Based Decision System

The functional architecture described in Section 1.2 allows the implementation of operators and the switching among them using different approaches, as for example state machines or AI production systems.

Previous implementations of the µA machine were done using state machines, which basically implemented a reactive decision-making system, based on simple reactions to external or internal events. Robots using this kind of decision mechanism usually show very primitive behaviors, and are not able to accomplish non-trivial goals on complex, dynamic, and incomplete domains. On the other side, deliberative decision-making systems are able to make decisions based upon past experience and by predicting future consequences of its actions. The system may even act and decide in a way that was not predicted by the designers, but that is actually valid and efficient in order to achieve the goal. But deliberative systems are usually computationally heavy. A way to take advantage of both kinds of systems is a hybrid decision-making system with a reactive component and a deliberative component. The system uses, normally, the decisions made by the former component. But, when it takes too long to decide, it uses the decision made by the reactive component.

So, in order to have a more abstract way to deal with decision-making and behaviour switching, the µA `machine` has been implemented using a distributed decision-making architecture supported on a logical approach to modelling dynamical systems [37], based on situation calculus, which is a second-order language specifically designed to representing dynamically changing worlds. All the changes to the world are result of named actions. A possible world history, which is simply a sequence of actions, is represented by a first-order term called a *situation*. There is a distinguished binary function symbol do; do(_, S) denotes the successor situation to S resulting from performing the action _. For example, put(x, y) might stand for the action of putting object x on object y, in which case do(put(A,B), S) denotes the situation resulting from placing A on B when the current situation is S. Notice that in situation calculus, actions are denoted by function symbols, and situations (world histories) are first-order terms. For example, do(score(A), do(takeBall(B), S_0)) is a situation term denoting the sequence of action [takeBall(B), score(A)].

Generally, the values of relations and functions in a dynamic world will vary from one situation to the next. Relations whose truth values vary from situation to situation are called *relational fluents* and are denoted by predicate symbols taking a situation term as their last argument. Actions have preconditions, necessary and sufficient conditions that characterize when the action is physically possible. World dynamics are specified by effect axioms, which describe the effects of a given action on the fluents - the causal laws of the domain. Axiomatizing a dynamic world requires more than just action preconditions and effect axioms: frame axioms are also necessary. These specify the action invariants of the domain, namely those fluents that remain unaffected by a given action. But the problem with frame axioms is that we can expect a vast number of them – the frame problem: only relatively few actions will affect the truth value of a given fluent; all other actions leave the fluent unchanged. This is problematic for a theorem proving system, as it must efficiently reason in the presence of so many frame axioms.

By appealing to earlier ideas of Haas, Schubert and Pednault, Reiter proposed a quasi-solution to the frame problem, through a systematic procedure for generating, from the effect axioms, all the frame axioms needed.

The hybrid architecture developed for the high level decision-making of our MRS comprises several components, from which the most important ones are: World Representation, Reactive Component, Deliberative Component and Behavior Selection, whereas the deliberative one uses the procedure proposed by Reiter. **Fig. 1.19**. presents this architecture.

The World Representation Component (WRC) is responsible to build a world model using sensorial data. From the sensory inputs and the static information about the game, the WRC builds the game model, which consists of basic information, like ball position and players' postures, and advanced information, such as cooperative decisions. The variables used to define the world model are stored in a *blackboard*, as described in Section 1.2.

Based on this information a more pictorial world model is build, and shared by all the robots. The idea is to focus the attention on the most important moving element in the game, the ball. But to make adequate and efficient decisions robots must see the world in a more abstract way. The idea is to divide the area surrounding the ball in six cones, and each cone in three different zones (near, middle, far). Then we classify every element of the game (opponents, teammates, goals, field lines, etc.) using this relative positioning, and work with things like ”near goal”, ”has line of pass”, etc.. This world model is inspired on an idea from the CMU (Carnegie Mellon

University) simulation soccer team [42]. **Fig. 1.20**. graphically presents an example of a possible world situation.

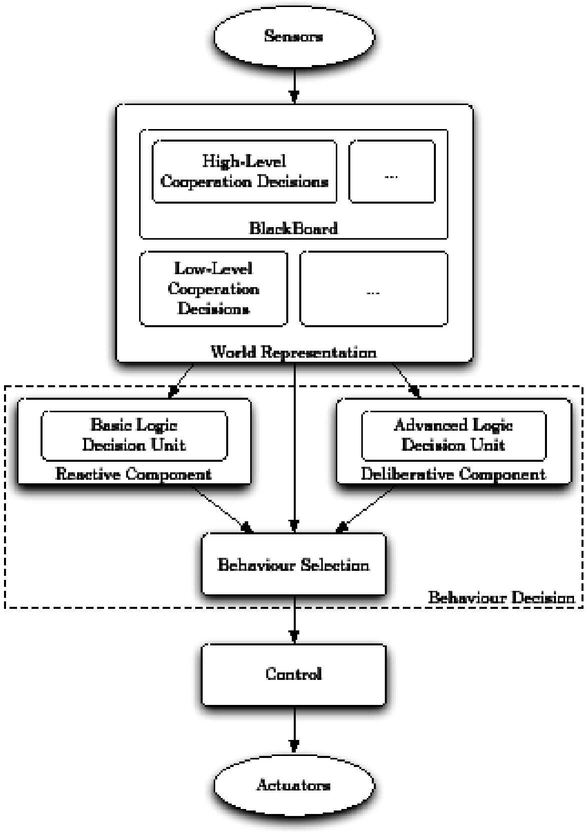


Fig. 1.19. Hybrid architecture

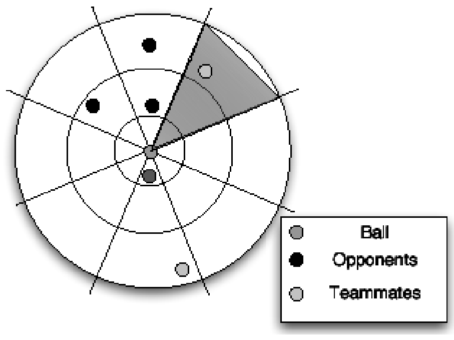


Fig. 1.20. World model

The reactive component has two main purposes: to guarantee a quick, real-time decision to be executed by the robot, and to react to unexpected events. On the robotic soccer domain, such things may happen very often: if a robot has the ball and plans to take it to the goal, and score, it needs to react to an unexpected event (like an opposite robot taking the ball away from it, the robot loosing the ball, or bumping against another robot). This component, called Basic Logic Decision Unit (BLDU) and supported on first-order logic statements, allows us to easily model the reactive behaviour of the robot. The BLDU runs in a cyclic manner and, at each iteration, it executes some operations related to the decision process and the world modeling. First, it broadcasts the status of the robot to its teammates. This includes the player role and the play mode (playing, paused, going to start position, etc). Then, it checks if it needs or wants to change its role. Finally, it decides the next behaviour to be executed (going to some place on the field, taking the ball to goal, etc).

The reactive decision process is based on a set of Prolog rules, constituted by a set of preconditions that must be all true in order for that rule to be applied. The set of rules of the BLDU has a pre-defined order, in the sense that the first rule following that order that has all pre-conditions true will be the one used to make a decision. A pre-condition is anything that can be represented by a logical formula, like having the ball, or being inside a pre-determined area of the field. In **Fig. 1.21**. there are two examples of rules used in BLDU. The first one is related with the defender role. This rule tells the control component to move the robot to a specific position on the field (given by X, Y, Theta) and it is applied if the robot is playing, it sees the ball but does not have it, the ball is outside the defenders zone, the robot is not near the position where he wants to go (a simple hysteresis), and there is not another robot in that position. The second rule is related to the attacker, and tells low-level control to score if the robot is playing, if the ball is in the attacker zone, if the robot can see the opposite goal, if it sees and have the ball, and if it is near the goal. This way it is very easy to design new behaviors, simply by adding new rules to the system. The designer just needs to be careful with the rules precedence. This is clearly an easier system to work with, compared to the state machine.

In order to compare the BLDU with the state machine implementation, we performed the following test: the robot starts facing the opposite goal, near the penalty mark. The robot must go back to the middle of the field, get the ball, return to the opposite goal and score. We applied this test ten times using each system (logic based decision system and the original state-machine decision system). The results are on **Fig. 1.22**.

```
basicBehaviour( goGeneric, Generic, X, Y,
Theta, defender) :-
  currentMode(play),
  visionSeeball,
  \+ hasBall,
  ballOutDefenderZone(now),
  plGetGenericConst(Generic),
  defenderTrackBall(X, Y, Theta, now),
  \+ xyInSideMargin(X, Y),
  \+ plRobotInPosition(X,Y).
```

```
basicBehaviour( score, 0, 0.0, 0.0,
0.0, attacker) :-
  currentMode( play ),
  \+ ballOutAttackerZone(now),
  \+ stateFinished( score ),
  visionSeeOthergoal(true),
  visionSeeball,
  hasBall,
  nearGoal.
```

Fig. 1.21. Two example rules of the Basic Logic Decision Unit

Not surprisingly the results obtained are similar for both systems, since the low-level control unit is the same. For the robot performance on this test, the control unit is much more important than which tool is used for behavior switching. The logic-based system has showed to be fast enough (even a little faster than the state machine) to handle the low-level control problems, like behavior switching, with the right timings. So, we concluded that the BLDU might replace the state machine without negative consequences, and leave room for many improvements like the one described next.

Decision System	Ball Losses	Path Length (m)	Time (s)	Ball Out
Logic	0.8	9.28	30.5	0.2
State Machine	0.7	9.40	33	0.1

Fig. 1.22. Average results for the tests using both decision systems

One good example of the simplicity and power of the BLDU is the management of player roles. We defined three roles for our players: `attacker`, `defender` and `full player`. It would be very hard to model all these behaviours using a state machine, but it would be even harder to switch roles in real-time, like BLDU does. This dynamic role switching is also an example of the kind of cooperation we intend to have with our architecture. The robot keeps checking if it needs or wants to switch roles. The need comes from two situations: if a robot stops playing (for instance, due to a referee decision, or a software crash), or if another robot decides to change its role. In the latter case, it may be necessary to switch roles in order to keep the strategy of the team.

But when the robot decides it wants to change roles? Imagine a `defender` in a situation that the ball enters the defensive midfield. It will try to approach it, and take it to the opponent goal. But, trying to score a goal is an `attacker`'s task, and, moreover, it would leave the team with no one protecting the goal. So, one of the teammates will switch to `defender` and go back in the field, protecting the goal again. The old `defender` may become an `attacker` after a possible shot; it does not need to return to its old position and lose time and battery power. **Fig. 1.23**. shows a dynamic change: when the black robot (`defender`) catches the ball at the defense, and takes it to the opponents' goal, the gray robot (`attacker`) switches to `defender` to replace his teammate.

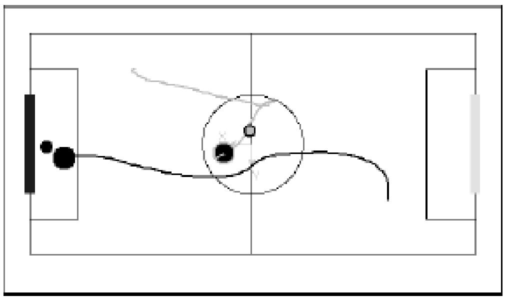


Fig. 1.23. Dynamic role exchange

The Deliberative Component, called Advanced Logic Based Unit (ALBU), is responsible to determine plans (sequences of behaviours) that allow the team to achieve something (like scoring on the opposite goal). The development of the ALBU has been made in GOLOG, a language built on top of Prolog with the purpose of programming intelligent robot behaviour.

This language allows us to use situation calculus in order to produce plans. The language semantics is defined through a macro-expansion into sentences of the situation calculus. GOLOG offers significant advantages over current tools for applications in dynamic domains like the high level programming of robots and software agents, process control, discrete event simulation, complex database transactions, etc.

More importantly, GOLOG programs are evaluated with a theorem prover. The user supplies precondition axioms, one per action, successor state axioms, one per fluent, a specification of the initial situation of the world, and a GOLOG program specifying the behavior of the agents in the system. Executing a program amounts to finding a ground situation term σ such that:

$$\text{Axioms} \models \text{Do}(\text{program}, S_0, \sigma).$$

i.e., the fluent Do(program, S_0, σ) is derivable from the axioms.

This is done by trying to prove

Axioms $|= (\exists_s)$ Do(program, S_0, s),
i.e., there exists a situation s such Do(program, S_0, s) is true. If a constructive proof is found, such a ground term do(a_n, ...do(a_2, do(a_1, S_0))...) is obtained as binding for the variable s, where S_0 denotes the initial situation. Then the sequence of actions [a_1, a_2, ..., a_n] is sent to the primitive action execution module.

Our objective was to develop a tool capable of planning and control task execution in a distributed environment. To do so we assumed that: the agents (robots) can generate, change and execute plans; a plan can be generate, and executed by one or more agents; decisions over the generated plans are based on hypotheses, i.e., assumptions over future states that cannot be guaranteed; and the agents have the capacity to communicate among them, and share information about plans or environment states. Since the GOLOG programming logic is oriented to a single agent we cannot apply it directly, rather we will have to be careful with the task synchronization among plan tasks, among team members, and take into account that our world model is based on sensors and information shared among the team members, and does not change only as result of our actions, since there are also other agents that can affect the environment. This last problem will be addressed in future work, for now when the environment is affected by an action on other agent in such a way that makes the plan invalid, the agent has to generate a new plan.

The Behaviour Selection (BS) module chooses between the decisions produced by Deliberative and Reactive components. It also handles plan execution and checks if it is still valid. If a plan is no longer valid (due to an action pre-condition being no longer true), it will discard the plan and use the reactive decision. This way, the robot may actually react to an unexpected event.

An example of a cooperative plan determined by the deliberative component for a soccer game situation where we have two robots, both starting at their mid-field, but the ball is near the opponent goal, is given next.

The plan for robot 1, denoted “bp”, is:

```
[ actionPass(bp,ph),
actionWaitFor(bp,ph,actionGo2Goal(bp)),
actionWaitFor(bp,bp,actionGetClose2Ball(bp)),
actionGetClose2Ball(bp) ]
```

and the plan for robot 2, denoted “ph”, is:

```
[ actionScore(ph),
actionWaitFor(ph,ph,actionTakeBall2Goal(ph)),
actionTakeBall2Goal(ph),
actionWaitFor(ph,ph,actionGo2Goal(ph)),
actionGo2Goal(ph),
actionHelp(ph,bp) ]
```

1.5.2 Distributed Planning and Coordinated Execution

The work described next was developed in the context of the RESCUE project, which aims at the development of novel methodologies for using robotic teams in rescue operations. Typically, a rescue operation within a situation of catastrophe involves several and different rescue elements (individuals and/or teams), none of which can effectively handle the rescue situation by itself. Only the cooperative work among all those rescue elements may solve it. Considering that most of the rescue operations involve a certain level of risk for humans, depending on the type of catastrophe and its extension, it is understandable why robotics can play a major role in Search and Rescue situations (S&R), especially teams of multiple heterogeneous robots.

The overall goal of the RESCUE project is to develop a robotic team, constituted by more than one robot, capable of autonomously handle a rescue operation. This project can be seen at different levels of abstraction, such as a technological level (e.g., hardware development), a level of control (e.g., motor control), a level of robot navigation, and a level of task planning, if an individual robot is considered. If we assume also the existence of a team of robots, new levels must be added, for instance a level of robot cooperation and a level of mission management. At these levels, the objectives are making robots cooperate to fulfill their common goals, both through cooperative planning and cooperative execution.

This work is mainly focused on the problem of distributed planning and task allocation in a multi-robot rescue system, assuming that teamwork (i.e., cooperative tasks) plays an important role on the overall planning system. However, all considerations, related with technology and utilization of real robots, were not an issue in this work. So our rescue team is composed of agents, virtual entities interacting within a simulated environment and capable of some intelligent actions, both individual and cooperative.

For that, an agent architecture has been developed, inspired on a Belief-Desire-Intention (BDI) architecture, considering that each agent interacts with others in the same rescue scenario, with the same interface and ontology. Moreover, the proposed architecture takes into account issues as agent heterogeneity, failures recover, cooperation, to name but a few. Besides that, agents equipped with this architecture are prepared to act in a non deterministic environment (where its state could change without any agent action), incomplete (meaning that only information agents have is acquired by their sensors which provided only incomplete data about the environment state), dynamic (meaning that planning decisions made for a certain environment state could be invalid when they are executed, claiming for some re-planning).

Since teamwork is a key aspect of this work, agents need to negotiate the execution of certain actions, either because an agent does not have the right skills to do it, or it evaluates that another agent could do it better (with a lower cost). To implement this, a Contract-Net system was developed and integrated in the agent architecture. This system allows agents to propose and negotiate contracts with other agents, and gives the necessary guarantees for maintaining "signed" contracts consistency (i.e., if an agent cannot fulfill a contract it must inform others involved in that contract).

The main decision process, the planner, was implemented based on a Hierarchical Decomposition Partial Order Planner (HDPOP) approach, with an important extension, the possibility to handle (plan) the resources needed for each of the tasks. The planner was developed using the STRIPS language and is supported on a variation of the well-known A* search algorithm, the Iterative Deepening A* (IDA*).

To experiment and evaluate the proposed planning system, a simplified version of a rescue simulator was also developed. This simulator allows creating virtual rescue scenarios where rescue teams should face building and forest fires, civilians trapped in collapsed buildings, and roads blocked. The rescue teams are composed of aerial and land robots, with different skills. The former could perform a survey of the affected region and are also capable of transporting victims to rescue spots. The latter may be a civil protection (CV) agent (responsible for organizing the rescue missions and contracting other agents), physician agent (capable of giving first aid assistance), firefighters or an agent capable of removing roadblocks.

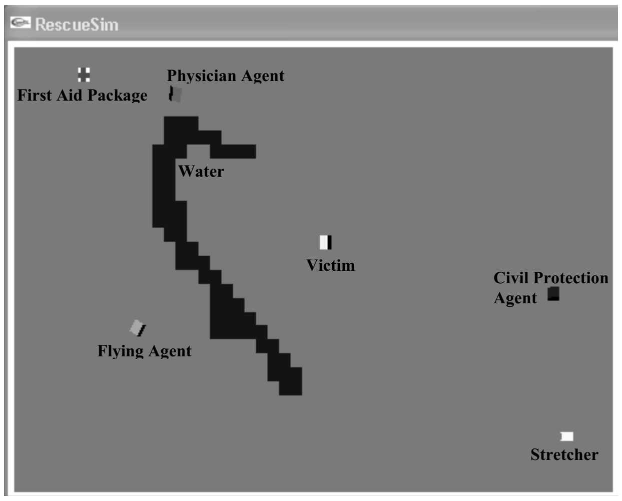
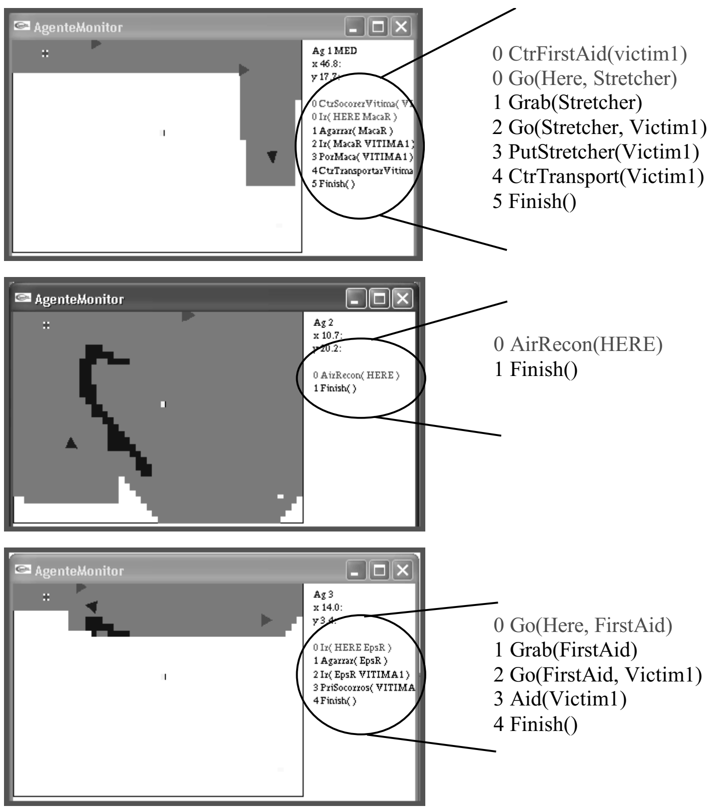


Fig. 1.24. A simulation scenario with three agents: a civil protection agent, a physician agent and a flying agent. There are also a first aid package and stretcher needed to rescue the victim

Fig. 1.24. presents a simulated rescue scenario where three agents have to cooperate to search and rescue a victim. After the victim being found, the CV agent generates a plan to rescue it, which includes contracting the physician agent, finding and getting the stretcher, putting the victim on it, and contracting the flying agent to transport the victim to a rescue spot. The physician agent makes a plan to get the first aid package, go near to and help the victim, and the flying agent generates a transport plan.

Fig. 1.25. shows the state of the three agents immediately after the victim has been found by the flying agent.

In general, the results obtained show that a distributed approach to a rescue problem is clearly an interesting solution when compared with a centralized one. One might lose some quality of the planning solutions, but gains more flexibility, redundancy and the possibility of parallelizing the planning process. One key word emerging from this work and its results was "delegation", meaning that agents should delegate as much as possible given other agents skills, particularly whenever planning is concerned.



0 CtrFirstAid(victim1)
0 Go(Here, Stretcher)
1 Grab(Stretcher)
2 Go(Stretcher, Victim1)
3 PutStretcher(Victim1)
4 CtrTransport(Victim1)
5 Finish()

0 AirRecon(HERE)
1 Finish()

0 Go(Here, FirstAid)
1 Grab(FirstAid)
2 Go(FirstAid, Victim1)
3 Aid(Victim1)
4 Finish()

Fig. 1.25. The state of the three agents immediately after the victim has been found by the flying agent (on the right is shown the current plan of each agent). Ag. 1 is the civil protection agent; ag. 2 is the flying agent and ag. 3 is the physician agent. The green areas represent the regions already explored by each agent; the red lines in the plans indicate the action(s) under execution

1.5.3 Relational Behaviours in Cooperative MRS

Our research on relational behaviours has been mainly driven by the application to soccer robots, but the motivation comes from the need to design, implement and test in real robots concepts from teamwork theory, originally developed for multi-agent systems.

One cooperation mechanism that we first implemented in 2000 consists of avoiding that two or more robots from the same team attempt to get the ball. A relational operator was developed to determine which robot should go to the ball and which one(s) should not. In the current implementation, each robot that sees the ball and wants to go for it uses a heuristic function to determine a fitness value. This heuristic penalizes robots that are far from the ball, are between the ball and the opposite goal and need to perform an angular correction to centre the ball with its kicking device. Each robot broadcasts its own heuristic value, and the robot with the smallest value is allowed to go for the ball, whereas the others execute a `Standby` behaviour. Another example of utilization of this mechanism is the decision to dynamically switch roles among players, e.g., the `defender` becomes an `attacker` when it acquires the right to get the ball, and correspondingly the `attacker` becomes a `defender`.

A relational behaviour is not seen in our research as a simple matter of relating the tasks performed by two or more robots from the team. We support relational behaviours on *teamwork theory* techniques, such as the Joint Commitment Theory (JCT) [7]. One such example is the implementation of a ball pass between two robots [46]. These behaviours have a general formulation based on the JCT and use the individual robot navigation methods. The robots are capable of committing to relational pass behaviour where one of the robots is the kicker and the other the receiver. If any of the robots ends the commitment, the other switches to an individual behaviour.

In **Fig. 1.26**., several individual behaviours can be found within the commitment. At any time the participants have to select the correct behaviour individually. Commitments among teammates are established at the *relational behaviour* level of the architecture described in Section 1.2. Behaviour selection is done in the *logic machine* module of the hybrid logic-based decision system explained in sub-section 1.5.1. The robot first chooses a *role*, next it selects a *commitment*, and finally the individual behaviour.

Predefined logical conditions can establish a *commitment* between two robots. Once a robot is committed to a relational behaviour, it will pursue this task until one or more conditions become false, or until the goal has been accomplished. The initiative for a relational behaviour is taken by one of the robots, which sets a *request* for a relational behaviour. A potential partner checks if the conditions to *accept* the request are valid. If so, the commitment is established. During the execution of the commitment the changing environment can lead to failure or success at any time. In that case the commitment will be ended.

In general, within a commitment three phases can be distinguished: `Setup`, `Loop` and End. During the set up and ending of a commitment, a robot is not executing a relational behaviour. The *logic machine* will not select any relational behaviour, and no commitment takes place during the primitive behaviour selection. The participant robots will select the behaviours concerning the commitment to achieve their joint goal only during the `Loop` phase. This selection process will now be explained for the `Pass` example in **Fig. 1.26**. Three individual behaviours can be found in the figure diagram: `standBy` for both participants, `aimAndPass` for the kicker, and `intercept` for the receiver.

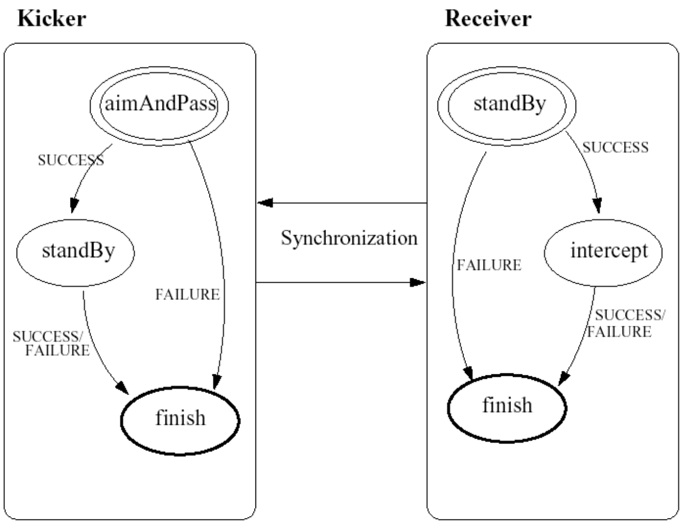


Fig. 1.26. Diagram representing the relational behaviour `Pass` resulting of the teamwork between the kicker and receiver robots

The pass commitment has been split up in several states, referred to as *commitment states*:

- *request* and *accept* in the `Setup` phase.
- *prepare* and *intercept* in the `Loop` phase.
- *done* and *failed* in the `End` phase.

Table 1.1. - Behaviour selection for all pass commitment states

phase	Setup		Loop		End	
state	*request*	*accept*	*prepare*	*intercept*	*done*	*failed*
Kicker	-	-	aimAndPass	standBy	-	-
Receiver	-	-	standBy	intercept	-	-

In general, the states in the `Setup` and `End` phase will be the same for any commitment, other than the `Pass` example given here. The `Loop` phase, however, is problem-dependent. Splitting it in several states allows the synchronized execution of the relational behaviour. Each commitment state is linked to (a set of) behaviours for both robots, as listed in Table 1.1. When the commitment proceeds as planned, the pass states will be run through sequentially, from *request* until *done*. An error at any time can lead to the state *failed*. New commitments for the same or another application can be created under the same framework.

To synchronize the behaviours, the participants use explicit (wireless) communication. Four variables, containing the identities of the participants and their commitment states, are kept in each participant version of the blackboard. Each of these four variables will be sent to the other participants in the relational behaviour when it is changed.

1.5.4 Optimal Decision Making for MRS Modelled as Discrete-Event Systems

Though not tested yet in real robots, formal work on Stochastic Discrete-Event Systems modelling of a multi-robot team has been carried out within our soccer robots project [10]. The environment space and each player (opponent and teammate) actions are discretized and represented by several finite state automaton models. Then, all finite state automata are composed to obtain the complete model of a team situated in its environment and playing an adversarial 2 vs. 2 player game. An example of several automata and their composition for this example is depicted in **Fig. 1.27**. Controllable (e.g., `shoot_p1`, `stop_p2`) and uncontrollable (e.g., `lost_ball`, `see_ball`) events (i.e., our robots actions) are identified, and exponential distributions are assigned to the uncontrollable event inter-event times.

Dynamic programming is applied to the optimal selection of the controllable events, with the goal of minimizing the cost function

$$\min_{\pi}\left[\int_0^{\infty} C\big[X(t),u(t)\big]dt\right]$$

where π is a policy, *X(t)* the game state at time *t*, and *u(t)* is a controllable event, with the cost of unmarked states equal to 1, and all the other states having zero cost. If the only marked states are those where a goal is scored for our team, and there are no transitions from marked to unmarked states, this method obtains the minimum (in a stochastic sense) time to goal for our team, constrained by the opponent actions and the uncertainty of our own actions. Some of the chosen actions result in cooperation between the two robots of the team.

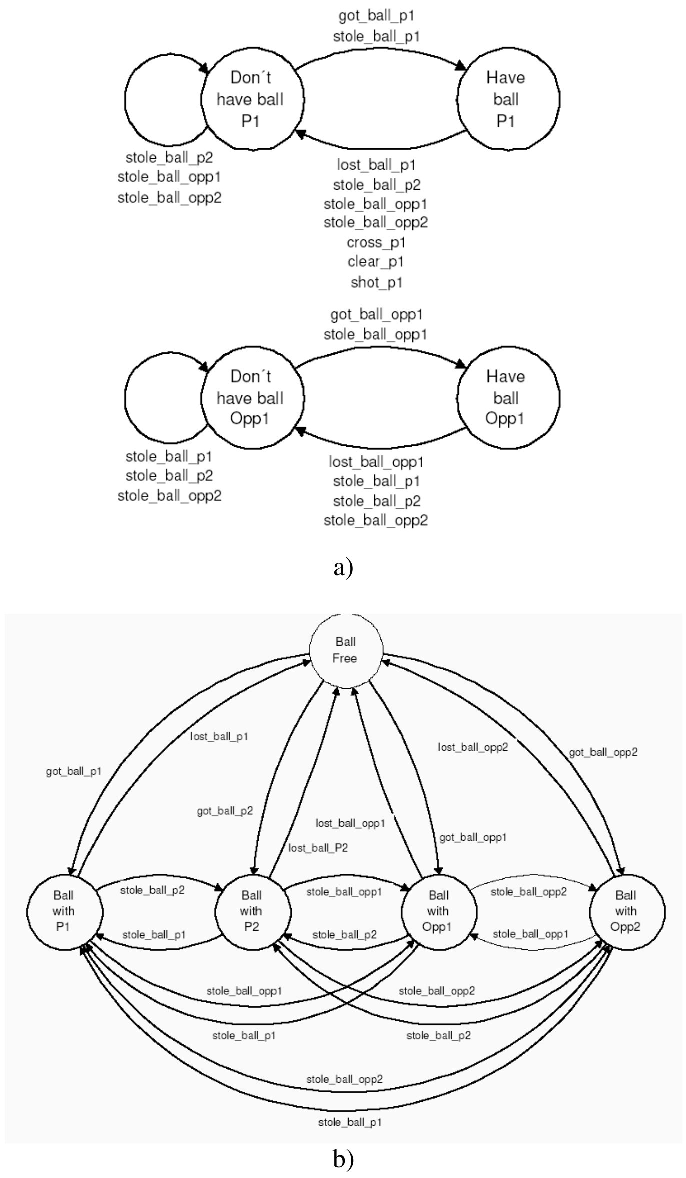


Fig. 1.27. Ball possession model: (**a**) at the top, player 1 ball possession model; at the bottom, opponent player 1 ball possession model; (**b**) finite state automaton that models the overall game ball possession, resulting from the parallel composition of the models in (a) for two players per team

1.6 Emotions and Multi-Robot Systems

Adequate decision-making under difficult circumstances (in unpredictable, dynamic and aggressive environments) raises some interesting problems from the point of view of the implementation of artificial agents. At the first sight, in such situations, the agent should (ideally) be capable of performing deductive reasoning rooted on well-established premises, reaching conclusions using a sound mechanism of inference, and acting accordingly. However, in situations demanding urgent action such an inference mechanism would deliver "right answers at the wrong moment." To circumvent this problem, some architectures have been proposed (e.g., reactive [5], hybrid [17]) together with planning algorithms capable of providing courses of action within limited lapses of time.

An interesting alternative mechanism of decision-making can be found in mammals which, when confronted with severe, demanding situations, respond "emotionally" to solve difficult problems. Unfortunately, the nearness between urgency and emotion has supported the common-sense belief that emotions should not play an important role in everyday rational decision-making.

However, recent research findings on the neurophysiology of human emotions suggest that human decision-making efficiency depends deeply on the emotions machinery. In particular, the neuroscientist António Damásio [9] claims that alternative courses of action in a decision-making problem are (somatically) marked as good or bad, based on an emotional evaluation. Only the positive ones (a smaller set) are used for further reasoning and decision purposes. This constitutes the essence of the Damásio's somatic marker hypothesis, where the link between emotions and decision-making is suggested as particularly strong for the personal and social aspects of human life. The Damasio's research has demonstrated that even in simple decision-making processes, the mechanism of emotions is vital for reaching adequate results. In another study about emotions, conducted by the neuroscientist Joseph LeDoux [19], it is recognized the existence of two levels in the sensorial processing, one quicker and urgent, and another slower but more informed.

Emotions have been considered, for decades, as something that lies on the antipodes of rationality. As a matter of fact, emotional behavior has been thought as characteristic of irrational animals and so should be avoided by human beings when reaching a certain degree of "perfection." However, consider the competence of certain mammals as dogs or cats: they not only survive in a demanding environment but they also perform tasks, learn, survive, adapt themselves and make adequate decisions even when faced with unfamiliar situations. And certainly they do not reason (at least in the sense that is accepted by the artificial intelligence community infer new knowledge, verbally represented, from existing one). What these animals exhibit is a *sensory-motor* intelligence which none of our robots possesses. According to J. Piaget, sensory-motor intelligence is "essentially practical that is, aimed at getting results rather than at stating truths - this intelligence nevertheless succeeds in eventually solving numerous problems of action (such as reaching distant or hidden objects) by constructing a complex system of action-schemes and organizing reality in terms of spatio-temporal and causal structures. In the absence of language or symbolic function, however, these constructions are made with the sole support of perceptions and movements and thus by means of a sensory-motor coordination of actions, without the intervention of representation or thought." [34].

The discussion concerning the relevance of emotions for artificial intelligence is not new. In fact, AI researchers as Aaron Sloman [40] and Marvin Minsky [31] have pointed out that a deeper study of the possible contribution of emotion to intelligence was needed. Recent publications of psychology [16] and neuroscience research results suggest a relationship between emotion and rational behaviour, which has motivated an AI research increase in this area. The introduction of emotions as an attempt to improve intelligent systems has been made through different ways. Some researchers use emotions (or its underlying mechanisms) as a part of architectures with the ultimate goal of developing autonomous agents that can cope with complex dynamic environments [47, 48, 41].

The ISLab research group has been working since 1997 on developing emotion-based agent architectures that incorporate our interpretation of artificial emotions. The *DARE* architecture joins together all the concepts that we have been studied and developed related with the application of emotional mechanisms in agents (both virtual and real robots). Although this research follows a prescriptive research perspective rather than a descriptive one, the developed architecture is essentially grounded on the above mentioned theories about emotions neurological configuration and application [48, 49, 23, 44, 38].

1.6.1 Emotion-based Agent Architecture

The basic idea underneath the DARE architecture is the hypothesis that mammals process stimuli *simultaneously* under two different perspectives: a *cognitive*, which aims at finding out *what the stimulus is* (by a some rational mechanism), and another one, *perceptual*, intending to determine *what the agent should do* (by the way of extracting relevant features of the incoming stimulus). As this latter process is much more rapid (in terms of computation) than the former, the agent can react even before having a complete cognitive assessment of the whole situation.

Following the suggestions of Damásio, a somatic marker mechanism should associate the results of both processing sub-systems in order to increase the efficiency of the recognition process in similar future situations. On the other hand, the ability of anticipating the results of actions is also a key issue as the agent should "imagine" the foreseeable results of an action (in terms of a somatic mark) in order to make adequate decisions.

The DARE architecture for an individual emotion-based agent includes three levels: stimulus processing and representation, stimulus evaluation and, action selection and execution. **Fig. 1.28**. represents the architecture with the main relationships among blocks represented by solid arrows. Dashed arrows represent accessing operations to the agent's memory or body state.

The environment provides stimuli to the agent, and as a consequence of the stimulus processing the agent decides which action should be executed. During this stimulus-processing-action iterative process, decisions depend not only on the current incoming stimulus and the internal state of the agent (body state) but also on the results got from previous decisions, stored in the agent's memory.

After the reception of a stimulus, a suitable internal representation is created and the stimulus is simultaneously analysed by two different processors: a *perceptual* and a *cognitive*. The perceptual processor generates a perceptual image that is a vector that contains the values of the relevant features extracted from the stimulus. For instance, for a prey the relevant features of the predator image might be the colour, speed, sound intensity, and smell, characteristics that are particular to the corresponding predator class. The definition of what are relevant features and corresponding values is assumed to be built-in in the agent. This perceptual, feature-based image, as it is composed of basic and easily extracted features, allows the agent to efficiently and immediately respond to urgent situations. The cognitive processor uses a cognitive image that is a more complex representation of the stimulus (for instance, if dealing with visual images, a cognitive image might be an image processed using computer vision techniques to identify relevant objects in it). The cognitive processing aims at performing a pattern matching of the incoming stimulus with respect to cognitive images already stored in memory. As this processor might involve heavy computation processing, the cognitive image is not suitable for urgent decision-making.

With the two images extracted from the stimulus, the process proceeds through a parallel evaluation of both images. The evaluation of the perceptual image consists of assessing each relevant feature included in the perceptual image. From this evaluation results what is called the perceptual Desirability Vector (DV). This vector is computed in order to establish a first and basic assessment of the overall stimulus desirability. In the perceptual evaluation, the DV is the result of a mapping between the desirability of each feature and the amount of the feature found in the stimulus. The information concerning the feature desirability is assumed to be built-in, and therefore defined when the agent is created. Of course, the mentioned mapping depends on the considered species (e.g., a lion and a bull assign different desirability vector values to the same colour).

The cognitive evaluation differs from the perceptual in the sense that it uses past experience, stored in memory. The basic idea is to retrieve from memory a DV already associated with cognitive images similar to the present stimulus. Since a cognitive image is a stimulus representation including all extractable features of it, two stimuli can be compared using an adequate pattern matching method. This process allows the agent to use past experience for decision making. After obtaining the perceptual and cognitive images for the current stimulus, when the evaluation of the perceptual image does not reveal urgency, *i.e.,* the resulting DV is not so imperative that would demand an immediate response, a cognitive evaluation is performed. It consists of using the perceptual image as a memory index to search for past obtained cognitive images similar to the current cognitive one.

One of the purposes of using perceptual information to index memory of cognitive images is to reduce search. It is hypothesized that it is likely to have the current cognitive image similar to others with the same dominant features. Each cognitive image in memory, besides having an associated perceptual image, also has the resulting DV from past evaluation. If the agent has been already exposed to a similar stimulus in the past, then it will recall its associated DV, being the result of the cognitive evaluation. This means that the agent associates with the current stimulus the same desirability that is associated with the stimulus in memory. If the agent has never been exposed to a similar stimulus, no similar cognitive image will be found in memory, and therefore no DV will be retrieved. In this case, the DV coming from the perceptual evaluation is the one to be used for the rest of the processing (decision-making).

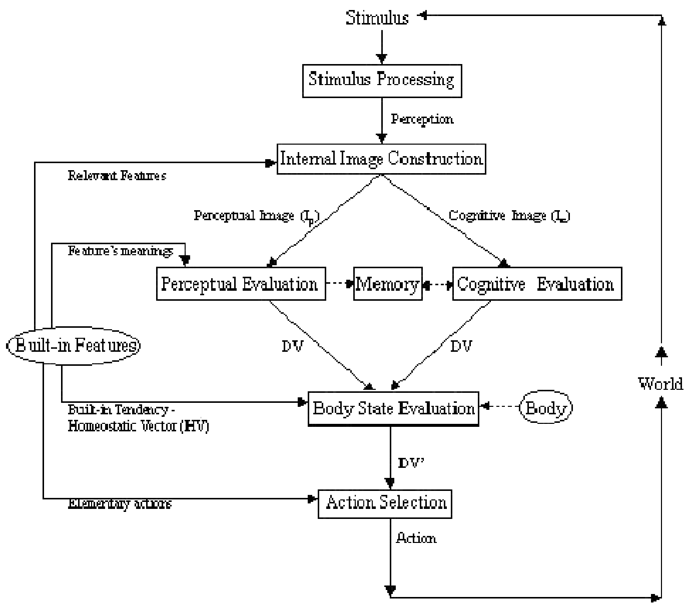


Fig. 1.28. A block diagram of the DARE architecture

In this architecture, the notion of body corresponds to an internal state of the agent, i.e., the agent's body is modeled by a set of pre-defined variables and the body state consists of their values at a particular moment. The internal state may change due to the agent's actions or by direct influence of the environment.

The innate tendency establishes the set of body states considered ideal for the agent, through the definition of the equilibrium values of the body variables – the Homeostatic Vector (HV). The built-in tendency can be oriented towards the maintenance of those values or the maximization/minimizationof some of them. In other words, this comprises a representation of the agent's needs. The body state evaluation consists of an estimate of the effects of the alternative courses of action, performing an anticipation of the possible action outcomes: "will this action help to re-balance a particular unbalanced body variable, or will it get even more unbalanced?" This action effects anticipation may induce a change on the current stimulus DV, reflecting the desirability of the anticipated effects according to the agent's needs. As the agent's decisions depend on a particular body state – the one existing when the agent is deciding, it will not respond always in the same manner to a similar stimulus. On the other hand, the existence of a body representation forces the agent to behave with pro-activeness–because its internal state drives its actions–and autonomy–because it does not rely on an external entity to satisfying its needs.

After finishing the evaluation process, the agent will select an adequate action to be executed. In the last step of evaluation the effects of all possible actions were anticipated based on the expected changes on the body state. The action with the best contribution for the agent's overall body welfare will be selected as the one to be executed next. It is assumed that there is a set of built-in elementary actions that the agent can execute. After the selected action being executed, the changes in the environment will generate a new stimulus to be processed.

The DARE architecture allowed the implementation of an autonomous agent, (i) where the goal definition results from the agent's behaviour and needs, i.e., it is not imposed or pre-defined; (ii) where the agent is capable of quickly reacting to environment changes due to the perceptual level processing; (iii) where the agent reveals adaptation capabilities due to the cognitive level processing; and finally (iv) where the agent is capable of anticipating the outcomes of its actions, allowing a more informed process of decision making.

However, as mentioned before, the link between emotions and decision-making seems particularly strong in the social aspects of human life, which is why some emotion theories, mainly in psychology, focus on the social aspects of emotion processes. The work presented in the next section tries to explore these notions and the importance of emotional physical expression on social interactions, as well as the sympathy that may occur in those interactions. The goal is to incorporate these concepts in MRS in order to improve the system efficiency and competence.

1.6.2 Emotion-based MRS

In what concerns emotion expression, it has been claimed that there is not another human process with such a distinct mean of physical communication, and more interesting it is unintentional. Some theories point out that emotions are a form of basic communication and are important in social interaction. Others propose that physical expression of emotion is the body preparation to act, where emotional response can be seen as a built-in action tendency aroused under pre-defined circumstances. This can also be a form of communicating to others what will be the next action. If the physical message is understood it may defuse emotions in others, establishing an interactive loop with or without actions in the middle.

The AI research concerning multi-agent systems relies mainly on rational, social and communication theories. However, the role of emotions in this field has been considered important by an increased number of researchers.

Linked to expressing emotions is the notion of sympathy defined as the human capability to recognize others' emotions. This capability is acquired by having consciousness of our own emotions. Humans can use it to evaluate others' behaviours and predict their reactions, through a mental model learned by self-experience or by observation that relates physical expression with feelings and intentions. Sympathy provides an implicit communication mean, sometimes unintentional, that favors social interactions.

In order to explore these concepts an extension of the DARE architecture for a multi-agent environment was developed [24]. The decision-making processes were extended for decision-making involving other agents. Agents represent others' external expression in order to predict their internal state, assuming that similar agents express the internal state in the same way (a kind of implicit communication). Sympathy is grounded on this form of communication, allowing more informed individual decisions, especially when these depend on others. On the other hand, it allows the agent to learn, not only by its own experience, but also by the observation of others' experience. The new DARE architecture also allows the modelling of explicit communication through the incorporation of a new layer, the symbolic layer, where relations between agents are represented and processed.

The DARE architecture was applied to an environment that simulates a simple market involving: producer agents, that own products all the time; supplier agents, that must fetch products from producers or other suppliers either for its own consumption or for selling to consumers; and consumer agents, that must acquire products from suppliers for its own consumption. Agents are free to move around the world, interact and communicate with others. Their main goal is to survive by eating the necessary products and, additionally, maximize money by selling products.

Fig. 1.29. shows a global view of the DARE architecture. Stimuli received from the environment are processed in parallel on three layers: perceptual, cognitive and symbolic. Several stimuli are received simultaneously, and they can be gathered from any type of sensor.

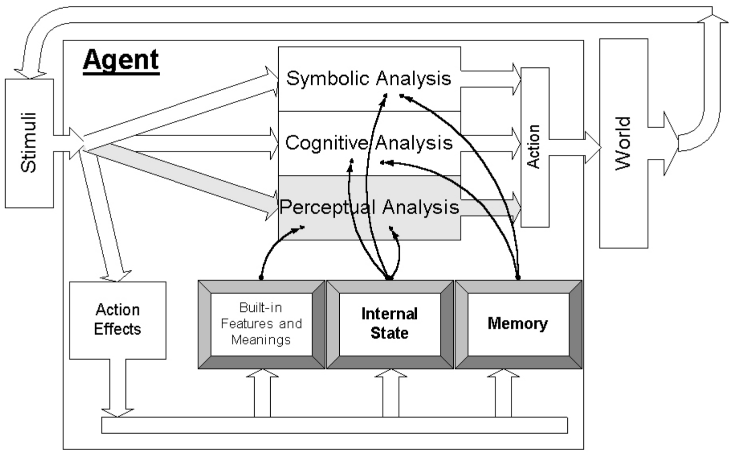


Fig. 1.29. Global view of the DARE architecture applied in a MRS environment

The perceptual and cognitive analyses are similar to the ones described in sub-section 1.6.1. The novelty is mainly in the introduction of a new level of processing – the symbolic analysis.

The symbolic layer was introduced aiming at the capture of concepts involved in communication and sympathy. This layer has the same conceptual foundation of the cognitive layer in what concerns the modelling of the somatic marker hypothesis, allowing the same kind of adaptation and learning. In the symbolic layer those concepts are applied to more abstract information extracted from stimuli in order to i) establish communication between agents, ii) represent explicitly other agents goals and interactions and iii) trigger reasoning. This extension leads eventually to emergence of imitation behaviours.

Besides possible imitation behaviours, this mechanism of observing expressions and actions of other agents also allows the agent to anticipate other agents' actions. Nevertheless, there is the possibility for the agent to make mistakes on the assessment of others' internal state. Depending on the application, an expression may not be directly mapped to a specific internal state but only to a set of internal states. Moreover, some changes of expression may not be a direct effect of the last action executed.

Overall, the extension of the DARE architecture revealed an interesting performance in the dynamic multi-agent environment where it was tested, showing similar capabilities on individual decision-making, flexibility and learning as its previous version, and new abilities to model the social role of emotions. For instance, some human-like behaviours were observed when agents interact among each others, e.g., a stealing behaviour, when a consumer agent has no money to buy products and the corresponding punishment behaviours by the other agents; an ordering behaviour, when a supplier does not have the product needed by a consumer, it tries to find another supplier or a producer that have the product in order to sell it later to the consumer agent that implicitly ordered it.

1.7 Conclusions

Cooperative Robotics within a Multi-Robot System is a modern research field, with applications to areas such as building surveillance, transportation of large objects, air and underwater pollution monitoring, forest fire detection, transportation systems, or rescue after large-scale disasters. In short, a population of cooperative robots behaves like a ``distributed'' robot to accomplish tasks that would be difficult, if not impossible, for a single robot. In this case, besides all subsystem integration problems posed by the development of a single robot capable of performing non-trivial tasks, the presence of multiple, possibly heterogeneous, robots and the need of having them behaving cooperatively establish new research challenges.

Problems such as the development of functional and software agent architectures, distributed world modelling, task planning, cooperative decision-making and social cognition have been approached mainly by the AI community for multi-agent systems. But, when a real multi-robot system is considered, many of the approaches developed by AI researchers have to be re-worked and new research topics as cooperative navigation, mutual localization and formation control emerge, as well as the problem of integrating continuous time and space with event-driven decision making and symbolic world modelling.

This chapter surveys several research problems addressed by the ISLab research group in the area of Multi-Robot Systems, building on AI concepts a Systems Theory standpoint.

However, MRS is still a young field, with many exciting challenges. Even though some steps towards mature results, especially those that provide formal methods applicable to different problems, have been taken, some of them described in this chapter, much is till to be done. Among several interesting research problems, we list here those that appear as serious candidates for our future research:

- *Cooperative Localization* in MRS based on probabilistic methods, so as to capture the uncertainty in the observation of one robot by its teammates, as well as the self-localization uncertainty. Methods other than the traditional Kalman filter and similar approaches should be sought, so as to avoid their well-known associated problems. This includes extensions to the Kalman filter and Monte-Carlo / Markov Localization like methods [14].

- *Robot Formation Guidance, Navigation and Control* is required in several situations (e.g., to keep the team robots within line of sight so as to ensure a communication path, to distribute the robots in a given area while keeping some desired team topology) and is currently a very active research field, with applications to land, aerial and underwater robots, but especially to space robots, where multi-aperture telescopes can be built based on several sub-telescopes carried by robotic spacecrafts tightly keeping their relative distances and orientations, instead of large unfeasible monolithic solutions, achieving the same resolution in remote sensing. A particular interesting topic concerns non-rigid formations, especially robot formations deformable in the presence of obstacles or in the presence of near collision situations.
- In adversarial environments, *modelling the opponent behaviour* (e.g., using Hidden Markov Models) is, *per se*, an interesting problem. Furthermore, opponent models are of utmost importance to solve problems modelled as stochastic games [4].
- *Cooperative Reinforcement Learning* is a vast field where we have already adventured, with results not reported here. The cooperation among robots from a team to explore an initially unknown environment raises several interesting questions, such as how to maximize learning minimizing communications, or what information should be shared (world models, policies, state evaluations?). Another approach is to model the problem as a stochastic game. In adversarial environments, it is important to have a model of the opponent behaviour to reach desired equilibria where our team wins. But even in non-adversarial games the optimal solution for a team may be an equilibrium point that must be learned.
- *Distributed Knowledge Representation and Reasoning under Uncertainty* is a relevant research issue for developing multi-agent systems, especially when the agents are real robots. An agent is typically situated some environment and usually carries are presentation or some prior knowledge of it. One of its goals should be keeping the best possible representation of the environment given its a priori knowledge and observations it makes on the environment. However, as some aspects of it are often unobservable and must be estimated indirectly, the relations among environment events are uncertain, the observations may be imprecise, ambiguous or noisy, and the agent might not have adequate

resources to observe or process all events, the reasoner's task is not one of deterministic inference but rather uncertain reasoning. One methodology that may be useful for this purpose is probabilistic reasoning based on Bayesian Networks.

- The mechanism of emotions seems to play two main roles: a *decisional*, influencing the way agents assess situations and make decisions, and a *communicational*, affecting the way agents express their internal state when confronted with their environment. In what concerns the decisional aspect the relevant issue is to determine whether the framework suggested by Damásio helps in the development of more competent agents. The experimental work with an agent equipped with the DARE architecture has shown some interesting characteristics: *purposefulness*, as the agent is capable of finding ways to fulfill its needs; *self-preservation*, as it is able to survive and circumvent threats; *efficiency*, as the mechanism of decision making is quick (in terms of computation); *learning*, as the agent is capable of learning useful associations which have improved its performance; *flexibility*, as the agent exhibits a different competence when faced with a differing instance of the environment. However, some aspects of this research have not yet been explored, namely: i) its application with multiple real robots, ii) allow agents to anticipate action effects on a long-range basis, and iii) incorporate rational (logical) inference mechanisms.

References

1. Arkin, R., (2002), "MissionLab: Multiagent Robotics Meets Visual Programming", Working notes of Tutorial on Mobile Robot Programming Paradigms, ICRA 2002, Wasington DC, USA.
2. Balch, T., (2002), "The TeamBots Environment for Multi-Robot Systems Development", Working notes of Tutorial on Mobile Robot Programming Paradigms, ICRA 2002, Wasington DC, USA.
3. Balch, T., Parker, L., editors (2002), *Robot Teams: From Diversity to Polymorphism*, A. K. Peters
4. Bowling M., Veloso, M., (2000), "An Analysis of Stochastic Game Theory for Multiagent Reinforcement Learning", Technical Report CMU-CS-00-165.
5. Brooks R., "Intelligence without representation". Artificial Intelligence, 47:139–159, 1991.
6. Cassandras, C.G., Lafortune, S., (1999), *Introduction to Discrete Event Systems*, The Kluwer International Series On Discrete Event Dynamic Systems. Kluwer Academic Publishers.
7. Cohen, P. R., Levesque, H. J., (1991), "Teamwork". Nous, Vol 35, pp. 487-512.

8. Desai, J., Kumar, V. Ostrowski, J., (1999), “Control of Changes in Formation of Multi-Robot Teams”. *In Proceedings of the 1999 International Conference on Robotics and Automation*, Detroit, USA.
9. Damásio, A. (1994), *Descartes’ Error: Emotion, Reason and the Human Brain*. Picador.
10. Damas, B., and Lima, P., (2004), "Stochastic Discrete Event Model of a Multi-Robot Team Playing an Adversarial Game", *5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles - IAV2004*, Lisboa, Portugal.
11. Drogoul, A., and Dubreuil, C. (1993), “A Distributed Approach to n-puzzle Solving”, *Proceedings of the Distributed Artificial Intelligence Workshop*
12. Durrant-Whyte, H. F., (1988), *Integration, Coordination and Control of Multi-Sensor Robot Systems*, Kluwer Academic Publishers, 1988.
13. Esposito, J. M., Kumar, V., (2002), “A Hybrid Systems Framework for Multi-robot Control and Programming”, Working Notes of Tutorial on Mobile Robot Programming Paradigms, ICRA 2002, Wasington DC, USA.
14. Fox, D., Burgard, W., Kruppa, H., Thrun, S. (2000), “A Probabilistic Approach to Collaborative Multi-Robot Localization”, Autonomous Robots, 8(3).
15. Frazão, J., and Lima, P., (2004), "Agent-Based Software Architecture for Multi-Robot Teams", *5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles - IAV2004*, Lisboa, Portugal.
16. Goleman, D. (1995), *Emotional Intelligence*, Bantam Books.
17. Kaelbling, L. P., (1986), “An Architecture for Intelligent Reactive Systems”. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning about Actions and Plans – Proceedings of the 1986 Workshop.*
18. Kosecka, J., Bajcsy R., and Christensen H. I. (1995), “Discrete Event Modeling of Visually Guided Behaviors”, International Journal on Computer Vision, Vol. 12, No.3, pp. 295-316
19. LeDoux, J., (1996) *The Emotional Brain*, Simon and Schuster.
20. Levesque, H., Reiter, R., Lesprance, Y., Lin, F., Scherl, R., (1997), “Golog: A Logic Programming Language for Dynamics Domains”. Journal of Logic Programming.
21. Lima, P., Ribeiro, M. I., Custódio, L., and Santos-Victor, J. (2003), “The RESCUE Project - Cooperative Navigation for Rescue Robots”, *ASER'03 - 1st International Workshop on Advances in Service Robotics,* March 13-15, 2003 - Bardolino, Italy.
22. Liu, J., Wu, J., editors (2001), *Multi-Agent Robotic Systems*, The CRC Press International Series on Computational Intelligence
23. Maçãs, M., Ventura, R., Custódio, L. and Pinto-Ferreira, C., (2001), “Experiments with an emotion-based agent using the dare architecture”, *Proceedings of the AISB'01 Symposium on Emotion, Cognition, and Affective Computing*, pages 105-112.
24. Maçãs, M., and Custódio, L., (2003), “Multiple Emotion-Based Agents using an Extension of DARE Architecture”, INFORMATICA, an International Journal of Computing and Informatics, Special Issue on Perception and Emotion Based Reasoning, pp. 185-196, Volume 27, Number 2.

25. Marques, C., and Lima, P., (2004), "Multi-Sensor Navigation for Non-Holonomic Robots in Cluttered Environments", IEEE Robotics and Automation Magazine, September 2004.
26. Melo, F.A., Lima, P., Ribeiro, M.I., (2004a), "Event-driven Modelling and Control of a Mobile Robot Population", *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8)*, Amsterdam, The Netherlands.
27. Melo, F.A., Ribeiro, M.I., Lima, P., (2004b), "Navigation Controllability of a Mobile Robot Population", *Proceedings of the RoboCup2004 Symposium*, Lisbon, Portugal.
28. Melo, F.A., Ribeiro, M.I., Lima, P., (2004c), "Blocking Controllability of a Mobile Robot Population", Technical Report RT-601-04, Institute for Systems and Robotics, Lisbon, Portugal.
29. Milutinovic, D., Lima, P., (2004), "Modeling and Control of a Large Size Robotic Population", *submitted to the* IEEE Transactions on Robotics.
30. Milutinovic, D., Carneiro, J., Athans, M., Lima, P., (2003), "A Hybrid Automata Model of TCR Triggering Dynamics", *Proceedings of the 11th Mediterranean Conference on Control and Automation,* MED 2003, June, Rhodes, Greece.
31. Minsky, M. (1988), *The society of Mind*. Touchstone.
32. Parker, L. (1998), "ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation", IEEE Transactions on Robotics and Automation*,* Vol. 14, No. 2, April
33. Perelson, A., Weisbuch, G., (1997), "Immunology for Physicists", Reviews of Modern Physics, Vol.69, No.4, 1219-1267, October.
34. Piaget, J., and Inhelder, B. (1969), *The Psychology of the Child.* Basic Books, Inc.
35. Pinheiro, P., and Lima, P. (2004), "Bayesian Sensor Fusion for Cooperative Object Localization and World Modelling", *8th Conference on Intelligent Autonomous Systems (IAS-8)*, Amsterdam, The Netherlands, May 2004.
36. Pires, V., Arroz, M., and Custódio, L. (2004), "Logic Based Hybrid Decision System for a Multi-robot Team", *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8)*, Amsterdam, The Netherlands.
37. Reiter, R. (2001), *Knowledge in Action*. MIT Press.
38. Sadio, R., Tavares, G., Ventura, R., and Custódio, L. (2001), "An emotion-based agent architecture application with real robots". In Emotional and Intelligent II: The Tangled Knot of Social Cognition – 2001 AAAI Fall Symposium.
39. Schmitt T., Hanek, R., Beetz, M., Buck, S., and Radig, B. (2002), "Cooperative Probabilistic State Estimation for Vision-Based Autonomous Moobile Robots", IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, October
40. Sloman, A., and Croucher, M. (1981), "Why robots will have emotions". In *Proc. 7th Int. Joint Conference on AI*, 197–202.
41. Staller, A., and Petta, P., (1998), "Towards a tractable appraisal-based architecture". In D. Cañamero, C. Numaoka, and P. Petta, editors, *Workshop:*

Grounding Emotions in Adaptive Systems Int. Conf. of Simulation of Adaptive Behaviour, from Animals to Animats, SAB'98, pages 56–61.

42. Stone, P., Riley, P. and Veloso, M., (1999). The CMUnited-99 Champion Simulator Team. In Veloso, M.; Pagello, E.; and Kitano, H., eds., RoboCup-99: Robot Soccer World Cup III. Berlin: Springer Verlag.
43. Tabuada, P., Pappas, G., Lima, P., (2005), "Motion Feasibility of Multi-Agent Formations", *accepted for publication in the* IEEE Transactions on Robotics.
44. Thrun, S., Fox, D., Burgard, W.,and Dellaert, F. (2001), "Robust Monte Carlo Localization for Mobile Robots", Artificial Intelligence, Vol. 128, No. 1/2, pp. 99-141
45. Vale, P. and Custódio, L., (2001), "Learning individual basic skills using an emotion-based architecture". Proceedings of the AISB'01 Symposium on Emotion, Cognition and Affective Computing, pages 105-112.
46. Vecht, B., Lima, P., (2004), "Formulation and Implementation of Relational Behaviours for Multi-Robot Cooperative Systems". *Proceedings of RoboCup 2004 Symposium*, Lisbon, Portugal.
47. Velásquez, J., (1998), "When robots wheep: Emotional memories and decision-making". In *Proceedings of AAAI-98*, pages 70–75. AAAI.
48. Ventura, R., Custódio, L., and Pinto-Ferreira, C., (1998a), "Artificial emotions - goodbye mr. spock!", In *Proceedings of the 2nd Int. Conf. on Cognitive Science*, pages 938–941.
49. Ventura, R.; Custódio, L., and Pinto-Ferreira, C., (1998b), "Emotions - the missing link?", In Cañamero, D., (Editor), Emotional and Intelligent: the Tangled Knot of Cognition. 1998 AAAI Fall Symposium, pages 170--175. AAAI.
50. Ventura, R., Aparício, P., Marques, C., Lima, P., Custódio, L., (2000), "ISocRob - Intelligent Society of Robots", Team Description Paper *in RoboCup-99: Robot Soccer World Cup III*, Springer-Verlag, Berlin, 2000
51. Weigel, T., Gutmann, J.-S., Dietl, M., Kleiner, A., and Nebel, B. (2002), "CS Freiburg:Coordinating Robots for Successful Soccer Playing", IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, October

2 Vision-Based Autonomous Robot Navigation

Quoc V. Do[1], Peter Lozo[2] and Lakhmi C. Jain[1]

1. Knowledge-Based Intelligent Engineering Systems Centre, University of South Australia, Mawson Lakes, S.A. 5095, Australia
2. Weapons Systems Division, Defence Science and Technology Organisation, P.O. Box 1500, Salisbury, S.A. 5108, Australia

2.1 Introduction

In the last three decades, there has been a rapid increase in the development of vision-based autonomous robots due to the advancement in computer technology. The ability to achieve real-time image processing was once considered as a pipe-dream is now made possible. However, the challenge still remains in the area of extracting relevant navigational data from 2-D image representations of the 3-D external environments that are robust against image distortions, occlusions and environmental conditions. Despite the challenge, autonomous robots today have found their way into our everyday life through commercially available robots such as autonomous vacuum cleaner robots, lawn mower robots, pool cleaner robots, robots toys [1] and prominently planetary exploration robots such as a robots series in the Mars Pathfinder project [2, 3]. These prominent examples of successful commercial and exploration robots are the results of thousands of researcher's contributions worldwide.

In general, vision-based robots must have a vision system to sense and observe the external environment. They may equip with additional sensors such as infrared, ultrasonic, laser and GPS to enhance their environmental perceptions. In contrast to vision systems, three major branches currently under intensive research are monocular, omnidirectional and stereo vision systems. Monocular and omnidirectional vision systems both consist of a single camera, while stereo vision systems have two cameras. The major difference among these vision systems is the way in which images are captured. Monocular and omnidirectional vision systems process single images obtained from a camera. Images obtained from the camera are 2-D images, which represent the 3-D external environment. As a result, depth information cannot be recovered from these images alone. Normally, additional sensors are required to help recover depth information. Furthermore, the major difference between monocular vision and omnidirectional vision systems is in the camera arrangement. In monocular vision, the camera is mounted horizontally with a field of view less than 180 degrees in front of the robot. Omnidirectional vision systems on the other hand, the camera is mounted vertically, pointed upward at a convex mirror [4, 5]. This arrangement allows the camera to observe a 360 degrees field of view around the robot. However, the images retrieved have a much lower resolution than in monocular vision systems.

Stereo vision systems on the other hand are completely different to monocular and omnidirectional vision systems. They are inspired by human vision system and designed to mimic the functioning of human's eyes, using a pair of images from two specially mounted cameras [6, 7]. Due to the vast nature of vision-based robots, this chapter limits to the discussion of monocular vision-based autonomous robots.

The central idea to monocular vision-based robots is to be able to recognise landmarks in the surrounding environment. Landmarks are environmental features that are familiar to the robots, which serve as navigational aids. Upon successfully recognising a landmark, the robot is able to approximate its current position and derive an optimum path to reach its goal. This chapter describes a selective visual attention landmark recognition (SVALR) architecture that uses the concept of *selective attention* from physiological study as a means for 2-dimensional shape landmark recognitions in complex clustered backgrounds.

This chapter is written with a brief background in monocular vision-based robots, then focusing on two neural networks, the adaptive resonance theory (ART) and selective attention adaptive resonance theory (SAART) neural networks for 2-D shape recognitions. This leads to the development of the SVALR architecture and its application in monocular vision-based autonomous robots. A small robot is designed and implemented to evaluate the SVALR architecture through real-time laboratory experiments.

2.2 Monocular Vision-Based Robots

In the area of monocular vision-based robot navigations, many approaches have been reported and classified into three distinct categories based on their level of dependency on a map of the external environment [8]; map-dependent robots, map-building robots and map-independent robots. In map-dependent robots, the robots are supplied with a map of the navigating environment priori to navigation. Similarly, map-building robots navigate based on a map but the map is not given to the robot. The robots have to create their own map of the external environment using their sensors. The robots then use the constructed map to achieve goal driven tasks. Map-independent robots on the other hand, do not use a map. These robots have a control information database, where control instructions are associated with various stimuli in the environment. During navigation, the robots base on these stimuli and extract the pre-programmed control instructions for navigation.

In general, vision-based robots have a vision system that perceives the external environments. There are five essential components in a vision system of an autonomous vision-based robot [9].

- ➢ **Maps:** The system requires some internal representation or knowledge of the external environment in order to perform goal driven tasks.
- ➢ **Data Acquisition:** The system collects images from a camera.
- ➢ **Feature Extraction:** The feature extraction stage extracts significant features from input images such as edge, texture and colour.
- ➢ **Landmark Recognition:** The system searches for possible matches between the features in the observed images and the expected landmarks pre-stored in memory with respect to some preset criteria.
- ➢ **Self-Localisation:** The self-Localisation stage calculates the robot's current position as a function of detected landmarks and its previous position. The system then derives an optimum path for the robot to traverse to reach its goal.

2.2.1 Maps

Maps are essential for navigating an environment and, therefore maps are a crucial element in autonomous robot navigations. Maps provide the robots with an essential navigational knowledge and awareness of the surrounding to guide the robot to desired locations. In the field of vision-based autonomous robots, there are essentially two major types of maps; geometrical and topological maps. Geometrical maps provide details of metrical information (exact co-ordinates and distances) between objects found in the environment, usually in the form of CAD (computer aid design) models [10, 11]. Topological maps on the other hand are simpler representations of the environment. They are inspired by human navigational maps. The environment is represented in a graphical form, which consists of nodes and arches [12, 13], where nodes represent significant entities in the environment and arches represent their relationships.

2.2.1.1 Geometrical Maps

There are many approaches to geometrical maps including CAD descriptions of the environment, occupancy maps and vector field histograms. CAD representation of geometrical maps is a system of co-ordinate and metrical information of objects found in the environment. Originally, geometrical maps were a 2-D CAD [10, 11] representation of the environment. Lebegue and Aggarwal later developed an automatic generation of CAD model [10, 14]. The robot uses a single camera to capture a sequence of images of a structured indoor environment. The robot constructs a CAD model of that environment based on the captured images. Alternative representations of geometrical maps are occupancy maps. The occupancy map represents the environment using a 2-D grid, where objects in the environment are represented by a 2-D projection of its volume into a horizontal plane [15]. The idea of an occupancy map was further extended by Borenstein, who developed a "Virtual Force Fields" map [16]. The environment is represented by a certainty grid where each cell has a certainty level that represents the certainty that the cell has been occupied [17].

2.2.1.2 Topological Maps

The topological maps are inspired by the way in which human navigates. One does not need to know the exact co-ordinates of every object in the surrounding to be able to successfully navigate an environment. Researches in Physiological studies have indicated that human selectively pay attention to significant and distinct feature in the environment which appears at critical points along the navigational route. These distinctive features are ones that human navigator tends to remember. This concept is modelled in topological maps [9, 12, 18]. Topological maps are much easier to build and have a graphical form with interconnected nodes and arches. Each node represents a distinct entity in the environment and arcs represent their relationships.

2.2.1.3 Integration of Topological and Geometrical (TG) Map

There are advantages and disadvantages for both topological and geometrical maps. Topological maps are much easier to build and they provide sufficient and essential information for navigation. However, topological maps fail to recognise nodes, which are previously visited [8]. Geometrical maps on the other hand, contain metrical information that provides accurate and efficient autonomous robot navigation. However, geometrical maps are harder and costly to build. The construction of a metrical rich geometrical map of an outdoors dynamic environment is almost impossible. In general, geometrical maps are limited to indoor robot navigations.

A number of researchers recently proposed methods for integrating geometrical and topological maps to utilize the advantages of both topological and geometrical maps. The idea is investigated by Yung and Zenlinsky [19], Tomono & Yuta [20], they developed an integration of a topological and geometrical map named a T-G Map. The T-G is a global topological map built by connecting local geometrical maps. It consists of nodes that represent geometrical entities, and arches that represent relationships between entities. Each geometrical local entity contain geometric information of objects in a local region of the environment such as doors, desks or a room for an indoor environment, and buildings, houses or trees for outdoor environment [21], [22]. Similarly, another recent contribution for integrating geometrical and topological maps is proposed by Mararka and Kuipers, which integrates a CAD model and a topological map of the environment [23]. The overall navigating environment is divided into distinct regions, such as rooms and hallways. A CAD map is used to provide metrical information of objects found in each room. A collection of rooms is used to create a topological map for high-level path planning.

2.2.2 Data Acquisition

Nowadays, image processing still requires significant processing power and considers as a slow process, even with fastest computer available. Furthermore, the construction of a robot platform that is able to carry a desktop computer can be quite expensive. Therefore, the designs of autonomous research robots are targeted at using remote autonomous robot navigation. Generally the autonomous robots are designed with the image module implemented off-board the robots [24], with video and data streams transmit between a local PC and the robot via wireless communication links. The robot is to be operating within the wireless communication range. However, other well funded research groups are able to build large robots with image processing capabilities implemented on-board the robot.

Regardless of the platform, the data acquisition process is similar. It involves the collection of raw visual information from a camera. In general the data acquisition process consists of a video camera (normally a CCD camera), which is connected to a frame grabber in the PC. The frame grabber slices the real-time video stream into frames, normally at a rate of 30 frames/sec but varies from different frame grabber hardware. A software module then processes the incoming frames to induce control commands for the robot.

2.2.3 Feature Extraction

Once the system has successfully acquired image frames from a real-life video stream, the main purpose of the feature extraction stage is to extract relevant information for the landmark recognition stage. Raw images obtain from standard CCD cameras are normally in colour (red, green, blue) or gray-level (black and white) images. The RGB colour images can be converted to other colour space such as YUV and HSV colour spaces. Depending on the format of raw images use, appropriate techniques have been developed to extract relevant information, which is used to match with previously learned features stored in memory. In general, raw images are subjected to a pre-processing stage, which involves some means of domain transformations, edge detections and image segmentations for extracting relevant edges, lines, corners and shapes. This information are extracted based some pre-set criteria or guided by other stages in the vision system.

2.2.4 Landmark Recognition

Using landmark recognitions for navigation is a concept that uses by human navigator. It indicates that only regions or objects present along a route that are distinctive from the background and appear at critical point long the route needed to be memorised, in order to successfully navigating that environment. For instance, objects around a point where the navigator needs to make a turn. Thus landmarks serve as navigational references with respect to a map of the environment. The navigator's ability to recognise landmarks enables the navigator to determine their current position in the environment and able to make intelligent decisions to reach desired locations. Similarly, landmarks are used in the same way in autonomous robots. Thus the landmark recognition stage plays a critical role in a vision system. It governs the system ability to recognise features in the surrounding environment that are essential to navigation.

Depending on the application domains, different environmental features are chosen to serve as landmarks. The key to landmark selections is to select environmental features that minimises image distortions with respect to scale, rotations, environmental conditions and object occlusions. Thus landmarks are designed to standout from other objects in the environment to aid the landmark recognition stage. Nevertheless, virtually anything in the environment can be chosen to serve as landmarks and categorised in three distinct categories of landmarks: point, scene and shape based landmarks.

The point-based landmarks are distinct points or group of points in the image that are purposely chosen to serve as landmarks. The central idea is to search or track the appearance of point-based landmarks in the incoming images over time. In general these point-based landmarks are corners, vanishing points and points with distinct colour and contrast in the environments.

Unlike point-based landmarks, scene-based landmarks consist of information of an entire image as a landmark. The robot may traverse a route and store a sequence of images along the route, memorising the environment. Due to memory limitations, the memorised images are reduced to a relatively small size compared to the full size images. Alternatively, the system only memorises images at critical points along the traversed route, such as situations where the robot is about to make a turn or encounter a distinctive event. The stored images are associated with control commands that focus on leading the robot to its final position.

For instance, a scene-based landmark recognition system is proposed by Matsumoto et al [25], which operated in two modes: a recording run and an autonomous run. In the recording run, the robot used a camera to capture a sequence of images at a fixed time interval. These images were used to construct a database or 'memory' of all the images observed during learning trial. Each image in the database is associated with motion commands that led the robot from the current position to the final destination. This is called View-Sequence-Route Representation (VSRR). In autonomous run, the robot compared each input image with the memorised images stored in memory to determine its current position.

Shape-based landmarks on the other hand, are 2-D objects in the environment, which are purposely chosen to serve for navigational aids. The range of objects use spread from small toys in indoor environment to shapes of houses and buildings in outdoor environment. The objects selected are domain dependent. They are generally chosen such that the objects are standout from the other objects in the background to aids the landmark detection stage. For instance, Cheng and Zelinsky used an image processing hardware to recognise circular objects in clean background such as tennis and soccer balls [26]. Luo et al. designed landmarks with retro-reflective material so that only desired landmarks are visible in the observed image due to the reflection of light [27]. Furthermore Li [28] used numerical signs (0-9) as landmarks which placed along the route. A genetic algorithm is used to recognise these numerical landmarks.

2.2.5 Self-localisation

The answer to the question “Where am I” is extremely important for mobile robots to achieve many independent tasks [29]. The answer to this question lies in the self-localisation process. Thus self-localisation is essential to robot navigation. It provides the pose and orientation of the robot at any given time during navigation with respect to the global environment. Upon successfully determining the pose and orientation, the robot generates or corrects an existing path in order to reach the final destination. This is known as path planning. It is a high level process in which the robot bases on its current knowledge and situation awareness of the surrounding environment to formulate an optimum path for the robot to follow to reach its goal. This optimum path is revised every time the robot recognises a landmark, as its current position and the environment are constantly changing during navigation. Some of the self-localisation methodologies are discussed in further details.

The most basic approach to robot self-localisation is dead-reckoning methods. The robot is equipped with internal sensors to measure the amount of wheel rotations, which is directly related to the robot travelled distance. The robot’s current pose and orientation are approximated with respect to its initial position. However, the measurements of the amount of wheel rotations suffer significant error due to wheel slippage and errors in data measurements. Eventually, the robot will lose track of its position and the self-localisation will fail [30] as these errors are accumulative. In order to overcome this problem, external sensors are used to gather extra information to correct the accumulative errors once it reaches a pre-defined threshold. As suggested by Hardt et al [31], self-localisation using dead reckoning method can be improved with the addition of gyroscopes, magnetic compass and ultrasonic sensors [32]. Alternatively combining vision sensor and dead reckoning [33], [29].

2.3 Real-time Visual 2-D Landmark Recognitions

This section describes the development of a 2-D visual landmark recognition architecture. It is based on two neural networks, adaptive resonance theory (ART) and selective attention adaptive resonance theory (SAART).

The new architecture is named selective visual attention landmark recognition (SVALR). It is developed based on the following assumptions:

- ➢ Single landmark detection (only one landmark is detected at a time).
- ➢ Landmarks are not occluded.
- ➢ Landmark recognition is based on contour or edge information.
- ➢ Single 2-D objects are used as landmarks.

2.3.1 Adaptive Resonance Theory (ART)

ART was originally introduced as a physical theory of cognitive information processing in the brain [42, 43]. The theory was derived from a simple feedforward real-time competitive learning system called Instar [47], in response to a problem that real-time competitive learning systems face the *plasticity-stability dilemma.* The dilemma is that a real-time competitive learning system must be plastic to learn significant novel events, while at the same time it must be stable to prevent the corruption of previously learned memories by erroneous activations. The adaptive resonance concept suggests that only the resonant state, in which the reverberation between feedforward (bottom-up) and feedback (top-down) computations within the system are consonant, can lead to adaptive changes. Since the introduction of ART in the late 1970’s and early 1980’s a large family of ART based artificial neural network architectures have been proposed. These include: ART-1 for binary inputs [44], ART-2 for binary and analog inputs [45], ART-3 for hierarchical neural architectures [46], ARTMAP for supervised self-organisation of memory codes [48, 49] and various other versions.

The ART model shown in Fig. 2.1 embeds bottom-up and top-down adaptive pathways in a competitive network that containing two subsystems that regulate learning: (i) an attentional subsystem where top-down expectancies (recalled memories) interact with the bottom-up information; and (ii) an arousal (orienting or vigilance) subsystem that is sensitive to the mismatch between the two. Interactions between these two subsystems ensure that memory modification occurs under exceptional circumstances. Memory can only be modified when an approximate match has occurred between the neural pattern at the input and the resultant pattern across F1. This state is called *adaptive resonance*.

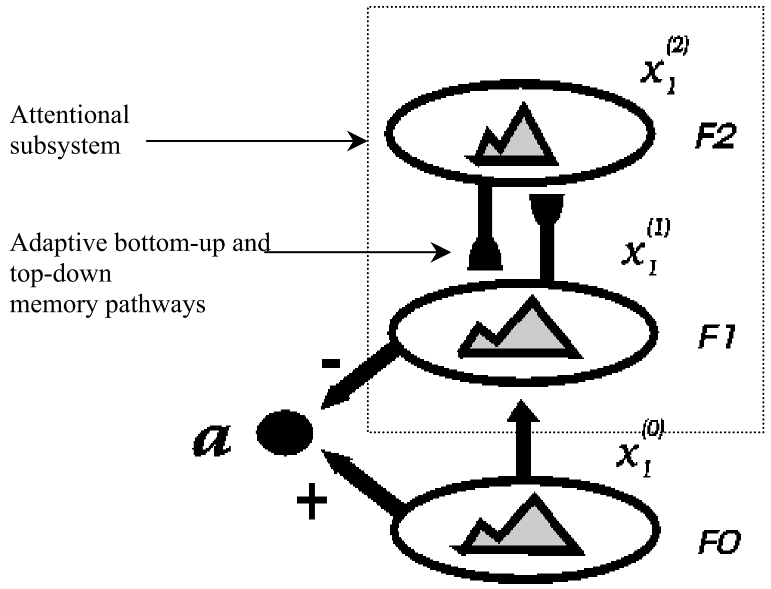


Fig. 2.1 Schematic of the ART model. F0, F1 and F2 are competitive neural Fields whose neural pattern of activity is represented by $x_1^{(0)}$, $x_1^{(1)}$ and $x_1^{(2)}$ respectively. If $x_1^{(0)}$ and $x_1^{(1)}$ match above the preset threshold level then resonance is established between F1 and F2 and this leads to learning in the memory pathways between the active cells in F1 and F2

2.3.2 Limitation of ART

The 2-D pattern recognition problem illustrates in Fig. 2.2 has exposed a limitation of ART's attentional subsystem. Considers a problem of recognising a 2-D shape in a cluttered input, it can be demonstrated via computer simulations [41] that if a conventional ART model (such as ART-2 or ART-3 neural network) has previously learned a 2D pattern, it will not be able to recognise that pattern when it is presented in a complex background of other patterns and clutters. In other words, the ART networks learn pattern x_1 as shown in Fig. 2.2(a). In Fig. 2.2(b), the pattern x_1 is now presented in a cluttered background. This is transferred to Field F1 to activate F2. In Fig. 2.2(c), before the network reaches the steady state, the top-down memory of the learned pattern is transferred to F1 where it replaces the previous activity in F1. A mismatch is detected between F0 and F1 and the network is reset leading to the recognition failure of x_1 in input x_2.

Fig. 2.2 illustrates the processing stages of the conventional ART model. As shown, whenever Field F0 is activated by an input pattern in which the known pattern is embedded, the network does not have the capability to selectively pay attention to the known portion of the input. Thus, even if the correct top-down memory is activated, the ART model fails to match it with the input because it compares the whole pattern across F0 with the whole pattern across F1. The next section shows how the model can be extended to enable selective attention to known portions of the input pattern.

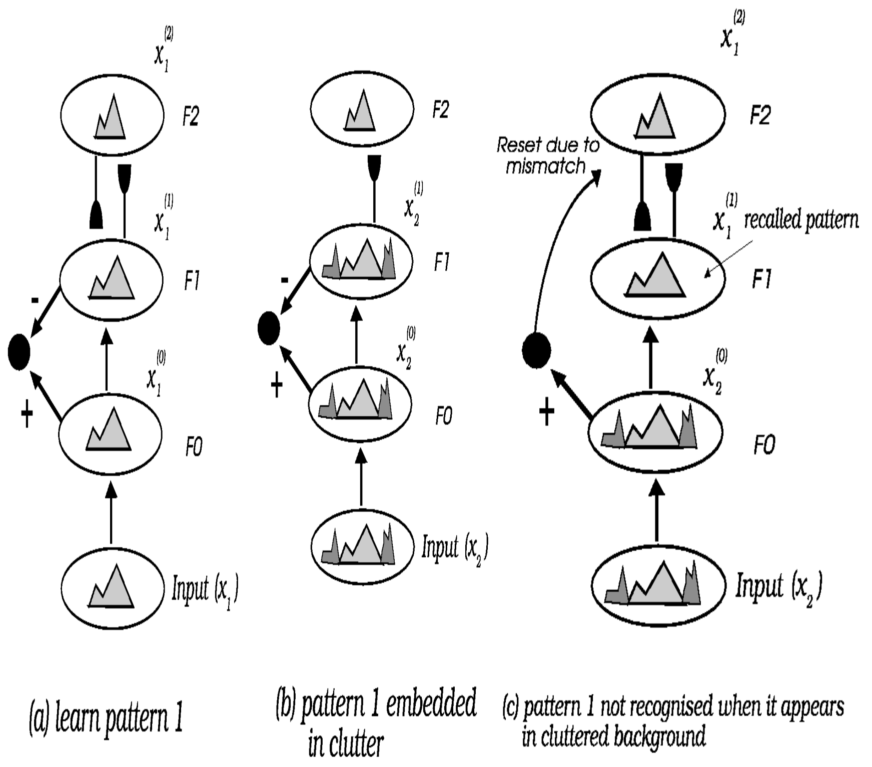


Fig. 2.2. Illustration of processing stages in a conventional ART model

2.3.3 Selective Attention Adaptive Resonance Theory

Instead of resetting the ART network as in Fig. 2.2, Lozo [42, 43] has proposed that the clutter problem can be solved by extending the capability of the attentional subsystem with an additional, functionally different set of top-down feedback pathways. These new pathways run from F1 to F0, allowing the recalled memory across F1 to selectively focus on the portions of the input, which it can recognize. This is illustrated in Fig. 2.3. The new model is named selective attention adaptive resonance theory (SAART) neural network. The feedback pathway from F1 to the input Field F0-A is modulatory, and acts on the signal transmission gain of the bottom-up input synapses of F0-A (and F0-B). Thus, each cell in F1 sends a top-down facilitatory gain control signal to the input synapse of the corresponding cell in F0-A. Lateral competition across F0-A will suppress the activity of all neurons whose input gain is not enhanced by the corresponding neurons in F1. These feedforward-feedback interactions enable *selective resonance* to occur between the recalled memory at F1 and a selected portion of the input at F0-A. Because the facilitatory signals and the competitive interactions in F0-A do not act instantaneously, the resonant steady state develops over a period of time during which the network may be in a highly dynamic state. To follow the progress of these interactions, it becomes necessary to measure the degree of match between the spatial patterns across the Fields F0-A and F1 as well as the time rate of change of this match. To protect the long-term memory from unwarranted modification by non-matching inputs during these rapid changes, long-term memory is updated when the system is in a stable resonant state. Similarly, the certainty of the network's pattern recognition response increases as the steady state is approached. Thus, what may initially be taken as a bad mismatch may eventually end up as a perfect match with a selected portion of the input pattern. The unmatched portion of the input will appear across Field F0-B, where it has direct access to the bottom-up memory (this feature of the model also enables familiar inputs to activate their memory directly by bypassing Field F1).

2.3.4 Limitations of SAART

The SAART neural network has demonstrated through numerous simulation by Lozo [41, 50-52] to be able recognise 2-D shapes when present in clustered backgrounds. However, the SAART neural network is a dynamic network and therefore very computationally intensive. Thus it is incapable of providing real-time landmark recognitions for robotics applications.

In order to apply SAART into robotics applications, it has to be re-engineered to achieve real-time landmark recognitions. The central concept to the SAART neural network is the addition of the modulatory top-down feedback pathways into the existing ART network, for selectively attending to relevant input data. Using this concept in combination with conventional image processing architecture a new landmark recognition architecture is developed. The new architecture is named selective visual attention landmark recognition (SVALR) architecture. The following subsequences describe in details the development of the SVALR architecture.

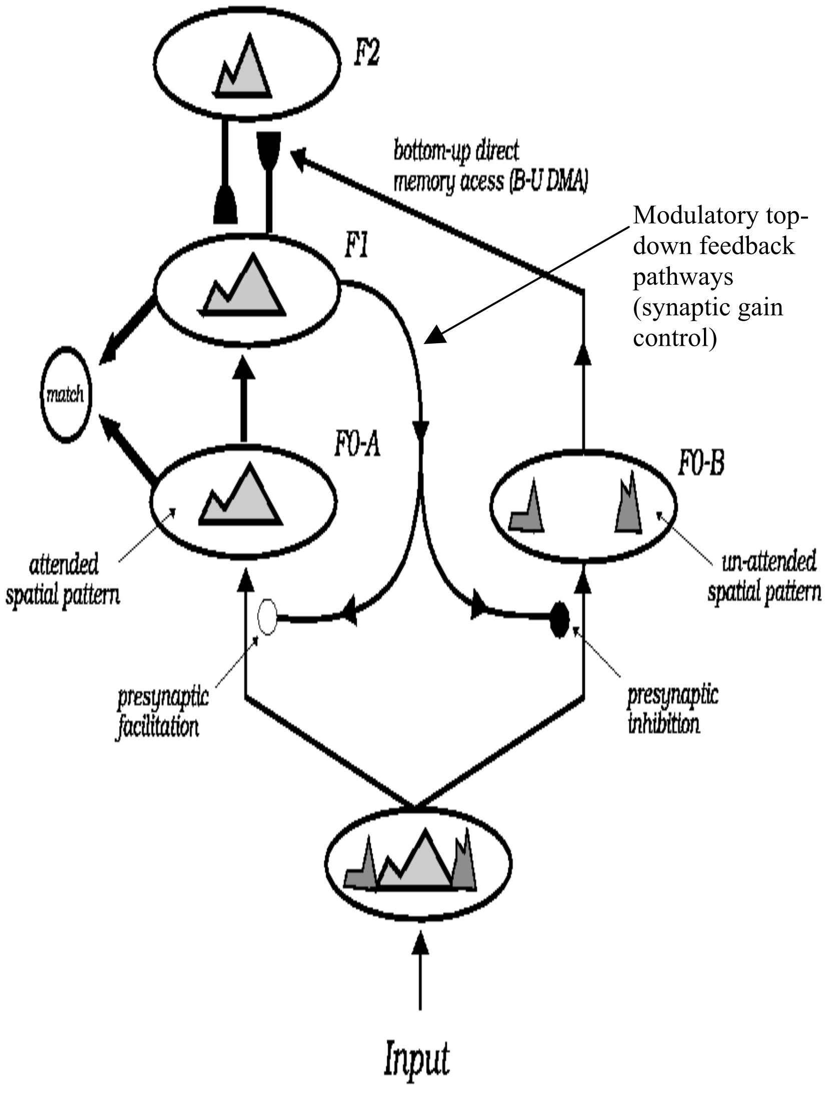

F2
bottom-up direct memory acess (B-U DMA)
Modulatory top-down feedback pathways (synaptic gain control)
F1
match
F0-A
F0-B
attended spatial pattern
un-attended spatial pattern
presynaptic facilitation
presynaptic inhibition
Input

Fig. 2.3. Schematic of the SAART neural model

2.3.5 Selective Visual Attention Landmark Recognition Architecture

The prominent characteristic of the SVALR architecture lies in the incorporation of selective visual attention in conventional image processing architecture. This enables architecture to selectively attending to relevant while ignoring irrelevant input data. As a result, enables the SVALR architecture to reliably achieve object background separation at the feature extraction stage, which leads to the ability of visual landmark recognitions in cluttered backgrounds. Selective visual attention is implemented using a mechanism named memory feedback modulation (MFM). The SVALR architecture is illustrated in Fig. 2.4.

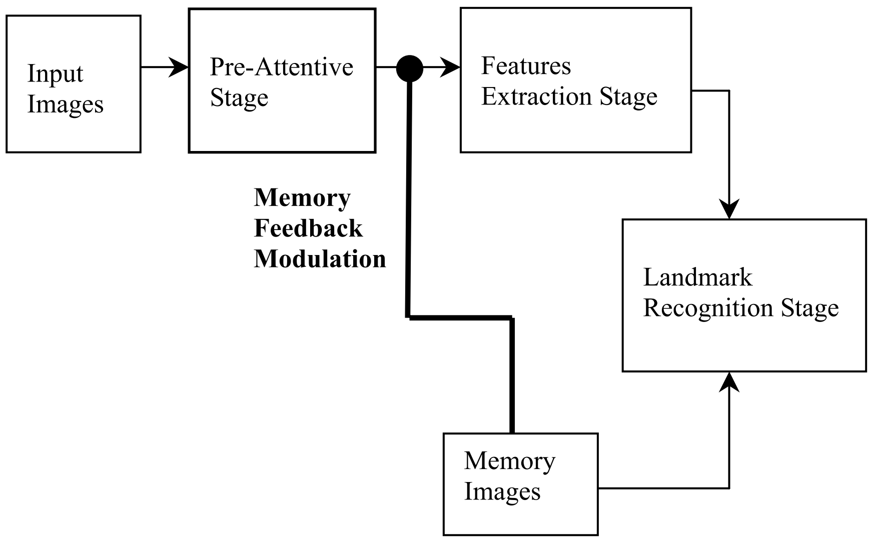

Input Images
Pre-Attentive Stage
Features Extraction Stage
Memory Feedback Modulation
Landmark Recognition Stage
Memory Images

Fig. 2.4. The selective visual attention landmark recognition architecture

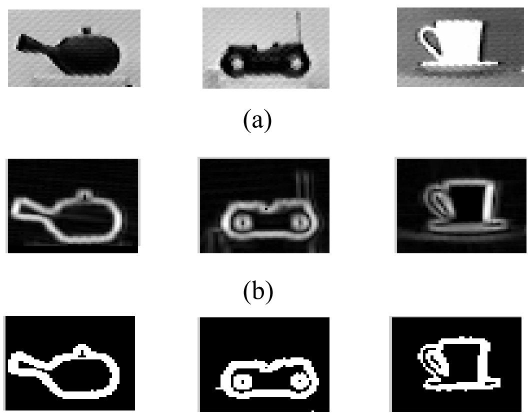
2.3.5.1 Memory Selection

The SVALR architecture recognises visual landmarks based on template matching, where an input pattern is matched with a pre-stored memory image of the target landmark. Two images are stored in memory for each visual landmark, a memory image and binary memory filters (BMF). Both images are considered as prior knowledge of the system about an external environment. Memory images are used in the recognition stage, while BMF filters are used in the MFM mechanism. These will be discussed in later sections.

Each landmark has its own corresponding memory image and BMF filter. There are three steps involved in creating memory images and BMF filters. The first step is to obtain a clean gray level image of objects that are chosen to serve as visual landmarks (place each object on a clean background with high level of contrast). The landmark must be adjusted to fit into a memory image of size 50x50 pixels by varying the distance between the camera and the chosen landmark. The second step is to apply Prewitt edge detection to each gray level image to produce an edge image of each landmark shown in Fig. 2.5(b). These images are used as memory images. The third step is to apply a small threshold to the Prewitt edge images using *eq.2.1* to produce BMF filters as show in Fig. 2.5(c).

$$E(i,j) = \left\{ \begin{vmatrix} > \tau & F(i,j) = 1 \\ < \tau & F(i,j) = 0 \end{vmatrix} \right\} \tag{2.1}$$

Where E(i,j) is the Prewitt edge detected image, τ is a small threshold and F(i,j) is the BMF filter.



(c)

Fig. 2.5. Samples of objects used as Landmarks. (**a**) Gray level image of landmarks in clean backgrounds. (**b**) Memory images and (**c**) Binary memory filters

2.3.5.2 Memory Feedback Modulation Mechanism

The core concept of the SVALR architecture is the memory feedback modulation (MFM) mechanism. The MFM mechanism has the ability to suppress irrelevant edge activities and enhance relevant activities in the input images, achieving object background separation at the feature extraction stage. This is achieved by selectively amplifying relevant regions in the input image using a BMF filter. This process is selectively guided by feedback information from memory using *eq.2.2*.

$$Ex(i,j) = S(i,j) + G(S(i,j) * F(i,j)) \tag{2.2}$$

Where F(i, j) is the feedback BMF filter, S(i, j) is a region within the input image, Ex(i, j) is the modulated image - the output of the feature extraction stage and G is a gain control

Notice that a BMF filter contains approximately 20% of high value pixels that describe the shape of a landmark. Therefore the edge activities in the BMF filter are used to selectively amplify pixels that corresponding to the landmark shape in the input image. Input pixels that have elementary alignment with active pixels in the MBF filter will experience high amplification. The amount of amplification is controlled by the gain control, G of *eq.2.2*. Similarly, input pixels that have elementary alignment with non-active pixels in the MBF filter will experience no amplification as the term $S(i,j) * F(i,j)$ in *eq.2.2* equal to zero. The regions that receive no amplification are potential background clutters. These background clutters are removed by a lateral completion. The lateral completion between pixels in the modulated image, Ex(i,j) is achieved by L2 normalisation. During the competition, high value pixels are experiencing a small amount of suppression by the smaller value pixels, while small value pixels are experiencing larger suppression from large value pixels simultaneously. Thus, pixels with small value will be suppressed to an extremely low level. If they are suppressed below a preset threshold, they are discarded using *eq.2.3*.

The prominent effect of the MFM mechanism is achieving object background separation. This is further illustrated in Fig. 2.6, where a selected input region in the input edge image, containing the target landmark embedded in a cluttered background. If this region is compared directly with the memory it will not result in match due to background clutters, the system will fail to recognise the target landmark. However, if this region undergoes memory feedback modulation to obtain object-background separation, obtaining a clean image, Ex(i,j), of the landmark without background clutters as shown at the bottom of Fig. 2.6, this will result in a perfect match with memory.

$$Ex(i,j) = \left\{ \begin{array}{ll} > \tau & Ex(i,j) = E(i,j) - \tau \\ < \tau & Ex(i,j) = 0 \end{array} \right\} \tag{2.3}$$

Where Ex(i, j) is the normalised form of the modulated region and τ is a small threshold

2.3.5.3 Landmark Recognition Stage

The landmark recognition stage matches each normalised region, Ex(i,j) with the corresponding memory image. The degree of match between the input region Ex(i,j) and the memory image is measured using the cosine angle between two 2D arrays. The equation used for the comparison is shown in *eq.2.4*.

$$Match = \frac{\sum\sum Ex(i,j) * M(i,j)}{\varepsilon + \sqrt{\sum\sum Ex(i,j) * Ex(i,j)}\sqrt{\sum\sum M(i,j) * M(i,j)}} \tag{2.4}$$

Where Ex(i,j) is the selected normalised input region, M(i,j) is the memory image, ε is a small constant to prevent the equation from divided by zero, when both Ex(i,j) and M(i,j) are blank images.

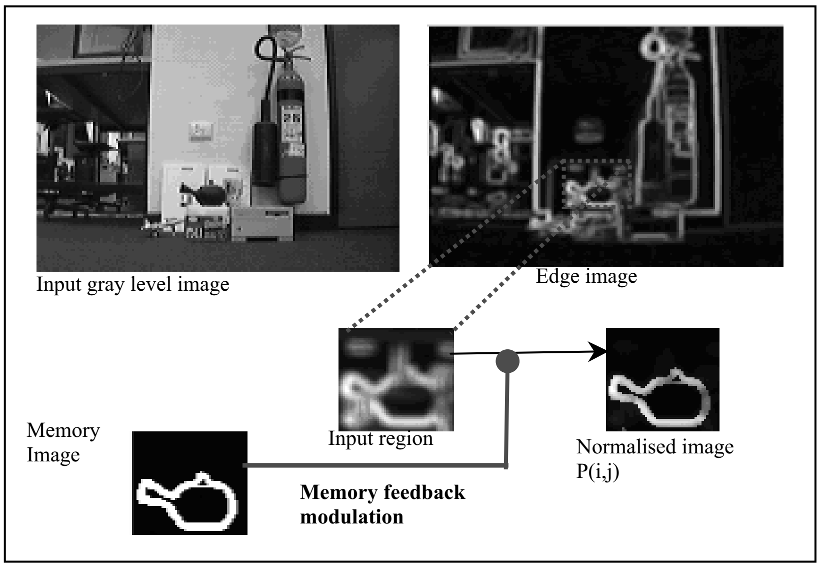


Fig. 2.6. The memory feedback modulation mechanism

The advantage of using the cosine between two images is that it provides a degree of match in a range from 0-1, where 1 represent 100% match. This makes it easier to set a match threshold. However, with every advantage there is an encounter disadvantage. It is found that by using the cosine angle between two images results in a high degree of match between planar regions in the input image and the memory image. This is due to the fact that planar regions in the input image are flat surfaces such as walls and floors. These regions have very low edge activity in the input edge image. Since the majority of pixels in the memory image are low value pixels, with approximately 20% of the pixels are high value pixels that describe the shape of a landmark. It is clear that there is a high level of similarity between the memory image and planar regions in the input image. Furthermore, when applying memory feedback modulation pixels in the input planar regions that have elementary alignment with active pixels in the BMF filter will be amplified and become strong edges. Thus, the comparisons between memory and these planar regions resulted in a high degree of matches leading to fault landmark detections as illustrated in Fig. 2.7(d).

In order to overcome the fault matches in planar input regions, an additional vertical shift component is introduced into the MFM equation, *eq.2.2*. The new memory feedback modulation equation is shown in *eq.2.5*. By carefully selecting the value for the vertical shift in the memory feedback modulation equation, this reduces the system sensitivity toward input planar regions and at the same time maintaining the overall correlation between the input image and memory as illustrated in Fig. 2.7(e).

$$Ex(i,j) = S(i,j)\left[1 + G * F(i,j)\right] + \Im \tag{2.5}$$

Where $\Im$ is a global vertical shift for reducing the system sensitivity in planar regions in the input image.

2.3.5.4 Landmark Searching Process

The searching process of the SVALR architecture is based on window-scan searching mechanism illustrated in Fig. 2.8(a). The searching process is initially started at the top left corner, with a search window of size equal to the size of the memory image. This search window slides cross the image horizontally (horizontal scan), upon completion the search window moves down vertically by one pixel and the horizontal scan repeated until every pixels position is searched. As the search window slides across the image, regions of 50x50 pixels are extracted for landmark recognition. The advantage of this searching method is that it ensures that the landmark cannot be missed if the landmark is in the image. However, the disadvantage of this searching method is that it is computationally intensive. In order to reduce the computational requirements, this search method is extended to incorporate a concept of pre-attentive stage in human visual system to obtain a fast searching mechanism.

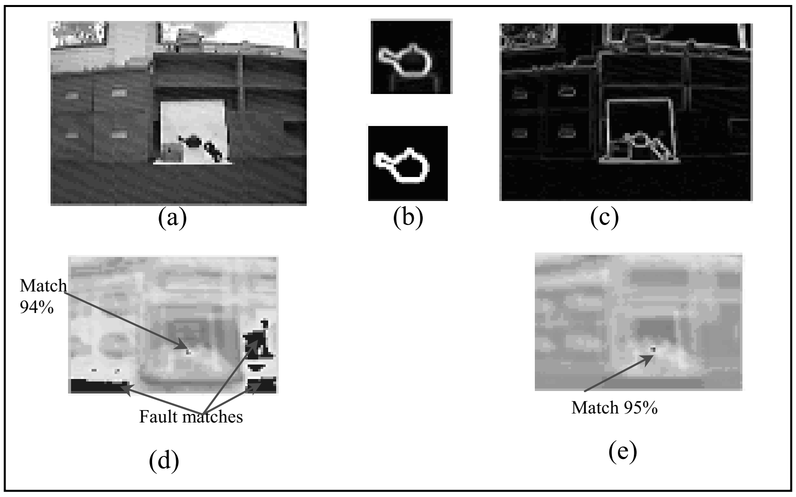


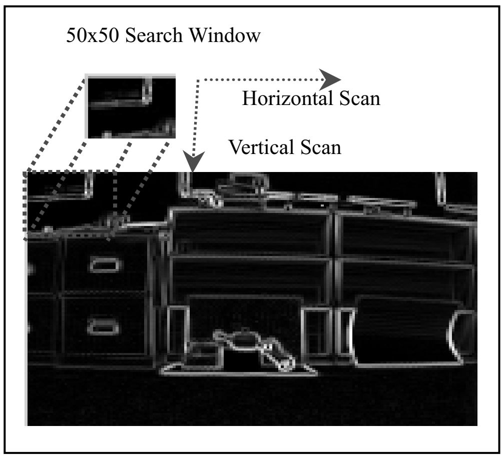
Fig. 2.7. System high sensitivity in planar regions (**a**) Gray-level input image, (**b**) Memory image and BMF filters, top and bottom respectively (**c**) edge input image, (**d**) High match obtained in planar regions and (**e**) System improvement with the addition of the global vertical shift

Human visual system tends to pay more attention to the most "eye-catching" regions. One tends to focus on regions of most conspicuous and highly features regions first. This suggests that human visual system perform objects recognition in two stages; a pre-attentive stage and an attentive stage [53, 54]. The pre-attentive stage is a quick, effortless process that identifies and allocates computational resources to the most relevant regions in the visual input. The attentive stage, on the other hand, is a slower and thorough process, which identifies and understands the physical features in the visual input.

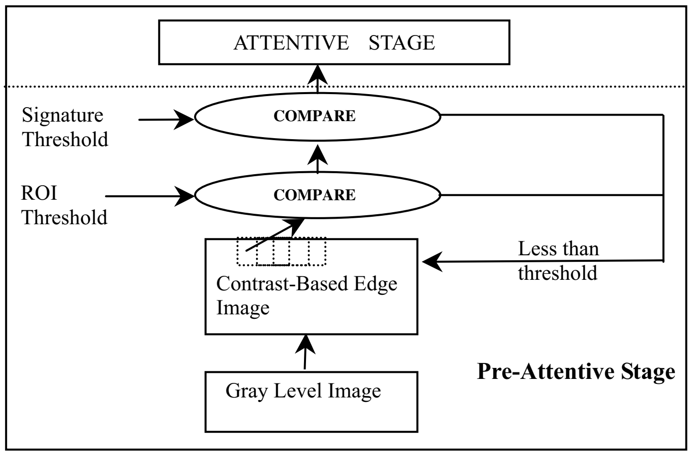
The developed pre-attentive searching mechanism models the pre-attentive processing stage of human visual system. Its core task is to selectively allocate available computational resource to the relevant regions in the input image. The overall pre-attentive stage is illustrated in Fig. 2.8(b). The effectiveness of the pre-attentive stage depends on the determination of potential regions (PR) [55]. There are two steps involved in determining PR regions. The first step is to determine the regions of interest (ROI). The knowledge of edge activities in the pre-stored memory images are used to set a ROI threshold, such that ROI regions are of the same size as the memory image and have edge activities greater than the ROI threshold.

The ROI threshold is calculated based on the knowledge of the landmark stored in memory. The following steps are used to determine the ROI threshold:

- Calculate the number of significant pixels within the memory image pixels with activity greater than 20% of the maximum pixel value is considered as significant pixels.
- The ROI threshold is set at 50% of the total number of significant pixels.



(a)



(b)

Fig. 2.8. The search mechanism. (**a**) The basic window-based search mechanism. (**b**) The pre-attentive landmark searching mechanism

ROI regions are further processed to determine whether they have a potential of matching with the memory image based on a signature threshold. The signature threshold is calculated from unique regions. These unique regions are smaller regions within the memory image that represent the internal unique features of each landmark. These regions are fixed regions and unchanged from a robot field of views as illustrated in Fig. 2.9. The edge activities in the unique regions are calculated to determine a landmark signature threshold. The total number of significant pixels in all of the selected unique regions is set as the signature threshold. The ROI regions that satisfy the signature threshold will be considered as potential regions and will be promoted into the attentive stage, where they are subjected to intensive processes for landmark recognition. Other ROI regions are discarded.

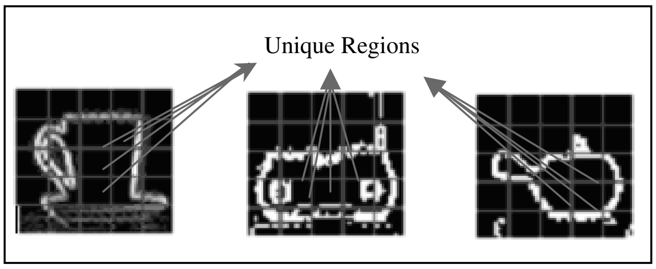


Fig. 2.9. Memory unique regions identification

2.4 Mobile Robotics Applications

This section describes an implementation of a monocular vision-based autonomous robot, which is used to demonstrate the applicability and robustness of the SVALR architecture. An additional extension to the SVALR architecture named SMIS mechanism is developed to cope with image distortions, size and view invariant real-time landmark recognitions.

2.4.1 Robot Navigational Topology

The navigational topology used in this robot mimics the way in which human navigates. One does not need a detailed geometrical map of the environment to successfully navigating an environment. Instead only distinctive things or infrastructures at critical points along the route are memorised. These features are essential to navigations and are known as landmarks. This is also known as topological navigations.

In order to use the concept of topological navigation, a topological map is required to be built and given to the robot prior to navigation. The construction of the topological map involves the placement of the selected landmarks in arbitrary locations in the laboratory, with measurement of their relative distances and directions. The locations of landmarks are denoted as nodes, where relative distances and directions are represented by arrows. This is illustrated in Fig. 2.10. Each node has its own corresponding memory image pre-selected as shown in Fig. 2.5.

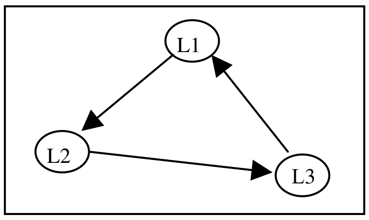


Fig. 2.10. A simple topological map consists of three landmarks

2.4.2 Robot Constructions

The robot has been designed and implemented in considerations as a prototype system with minimal cost. This research has a potential to be implemented on a larger scale outdoor robot by the Defence Science and Technology Organisation (DSTO), which will use real-life outdoor landmarks such as cars, road signs, trees, buildings and houses.

Therefore, the robot is designed and constructed based on an existing framework of a small wireless remote control toy car with additional electronics hardware. On-board the robot, there are a PC/104 AMD GX1-300 MHz embedded PC with a serial 4-ports extension module, a TCM2 magnetic compass and an odometer to measure the robot's heading with respect to the Earth's magnetic field and distance travelled by the robot respectively. The embedded PC is running under a Linux operating system. Additionally, the robot is equipped with three GP2YA02Yk infrared range sensors for obstacle detection. The hardware implementation of the robot is shown in Fig. 2.11.

The navigational algorithm is implemented on-board the robot using C++ programming language. The algorithm has a motor control selector and three distinct independent behaviours: navigation, obstacle detection and self-localization behaviours as illustrated in Fig. 2.12. The navigation behaviour controls the robot's speed and heading based on the compass and odometer readings. It ensures that the robot navigates in the direction given in the provided map. Upon a detection of an obstacle using infrared sensors, the robot is stopped. Future work will implement an obstacle avoidance behaviour. The self-localization behaviour is executed whenever a pre-specified visual landmark is detected.

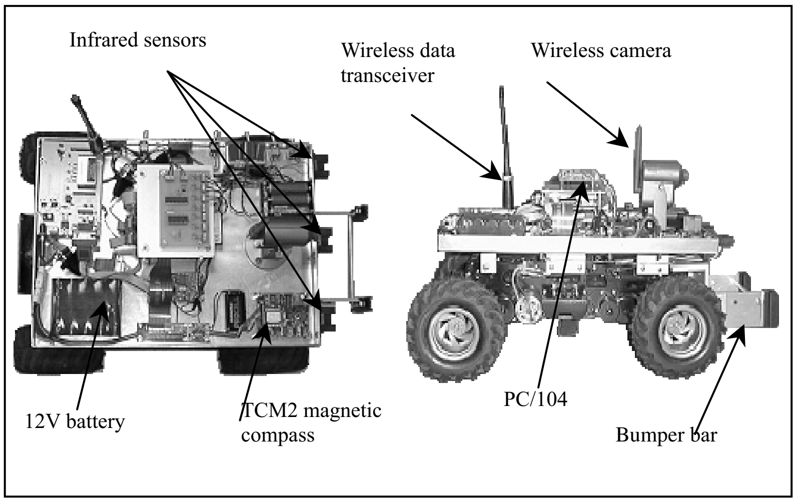


Fig. 2.11. The monocular vision-based autonomous robot

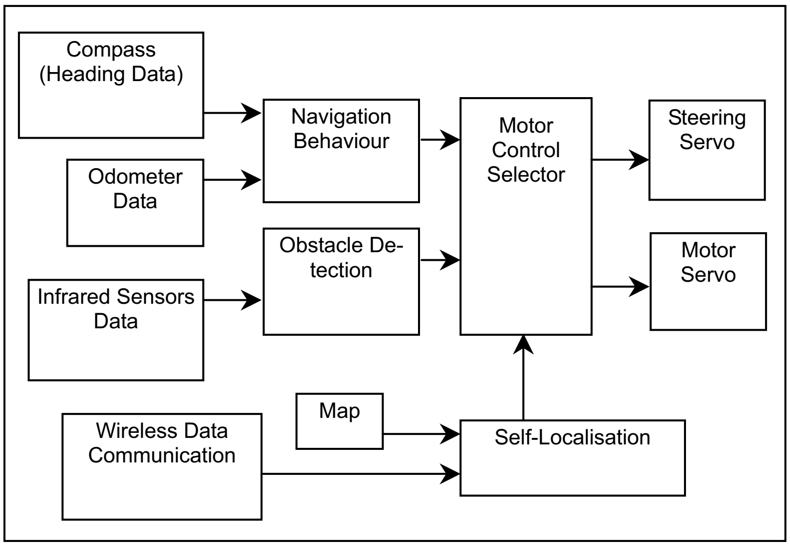


Fig. 2.12. The robot navigation control block diagram

2.4.3 Landmark Recognitions

In autonomous robot navigations, it is critical that the vision system is able to achieve reliable and robust visual landmark recognitions in real-time. Fault landmark recognitions will lead to *'the robot is lost'* situation, where the robot loses its perception and its current location in the environment. In general, fault recognitions are cased by image distortions. Therefore, the challenge is to develop techniques to overcome image distortions due to noises introduced through wireless video links. Furthermore, as the robot navigates the size and shape of landmark are changing constantly. These changes are directly proportional to the robot's speed and the robot's approaching angles with respect to the target landmark during navigation. The following sections describe techniques used to overcome image distortions, and changes in landmark's size and shape.

2.4.3.1 Distortion Invariant Landmark Recognition

The SVALR architecture recognises landmarks based on their shapes. Therefore if the shape of a landmark is affected by noises causing image distortions, changing the size and shape of the landmark will result in a recognition failure. The architecture employs two concepts named band transformation and shape attraction to overcome image distortions and small change in the landmark's size and shape.

The central idea to band transformations is to thicken the shape of the landmark by means of a Gaussian filter [56] or an averaging mask [57] using *eq.2.6*. This will produce a blurred edge image. The blurred image is then subjected to a shape attraction process. The shape attraction process uses the memory template to selectively attract the corresponding edge activities in the blurred shape and project them into the original undistorted shape. The concept of shape attraction is further illustrated in Fig. 2.13.

$$IB(i,j) = \left[\sum_{r=0}^{r=5} \sum_{c=0}^{c=5} I(i+r, j+c) \right] / (r * c) \tag{2.6}$$

Where *IB* is the burred image, *r* & *c* are the size of the averaging window and *I* is the input edge image.

2.4.3.2 Sizes and Views Invariant Landmark Recognition

The SVALR architecture requires the recognition of landmarks that are continuously changing in size and shape during navigation. This leads to the development of a simultaneously multiple-memory image search (SMIS) mechanism. This mechanism is capable of providing real-time size and view invariant visual landmark recognition [58]. The central idea to the SMIS mechanism is to pre-store multiple memory images of different landmark's sizes and from different views and compares each input image with multiple memory templates.

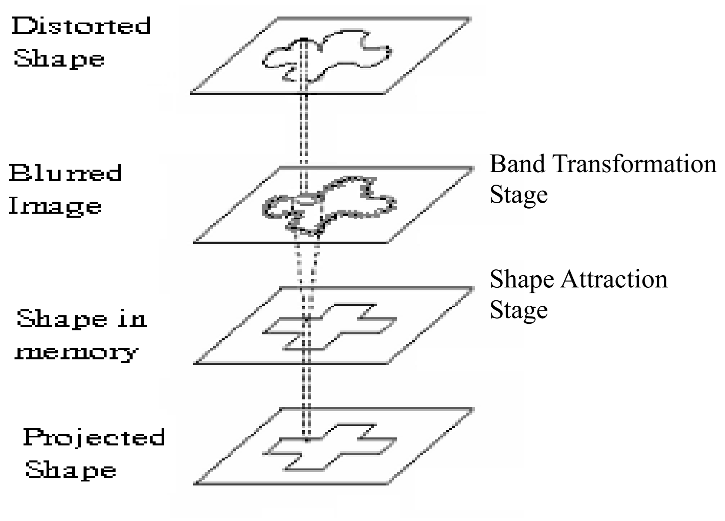


Fig. 2.13. The shape attraction process

Through experiment it was found that the landmark's size is directly proportional to the distance between the landmark and the robot. As a result, the shape attraction method is capable of providing a small detectable region for each memory image stored in memory as illustrated in Fig. 2.14(a). The first memory image is taken from a distant, K_1, away from the landmark. This provides a detectable region around the location, $X_{1,}$ and a detectable angle, α. Thus, multiple memory images can be selected with adjacent detectable regions joined together to provide landmark recognition over larger distances and hence larger changes in landmark's size. Therefore, by storing multiple memory images of different landmark's sizes, with detectable regions joined together will provide the system with full size invariant landmark recognitions. The numbers of memory images required depend on the rate of change in the landmark's size, which is directly proportional to the robot's speed.

Similarly, each memory image provides a detection angle, α. Therefore, multiple views covering 360^0 around the landmark with the angle between these views equal to α are stored for each landmark to provide full view invariant landmark recognitions as shown in Fig. 2.14(b). The number of views required to cover 360^0 are given by the *eq.2.7.*

$$\text{No. of views} = 3600/ \qquad (2.7)$$

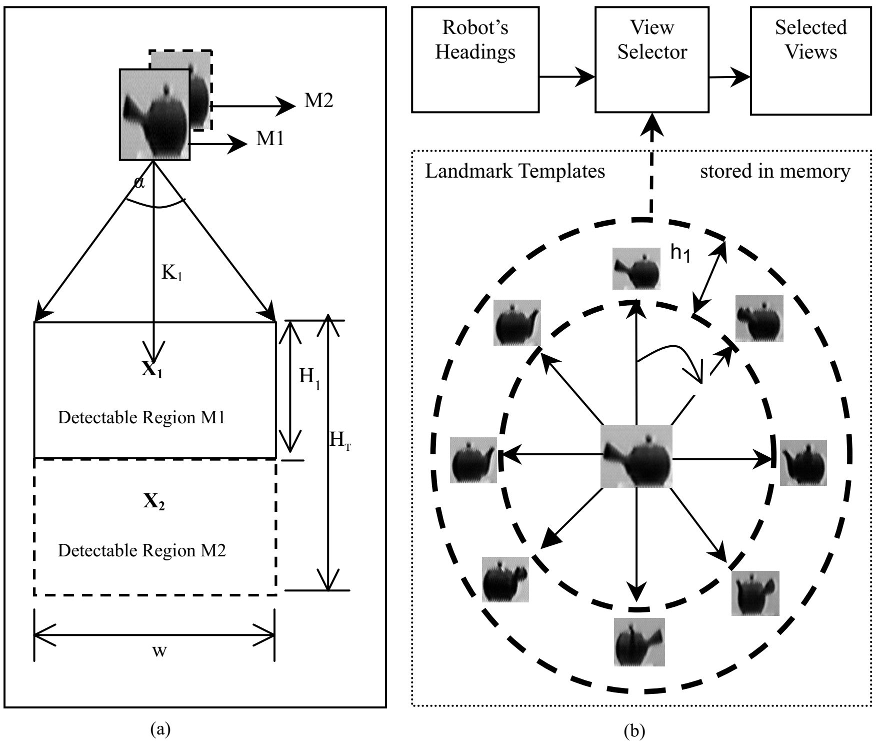


Fig. 2.14. The SMIS mechanism for achieving size and view invariant landmark recognitions. (**a**) Size invariant landmark recognition using two memory images. (**b**) View invariant landmark recognition using views memory images

The central idea of the SMIS mechanism is to search for multiple memory images simultaneously. Thus, it allows the SVALR architecture to recognise landmarks of different sizes and from different views. However, the SMIS mechanism is very computational intensive as many views are evaluated simultaneously. Therefore the SMIS mechanism employs a view selector to select a limited number of views to use in the searching process for reducing the computational requirement. The view selector determines appropriate views based on the robot's heading, which is provided by the magnetic compass on-board the robot via wireless data link, illustrated on the top in Fig. 2.14(b). This reduces to only the current view and two left-right adjacent views are activated instead of simultaneously searching through all the views associated with a landmark.

2.4.3.3 Light Invariant Recognition

The SVALR architecture processes input images based on pre-processed edges or boundary information of an edge detection stage. Therefore, the efficiency of the architecture is directly depending on the quality of edge information obtained. Common edge detection methods such as Sobel, Prewitt and Robinson edge detections, all detecting edges based on the differences between the sums of pixels values on the left and right regions of a target pixel. This is generally achieved by applying an appropriate edge detection convolution mask. The strength of the detected edges using these methods is directly affected by the amount of light in the environment. Changes in light intensity have an immediate impact on the strength of edges obtained at the edge detection stage. This section describes a new method for edge detection named contrast-based edge detection. This edge detection method enables the SVALR architecture to recognise landmarks under different lighting conditions.

The contrast-based edge detection is developed based on Grossberg's theory on shunting-competitive neural networks [59, 60]. The equation for dynamics competition of biological neurons is given in *eq.2.8.* Where A is the rate of decay, B and D are some constant that specify the range of neurons activities. E_{ij} and C_{ij} are excitatory and inhibitory inputs respectively.

$$\frac{dx_{ij}}{dt} = -Ax_{ij} + (B - x_{ij})E_{ij} - (x_{ij} + D)C_{ij} \tag{2.8}$$

At equilibrium $\frac{dx_{ij}}{dt} = 0$, there exists a steady state solution for the

neuron, x_{ij} given in *eq.2.9.*

$$-Ax_{ij} + (B - x_{ij})C_{ij} - (x_{ij} + D)E_{ij} = 0 \tag{2.9}$$

$$x_{ij} = \frac{BC_{ij} - DE_{ij}}{A + C_{ij} + E_{ij}} \tag{2.10}$$

In order to design the contrast-based edge detection, the C_{ij} and E_{ij} terms are replaced with the left and right columns of an edge detection mask instead of the excitatory and inhibitory inputs in dynamic competitive neurons as shown in Fig. 2.15. Since B and D are constants. Let B &

D =1, this gives the contrast-based edge detection equation as shown in *eq.2.11.*

$$x_{ij} = \frac{\sum | C_{ij} I_{ij} | - \sum | E_{ij} I_{ij} |}{A + \sum | E_{ij} I_{ij} | + \sum | C_{ij} I_{ij} |} \tag{2.11}$$

Where A is a small constant preventing the equation from dividing by zero and I_{ij} is the input gray level image. Notice that both Sobel and Robinson edge detection masks can be used in the contrast-based edge detection.

In general, the contrast-based edge detection uses a conventional edge detection convolution mask for detecting the difference between neighbouring left and right regions of a target pixel. The calculated difference is divided by the total sum of all edge activities from both left and right regions within the edge detection mask

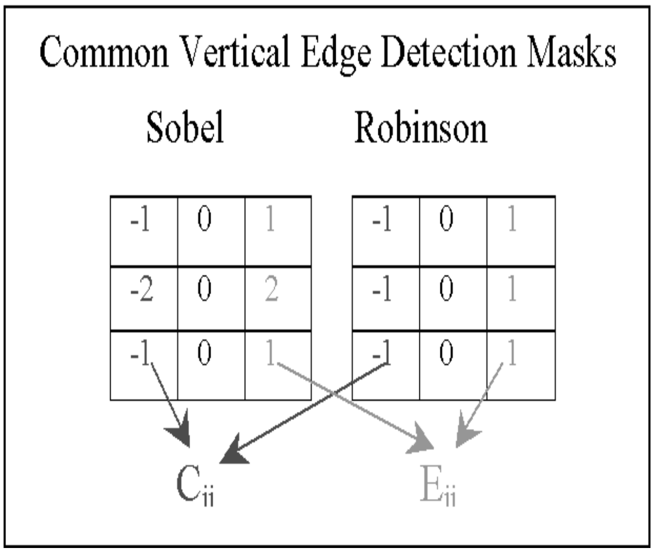


Fig. 2.15. Contrast-based vertical edge detection masks

2.4.3.4 Final SVALR Architecture

The final SVALR architecture is illustrated in Fig. 2.16. Initially, a gray level image is pre-processed using the contrast-based edge detection to generate an edge image. This image is blurred using a 5x5-averaging window for achieving distortion and small size and view invariant landmark recognitions using shape attraction. A window-based searching mechanism is employed to search the entire input blurred image for a target landmark. The search window is sized, 50x50 pixels, each region within the search window is processed in the pre-attentive stage using both the ROI and the signature thresholds as illustrated at the bottom of Fig 2.16. The selected regions are passed into the attentive stage, where they are modulated by the memory feedback modulation given in *eq.2.5.* Then, lateral competition between pixels within the selected region is achieved by applying L2 normalisation. This results in a filter effect, which enhances common edge activities and suppresses un-aligned features between the memory image and the input region achieving object background separation.

The SMIS mechanism selects appropriate views based on the robot's heading as illustrated on the top of Fig. 2.16. It is found that the SVALR architecture requires a minimum of two memory images for each view and eight views to achieve size and view invariant landmark recognition respectively. This is achieved at a moderate robot's speed. The selected memory images are used to compare with the selected input region. The matching between selected input regions and corresponding memory images are determined based on two criteria.

Firstly, each selected input region is compared with two selected memory images (belonging to one view) separately using the cosine between two 2-D arrays. The cosine comparison results with a match value range from 0 to 1, where 1 is the 100% match, which is evaluated against a match-threshold of 90% match. If either results are greater than the match threshold, then the second criterion is evaluated. The second criterion is based on a concept of top-down expectancy from physiological study. Based on a given map, the landmark is expected to appear at a certain distance and direction. These two constraints are used to further enhance the robustness of the landmark recognition stage. Therefore, a match only occurs when the robot has travelled a minimum required distance and heading in the approximate expected direction.

2.5 Results

The autonomous mobile robot is evaluated in indoor laboratory environment. The robot is provided with a topological map, which consists of the relative directions and approximate distances between objects placed on the laboratory floor. A number of autonomous navigation trials were conducted to evaluate the SVALR architecture ability to recognise landmarks in clean, cluttered complex backgrounds and under different lighting conditions. Four trials were selected for discussion in this chapter.

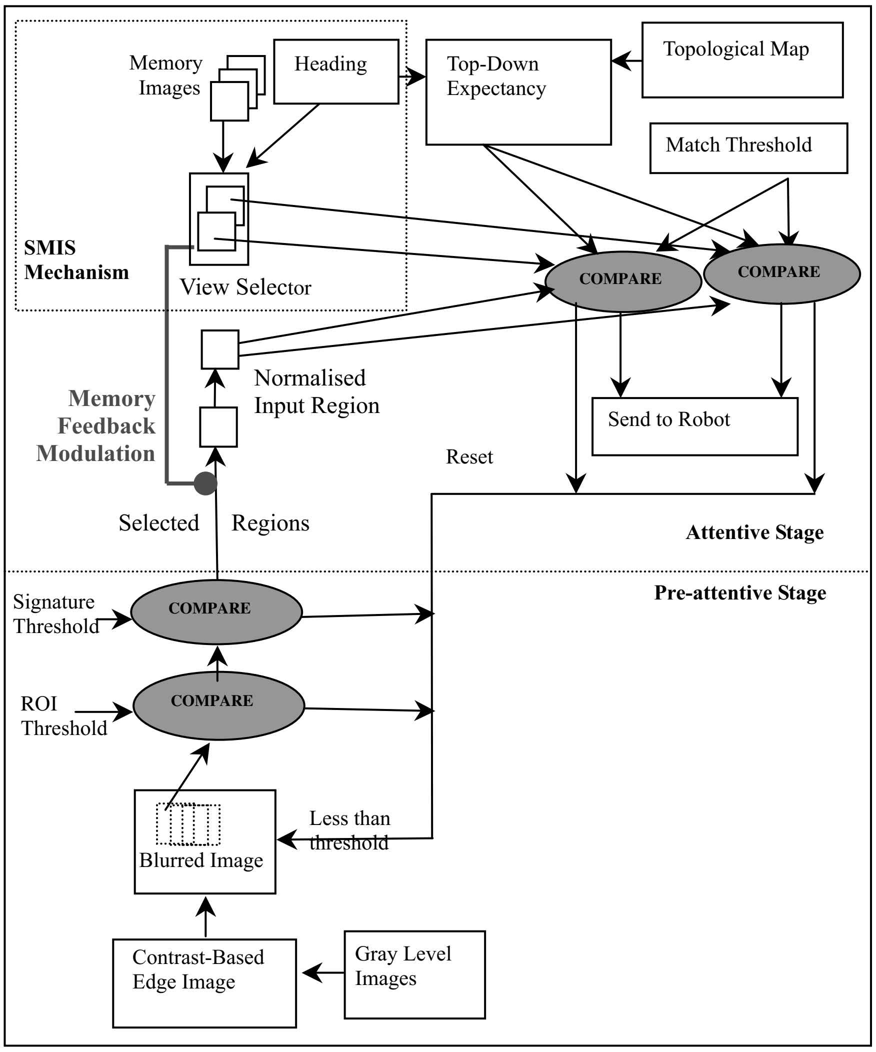


Fig. 2.16. The Selective Visual Attention Landmark Recognition Architecture

In the first trial, objects that are chosen to serve as landmarks were placed in front of clean backgrounds and critical points where the robot needs to make a turn. During navigation each input images is pre- processed by the contrast-based edge detection and then blurred using a 5x5 averaging window for achieving distortion invariant landmark recognition as shown in Fig. 2.17(b) and Fig. 2.17(c) respectively. The landmark search and recognition is performed on the blurred image, where each 50x50 region is compared with the memory image. The results are converted into a range from 0-255 and displayed as an image form in Fig. 2.17(d). The region with highest intensity represents the highest match with the memory image. The dot indicated by an arrow at the bottom centre of Fig. 2.17(d) highlights location of maximum match value and is greater than the match threshold (location where the object is found). This location is sent to the robot via wireless data link. The navigational algorithm on-board the robot based on the provided map to perform self-localisation and more toward the next visual landmark. In this trial, the results show that the SVALR architecture is capable of recognising all visual landmarks in clean backgrounds, successfully performing self-localisation and autonomous navigation. Finally, black regions in Fig. 2.17(d) are ones that have been skipped by the pre-attentive stage. This has increased the landmark searching process significantly [55].

In the second trial each landmark is placed in front of a complex background with many other objects behind it. This is to demonstrate the SVALR architecture ability to recognise landmarks in cluttered backgrounds. Similarly, incoming images are processed as discussed previously with the landmark recognition results illustrated in Fig. 2.18, which shows a sample processed frame during navigation. The dot indicated by the arrow highlights the location where the landmark was found in Fig. 2.18(d). The robot is able to traverse the specified route, detecting all visual landmarks embedded in complex backgrounds.

In the third trial, the same experimental setup as trial two was used except all the lights in the laboratory were turned off with windows remain open to simulate sudden change in image conditions. All landmarks are placed in complex cluttered backgrounds. A sample processed frame during the navigation is illustrated in Fig. 2.19. Similarly, the system is able to successfully traverse the route, recognising all landmarks under insufficient light conditions and embedded in cluttered backgrounds.

2.6 Conclusion

This chapter has provided an insight into autonomous vision-based autonomous robot navigations, focusing on monocular vision and navigation by 2D landmark recognitions in clean and cluttered backgrounds as well as under different lighting conditions. The essential components of monocular vision systems are described in details including; maps, data acquisition, feature extraction, landmark recognition and self-localisation. Then a 2-D landmark recognition architecture named selective visual attention landmark recognition (SVALR) is proposed based on a detailed analysis of how the Adaptive Resonance Theory (ART) model may be extended to provide a real-time neural network that has more powerful attentional mechanisms. This leads to the development of the Selective Attention Adaptive Resonance Theory (SAART) neural network. It uses the established memory to selectively bias the competitive processing at the input to enable landmark recognitions in cluttered backgrounds. Due to the dynamic nature of SAART, it is very computationally intensive. Therefore the main concept in the SAART network (top down presynatic facilitation) is re-engineered and is named memory feedback modulation (MFM) mechanism. Using the MFM mechanism and in combination with standard image processing architecture leading to the development of the SVALR architecture.

A robot platform is developed to demonstrate the SVALR architecture applicability in autonomous vision-based robot applications. A SMIS mechanism was added to the SVALR architecture to cope with image distortions due to wireless video links and the dynamic changing in landmark's size and shape. The SMIS mechanism uses the concepts of band transformations and the shape attraction to achieve image distortions, small size and view invariant landmark recognitions. The experiments show that the SVALR architecture is capable of autonomously navigating the laboratory environment, using the recognition of visual landmarks and a topological map to perform self-localisation. The SVALR architecture is capable of achieving real-time 2-D landmark recognitions in both clean and complex cluttered backgrounds as well as under different lighting conditions.

The SVALR architecture performance is based on the assumptions that all visual landmarks are not occluded and only one landmark is searched for and recognised at a time. Thus the problems of partial and multiple landmarks recognitions haven't been addressed. Furthermore, the robot platform is designed and implemented with the primary purpose of validating the SVALR architecture and therefore omitting the obstacle avoidance capability. In addition, the memory used in the SVALR architecture is preselected prior to navigation and cannot be changed dynamically. This give rises to a need for developing an obstacle avoidance capability and an adaptive mechanism to provide some means of learning to the SVALR architecture to cope with real-life situations, where landmarks move or change its shape and orientations dynamically. These problems are remained as future research.

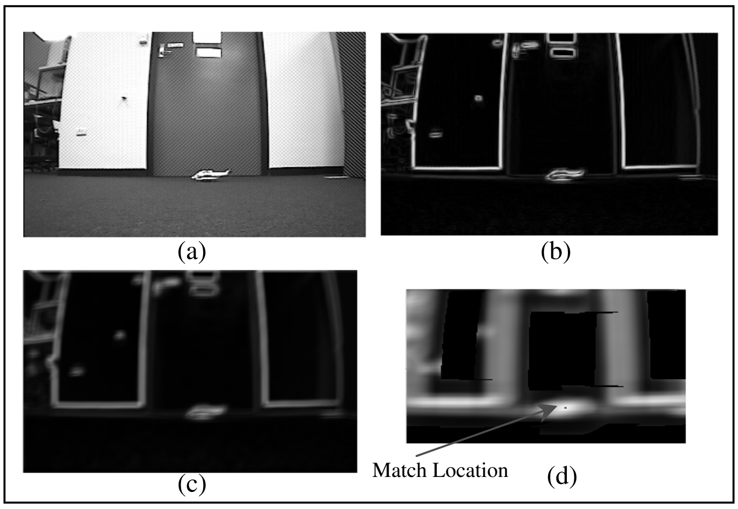


Fig. 2.17. A processed frame from the first trial, encountered in the navigational phase. (**a**) Grey level image, (**b**) Sobel edge image, (**c**) Blurred image using a 5x5-averaging mask and (**d**) The degree of match of the input image with the memory image at each location

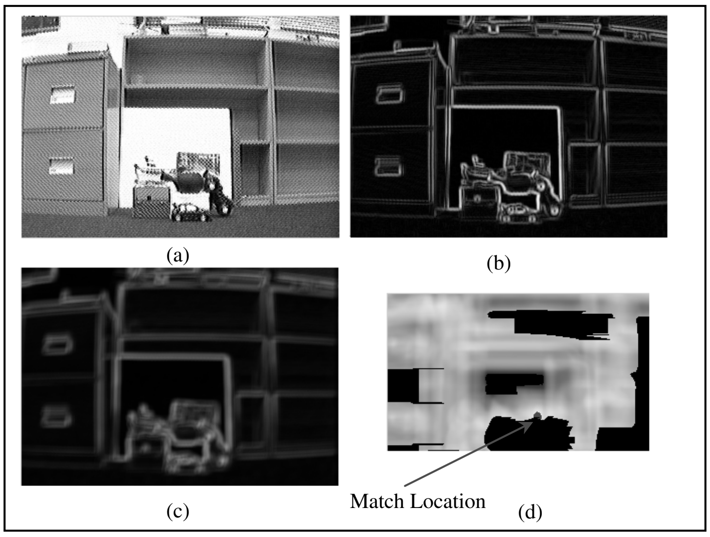


Fig. 2.18. A processed frame from the laboratory environment, encountered in the navigational phase. (**a**) Grey level image, (**b**) Sobel edge image, (**c**) Blurred image using a 5x5-averaging mask and (**d**) The degree of match of the input image with the memory image at each location

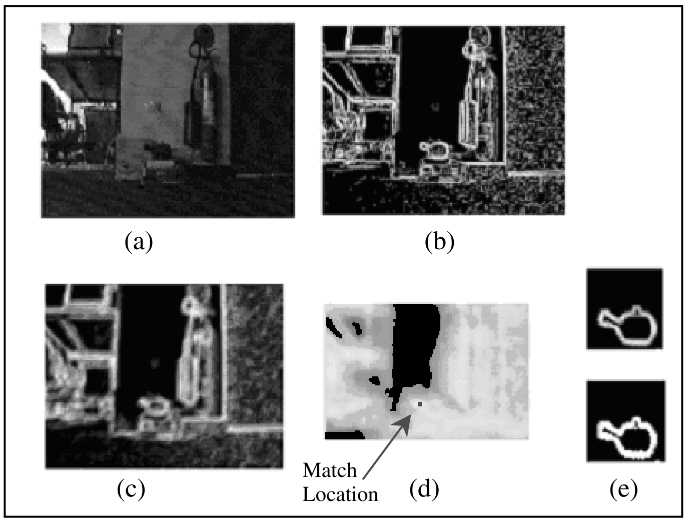


Fig. 2.19. A sample processed frame during the second trial with all light turned off, with minimal light enters from the laboratory windows. (**a**) Gray level input image, (**b**) contrast-based edge detection, (**c**) Blurred image, (**d**) degree of matches, converted into image scale and (**e**) Memory image and BMF filter, top and bottom respectively

Acknowledgements

We would like to thank the Australian Defence Science and Technology Organisation (DSTO) for supporting this research (contract No. 4500 177 390). Furthermore, we sincerely appreciate the technical assistance from Patrick O'Sullivan and Tony Gelonese in the School of Electrical and Information Engineering, University of South Australia, and Paul Munger and Adrian Coulter of DSTO.

References

1. P. J. M^{c}Kerrow, "Where are all the Mobile Robots?," in *Studies in Fuzziness and Soft Computing*, *Applied Intelligent Systems*, J. Fulcher and J. LAckhmi C, Eds.: Springer, 2004, pp. 179-200.
2. B. K. Muirhead, "Mars Pathfinder flight system integration and test," *in Proc. The IEEE Conference on Aerospace*, pp.191-205, 1997.

3. L. Pedersen, M. Bualat, C. Kunz, S. Lee, R. Sargent, R. Washington, and A. Wright, "Instrument deployment for Mars Rovers," *in Proc. IEEE International Conference on Robotics and Automation, Proceedings. ICRA '03*, pp.2535 - 2542, 2003.
4. N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor, "Omni-directional vision for robot navigation," *in Proc. IEEE Workshop on Omnidirectional Vision*, pp.21-28, 2000.
5. R. Ghurchian, T. Takahashi, Z. D. Wang, and E. Nakano, "On robot self-navigation in outdoor environments by color image processing," *in Proc. The 7th International Conference on Control, Automation, Robotics and Vision, ICARCV'02.*, pp.625-630, 2002.
6. D. Murray and C. Jennings, "Stereo vision based mapping and navigation for mobile robots," *in Proc. IEEE International Conference on Robotics and Automation*, pp.1694-1699, 1997.
7. M. Xie, C. M. Lee, Z. Q. Li, and S. D. Ma, "Depth assessment by using qualitative stereo-vision," *in Proc. IEEE International Conference on Intelligent Processing Systems, ICIPS '97*, pp.1446-1449, 1997.
8. G. N. DeSouza and A. C. Kak, "Vision for Mobile Robot Navigation: A Survey," *IEEE Transactions on Pattern Analysis and Machine Interlligence*, vol. vol.24, pp. 237-267, 2002.
9. D. Kortenkamp and T. Weymouth, "Topological Mapping for Mobile Robots Using a combination of Sonar and Vision Sensing," *in Proc. Proc. 12th Nat'l Conf. Artificial Intelligence,*, pp.979-984, 1995.
10. X. Lebegue and J. K. Aggarwal, "Automatic creation of architectural CAD models," *in Proc. The 1994 Second CAD-Based Vision Workshop*, pp.82-89, 1994.
11. X. Lebegue and J. K. Aggarwal, "Generation of architectural CAD models using a mobile robot," *in Proc. IEEE International Conference on Robotics and Automation*, pp.711-717, 1994.
12. V. Egido, R. Barber, M. J. L. Boada, and M. A. Salichs, "Self-generation by a mobile robot of topological maps of corridors," *in Proc. The IEEE International Conference on Robotics and Automation, Proceedings. ICRA '02.*, pp.2662 -2667, 2002.
13. T. Duckett and U. Nehmzow, "Exploration of unknown environments using a compass, topological map and neural network," *in Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp.312-317, 1999.
14. X. Lebegue and J. K. Aggarwal, "Significant line segments for an indoor mobile robot," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 801-815, 1993.
15. H. P. Moravec and A. Elfes, "High Resolution Maps from Wide Angle sonar," *in Proc. Proc. IEEE Int'l conf. Intelligent Robotics and Automation*, pp.116-121, 1985.

16. J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fact mobile robots," *in Proc. IEEE Transactions on Systems, Man and Cybernetics*, pp.1179 - 1187, 1989.
17. J. Borenstein and Y. Koren, "High-speed obstacle avoidance for mobile robots," *in Proc. Intelligent Control, 1988. Proceedings., IEEE International Symposium on*, pp.382 -384, 1988.
18. A. Bandera, C. Urdiales, and F. Sandoval, "An hierarchical approach to grid-based and topological maps integration for autonomous indoor navigation," *in Proc. The IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.883-888, 2001.
19. D. Jung and A. Zelinsky, "Integrating spatial and topological navigation in a behaviour-based multi-robot application," *in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS '99*, pp.323 -328, 1999.
20. M. Tomono and S. Yuta, "Mobile robot localization based on an inaccurate map," *in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.399 -404, 2001.
21. M. Tomono and S. Yuta, "Indoor Navigation Based on an Inaccurate Map Using Object Recognition," *in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.619 - 624, 2002.
22. M. Tomono and S. Yuta, "Mobile robot navigation in indoor environments using object and character recognition," *in Proc. ICRA '00. IEEE International Conference on Robotics and Automation*, pp.313 -320, 2000.
23. A. Murarka and B. Kuipers, "Using CAD drawings for robot navigation," *in Proc. IEEE International Conference on Systems, Man, and Cybernetics*, pp.678-683, 2001.
24. G. Cheng and A. Zelinsky, "Real-time visual behaviours for navigating a mobile robot," *in Proc. The International Conference on Intelligent Robots and Systems, IROS 96*, pp.973-980, 1996.
25. Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using view-sequenced route representation," *in Proc. Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, pp.83-88, 1996.
26. G. Cheng and A. Zelinsky, "Real-time visual behaviours for navigating a mobile robot," *in Proc. Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, pp.973-980, 1996.
27. R. C. Luo, H. Potlapalli, and D. W. Hislop, "Neural network based landmark recognition for robot navigation," *in Proc. The 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation*, pp.1084 - 1088, 1992.
28. H. Li and S. X. Yang, "Ultrasonic sensor based fuzzy obstacle avoidance behaviors," *in Proc. IEEE International Conference on Systems, Man and Cybernetics*, pp.644-649, 2002.
29. E. Krotkov, "Mobile robot localization using a single image," *in Proc. IEEE International Conference on Robotics and Automation*, pp.978-983, 1989.
30. Y. Watanabe and S. Yuta, "Position estimation of mobile robots with internal and external sensors using uncertainty evolution technique," *in Proc. The*

International IEEE Conference on Robotics and Automation, pp.2011-2016, 1990.
31. H.-J. von der Hardt, D. Wolf, and R. Husson, "The dead reckoning localization system of the wheeled mobile robot ROMANE," *in Proc. Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on*, pp.603-610, 1996.
32. C.-C. Tsai, "A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements," *in Proc. Instrumentation and Measurement Technology Conference, 1998. IMTC/98. Conference Proceedings. IEEE*, pp.144-149, 1998.
33. H. Makela and K. Koskinen, "Navigation of outdoor mobile robots using dead reckoning and visually detected landmarks," *in Proc. Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pp.1051-1056, 1991.
34. J. Moran and R. Desimone, "Selective attention gates visual processing in the extrastriate cortex," *Science*, vol. 229, pp. 782-784, 1985.
35. B. C. Motter, "Focal attention produces spatially selective processing in visual cortical areas V1, V2, and V4 in the presence of competing stimuli," *Journal of Neurophysiology*, vol. 70, pp. 909-919, 1993.
36. L. Chelazzi, E. K. Miller, J. Duncan, and R. Desimone, "neural basis for visual search in inferior temporal cortex," *Nature*, vol. 363, pp. 345-347, 1993.
37. R. Desimone, M. Wessinger, L. Thomas, and W. Schneider, "Attentional control of visual perception: Cortical, and subcortical mechanisms," *Cold Spring Harbour Symposium in Quantitative Biology*, vol. 55, pp. 963-971, 1990.
38. R. Desimone and J. Duncan, "Neural Mechanisms of selective visual attention," *Annual Review of Neuroscience*, vol. 18, pp. 193-222, 1995.
39. R. Desimone, "Neural mechanisms for visual memory an their role in attention," *in Proc. Proceedings of the National Academy of Sciences*, USA, pp.13494-13499, 1996.
40. P. Lozo, "Selective attention adaptive resonance theory (SAART) neural network for neuro-engineering of robust ATR systems," *in Proc. IEEE International Conference on Neural Networks*, pp.2461-2466, 1995.
41. P. Lozo. 1997, Neural theory and model of selective visual attention and 2D shape recognition in visual clutter, PhD Thesis, Department of Electrical and Electronic Engineering. Adelaide, University of Adelaide
42. S. Grossberg, "Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, and illusions," *Biological Cybernetics*, vol. 23, pp. 187-202, 1976.
43. S. Grossberg, "How does a brain build a cognitive code?," *Psychological Review*, vol. 87, pp. 1-51, 1980.
44. G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54-115, 1987.

45. G. A. Carpenter and S. Grossberg, "ART 2: Self-organization of stable category recognition codes for analog input patterns.," *Applied Optics*, vol. 26, pp. 4919-4930, 1987.
46. G. A. Carpenter and S. Grossberg, "ART 3: Hierarchical search using chemical transmitters in self-organising pattern recognition architectures.," *Neural Networks*, vol. 3, pp. 129-152, 1990.
47. S. Grossberg, "Neural expectation: Cerebellar and retinal analogs of cells fired by learnable or unlearned pattern classes," *Kybernetik*, vol. 10, pp. 49-57, 1972.
48. G. A. Carpenter, S. Grossberg, and J. Reynolds, "ARTMAP: a self-organizing neural network architecture for fast supervised learning and pattern recognition," *in Proc. International Joint Conference on Neural Networks, IJCNN-91-Seattle*, pp.863-868, 1991.
49. G. A. Carpenter, S. Grossberg, and J. H. Reynolds, "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Networks*, vol. 4, pp. 565-588, 1991.
50. P. Lozo, "Neural Circuit For Matchhnismatch, Familiarity/novelty And Synchronization Detection In Saart Neural Networks," *in Proc. Thr Fourth International Symposium on Signal Processing and Its Applications, ISSPA'96*, pp.549-552, 1996.
51. P. Lozo and N. Nandagopal, "Selective transfer of spatial patterns by presynaptic facilitation in a shunting competitive neural layer," *in Proc. The Australian and New Zealand Conference on Intelligent Information Systems*, pp.178-181, 1996.
52. P. Lozo, "Neural Circuit For Self-regulated Attentional Learning In Selective Attention Adaptive Resonance Theory (saart) Neural Networks," *in Proc. The Fourth International Symposium on Signal Processing and Its Applications, ISSPA-96*, pp.545-548, 1996.
53. B. Juesz and J. R. Bergen, "Texons, the Fundamental elements in pre-attentive vision and perception of textures," *Bell System Technical Journal.*, vol. 2, pp. 1619-1645, 1983.
54. E. W.-S. Chong. 2001, A Neural Framework for Visual Scene Analysis with Selective Attention, PhD Thesis, Department of Electrical and Electronic Engineering, University of Adelaide
55. Q. V. Do, P. Lozo, and L. Jain, "A Fast Visual Search and Recognition Mechanism for Real-time Robotic Applications," *in Proc. The 17th Australian Joint Conference on Artificial Intelligence*, Cairns, Australia, pp.937-342, 2004.
56. J. Westmacott, P. Lozo, and L. Jain, "Distortion invariant selective attention adaptive resonance theory neural network," *in Proc. Third International Conference on Knowledge-Based Intelligent Information Engineering Systems*, USA, pp.13-16, 1999.
57. P. Lozo, J. Westmacott, Q. V. Do, L. Jain, and L. Wu, "Selective Attention Adaptive Resonance Theory and Object Recognition," in *Studies in Fuzziness and Soft Computing*, *Applied Intelligent Systems*, J. Fulcher and L. C. Jain, Eds.: Springer, 2004, pp. 301-320.

58. Q. V. Do, P. Lozo, and L. C. Jain, "Autonomous Robot Navigation using SAART for Visual Landmark Recognition," *in Proc. The 2nd International Conference on Artificial Intelligence in Science and Technology*, Tasmania, Australia, pp.64-69, 2004.
59. S. Grossberg and D. Todorovic, "Neural dynamics of 1-D and 2-D brightness perception: A unified model of classical and recent phenomena," *Perception and Psychophysics*, pp. 241-277, 1988.
60. S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Networks*, vol. 1, pp. 17-61, 1988.

3 Multi View and Multi Scale Image Based Visual Servo For Micromanipulation

Rajagoplalan Devanathan[1], Sun Wenting[1], Chin Teck Chai[1], An-drew Shacklock[2]

1. School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798. appa_offfice@yahoo.com, ewtsun@pmail.ntu.edu.sg, etcchin@ntu.edu.sg
2. Singapore Institute of Manufacturing Technology 71 Nanyang Drive Singapore 638075 andrewps@SIMTech.a-star.edu.sg

3.1 Introduction

In this article, we present vision-based techniques for solving some of the problems of micromanipulation. Manipulation and assembly at the micro scale is a critical issue in a diverse of industries as the trend for miniaturization continues. We are also witnessing a proliferation of biomedical applications that require precise manipulation of delicate living material. However, there are many problems and uncertainties encountered when working at the micro scale. There is therefore a dependence on human interaction for reduction of this uncertainty. There is an urgent need to reduce this dependency or at lease enhance the performance of operators in tasks which are unsuitable for automation. Many promising businesses in the biomedical sector are struggling due to problems of yield and productivity, whereas in the MEMS industry devices never leave the research laboratories because the practicalities of manufacture remain unsolved.

The work presented here is part of a program of collaborative research at Nanyang Technological University (NTU) and the Singapore Institute of Manufacturing Technology (SIMTech). The aims are to characterize and understand the uncertainties as well as build demonstration systems that implement solutions to the problems of micromanipulation. A key to this is the interaction of humans and systems across the large differences of scale. The strategy is to find optimum division of tasks according to the relative capabilities of human and machine/system. We want the human to concentrate on high level task decisions and data interpretation whilst the machine handles the tracking and precision manipulation at a lower level. In the longer term, the autonomous system will take on more of the perception and decision making functions as levels of automation increase.

There is a clear role for visual servoing when we want the machine to finish the fine positioning task initiated as a high level command. The distinction in this work, is that the command and servoing can be take place over different views at differing scales. This is similar to a coarse-fine motion strategy except that the availability of multiple data is exploited in the reduction of overall system uncertainty. In the ensuing sections, the multiple view and multiple scale algorithms will be presented. Before that, it is necessary to clarify what is understood by the term micromanipulation and highlight the difficulties that make manipulation tasks so difficult at the micro scale.

Understanding of the term micromanipulation varies because applications are diverse and the dimensions of object and work volume differ in scale. Micromanipulation is commonly defined in terms of the object as ``the controlled movement of entities with dimensions ranging from 1 μm to 1 mm scale using any method."

This definition is very broad in scope and it embraces objects that are still visible to the human eye. Applications of micromanipulation include the handling of biological cells and even DNA and molecules. Some argue that once the dimensions of the objects are less than 1 μm the realm becomes nanomanipulation.

An explanation from a biomedical perspective relates to the dimensions of the tools and the fact that the task is performed under the view of a microscope. A dictionary definition is ``the technique of using delicate instruments, such as microneedles and micropipettes, to work on cells, bacteria, etc., under high magnification, or of working with extremely small quantities in microchemistry." In micro-injection [1] the tool is fixed on the micromanipulator, which has multiple degrees of freedom, and is guided to pierce the target micro object. The task is difficult because the objects (cells, seeds etc) are small and delicate.

The term is also applied to manipulation tasks to position micro ojects. In a micro assembly task, the alignment is precise but the objects may be at the meso scale. For example, in 3D micro assembly tasks, between 4 or 6 degrees of freedom are required. In [2], the tasks are distributed are ditributed on a 100 mm wafer, and the assembly tolerance is typically in the order of microns. The micromanipulator has to traverse a long range and achieve high resolution as well. Another aspect of micromanipulation is

found in the assembly of structures from many micro sized parts. [3] described the approach to design and fabricate scaffold/cell constructs for tissue engineering.
There are many problems which make micromanipulation very difficult for both man and machine. Generally, it is the uncertainty resulting from huge scale differences that cause the major problems.

Micro physics Objects behave very differently at the micro scale when compared to our physical experiences of handling. As object dimensions fall below 100^{1}m forces other than gravity start to dominate and govern the behavior of the object.

Perception Perception is troublesome because the observation is remote. Detecting, sensing and visualization are all very difficult.

Environmental effect small changes in temperature induce great effects in micromanipulation, which are negligible at the macro scale. Humidity and extraneous particles (dust) both cause serious problems.

To reduce the uncertainties in micromanipulation, a common approach is to: (1)control the environmental variables with clean rooms, humidity and temperature control; (2) increase the precision for mechanisms, tools and fixtures, which is associated with necessary procedures of recalibration and re-configuration for different applications. Development of elements for achieving high performance requires different principles and designs for different tasks [4, 5]. Efforts in these directions are certainly necessary but they increase cost and conflict with the need to increase flexibility (ease of reconfiguration) and productivity. As the scale decreases, uncertainties caused by practical limits of these devices still needs to be compensated. So complementary methods are needed to achieve reconfigurability and ease of use.

Another approach is to accept that there will be uncertainty and learn how to cope with it. For example by sensing and adapting the task strategy accordingly. However this takes us back to the aforementioned problems of perception. Vision and haptic are the two main sensing techniques for manipulation. Both play an important and complementary role in micromanipulation but this program of work concentrates on visual techniques. The aim is to develop automatic system that facilitate human operator interaction. The man-machine interface(MMI)is the most apparent feature of the system but its success depends on the underlying understanding of the uncertainties of the complete system and task. The work has three main aspects:

- Visualisation and interface tools.
- Visual servoing.
- Automatic determination of system parameters.

The first part is based on ARGUS, computational software based on the algebraic geometry of multiple views. This software resolves uncertainty across multiple viewpoints and frames but does not implement any control. Control is the responsibility of the visual servo modules. The distinction between these modules is not clear cut as they work together to provide the evidence needed to reduce uncertainty and tune system parameters. It is this integration that is so important in handling the problems of differing scales and multiple views.

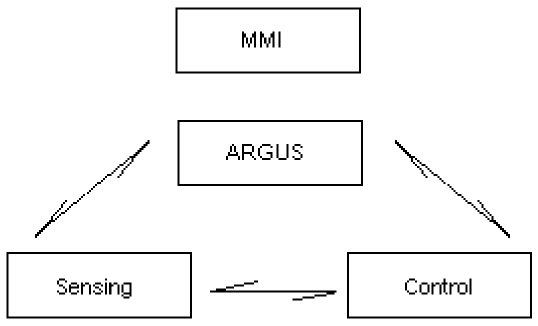


Fig. 3.1. Illustration of the System Hierarchy

In the following sections, we introduce the vision based approaches used to provide the human operator assistance for solving the above mentioned problems. In vision based methods, multiple views which consist of macro projective image and microscopic image provide global information beyond the limited field of view as well as detailed information to provide suffcient resolution for precision. We will describe this multiple view multiple scale image based visual servo. In this vision based method, feature detection, correspondence finding and correction, and motion estimation from images are very important. Many of the techniques are beyond the scope of this article but when necessary reference will be made to these functions too. The chapter is structures as follows as follows. In section 3.2, the difficulties for micromanipulation are listed. The problems of existing vision based methods are covered in section 3.3. The approaches for multiple view multiple scale image based visual servo is developed in sec tion 3.4. The simulation set up will be introduced in section 3.5. The experiment results are presented in section 3.6. Conclusions are drawn in section 3.7.

3.2 Difficulties in Micromanipulation

The development of an automated and efficient manipulation system is demanded to improve the industrial productivity and release the burden for the human operators. However, there are several problems concerning micromanipulation.

3.2.1 Scaling Effect

When the objects are less than 1mm in size, the physics that dominates is completely different [6]. The conventional manipulation can be modelled based on Newtonian mechanics, however, when the scaling decreases physical phenomena in micro world become substantially different from macro world, which make system performance of conventional techniques degrade or even fail. For this reason, the physical differences and their effect in micromanipulation systems have to be considered. Many surface forces as van der Waal's, electrostatic, and surface tension become dominant over gravity in micro scale. Van der Waals forces are caused by quantum mechanical effects. Electrostatic forces are due to charge generation or charge transfer during contact. Surface tension effects arise from interactions of layers adsorbed moisture on the two surfaces. When conducting manipulation in conventional world, we can place and pick up object as desired, while in micro world, the object will stick to the gripper due to the surface forces, see **Fig. 3.2**; free standing micro structures tends to stick to the substrate after being released during processing. Attempts to reduce the adhesive forces in micro world can be found in [7, 8].

Environmental conditions, such as temperature and humidity can also influence the adhesion forces and microbiologically properties of micro parts and cause many uncertainties [9].

Besides, when manipulating on several objects, the area may in the order of several millimeters, while the requirement of accuracy may be in the order of nanometer. if we transport the end effector between objects and manipulate on different objects, the manipulator must have centimeter or-der moving mechanism and nanometer order position accuracy. There will be need for tradeoff between efficiency and accuracy [10].

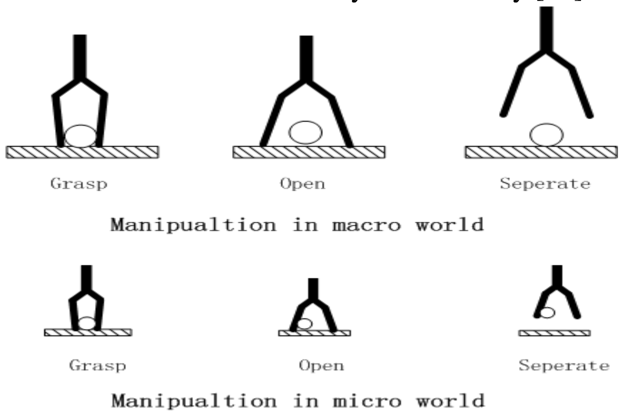


Fig. 3.2. Manipulation in macro/micro world

3.2.2 Spatial Uncertainty

Spatial uncertainty means that objects are not where we expect them. Spatial uncertainty causes many difficulties in manipulation of micro-scale objects. One cause of spatial uncertainty in micromanipulation is thermal drift between the tip and the sample. For the AFM, working at room temperature, in ambient air and without careful temperature and humidity control, a typical value for drift velocity is 0.05 nm/s. So after a certain period of time for scanning, the object will drift a distance that is approximately the size of the particles usually manipulated [11]. Hysteresis, creep, and other nonlinearities also cause problems not only in positioning error but also in instability.

3.2.3 Perception

Perception is another problem. Observed through microscope, the depth information of the object would be lost, the field of view becomes very small and much data is out of view. The perspective relation which we can make judgment of the spatial information does not hold, making the image ambiguous and confusing. In micromanipulation, observer is removed from the task, so the uncertainty of sensors has great effect on the operation and decision making, thus precision becomes very difficult to be achieved.

Furthermore, the operator is in macro world, while the object is in micro scale, so the propagation of errors and uncertainty over scale becomes crucial for micromanipulation. However, this is not a fully understood area.

Uncertainty effects and imprecision can be compensated using feedback control. [12, 13] proposed non-linear models for close loop control of piezoelectric actuators, [14, 15, 16] developed different position feedback techniques with calibration, visual servo, etc. Bilateral control, which reflect back the forces of operation environment to the operator, is reported can aid the operator to improve performance and even perform tasks that otherwise are beyond his capabilities [17, 18]. Sensors are needed to detect position errors, then suitable control laws are developed for compensation. A sensor based system can improve precision and reduce the need of expensive mechanisms and fixtures. Visual and haptic are two main sensors for micromanipulation. Haptic interface allows operator to feel and control the forces in the micro world [19], and compensate frictions [20]. while vision based method prevents any mechanical contact of the measurement system, capture multi dimensional nature of scene, easy to store, retrieve and memorize, besides, vision based method has the ability to bridge long distance transportation, making it suitable to be coded for tele-operation. And also because vision is a more mature and better understood technology, we will concentrate on visual sensing in this chapter.

3.3 Vision Based Methods

Vision can provide several functions to assist the operator for micromanipulation: It can detect features in the image; verify the input data and parameter estimation; and aids automatic tracking of feature and guided search.

However, vision strategies also suffer at this scale because the high magnification results in a very small field of view (FOV) and very small depth of field. It is therefore difficult to obtain a clear image if the object of interest is not planar or is subject to movement. If the amplitude of vibration of the object is large it may be impossible to obtain an image. If the sensor itself vibrates the problem is greatly magnified.

Often it is difficult to obtain any image of the region of interest (ROI) be cause it is occluded by tools and fixtures. Even if the ROI is imaged, there is still the problem of identifying where on the object the region corepond s to. The region may be very small in comparison with the working area (or volume).

The uncertainties can be reduced by calibration. F. Arai and T. Fukuda tried to compensate uncertainty by calibrating the absolute position by relative movement of the manipulator [21, 22]. They calibrated a three dimensional tool position against misalignment of the system components and tool exchange with the geometrical error directly. Visual feedback is used to detect the position of the micro tool tip, the error of the stepping motor stage is measured by the linear scale. In [23], a method to calibrate the orientation of the tool tip is proposed.

People are also trying to model the uncertainties with virtual models. In [14, 15, 16], virtual reality (VR) was developed for micromanipulation. Difficulty of manipulating in 3D space with 2D microscopic image information was reduced by virtual reality [15, 16] with parallel to calibration. However, modeling the micro object with virtual reality itself already includes many uncertainty. However, to model the physics and the micro object itself is very difficult due to the lacking of well understood knowledge of micro physics. The parameters for modeling become uncertain, and will change due to the problems listed in the last section. So the difference between the model and the real situation will lead to imprecision for manipulation task.

Comparing to VR, augmented reality(AR) provide visual augmentation to a real world environment, unlike VR which replaces the real environment, AR enhance the real world view of the user with real images. The validity of the model can be seen, the limitation of the real images can be overcame. In the following section, the augmented reality will be introduced to our method.

Visual servo is another technique to compensate uncertainties. Several visual servo strategies have been successfully implemented in micromanipulation. [24, 25] present a visual servo system with optical microscope which does not use the system calibration and a model of the observed scene. Since the single field-of-view of optical microscope is limited to a very small area, the method does not provide information sufficient enough to solve ambiguities in the scene, so systems with multiple views are developed. Multiple magnification based micro vision feedback system was presented in [26, 27], in which pattern matching was preprocessed on a low magnification vision data to position the object in the center of a high magnification vision data. In [1, 28, 29,30], stereo microscopic images provide information for visual feedback. A micromanipulation system was proposed in [31], in which supervisory logic-based controller that selects feedback from multiple visual sensors in order to execute a micro assembly task is used.

In the next section, the proposed method will be presented.

3.4 Multi View Multi Scale Image Based Visual Servo

3.4.1 System

In the concept system, images from the microscope and other cameras are made available to the operator with graphical enhancement of visual cues and outof-view data. The workstation schematic is illustrated in **Fig. 3.3**. The manmachine-interface (MMI) provides the following functionality:

1. Subpixel feature referencing for operator interaction on perspective view points.
2. Out-of-view reconstruction on microscope views.
3. Map-type views using geometric primitives reconstructed from image data.
4. Issue of motion commands using the local coordinate frame of the chosen view (i.e. image or map coordinates).

The visualization system performs precise tracking and estimation so that commands can be executed based on features that are determined at a resolution beyond the specification of the camera and display. The MMI also overcomes many of the problems of microscope visualization such as loss of information from limited depth-of-field and field-of-view. However, these concepts will fail unless particular care is taken to ensure reliable modeling and transformation of data. The total system will have increased uncertainty because priority is given to user-preferences over rigidity of fixtures and component lay-out.

In the experiment setup, the sample is located on a 3 degrees of freedom (DOF) stage, and observed from the optical microscope which is mounted by a CCD camera. Another CCD camera is positioned arbitrarily in the 3D space to get the full view of the work space.(See **Fig. 3.4**)

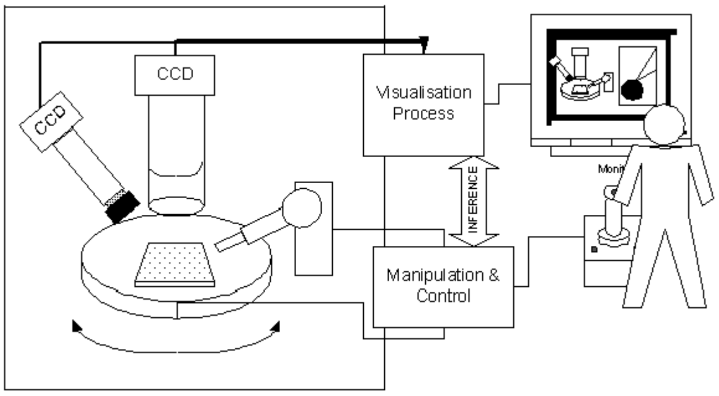


Fig. 3.3. The Concept of Micro-Assembly Workstation

The proposed strategy is that visual methods will be used for object tracking, identification and localization within a `Coarse-Fine' strategy. Visual servoing will be used to provide the precise 2D servoing needed to compensate for system uncertainty. Vision will also form the core of the Man-Machine-Interface (MMI). The real images from the microscope and tracking cameras will be made available to the operator with graphical enhancement of visual cues and out-of-view data. This will assist the operator in interpretation and command issue thus increasing productivity and reducing fatigue.

The system concept is summarised as follows. One (or more) standard CCD camera(s) provides views of the object (and global scene). These views are used to track the motion of the sample and tools relative to the microscope viewing window. Another camera integrated with the microscope provides the fine de-tail for precise tracking of motion.

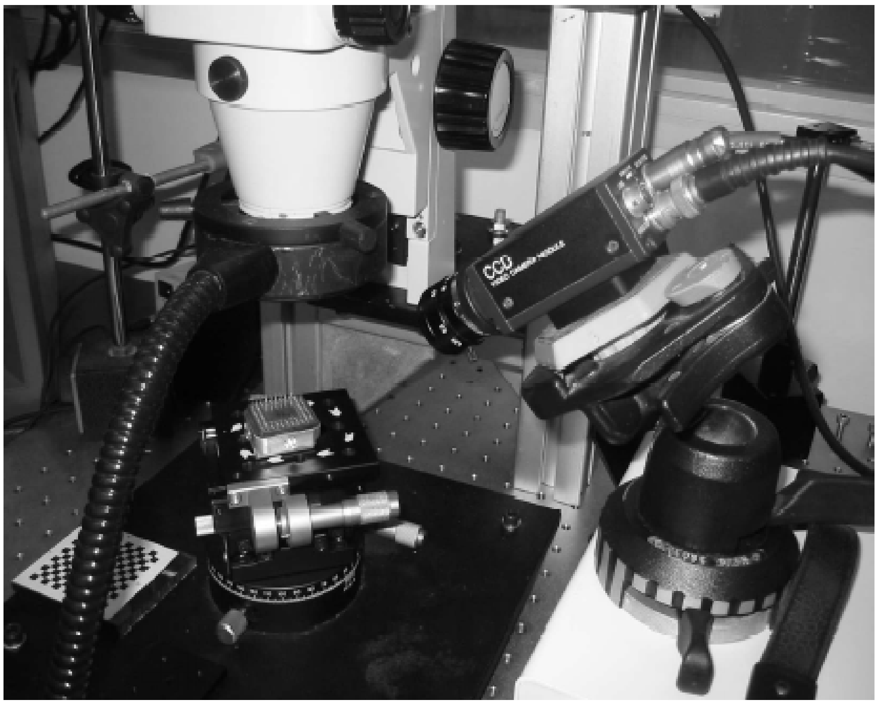


Fig. 3.4. System Setup

3.4.2 Methodology

Visual control of manipulators promises substantial advantages when working with targets whose position is unknown or with manipulators which may be flexible or inaccurate. Visual servoing control structures have been categorized as being either image-based or position based [32]. The essence of image-based feedback is the image Jacobian $\mathbf{J}_v$, which is a linear transform relating the velocity of image feature motion to the velocity of the motion in 3D space with respect to camera coordinates.

In our case, the target region is not in the field of view of the microscope at first. So the image based visual servo is started with the macro image from the macro camera. This is an eye-to-hand configuration [33], which should consist of a transform of the velocity screw

($\dot{r} = [T_x, T_y, T_z, \omega_x, \omega_y, \omega_z]^T$) of manipulator motion from camera coordinate system to world coordinate system.

The eye-in-hand image Jacobian (3 degree of freedom) relationship for the macro visual servoing is:

$$\dot{\mathbf{x}} = \mathbf{J}_v \dot{\mathbf{r}} \tag{1}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{Z_c} & 0 & -\frac{\lambda X_c}{Z_c^2} \\ 0 & \frac{\lambda}{Z_c} & -\frac{\lambda Y_c}{Z_c^2} \end{bmatrix} \begin{bmatrix} {}^cT_x \\ {}^cT_y \\ {}^cT_z \end{bmatrix} \tag{2}$$

where $\dot{\mathbf{x}}$ is the derivative of image feature, $[{}^cT_x, {}^cT_y, {}^cT_z]^T$ is the control vector with respect to the camera coordinates. We use the control law [24] below:

$$\begin{bmatrix} {}^cT_x \\ {}^cT_y \\ {}^cT_z \end{bmatrix} = -k\hat{\mathbf{J}}_v^{\,+}(\mathbf{x} - \mathbf{x}^*) \tag{3}$$

$\hat{\mathbf{J}}_v^{\,+}$ is the pseudo inverse of the estimated image Jacobian in macro view, k is the proportional control gain, $\mathbf{x}^*$ is the target feature coordinates in macro image. Note that $[\mathbf{R},\ \ \mathbf{t}]$ defines a mapping from camera frame c to the target frame ω . The control vector can be converted to $[{}^{\omega}T_x, {}^{\omega}T_y, {}^{\omega}T_z]^T$ with respect to the target frame by:

$$\dot{\mathbf{r}} = \begin{bmatrix} {}^{\omega}\mathbf{T} \\ {}^{\omega}\Omega \end{bmatrix} = \begin{bmatrix} \mathbf{R}{}^c\mathbf{T} - \mathbf{R}{}^c\Omega \times \mathbf{t} \\ \mathbf{R}{}^c\Omega \end{bmatrix} \tag{4}$$

In this case, we are considering 2 degrees of freedom(DOF), hence, from the above transform, we have:

$$\dot{\mathbf{r}} = \begin{bmatrix} {}^wT_x \\ {}^wT_y \\ {}^wT_z \end{bmatrix} = \mathbf{R}{}^c\mathbf{T} - \mathbf{R} \times \mathbf{t} \tag{5}$$

Force T_z to be 0 (assume the motion is planner), the velocity screw of 2DOF can be generated as:

$$\dot{\mathbf{r}} = \begin{bmatrix} {}^wT_x \\ {}^wT_y \end{bmatrix} = \mathbf{R}_{xy}{}^c\mathbf{T}_{xy} - \mathbf{R}_{xy} \times \mathbf{t}_{xy} \tag{6}$$

We can obtain the micro image Jacobian similar to that of macro image [35]:

$$\dot{\mathbf{x}}' = \mathbf{J}_v{}' \dot{\mathbf{r}} \tag{7}$$

$$\dot{\mathbf{x}}' = \begin{bmatrix} \dot{x}' \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \beta & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} {}^wT_x \\ {}^wT_y \end{bmatrix} \tag{8}$$

where $\beta = \alpha / s$ α is the total magnification of the microscope, and s is the effective size of micro image pixel. So the micro image Jacobian can be estimated as a constant.

We use the micro image features and micro image Jacobian to update the estimation of the stage position when correspondence can be found.

$$\mathbf{X}(k) = \mathbf{X}(k-1) + \hat{\mathbf{J}}_v{}'^{+} (\mathbf{x}'(k) - \mathbf{x}'(k-1)) \tag{9}$$

When the feature is difficult to be registered to the global view image, area based techniques can be used to estimate $\mathbf{x}'(k) - \mathbf{x}'(k-1)$.

When the interested object enters the switch area, fine positioning can be carried out. Micro image based visual servo is first undertaken with microscope image features. As the microscope coordinate is aligned with the target frame, this is an eye-in-hand configuration. We can get the velocity screw with respect to the world coordinates:

$$\begin{bmatrix} {}^wT_x \\ {}^wT_y \end{bmatrix} = -k' \hat{\mathbf{J}}_v{}'^{+} (\mathbf{x}' - \mathbf{x}'^{*}) \tag{10}$$

$\hat{\mathbf{J}}_v{}'^{+}$ is the pseudo inverse of the estimated image Jacobian in micro view, k' is the proportional control gain, $\mathbf{x}'^{*}$ is the target feature coordinates in micro image.

This time, the macro view image will be used to constrain the the sample object to be in the field of view regardless of vibration and drift. This is formulated as:

$$\boldsymbol{\theta} = \mathbf{J}_v^{\,*}\boldsymbol{\Delta} \tag{11}$$

where

$$\mathbf{J}_v^{\,*} = \begin{bmatrix} \dfrac{\lambda}{Z^*} & 0 \\ 0 & \dfrac{\lambda}{Z^*} \end{bmatrix} \tag{12}$$

and Z^* is an approximate value of Z_c at the desired target position with respect to the macro view camera, $\boldsymbol{\Delta}$ is the maximum distance micro view can cover in world space.

During the fine process, when the distance between current and former image features in macro view exceed $\boldsymbol{\theta}$, the process will be forced back to coarse positioning to relocate the interested sample. The positioning task will not switch to fine stage until the sample is relocated in the field of view.

3.4.3 Image Tracking

The multi view multi scale method is based on the estimation of motion from image scenes between macro and micro views. In practice, these are very difficult. In this section, we will introduce image tracking methods.

Optical flow is a commonly used method in object tracking [35, 36, 37]. The optical flow based algorithms extract a dense velocity field from an image sequence assuming that image intensity is conserved during the displacement. This conservation law is expressed by a spatiotemporal differential equation which is solved under additional constraints of different form.

Suppose that the image intensity is given by $I(x,t)$, where the intensity is now a function of time t, as well as of displacement x

Now, suppose that part of an object is at a position (x_1, x_2) in the image at time t, and that by a time τ later it has moved through a distance $\mathbf{d} = \begin{bmatrix} u \\ v \end{bmatrix}$, in the image.

By Taylor expansion, the intensity can be presented as:

$$I(\mathbf{x}+\mathbf{d}, t+\tau) \cong I(\mathbf{x}, t) + \nabla_x \cdot \mathbf{d} + \frac{\partial I}{\partial t}\tau + \cdots \tag{13}$$

where the dots stand for higher order terms.

Given a feature window W, we want to find the displacement which minimizes the sum of squared differences:

$$\varepsilon = \sum_W (\nabla_x \cdot \mathbf{d} + \frac{\partial I}{\partial t}\tau)^2 \tag{14}$$

By imposing that the derivatives of ε with respect to d are zero, we obtain:

$$\sum_W \begin{bmatrix} I_1^2 & I_1 I_2 \\ I_1 I_2 & I_2^2 \end{bmatrix} \cdot \mathbf{d} = -\tau \cdot \sum I_t \cdot \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \tag{15}$$

where

$$I_i = \frac{\partial I}{\partial x_i}, i = 1,2$$

$$I_t = \frac{\partial I}{\partial t} \tag{16}$$

We can compute $\mathbf{d} = \begin{bmatrix} u \\ v \end{bmatrix}$ from (15). Optical flow can perform well in short motion but it's not suitable for long distance as the assumption that the image intensity is conserved will not hold. So we are looking for more robust methods for image tracking, Markov Random Fields is a promising one. The tracking result with optical flow is shown in **Fig. 3.5**.

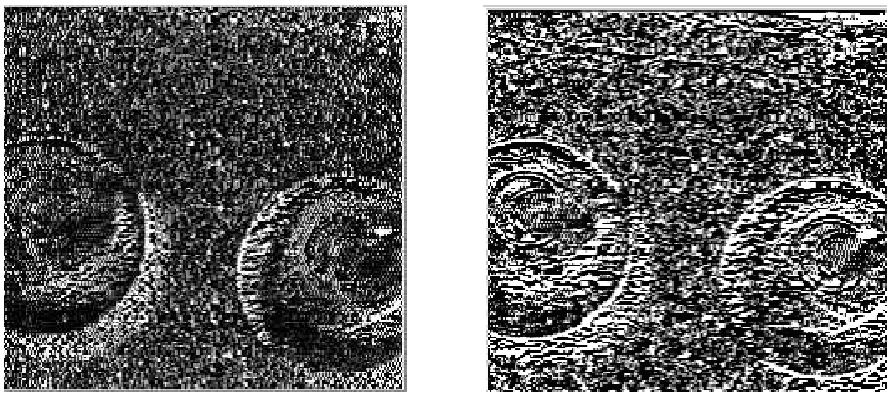


Fig. 3.5. *Left:* Optical Flow in X., *Right:* Optical Flow in Y

Markov Random Fields Markov Random Fields theory is a branch of probability theory for analyzing the spatial or contextual dependencies of physical phenomena. It was first used in visual labelling to establish probabilistic distributions of interacting labels [38]. Resent research has shown promising application to recovering of motion information under various environments [39, 40, 41]. Markov network is used to propagate likelihoods to best explain image data by inferring the underlying scene.

The problem of estimating displacement between image frames has been introduced to motion vector space as early as 1980s [42].

The observed image, g, which is related to the true underlying image, I, by some random transformation is considered to be a sample of a random field, G.

Disregarding occlusions and newly exposed areas, for every point in the preceding image, $t = t - 1$, there exists a corresponding point in the following image, $t = t$. Let the 2-D projection of the straight lines connecting these pairs of points be referred to as the displacement field, U, associated with the underlying image I. The true displacement field $\tilde{u} = (u(i,j), v(i,j))$, is a set of 2-D vectors such that for all x, the preceding point has moved to the following point $x(i + u(i,j), j + v(i,j), t)$ [43]. $\tilde{u}$ is assumed to be a sample from a random field U. Let $\hat{u}$ be an estimate of $\tilde{u}$ and u denote any sample field from U. (This relationship is shown in **Fig. 3.6**). By MRF, we can use the random field G, to find the displacement $\hat{u}$ between images from U.

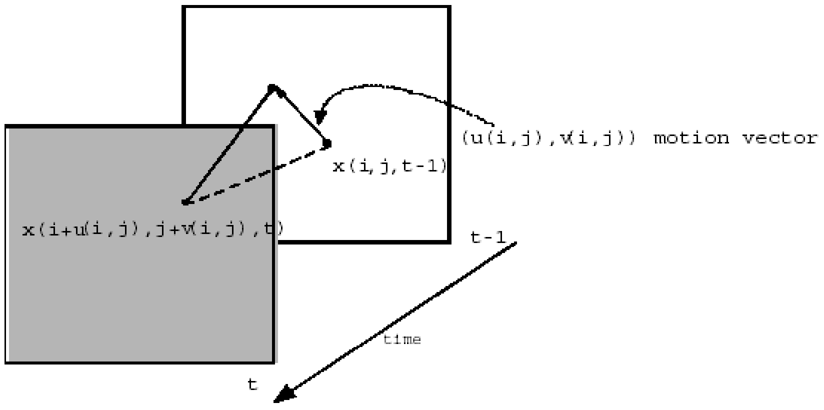


Fig. 3.6. Illustration of Motion Vector

The scene can be defined as the displacement space. Image sequences are connected with underlying scene patches; scene patches also connect with neighboring scene patches, while the neighboring relationship is with regard to different positions. The posterior distribution is modeled through the Gibbs distribution $P(\mathbf{d})$:

$$P(\mathbf{d}) = \frac{1}{Z}\exp(-E(\mathbf{d})) \tag{17}$$

where $\mathbf{d}$ is the matrix of all displacements $d_{i,j}$ and Z is a normalizing factor. The posterior distribution of displacement(P) between two images (I_1, I_2) can be derived from the prior(P_p) and measurement(P_m) models using Bayes' rule:

$$P(\mathbf{d} \mid I_1, I_2) \propto P_p(\mathbf{d})P_m(I_1, I_2 \mid \mathbf{d}) \tag{18}$$

which can be written as a matching energy function:

$$E(\mathbf{d}) = -\lg P(\mathbf{d} \mid I_1, I_2) \tag{19}$$

By maximizing $P(\mathbf{d} \mid I_1, I_2)$ (minimizing $E(\mathbf{d})$), the proper solution of displacement ($\mathbf{d}$) can be found. E_0 is modeled as the initial matching cost for iteration by:

$$E_0(i,j,d_{i,j}) = \rho_M(I_2(x_i + d_x, y_i + d_y) - I_1(x_i, y_i)) \tag{20}$$

where ρ_M is a contaminated Gaussian model (a mixture of a Gaussian distribution and a uniform distribution) [44], (x_i, y_i) refers to the pixel coordinates in image, and d_x, d_y is defined as the first and second element of $d_{i,j}$. The prior model is developed based on the Markov Random Fields theory that: if the joint probability distribution of all interacting neighbors is known, the local probability distribution of a site is completely determined. For facilitation, the smoothed probability distribution is generated:

$$p_s(i,j,d_{i,j}) = \sum_{d'} \exp(-\rho_P(d' - d_{i,j}))P(i,j,d') \tag{21}$$

where ρ_P is also a contaminated Gaussian model [44] and d' represents the neighbor site of $d_{i,j}$. The smoothed energy is:

$$E_s(i,j,d_{i,j}) = -\lg p_s(i,j,d_{i,j}) \tag{22}$$

$$E(i,j,d_{i,j}) = E_0(i,j,d_{i,j}) + \mu\left[E_s(i,j,d_{i,j}) + \sum_{(k,l)\in N_4} E_s(i+k, j+l, d_{i+k,j+l})\right] \tag{23}$$

where μ decides the speed of the process. This is also mentioned as a special nonlinear diffusion [44]. The statistical models of MRF characterize images, and allow computations of distances, yet are relatively insensitive to translation. In fact, MRF relates the spacial and temporal information together, to find the most likely displacement between image frames.

3.5 Simulation Setup

In this section, the simulation environment is set up for verification of the algorithm. We use simulation to control the noise and propagate noise across different views.

The simulation environment is shown in **Fig. 3.7**, **Fig. 3.8**, and **Fig. 3.9**. The rectangle in Cartesian space and macro view image is the view from microscope,whichisshownin thesimulatedmicroviewimage. The initial and target position of the interested object is also drawn in the image

The relation between world space and image spaces can be formulated by camera models. The related camera models are listed as follows.

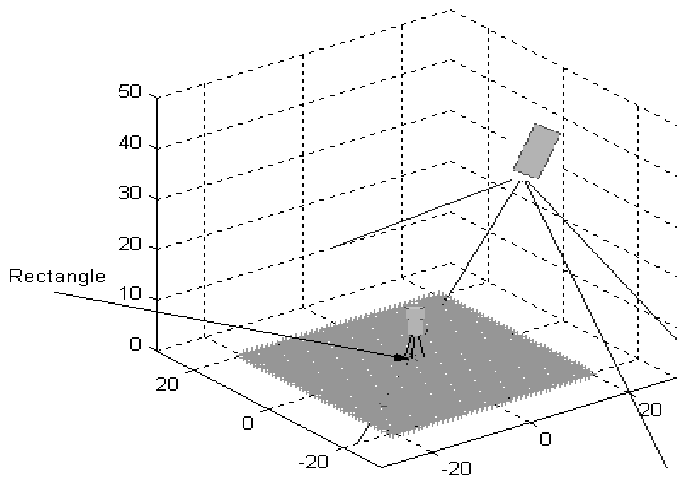


Fig. 3.7. The Simulated Cartesian Space

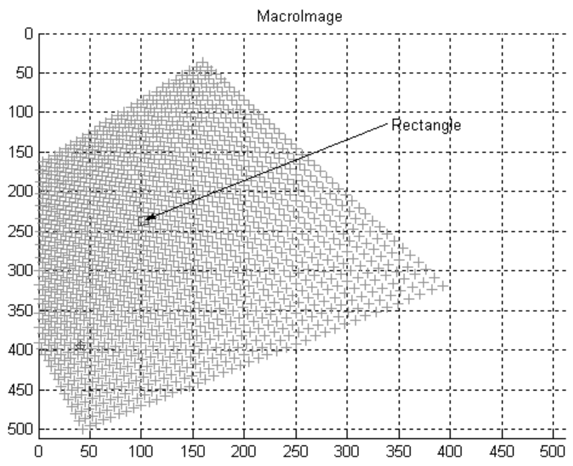


Fig. 3.8. The Simulated Macro Image

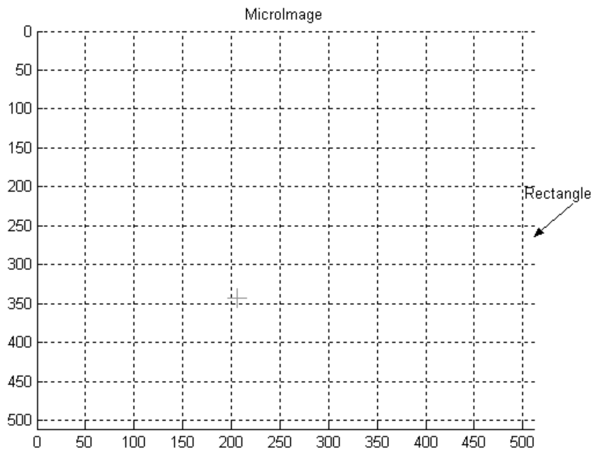


Fig. 3.9. The Simulated Micro Image

Macro View Modeling The camera model is shown in **Fig. 3.10**. Suppose there is a point $P(X_c, Y_c, Z_c)$ with respect to the camera coordinates in 3D work space. The corresponding point in the macro camera image p is described by the pixel value (x, y), so we get:

$$x_s = x - x_p = \frac{\lambda X_c}{Z_c} \tag{24}$$

$$x_s = x - x_p = \frac{\lambda X_c}{Z_c} \tag{25}$$

$$\lambda = {}^{f}\!/_{s} \tag{26}$$

where (x_s, y_s) is the new coordinates in the image, f is the focal length, s is the effective size of the pixel, and (x_p, y_p) is the principal point.

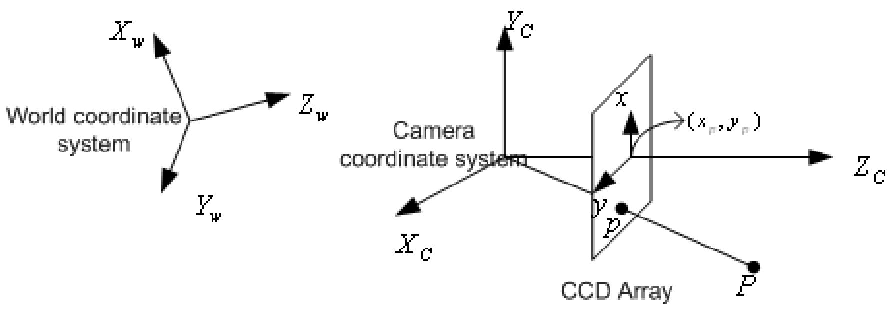


Fig. 3.9. Illustration of camera model

Micro View Modeling The simplified ray diagram for a typical optical mi croscope is shown in **Fig. 3.10**. g is called the optical tube length and is the distance between the posterior principal focal plane of the objective and the anterior principal focal plane of the projective eyepiece. For typical microscopes g is a constant. f_0 is the posterior objective focal length, f_t is the projective eyepiece focal length. c is the distance between the CCD receptor and posterior principal focal plane of the projective eyepiece.

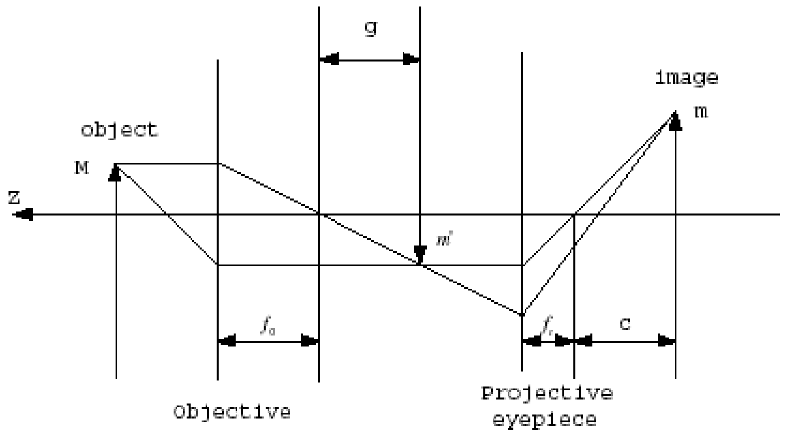


Fig. 3.10. Simplified Ray Diagram for Typical Optical Microscope

The intermediate image is projected at a distance g behind the posterior principal focus of the objective:

$$m' = \frac{Mg}{f_0} \tag{27}$$

$$m = \frac{m'c}{f_t} \tag{28}$$

then the total linear magnification is given by

$$\alpha = \frac{m}{M} = \frac{gc}{f_0 f_t} \tag{29}$$

m is the point in image plane with coordinates of $[x, y]^T$, M is the point in 3D work space with the coordinates of $[X, Y, Z]^T$.

The above transformations between world space and image spaces are 3 by 4. For simplicity, we assume the manipulator is operating on planar objects. Thus the projection between a world point X and an image point x can be formulated as 2D -2D projective mapping.

$$x = HX \tag{30}$$

where H is the homography mapping, and is invertible. The proposed method uses image features to control, thus inheriting the advantages of image basedvisual servo (reduse computation delay eliminate the necessity for image interpretation, and eliminate errors due to sensor modeling and camera calibration), while at the same time, the position error in 3D space is implied in the transform homography from images to 3D work space.

3.6 Experimental Results

In this section, the experiment results of the multi view multi scale method will be presented.

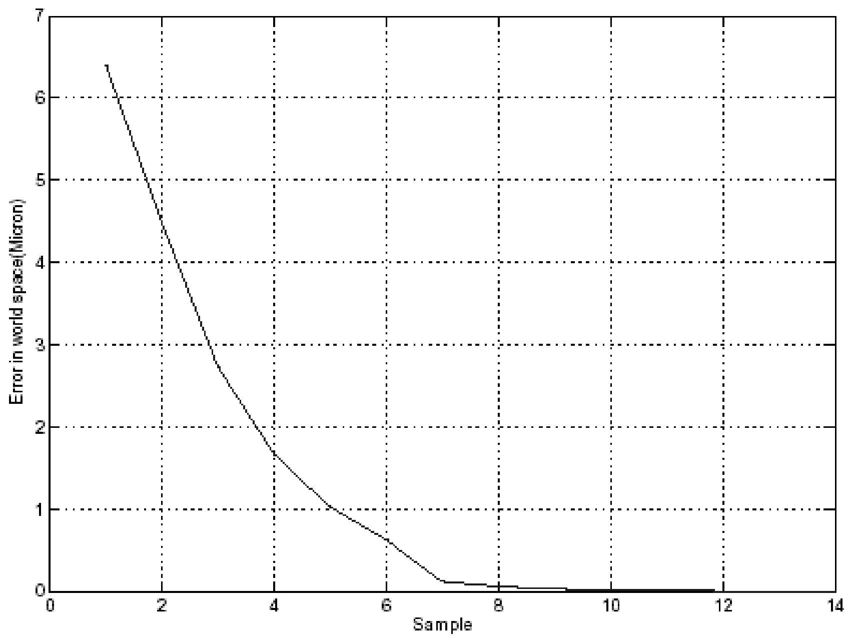


Fig. 3.11. The Simulation Results of the proposed MVMS Method w/o Image Noise

Fig. 3.11 shows the simulation process, the position reached a near neighborhood of the target very fast after the 7th step. This is achieved by coarse tracking with macro view image, but the transformational matrix is updated and regulated with information from micro view. The stage is moved to let the interested object approach the micro field of view, **Fig. 3.12** shows the tracking result.

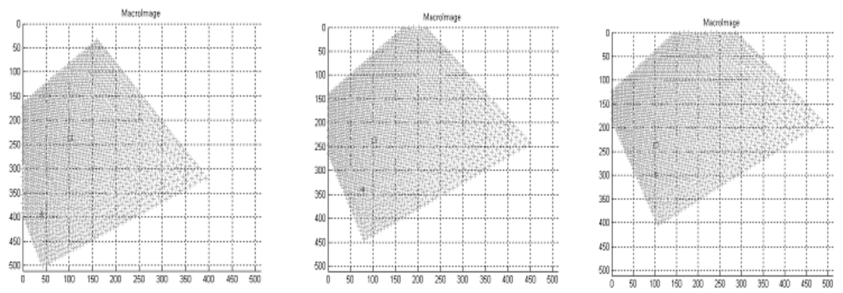


Fig. 3.12 Servoing Result with Macro Image Features

Then MVMS becomes slower for the fine tracking, when the interested object enters the micro field of view, see Image 1 in **Fig. 3.13**. During the fine tracking, visual servo is done by micro view image, but with constraint from macro view, which confines the tracking to be within the micro field of view. See **Fig. 3.13**. The circle is the predefined switching area, the red cross is the interested object.

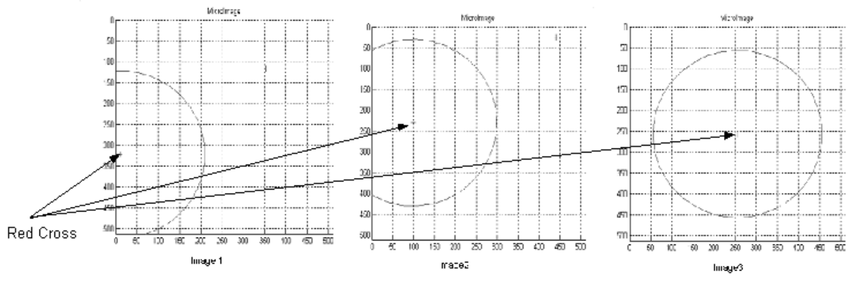


Fig. 3.13. Servoing Result with Micro Image Features

To quantify the sensitivity of the proposed method to noise, image noise of 0.1,0.2,0.3 standard deviation are added to the system in F**ig. 3.15** respectively, the sensitivity of vibration and other disturbances are compensated by testing the boundary condition in every iteration, while the multi view and multi scale scheme is still carried on to update the transforms. The sensitivity of image noise to the system is also shown.

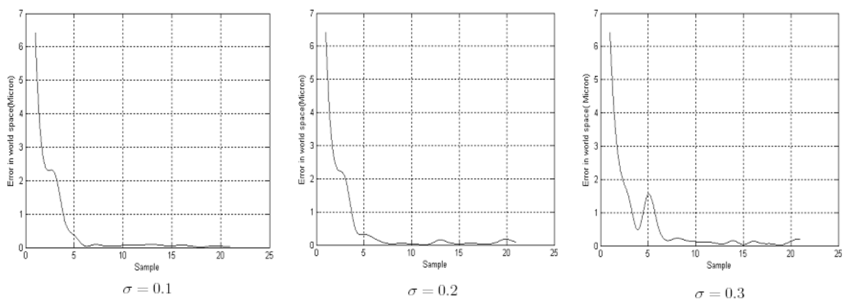


Fig. 3.14. The Simulation Results of MVMS Method

Testing sensitivity of the problems that we have highlighted relies on the features. It is important to detect the features robustly, find correspondence across views and track these features. ARGUS software can provide the robust feature detection with 0.4 pixels noise level even with occlusion, see **Fig. 3.15** and **Fig. 3.16**.

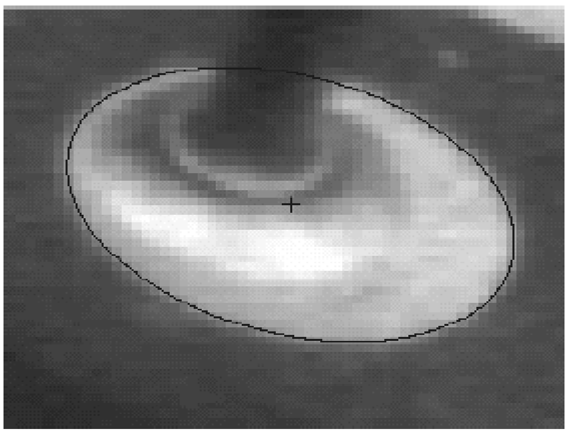


Fig. 3.15. Estimation of features (ellipse and centroid detection)

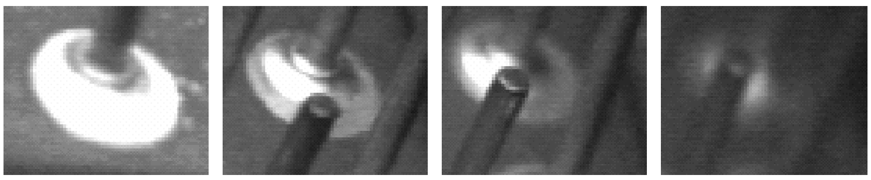


Fig. 3.16. A selection of close-ups illustrating degrading focus and contrast as well as occlusion

To estimate motion from images, the motion estimation performance from successive frames of images is tested. **Fig. 3.17** shows the comparative results among performance of optical flow (OF) and Markov Random Fields (MRF). The root-mean-square(RMS) error is compared.

From the results, the MRF method shows good estimation accuracy even for long displacement compared to the optical flow method. The experiment result shows that after 10 frames, the estimation error of optical flow increases sharply, that is because the optical flow based algorithms extract a dense velocity field from an image sequence assuming that image intensity is conserved during the displacement. This conservation law does not hold when the error of distance estimation between images accumulates, and the distance becomes large. Successive frames of the same scene are typically views of the same objects whose images are shifted in the frames due to the motion. By MRF we can assess that displacement despite very different local frame pixel values after large motion, since such frames are still similar. The statistical models of MRF characterize images, and allow computations of distances, yet are relatively insensitive to translation. In fact, MRF relates the spatial and temporal information together, to find the most likely displacement between image frames.

3.7 Conclusion

The experiments show the validity of the proposed method. By using both macro view and micro view at the same time, the proposed method appears to be good at solving some of the problems caused by uncertainty in micromanipulation system as demonstrated by the experiments. This method eliminates the burden to compute 3D position, but it keeps the position information in the transformational matrix, achieving precision without complex computation. The efficiency is achieved by coarse to fine tracking. The reliability is achieved by compensation between two sensors and updating the transforms among the two views and the work space. The results also shows that the complimentary nature between macro and micro views due to the consistency of motion structure is essential for enhancing precision without considering too much of the physics and kinematic uncertainty sources. The research work of reducing uncertainty with multiple view and multiple scale data based on human machine cooperation is still undergoing.

Acknowledgements

This work was carried out as part of the collaboration with Project ARGUS at SIMTech. The authors would like to thank Luo Hong for her helpful comments and discussions.

References

1. Sano T and Yamamoto H (2000) Study of micromanipulation using stereoscopic microscope, *Proceedings of 17th IEEE Technology Conference on Instrumentation and Measurement*, 2000, 3: 1227 –1231.
2. Yang G, Gaines J and Nelson B (2001) A flexible experimental workcell for efficient and reliable wafer-level 3d micro-assembly, *Proceedings of IEEE International Conference on Robotics and Automation(ICRA-2001),*1: 133-138.

3. Zhang H, Burdet E, Hutmacher D, Poo A.N, Bellouard, Y, Clavel R and Sidler T (2002) Robotic micro-assembly of scaffold/cell constructs with a shape memory alloy gripper, *Proceedings of IEEE International Conference on Robotics and Automation(ICRA-2002),*2: 1483-1488.
4. Breguet JM and Clavel R (1998) Stick and slip actuators: design, control, performances and applications, *Proceedings of the 1998 International Symposium on Micromechatronics and Human Science*(MHS –98), 89-95.
5. Fatikow S, Zollner J, Santa K, Zollner R and Haag A (1997) Flexible piezoelectric micromanipulation robots for a microassembly desktop station, *Proceedings of 8th International Conference on Advanced Robotics(ICAR –97),* 241-246.
6. Fearing R S (1995) Survey of sticking effects for micro parts handling, *Proceedings of 1995 IEEE/RSJ International Conference on Intelligent Robotics and Systems,* 2: 212-217.
7. Arai F, Andou D and Fukuda T (1995) Micro manipulation based on micro physics -strategy based on attractive force reduction and stress measurement, *Proceedings of IEEE/RSJ International Conference on Intelligent Robotics and Systems,* 2: 236-241.
8. Arai F, Andou D, Nonoda Y, Fukuda T, Iwata H, and Itoigawa K (1996) Micro endeffector with micro pyramids and integrated piezoresistive force sensor, *Proceedings of IEEE/RSJ International Conference on Intelligent Robotics and Systems,* 2: 842-849.
9. Zhou Q, Corral C, Esteban P, Albut A and Koivo H (2002) Environmental influences on microassembly, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems EPFL*, October 2002, 1760-1765.
10. Fukuda T and Arai F (1998) Micromanipulation and robotic technology, *Proceedings of the 1998 International Conference on Modeling and Simulation of Microsystems*, 11-16.
11. Requicha A A, Meltzer S, Arce F T, Makaliwe J, Siken H, Hsieh S, Lewis D, Koel B, and Thompson M (2001) Manipulation of nanoscale components with the afm:principles and applications, *Proceedings of IEEE International Conference On Nanotechnology*, *Maui, HI, October 28-30.*
12. Ge P and Jouaneh M (1997) Generalized presach model for hysteresis nonlinearity of piezoceramic actuators, *Precision Engineering*, 2: 99-111.
13. Celanovic M N (1997) Modeling piezoelectric stack actuators for control of micromanipulation*, Proceedings of IEEE Control Systems*, 3: 69-79.
14. Kawaji F A and Fukuda T (2001) 3d attitude control system for bio-micromanipulation, *Proceedings of International Symposium on Micromechatronics and Human Science (MHS-2001),* 197-202.
15. Arai F, Kawaji A, Luangjarmekom P, Fukuda T, and Itoigawa T (2001) Three dimensional bio-micromanipulation under the microscope, *Proceedings of IEEE International Conference on Robotics and Automation,*1: 604-609
16. Arai F, Sugiyama T, Luangjarmekorn P, Kawaji A, Fukuda T, Itoigawa K and Maeda A (2000) 3d viewpoint selection and bilateral control for bio-micromanipulation, *Proceedings of IEEE International Conference on Robotics and Automatio*n, 1: 947-952.

17. Johansson S (1993) Hybrid techniques in microrobotics, Proceedings of 1st IARP workshop on micro robotics and systems, June 1993.
18. Tanikawa T (1995) Two-finger micro hand, Proceedings of IEEE International Conference on Robotics and Automation, 1674-1679.
19. Fahlbusch S, Shirinov A and Fatikow S (2002) Afm-based micro force sensor and haptic interface for a nanohandling robot, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*,1772-1777, 2002.
20. Ando N, Szemes P, Korondi P, and Hashimoto H (2002) Friction compensation for 6dof cartesian coordinate haptic interface, *Proceedings of International Conference on Intelligent Robots and System IEEE/RSJ*, 3: 2893-2898.
21. Arai F, Kawaji A, Sugiyama T, Onomura Y, Ogawa W, Fukuda T, Iwata H and Itoigawa K (1998) 3d micromanipulation system under microscope, *Proceedings of International Symposium on Micromechatronics and Human Science (MHS-98),* 127-134.
22. Kawaji A, Arai F and Fukuda T (1999) Calibration for contact type of micromanipulation, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2: 715-720.
23. Li G Y and Xi N (2002) Calibration of a micromanipulation system, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System*, 2: 1742 -1747.
24. Mezouar Y and Allen P (2002) Visual servoed micropositioning for protein manipulation tasks, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System*, 2: 1766-1771.
25. Feddema J T and Christenson T R (1999) Parallel assembly of high aspect ratio microstructures, *Proceedings of SPIE Conference on Microrobotics and Micromanipulation,* September, 153-164.
26. Lee S, Kim K, Kim D, Park J O and Park G T (2001) Recognizing and tracking 3d-shaped micro parts using multiple visions for micromanipulation, *Proceedings of IEEE International Symposium on Micromechatronics and Human Science*, 203-210, 2001.
27. Le S J, Kim K, Kim D H, Park J O and Park G T (2002) Multiple magnification images based micropositioning for 3d micro assembly, *Proceedings of Seventh International Conference on Control, Automation, Robotics And Vision*, Dec 914-919.
28. Yamamoto H and San T(2002) Study of micromanipulation using stereoscopic microscope, *IEEE Transactions on Instrumentation and Measurement,* 51: 182 -187.
29. Yamamoto H and Sakiyama J (2002) Stereoscopic visual servo system for microinjection, *Proceedings of. 19th IEEE Instrumentation and Measurement Technology Conference (IMTC-2002),* 2: 1109-1112.
30. Yamamoto H and Sano S T (2001) Automatic microinjection system using stereoscopic microscope, Proceedings of International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 83-86.

31. Ralis S J, Vikramaditya B, and Nelson B J (2000) Micropositioning of a weakly calibrated microassembly system using coarse-to-fine visual servoing strategies, IEEE Transactions on Eletronics Packaging Manufacturing, 23(2), 123-131.
32. Sanderson A C and Weiss L E (1980) Image-Based Servo Control using relational graph error signals, *Proceedings of the IEEE International Conference on Robotics and Automation*, 1074-1077.
33. Flandin G, Chaumette F and Marchand E (2000) Eye-in-hand/eye-to-hand cooperation for visual servoing, *Proceedings of IEEE International Conference on Robotics and Automation*, April, 2741-2746.
34. Vikramaditya B, Micropositioning using active vision techniques (1997) Master's thesis, Mechanical Engineering, University of Illinois at Chicago.
35. Bors A G and Pitas I (2000) Prediction and tracking of moving objects in image sequences, *IEEE Transactions on Image Processing*, Aug, 9: 1441-1445.
36. Gern A, Moebus R and Franke U (2002) Vision-based lane recognition under adverse weather conditions using optical flow, *Proceedings of IEEE Intelligent Vehicle Symposium*, 2: 652-657.
37. Dang T, Hoffmann C, and Stiller C (2002) Fusing optical flow and stereo disparity for object tracking*, Proceedings of IEEE fifth International Conference on Intelligent Transportation Systems*, September, 112-117.
38. Li S Z (1995) Markov Random Field Modeling in Computer Vision. Springer-Verlag Tokyo.
39. Tao H, Sawhney H S and Kumar R (2000) Dynamic layer representation with applications to tracking, *Computer Vision and Pattern Recognition*, 2: 134-141.
40. Rittscher J, Kato J, Joga S, and Blake A (2000) A probabilistic background model for tracking, *Proceedings of European Conf. Computer Vision*, 336-350.
41. Kamijo S, Ikeuchi K and Sakauchi M (2001) Segmentations of spatio-temporal Images byspatio-temporal markov random field model, *LNCS 2134*: 298-313.
42. Tziritas G (1988) Displacement estimation for image predictive coding and frame motion-adaptive interpolation, *Proceedings of SPIE Visual Communications and Image Processing*, 936-941.
43. Larsen R (1993) Estimation of motion vector fields, *Proceedings of 2nd Danish Conference on Pattern Recognition and Image Analysis* (DANKOMB, DSAGM yearly meeting), Aug. 25-26, 37-42.
44. Scharstein D (1998) Stereo matching with nonlinear diffusion, *International Journal of Computer Vision*, 2: 155-174.

4 Path Planning in Dynamic Environments

Roman Smierzchalski[1] and Zbigniew Michalewicz[2]

1. Gdynia Maritime University, Faculty of Marine Electrical Engineering, Morska 83, 81-225 Gdynia, Poland, roms@wsm.gdynia.pl.

2. School of Computer Science, University of Adelaide, Adelaide, SA, 5005, Australia; Institute of Computer Science, Polish Academy of Sciences, Ordona 21, 01-237 Warsaw, Poland; and Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland, zbyszek@cs.adelaide.edu.au.

4.1 Introduction

The *motion planning* problem for mobile robots is typically formulated as follows: given a robot and a description of an environment, plan a path of the robot between two specified locations, which is collision-free and satisfies certain optimization criteria. Traditionally there are two approaches to the problem: off-line planning, which assumes perfectly known and stable environment, and on-line planning, which focuses on dealing with uncertainties when the robot traverses the environment. On-line planning is also referred to by many researchers as the navigation problem. Additional difficulty in approaching navigation problem is that some environments are dynamic, i.e., the obstacles which are present there, need not be static. In this chapter we consider a particular instance of a navigation problem, namely, a problem of computing a near-optimum trajectory of a ship. By taking into account certain boundaries of the maneuvering region, along with navigation obstacles and other moving ships, the problem of avoiding collisions at sea was reduced to a dynamic optimization task with static and dynamic constrains. The chapter presents a modified version of the Evolutionary Planner/Navigator algorithm, υEP/N++, to address the problem. The introduction of a time parameter, the variable speed of the ship, and time-varying constraints representing movable ships, are the main features of the new system. Sample results, having the form of ship trajectories obtained using the program for navigation situations are also presented.

The chapter is organized as follows. In Section 4.2, the definition of the ship navigation environment is provided. Some aspects of evolutionary algorithm used in our implementation are discussed in Section 4.3, while Section 4.4 presents an example of a modified on-line version of the evolutionary safe path search algorithm. Section 4.5 concludes this chapter.

4.2 Background

The *motion planning* problem for mobile robots is typically formulated as follows [19]: given a robot and a description of an environment, plan a path of the robot between two specified locations, which is collision-free and satisfies certain optimization criteria. Traditionally there are two approaches to the problem: off-line planning, which assumes perfectly known and stable environment, and on-line planning, which focuses on dealing with uncertainties when the robot traverses the environment. On-line planning is also referred to by many researchers as the *navigation* problem.

A great deal of research has been done in motion planning and navigation (see [19] and [6] for surveys). However, different existing methods encounter one or many of the following difficulties:

- high computation expenses,
- inflexibility in responding to changes in the environment,
- inflexibility in responding to different optimization goals,
- inflexibility in responding to uncertainties,
- inability to combine advantages of global planning and reactive planning.

In order to address these difficulties, we initiated the study of an Evolutionary Planner/Navigator (EP/N) system [7,18,17]; the inspiration to use evolutionary techniques was triggered by the following ideas/observations:

- randomized search can be the most effective in dealing with NP-hard problems and in escaping local minima,
- parallel search actions not only provide great speed but also provide ground for *interactions* among search actions to achieve even greater efficiency in optimization,
- intelligent behavior is the result of a collection of simple reactions to a complex world,

- a planner can be greatly simplified, much more efficient and flexible, and increase the quality of search, if search is not confined to be within a specific map structure,
- it is more meaningful to equip a planner with the flexibility of changing the optimization goals than the ability of finding the absolutely optimum solution for a single, particular goal.

The EP/N embodied the above ideas by incorporating some problem specific knowledge into evolutionary algorithm. With such an approach, the EP/N is pursuing all the advantages as described above. Less obvious though, is that with the unique design of chromosome structure and genetic operators, the EP/N does not need a discretized map for search, which is usually required by other planners. Instead, the EP/N "searches" the original and continuous environment by generating paths by various evolutionary operators. The objects in the environment can simply be indicated as a collection of straight-line "walls". This representation accommodates both known objects as well as partial information of unknown objects obtained from sensing. Thus, there is little difference between off-line planning and on-line navigation for the EP/N. In fact, the EP/N unifies off-line planning and on-line navigation with the same evolutionary algorithm and chromosome structure.

In this chapter we discuss a generalization of EP/N: its version, called υEP/N++, to address additional issues present in dynamic environments. The introduction of a time parameter, the variable speed of the ship, and time-varying constraints representing movable ships, are the main features of this version. The system was tested for particular environments: navigation of ships in collision situations.

When determining a safe trajectory for so-called *own ship*, we look for a trajectory that balances the cost of necessary deviation from a given route, or from the optimum route leading to a destination point, and the safety of passing all static and dynamic obstacles, called here *strange ships* (or *targets*). In this chapter the following terminology is used: the term *own ship* means the ship, for which the trajectory must be generated, and *strange ship* or *target* mean other ships in the environment, which must be avoided. All trajectories, which meet safety conditions (thus the risk of collision is reduced to a satisfactory degree) constitute a set of permissible trajectories. The safety conditions are, as a rule, defined by the operator, based on the speed ratio between the ships involved in the passing maneuver, as well as the actual visibility, weather conditions, navigation area, maneuverability of the ship, etc. The simplest way of determining the safe trajectory seems to use additional device – a decision supporting system, which would make an extension of the conventional Automatic Radar Plotting Aids (ARPA) system.

4.2.1 Previous Work

Dove et al. [2] presented a guidance concept for a ship entering a harbor, in which two autonomous systems (VTS and the system on board of the ship) were applied for evaluating the trajectory along given seaways, with simultaneous evaluation of the time correlation of subsequent positions of the ship, taking into account dynamic characteristics of the ship. The guidance principles were defined using an adaptation multi-dimensional optimum controller basing on the square quality factor, and taking into account both the minimization of the actual position and course deviation for a given time instant, and the minimization of the overall economic costs of guidance, represented by rudder positions and engine activity. In that paper, a non-linear discrete model of the ship was assumed. Process state variables were estimated using a linear Kalman filter. Dove et al. concept was developed further by Burns [1], who extended the guidance problem to a set of ships moving along a given voyage route. The proposed reactions in the collision situation were found with the theory of fuzzy sets. An autonomous ship guidance system was presented by Iijima and Hagiwara [5]. In order to evaluate the collision situation, make a decision, and give maneuvering orders, the authors developed the computer expert system. The system was tested on a training maneuvering ship in Tokyo Bay. During 1990-1994, a complex application prototype was worked out based on expert system technology applied for oceanic and coastal navigation. The integrated intelligent system consisted in a number of sub-systems which executed particular functions (e.g., optimum navigation, course planning, and automatic anchorage). The sub-systems were connected, via local area network, to the "Captain Expert"" – an expert system based on the knowledge and experience of navigators with long practice. The system was developed for guiding ships in oceanic an coastal navigational conditions in a fully automated manner, without crew interference, only being in touch with land-based services. A collision avoiding system restricted waters was developed by Hayashi et al. [4]. The system made use of an electronic map and the radar operation for evaluating the actual position and giving an assessment of the overall navigational situation. Sudhendar and Grabowski [16] discussed possible directions of further development, formulated requirements for an intelligent pilot system, and presented the actual state of work intended to meet particular requirements of coastal services in the United States and Canada. The inner structure of the expert system was discussed on the basis of a piloting system for the St. Lawrence Seaway in Canada. A detailed analysis of models and the synthesis of algorithms for safe, optimum steering were described in [13]. In their works the problem of determining a safe trajectory as a non-linear programming task was formulated, where a kinematics model of the own ship was applied. Another possible approach to this problem is the reduction of the solution space to a finite-dimensional one by creating so-called digitized matrix of permissible maneuvers for a given collision situation and a certain time instant [13]. In [9,10,11] the problem of avoiding collisions was formulated as the multi-criteria optimization task. Three separate criteria were used. The attempt to estimate the safe trajectory using classifier systems was presented in [3]. The collision situation was modeled as a fuzzy process with many inputs; for selecting the steering rules the authors made use of a fuzzy classifier system.

In the overwhelming majority of the reviewed publications on automatic ship guidance, the navigational process in the areas of intensive traffic was supported by expert systems. In this work, we report on experiments with evolutionary system, which takes into consideration the motion of other ships; see also [12,14,15].

4.2 Environment

The ship sails in an environment with some natural constraints (e.g., lands, canals, shallow waters) as well as other constraints resulting from formal regulations (e.g., traffic restricted zones, fairways, etc). These constraints are assumed stationary and are defined by polygons – in a similar manner to that used in creating electronic maps. When sailing in a stationary environment, the own ship meets other sailing strange ships/targets (some of which constitute a collision threat).

The degree of the threat of collision with dangerous targets is not constant and depends on the approach parameters: D_{CPA} (Distance at Closest Point of Approach) and T_{CPA} (Time of Closest Point of Approach), as well as on the speed ratio of both ships, and the distance and bearing of the target.

It is assumed that the dangerous target is each target that has appeared in the area of observation and can cross the estimated course of the own ship at a dangerous distance. Actual values if this distance depend on the assumed time horizon. The ranges of 5-8 nautical miles in front of the bow, and 2-4 nautical miles behind the stern of the ship are assumed. In the evolutionary task, the targets threatening with a collision are interpreted as moving dangerous areas having shapes and speeds corresponding to the targets determined by the ARPA system. The moving constraints represent approaching ships, and the shape of each constraint depends on the safety conditions: on an assumed value of the safe approach distance (D_{safe}), assumed safe distance, speed ratio, and bearing of the moving target. A safe distance is selected by the operator depending on the weather conditions, sailing area, and speed of the ship. When planning the safe trajectory, the evolutionary algorithm should take into account both the fixed constraints, and the areas of danger representing the moving targets, which dynamically change their locations [12,14]. Figures 4.1 and 4.2 display models of the environment where:

- fixed navigation constraints are modeled using convex and concave polygons,
- moving targets are modeled as moving hexagons,
- the dimensions of the own-ship are neglected due to small length of the own-ship with respect to the maximum length of the areas representing the moving targets.

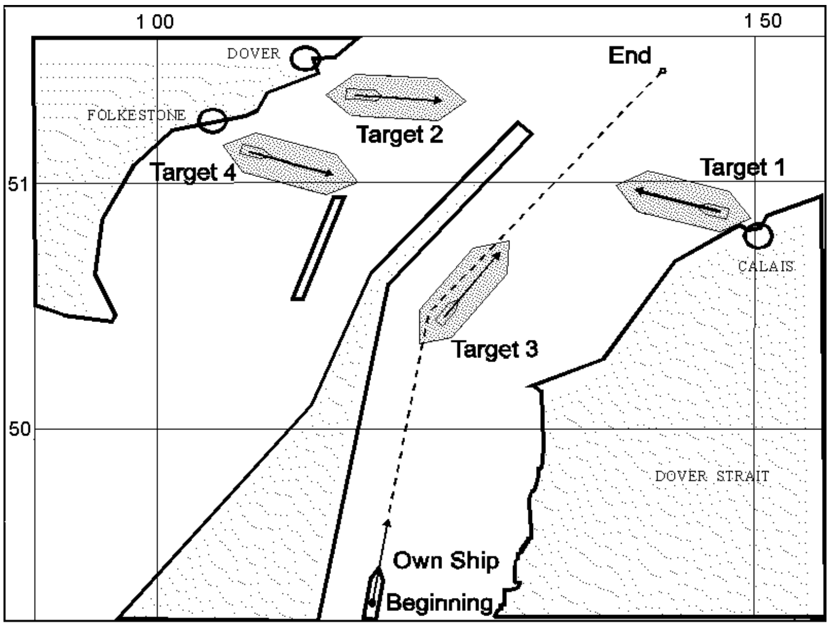


Fig. 4.1. Navigation situation in Dover Straits. There is an own-ship, four strange ships, and several navigational constraints

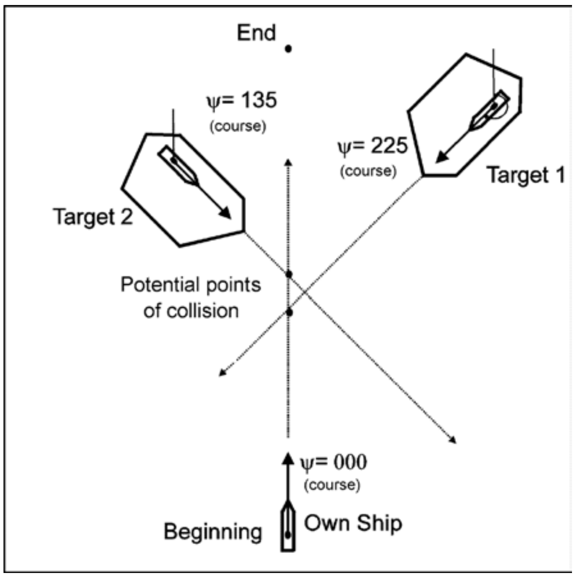


Fig. 4.2. Approaching two moving targets – hexagon constraint shapes

4.2.1 Planning the Trajectory in a Collision Situation

According to transport plans, the own ship should cover a given route R_0 in some assumed time. On the other hand, it has to move safely down a given trajectory, i.e., it must avoid navigation obstacles and cannot come too close to other moving targets. Estimation of a ship's trajectory in a collision situation represents a difficult trade-off between a necessary deviation from a given course and the safety of sailing. Hence it is a multi-criterion planning problem which takes into account the safety and economy of the ship motion.

The estimation of the own ship trajectory in the collision situation consists of determining a path, S, as the part of the given route R_0, from the present location (starting point) $(x_0, y_0) \in R_0$ to the actual end point $(x_e, y_e) \in R_0$. This path has the form of a sequence of elementary line segments s_i $(i = 1,...,n)$, linked with each other in turning points (x_i, y_i). The choice of the actual starting and end point depends on an assumed sensible horizon and is made by the operator. The boundaries of the environment are defined as

$$E = \{\boldsymbol{x} \in R^2 : a_i \leq x_i \leq b_i \ \textit{for} \ \ i=1,2\} \tag{1}$$

O_stat_j (j = 1,...,k) and $O\_dyn_j(t)$ ($j = k+1,...,l$) represent the sets of static and dynamic constraints, respectively. Note that each dynamic constraint, $O\_dyn_j(t)$ is time-dependent; i.e., it defines different sub-areas of E for different values of t. Clearly, static constraints represent time-independent constraints (e.g., lands, canals, restricted zones, etc), whereas dynamic constraints represent strange-ships.

The space $SF(t)$ of safe (anti-collision) paths is defined as

$$SF(t) = E - \bigcup_{j=1}^{k} O\_statj - \bigcup_{j=k+1}^{l} O\_dyn\,j(t) \tag{2}$$

In other words, a path S is safe (i.e., it belongs to the set of safe paths $SF(t)$ if any segment s_i (i = 1,...,n) of S stays within the limits of environment E, does not cross static constraints O_stat_j, and at the time instances t determined by the current locations of the own ship, does not come in contact with moving areas $O\_dyn_j(t)$ representing targets. Paths which cross the restricted areas generated by static and dynamic constrains are called unsafe, or dangerous paths.

The task of estimating the own-ship trajectory in a collision situation (so-called the *steering goal*) is performed as an evolutionary search for safe paths in the permissible space E, with subsequent selection of a near-optimum path S^* from the set SF with respect to the fitness function (defined by the path cost).

4.3 Evolutionary Algorithm uEP/N++

A crucial step in the development of an evolutionary trajectory planning system was made by the introduction of dynamic parameters: the time, and the moving constraints. In the evolutionary algorithm for trajectory planning eight genetic operators were used, which were: soft mutation, mutation, adding a gene, swapping gene locations, crossing, smoothing, deleting a gene, and individual repair [14]. The level of adaptation of the trajectory to the environment determines the total cost of trajectory. The trajectory costs include both the safety cost $Safe\_Cost(S)$ and that connected with the economy $Econ\_Cost(S)$ of the ship motion along the trajectory of concern. The total cost of the trajectory is defined as:

$$Total\_Cost(S) = Safe\_Cond(S) + Econ\_Cond(S) \tag{3}$$

The safety conditions are met when the trajectory does not cross fixed navigational constraints, nor moving areas of danger. The actual value of the safety cost function *Safe_Cost*(*S*) is evaluated as the maximum value defining the quality of the turning points s_i with respect to their distance from the constraints:

$$Safe\_Cond(S) = \quad wc * clear(S), \tag{4}$$

where: $clear(S) = \max_{i=1}^{n} c_i$, w_c is weight coefficient, c_i is the difference in length between the distance to the constraint-closest turning point s_i and the safe distance *d*. The trajectory cost connected with economic conditions *Econ_Cost*(*S*) includes: total length of trajectory *S* consisting of *n* line sections s_i, function of maximum turning angle between particular trajectory sections at turning points s_i time needed for covering the trajectory *S*. The total cost of the trajectory adaptation to the environment, resulting from the economic conditions, is equal to:

$$Econ\_Cond(S) = wd\ dist(S) + ws\ smooth(S) + wt\ time(S), \tag{5}$$

where: w_d, w_s, w_t are weight coefficients.

In many cases, the most effective maneuver for the own ship seems to be where the course changes and the speed is reduced. The speed reduction of the own ship can make it possible to pass a target without significant changes in its course. The analysis of examples where the speed of the own ship was initially assumed constant clearly shows the need for making this parameter variable along particular trajectory sections. In practice the speed is modified using an additional genetic operator: the speed mutation [13,15]. A set of permissible speed values was defined as υ = {3.6, 8.6, and 13.6 knots}; the mutation operator can select from this set an appropriate speed for any trajectory section under consideration. Those speeds correspond to the following telegraph settings: slow ahead, half ahead, and full ahead. Additionally, the total time of trajectory passing, *time*(*S*), was added to the function of the trajectory fitness, which took into consideration changes in the own ship's speed.

The next aspect of the evolutionary algorithm υEP/N++ is the on-line work, in which changes in parameters of motion of particular targets are taken into account. An interesting experiment was to check how the algorithm would determine the passing trajectory when one of the targets reduced its speed or its course.

The operation of the evolutionary trajectory planning algorithm system has been examined for a number of collision situations. Tests of the modification version of algorithm which changes the own ship's speed along the trajectory sections were discussed in previous works [13,14,15]. This chapter is focused on the discussion of experiments of on-line algorithm version. In all experiments reported here, the population size is 40, i.e., the evolutionary system processes 40 paths.

4.4 Simulation Studies

In our initial experiments the own-ship moves with a speed υ (along the safe path *S*) from the starting point (x_0, y_0) to the end point (x_e, y_e), and at the initial instant t_0, the motion of the strange-ships is defined as uniform. For each target, its motion is represented by the following parameters: bearing, distance, speed, and course, estimated by the ARPA system. The path of the own ship has the form of a sequence of elementary line segments s_i (i = 1,...,n), linked with each other in turning points (x_i, y_i).

These initial experiments considered the speed of the own ship constant; however, it is possible to gain additional efficiency while varying the speed. We return to this issue later in the chapter.

It is relatively easy to initialize the population of paths: each path (individual) can be generated randomly. Next, each path is evaluated. To determine whether a path is safe, the path is examined with respect to the set of static and dynamic constraints. The instantaneous locations of the dynamic areas with respect to the evaluated path depend on time t_c, determined by the first crossing point (x_c, y_c) between the own ship's path *S* and the trajectory of the target. For example, in the Figure 4.3, these crossing points are the points of the biggest collision threat for paths 1, 2, and 3. Having known the length of the line segment from the starting point (x_0, y_0) to the crossing point (x_c, y_c) and assuming that the own ship will keep moving with the uniform speed υ, it is possible to determine time t_c which the own ship needs in order to cover this distance.

After time t_c, the instantaneous location of the target with respect to the own ship is modeled as a dangerous area of hexagonal shape. Referring again to Figure 4.3 three locations of the target (at times t_1, t_2, and t_3) are given for three paths; note that the path segment of path 1 between the own ship and the intersection with the trajectory of the target is the longest one (i.e., longer than similar path segments of paths 2 and 3); consequently, t_1 is larger than t_2 and t_3, and the hexagonal shape of the target for t_1 is the leftmost one. Of course, as explained earlier, the detailed shape and dimensions of the hexagon depend on the safety conditions given by the operator.

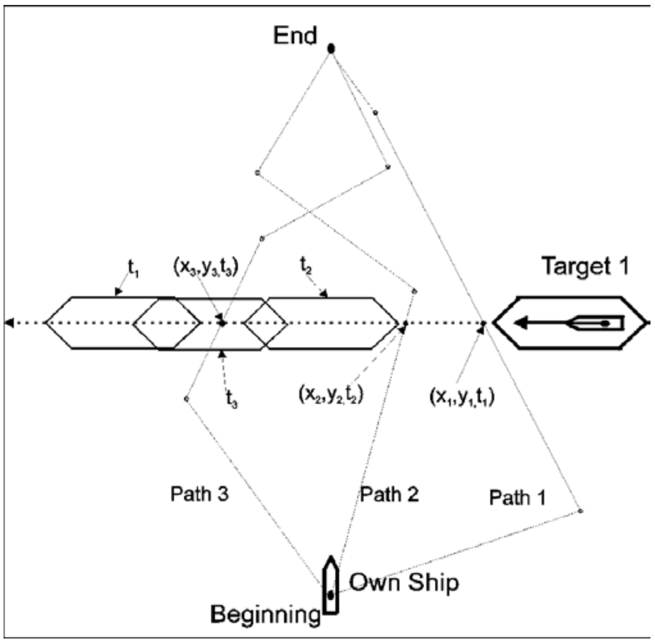


Fig. 4.3. Crossing paths and the corresponding dangerous areas

After the paths are evaluated, selected paths are modified by specialized set of operators (for details, see [18]).

The values assumed in here are the following:

- the distance in front of the bow which guarantees avoidance of the collision is equal to $3D_{safe}$ (in practice, safe distance D_{safe} is taken from the range between 0.5 and 3.0 nautical miles)
- the distance behind the stern is equal to D_{safe}
- the width of the dangerous area on each side of the own ship is chosen with the preference of the ship's passage behind the stern of the target, which depends on the course and bearing of the target.

4.4.1 Simple Example

Before we present the results of the evolutionary system on several test cases, we provide two simple examples, where the set of static constraints is empty and the dynamic constraints are defined by one or two strange-ships, respectively.

The first example (Figure 4.4) shows the situation when the own ship approaches a single target on its right side. As usual, time horizons for collision avoidance are around 30 minutes, we assumed $x = y = 8$ nautical miles. The population consisted of 10 individuals (paths), and the system converged after 300 generations. It is clear that the own ship, steering along the developed trajectory, will pass the target safely, passing it behind the stern. It is interesting to note that initially (see *Generation=50*) the own ship tried to move "left" (somewhat along the target), but clearly, much better maneuver is to go slightly right (as it is the case for *Generation=300*).

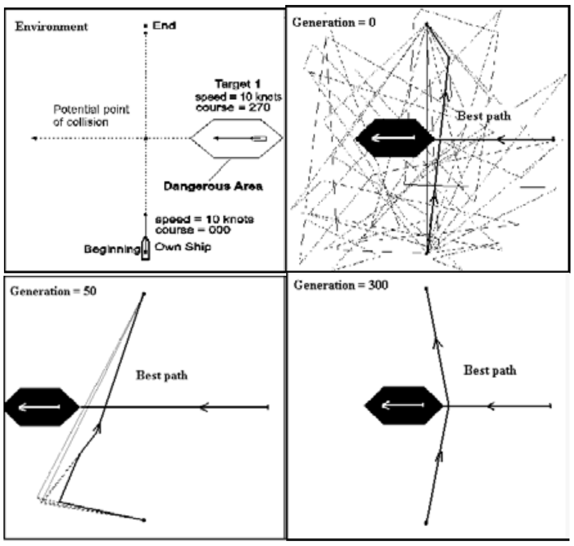


Fig. 4.4 Evolution of paths for the case of approaching one moving target

Note that in all three motion diagrams of Figure 4.4 (as well as in all further figures) the locations of the dynamic areas are shown (black hexagons) with respect to the best path, these locations depend on time determined by the first crossing point between the own ship's path and the trajectory of the target).

4.4.2 On-line Path Planning

In order to test the operational correctness of the on-line version of algorithm υEP/N++, certain trajectories were calculated using the off-line and on-line versions and then compared with each other. The test was divided into three phases. During the first phase, paths obtained in the off-line mode were tested. Then, during the next phase, the real motion of the own ship was studied; the ship was traveling along the trajectory assumed in the off-line mode. For the on-line version (phase number three), a quality assessment for the calculated trajectory was made on the basis of on-line calculations performed after changing the parameters of motion of one or more dynamic constraints – targets. The comparison was made for two sample environments with a relatively high level of complexity.

The first case presents the navigational situation in which the own ship passes around three islands and four moving targets from different directions and at different speeds. Input parameters for the simulation are shown in Figure 4.5. The speed of the own ship was defined as equal to 3.6, 8.6, or 13.6 knots. The progress of trajectory adaptations, made in the off-line mode for the case of the own ship meeting four moving objects in a collision situation, is shown in Figure 4.5, after 200, 500, and 1,000 generations, respectively.

During the computational process, after 1000 generations (computing time 12 seconds on a standard PC) no trajectory changes in the population were observed. Moving along the determined trajectory with changing speed, the own ship can pass the targets in front of their bows or behind their sterns, sailing between the islands. The ship speed is changed along subsequent trajectory sections. Initially, the ship reduces the speed to pass first two targets and then, after sailing between the islands it sails faster as it is not restricted by excessive approach to Target 3 and Target 4, at the same time making it possible to reach the final destination point in the shortest time. The execution of the proposed trajectory gives the optimum cost of trajectory passing with respect to the safety and economic criteria.

The next phase examines the real motion of the own ship traveling along the assumed (in off-line mode) trajectory and passing targets (Figure 4.6). During this phase it is possible to assess more accurately the correctness of positions of the passed targets with respect to the own ship. Figure 4.6 presents the navigational situation in the real motion at 10, 30, 40, and 50 minutes.

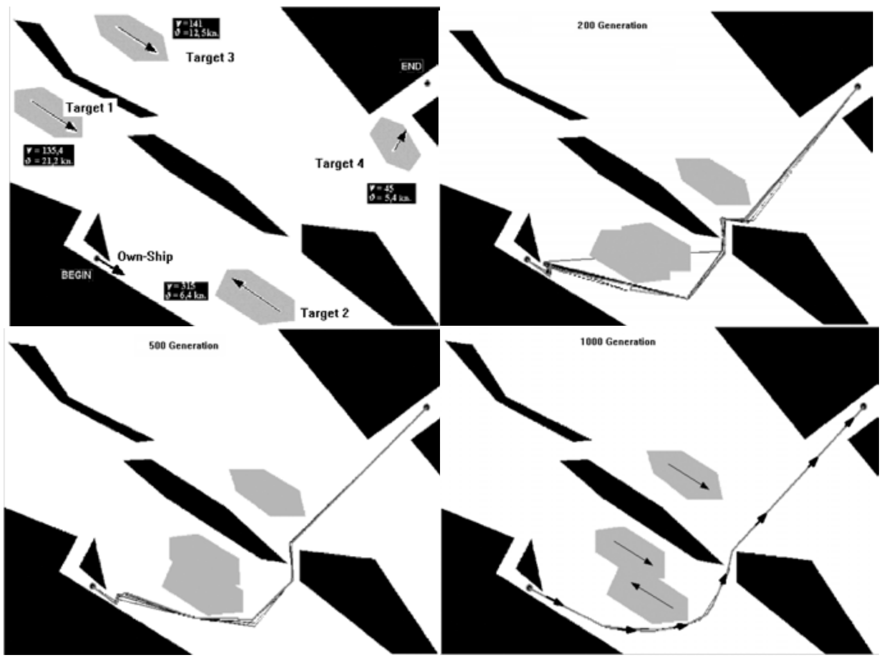


Fig. 4.5 Trajectory evolution in the off-line mode after 200, 500 and 1000 generations, respectively, for the case of approaching four moving targets in the presence of static navigation constraints (population 40 paths). The speed of own ship varies

In contrast to the relative presentation (Figure 4.5) the positions of the targets here (shaded areas representing dynamic constraints) and the position of the own ship on the trajectory are determined at times 10, 30, 40, and 50 minutes, which elapsed after the simulation has started. Additionally, the system simulates possible changes of parameters of motion (speed or/and course) of the targets. Changing parameters of the targets creates a new navigational situation – a new environment which triggers the adaptation of the own ship trajectory calculated in the on-line mode. Switching to the on-line mode, the system υEP/N++ adapts trajectory to the new environment.

For the test environment defined in Figure 4.5, the algorithm switched to the on-line mode at time of 20 minutes. Then selected environment parameters were changed, namely the course of the target seen in the right-hand part of the screen was changed from 45 degrees to 210 degrees, and the speed of motion of the target seen in the upper left part of the screen was changed from 5.2 to 17.6 knots. Figure 4.7 shows a newly calculated safe trajectory (after 1,000 generation) which has taken into account the above changes. After comparing Figures 4.5 and 4.7, it is clear that the path calculated in the on-line mode is similar to the trajectory obtained in the off-line mode, with certain differences resulting from trajectory corrections made due to environment changes introduced. This demonstrates the operational correctness of the on-line version.

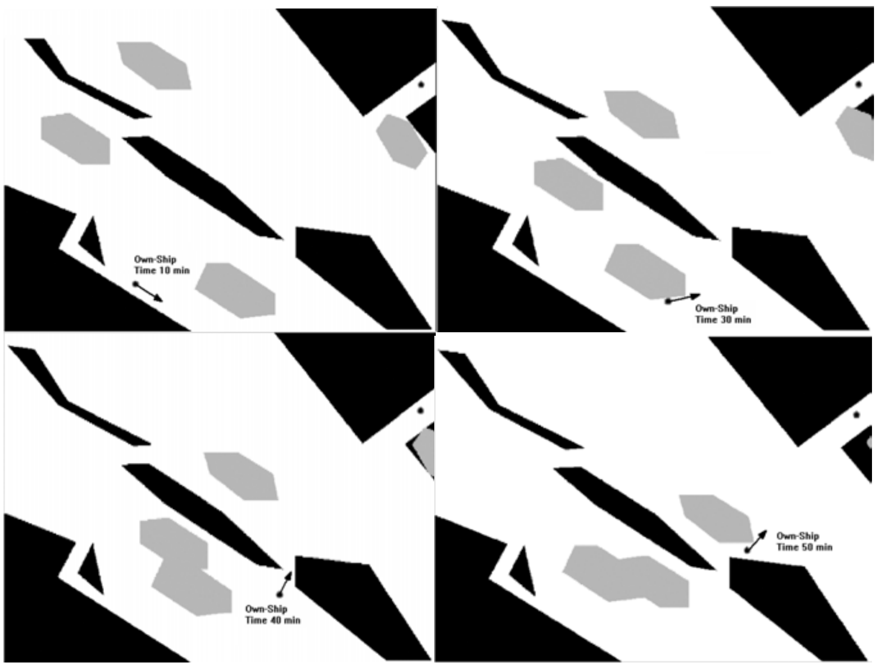


Fig. 4.6 Navigational situation in real motion at 10, 30, 40, and 50 minutes after start

The second case (Figure 4.7, left) presents a situation when the own ship leaves the channel and meets three targets moving from different directions and at different speeds. Input parameters for the simulation made for this environment are shown in Figure 4.8 (left). The speed of the own ship is equal to υ= 18.7 knots. The adaptation of the own ship trajectory calculated in the off-line mode (first phase) for collision meeting of three moving objects is shown in Figure 4.8 (right), after 1,000 generations.

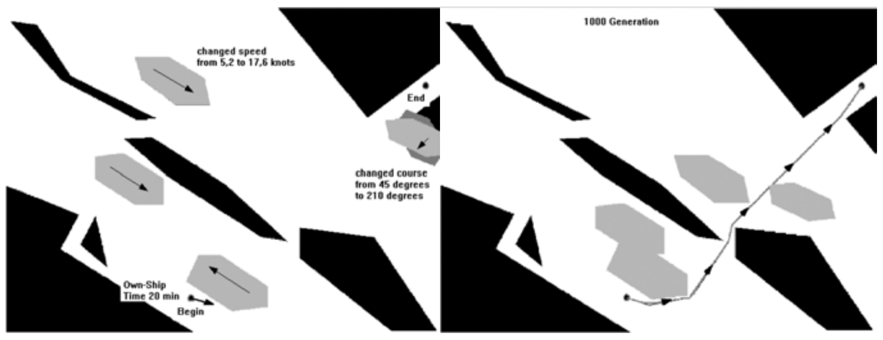


Fig. 4.7 Changing parameters of motion of two targets at time of 20 minutes (left) and newly calculated safe trajectory after 1,000 generations (right).

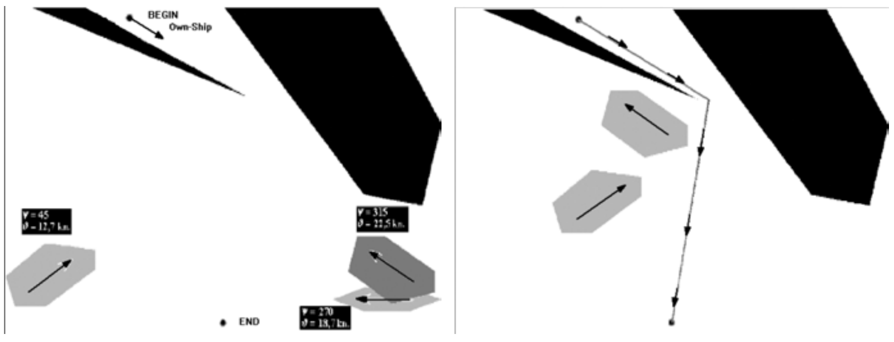


Fig. 4.8 Situation of meeting with the own ship and three targets (left). Trajectory evolution in the off-line mode after 1,000 generations (right)

Figure 4.9 refers to the second phase of the algorithm operation. In this phase the real motion of the own ship traveling along the assumed (in off-line mode) trajectory and passing strange targets is presented. For the tested environment and navigational situation, the positions of own ship and targets are displayed at times of 10 and 20 minutes after the start.
In order to examine the operation of the on-line mode in the test environment, at 20 minutes, selected parameters were changed in the motion of the target seen in the lower left-hand part of the screen. The speed was changed from 12.7 knots to 25 knots, and its course from 45 degrees to 103 degrees. The time of switching the system to the on-line mode is shown in Figure 4.10 (left). The newly calculated safe trajectory taking into account changes introduced to the motion of the target is shown in Figure 4.10 (right).

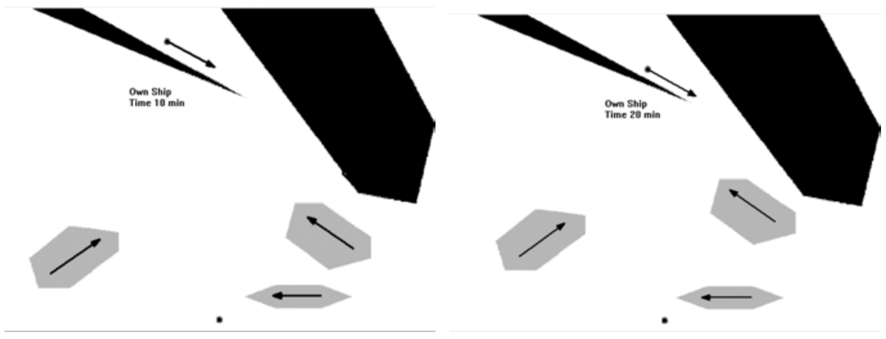


Fig. 4.9 Navigational situation in the real motion at times of 20 and 30 minutes after start

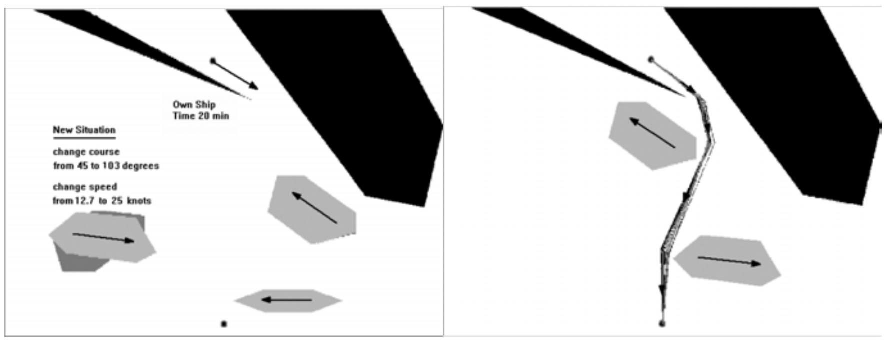


Fig. 4.10 Changing parameters of motion of one target at time of 20 minutes (left) and newly calculated safe trajectory after 1,000 generations (right)

For the test environment, the comparison of the solutions shown in Figure 4.8 (right) and Figure 4.10 (right) leads to the conclusion that the path calculated in the on-line mode is safe and close to optimum. This demonstrates the correctness of operation of the on-line program version. It should be stressed here that two most complex environments were selected for the present tests, out of all environments earlier studied by the authors (see [13,14,15]).

4.5 Conclusions

The evolutionary method of estimating the safe and optimum passing path being the own ship's trajectory in the environment with static and dynamic constraints is a new approach to the problem of avoiding collisions at sea.

A number of preliminary tests presented here make it possible to formulate the following conclusions:

- evolutionary algorithms can be effectively used for solving problem of avoiding collisions at sea when the environment is modeled as a set of polygons representing navigation constraints and moving targets,
- the task of evolutionary estimation of the own ship trajectory in a collision situation is reduced to an adaptive search for a set of safe paths *S* in a permissible space *X*, with subsequent selection of the optimum trajectory with respect to the fitness function,
- the strange ship (target) is modeled in the evolutionary environment as a dynamic constraint, a moving area of danger having a hexagonal shape. The detailed shape and dimensions of the hexagon depend on safety conditions and parameters of motion entered by the operator.

The introduction of additional elements (on-line mode) to the present program in order to include other environment changes does not impose any significant problems for the evolutionary path planning, and undoubtedly make the process more similar to real navigation situations. Each newly occurred situation can be in a natural way added to the operational diagram of the evolutionary algorithm.

Acknowledgements

The work was supported by the State Committee for Science Research in Poland (grant no. 3 T11A 003 26).

References

1. Burns R. S., An Intelligent Integrated Ship Guidance System. 2nd IFAC Workshop *Control Applications in Marine Systems*, pp. 199-208, Genova, Italy 1992.
2. Dove M. J., Burns R. S., Stockel C.T., An Automatic Collision Avoidance and Guidance System for Marine Vehicles in Confined Waters. *Journal of Navigation*, Vol. 39, p. 180,1986.
3. Furuhashi T., Nakaoka K., and Uchikawa Y., A Study on Classifier System for Finding Control Knowledge of Multi-Input Systems, (F. Herrera and J.L. Verdegay, Editors), *Genetic Algorithms and Soft Computing*, Physica-Verlang, 1996.

4. Hayashi, S., Kuwajima, S. Sotooka, K., Yamakazi H. and Murase H. A stranding avoidance system using radar image matching: development and experiment. *Journal of Navigation*, Vol. 44, p. 205, 1991.
5. Iijima Y. and Hayashi S. Study towards a twenty-first century intelligent ship. *Journal of Navigation*, Vol. 44, p. 184, 1991.
6. Latombe, J.C., *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
7. Lin, H.-S., Xiao, J., and Michalewicz, Z., Evolutionary Algorithm for Path Planning in Mobile Robot Robot Environment, In *Proc. First IEEE Conference on Evolutionary Computation*, Orlando, Florida, June 1994, pp. 211-216.
8. Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs. Spriger-Verlang, 3rd edition, 1996.
9. Smierzchalski R., The Application of the Dynamic Interactive Decision Analysis System to the Problem of Avoiding Collisions at the Sea, {*in Polish*} 1st Conference Awioniki, Jawor, Poland,1995.
10. Smierzchalski R., The Decision Support System to Design the Safe Maneuver Avoiding Collision at Sea. ISAS'96, Orlando, USA, 1996.
11. Smierzchalski R., Multi-Criterion Modeling the Collision Situation at Sea for Application in Decision Support, In *Proceedings of MMAR'96*, Miedzyzdroje, Poland, 1996.
12. Smierzchalski R., Trajectory planning for ship in collision situations at sea by evolutionary computation, In *Proceedings of the IFAC MCMC'97*, Brijuni, Croatia, 1997.
13. Smierzchalski R., Analysis and Synthesis of Navigator Decision Support Algorithms in Collision Situation at Sea, (*in Polish*), Academic Press, Gdynia Maritime University, Gdynia 1999.
14. Smierzchalski R. and Michalewicz, Z., Adaptive Modeling of a Ship Trajectory in Collision Situations at Sea, In *Proccedings of the 2nd IEEE World Congress on Computational Intelligence*, ICEC'98, Alaska, USA 1998, pp. 364--369.
15. Smierzchalski R. and Michalewicz, Z., Modeling of a Ship Trajectory in Collision Situations at Sea by Evolutionary Algorithm, *IEEE Transaction on Evolutionary Computation*, Vol.4, No.3, pp.227-241, 2000.
16. Sudhendar H., Grabowski M., Evolution of Intelligent Shipboard Piloting Systems: A Distributed System for the St Lawrence Seaway. *Journal of Navigation*, Vol. 49, No.3, 1996.
17. Xiao, J. and Michalewicz, Z., An Evolutionary Computation Approach to Planning and Navigation, chapter in Soft-Computing and Mecha *tronics*}, K. Hirota and T. Fukuda (Editors), Physica-Verlag, 1999.
18. Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K., Adaptive Evolutionary Planner/Navigator for Mobile Robots, *IEEE Transactions on Evolutionary Computation*, Vol.1, No.1, 1997, pp.18-28.
19. Yap, C.-K., Algorithmic Motion Planning, In Advances in Robotics, Vol.1: Algorithmic and Geometric Aspects of Robotics}, J.T. Schwartz and C.-K. Yap Ed., pp.95-143, Lawrence Erlbaum Associates, 1987.

5 Intelligent Neurofuzzy Control of a Robotic Gripper

J.A. Domínguez-López, R.I. Damper, R.M. Crowder and C.J. Harris

Department of Electronics and Computer Science, University of Southampton, Southampton, UK.
{jadl00r|rid|rmc|cjh}@ecs.soton.ac.uk.

5.1 Introduction

One of the major challenges of robotics is the grasping and manipulating of objects in an unstructured environment, in particular where the physical properties of the object are not known *a priori*. The resultant uncertainty makes it difficult to control contact forces, and the relative position between the object and the gripper's point of contact. As part of the grasping process, force control is required. This will avoid the risk of the object slipping out of the end effector as well as any possible damage to the object. The means of defining the required grasp force is crucial and can be posed as an optimisation problem, [1].

Various techniques have been applied to solve this problem [2-4] Some approaches are analytic and cannot be easily implemented in real-time applications, when dynamic adaptation to external disturbances is an important requirement. Also, the analytic approach cannot be used if variables such as the object's weight and end effector acceleration are unknown. To overcome this situation, other approaches using fuzzy controllers have been developed using a number of different sensors to measure the physical variables which provide feedback to the system. Fuzzy systems have a number of advantages over traditional techniques that make them an attractive approach to solve this type of problem. Some of these general advantages are their ability to model complex and/or non-linear problems, to mimic human decisions handling vague concepts, rapid computation due to intrinsic parallel processing, capacity to deal with imprecise information, improved knowledge representation and better uncertain reasoning than traditional techniques. However, fuzzy systems have also several disadvantages including mathematical opaqueness, they are highly abstract and heuristic, and they need an expert (or operator) for rule discovery. They are unable to adapt by themselves in response to changes in process parameters because in a purely fuzzy system the parameters do not appear in an analytical form so they cannot be easily modified for learning and tuning the fuzzy rules, [5]. Nevertheless, fuzzy control can be used in conjunction with powerful automatic learning methods (i.e., neural networks) as a neurofuzzy system [6-8].

In this chapter, we explore these issues using a very simple two-fingered gripper as an experimental system. Work is conducted both using a real gripper and, because of the high degree of flexibility it affords, in software simulation. The structure of the chapter is as follows. In Section 5.2, we describe these two experimental systems. Section 5.3 then outlines aspects of neurofuzzy systems as they impact on this study. Since machine learning methods to allow such systems to adapt to their environment are a major concern, these are discussed in some detail in Section 5.4. Results of applying our methods to the design of an adaptive neurofuzzy controller for the real gripper are described in Section 5.5. We then turn to the simulation of the gripper mounted on a six degree of freedom robot in Section 5.6, before concluding in Section 5.7.

5.2 Experimental Systems

The work reported in this chapter has used two different experimental systems: a simple, low-cost two-fingered gripper and a simulation system written by the first author. Ideally, we would have liked to do all our work with real robotic systems but this is inflexible and expensive in terms of hardware. Hence, the simple system was used to prove our ideas which were then further explored in simulation.

5.2.1 Two Fingered Gripper

The simple, low-cost, two-finger gripper is shown in Figure 5.1; which has one degree of freedom: the fingers can either open or close. It is fitted with a slip sensor [9] and force sensors. The slip sensor is located on one finger and is based on a rolling contact principle. Slip induces rotation of a stainless steel roller on a spring mounting, which is sensed by an optical shaft encoder. The slip sensor has an operational range of 0 to 80 mm s^{-1} and sensitivity of 0.5 mm s^{-1}. The applied force is measured using a strain gauge bridge on the other finger. The force sensors have a range of 0 to 2500 mN with a resolution of 2 mN. Control of the end effector was achieved using a personal computer (Pentium 75 MHz, 64 MB RAM) fitted with a high-speed analogue input/output card (Eagle Technologies PC30GAS4). Control software ran under the MS-DOS operating system. The sampling period for the system was set conservatively to 17 ms to allow adequate time for all processing to be complete between consecutive samples.

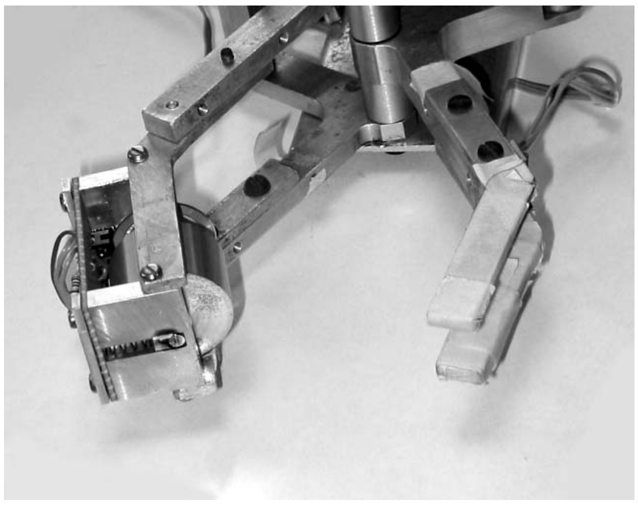


Fig. 5.1 The experimental, two-fingered gripper. The slip sensor appears on the left of the picture, attached to one of the fingers. The force sensors (strain gauges) are attached approximately midway along the finger shown on the right of the picture

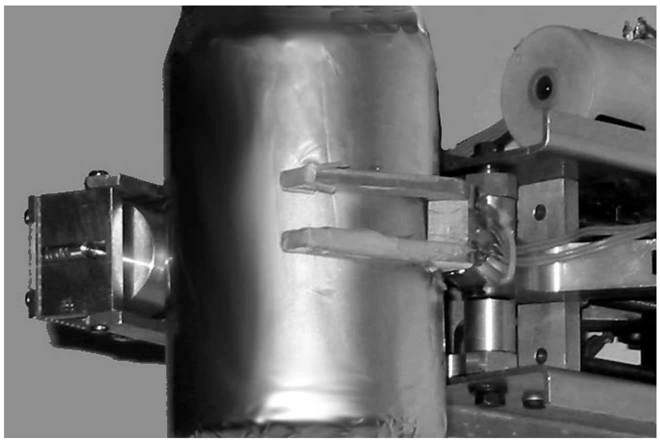


Fig. 5.2 A metal can being gripped by the experimental system

Figure 5.2 shows metal can being gripped by the experimental system. The roller of the slip sensor is clearly visible, together with its spring mounting which prevents the sensor from seizing during gripping. This metal can was found to be an effective vehicle for our experiments, as the total weight of the gripped object could be conveniently manipulated by placing various weights in it (see Section 5.5).

5.2.2 Simulation

In this part of the work, a simulated gripper was subjected to a range of identical load disturbances. The gripper was the simple two-fingered, single degree of freedom system with slip and force sensors, as described above. Full details of the simulation's kinematics equations can be found in Appendix A of [10]. The simulation was validated against results of the real gripper handling a range of weights.

The gripper was simulated holding a 0.1kg object. An external transient force of 10 N was applied to the load 3 seconds into the simulation, and a second force of -10 N was applied at 5 seconds. Between 6 and 10 seconds, the gripper was moved, thereby applying acceleration forces along its z-axis.

5.3 Neurofuzzy Systems

Neurofuzzy systems combine the advantages of fuzzy systems, such as transparent representation of knowledge, and those of neural networks, which deal with implicit knowledge that can be acquired by means of learning [8, 11, 12]. They mimic human decision processes, in that they manage imprecise, partial, vague or imperfect information. Also, they are able to resolve conflicts by collaboration and aggregation. Moreover, they have self-learning, self-organising and self-tuning capabilities, with no requirement for prior knowledge of relationships of data although they can use this if it is available. They have fast computation using fuzzy number operations. As do fuzzy systems, neurofuzzy systems embody rules (defined in a linguistic way) so that system operation can be interpreted in a transparent fashion.

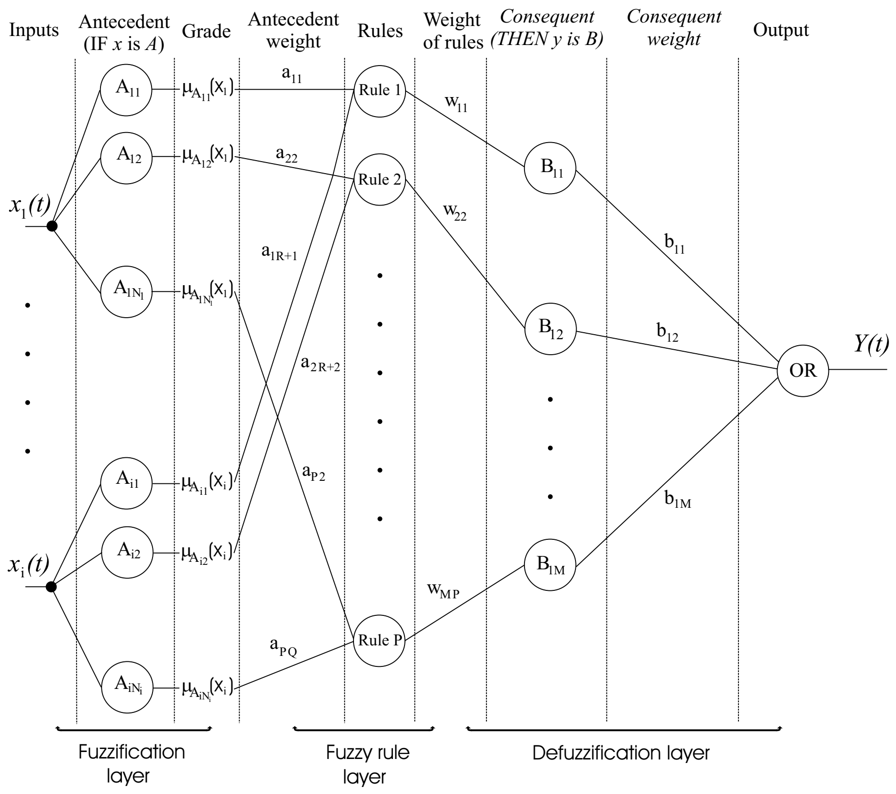


Fig. 5.3 Architecture of a typical neurofuzzy system. The number of rules is given by $P = \prod_i N_i$ where i is the number of inputs and N_i is the number of antecedent fuzzy sets for input i while M is the number of consequent fuzzy sets. In addition, $Q = \sum_i N_i$ and $R = \sum_{k=0}^{i-1} N_k$

Figure 5.3 shows the implementation of a neurofuzzy system as a feed forward network with three layers, namely: the fuzzification layer, the fuzzy rule layer and defuzzification layer. The mapping between the first two layers is nonlinear and linear between the last two layers [13]. The fuzzification layer consists of two components: the measurements x_i, which are the input signals from the system under control and the environment, and the computation of the antecedent membership function value, $\mu_{A_{ij}}(x_i)$ termed the grade of membership. The fuzzy rule layer then calculates the rule firing strength, which indicates how well the conditions in the antecedent are satisfied. An input x *fires* a rule to a degree, $w \in [0,1]$. The calculation of the rule firing strength is performed using the membership function values for fuzzy sets used in the rule antecedent.

It is dependent on how the antecedents are combined (i.e., using AND, OR). The product combiner is the fuzzy implication method, used because it preserves the original shape of the fuzzy set and it is differentiable. Finally, the defuzzification layer converts the values delivered by the fired rules at the previous layer into analogue output values. Centre of sums (CoS) and centre of gravity (CoG) are the most often used defuzzification techniques. The CoS approach has some advantages over CoG; in particular, CoS is faster because it is simpler to program [14]. The resulting defuzzified values are combined by some averaging method (denoted OR in Figure 5.3).

The antecedent and consequent weights (a_{ki} and b_{mn} respectively), and the rule confidences (w_{ij}) can be updated. In fact, there are two main possibilities for learning: adjust these weights, or adjust the shape of the membership functions. The latter can be achieved only if the membership functions are parametric. In this work, we fix all antecedent and consequent values, and membership functions; only rule confidences are learned.

5.4 Training Methods

A neurofuzzy controller is able to learn about its environment through a process of adjusting its weights and bias levels. The learning paradigms used to tune the network weights are commonly divided into three main classes: supervised, unsupervised and reinforcement [15]. These learning algorithms can be used individually or in conjunction[16, 17]. In supervised learning, a *teacher* gives the system a representative collection of input-output examples constituting knowledge of the problem environment. For unsupervised learning, no such specification is made of which output should be associated with particular inputs. Rather, the system bases its free-parameter updating on some quality measure of the representation of the learning goal. Reinforcement learning (RL) is a broad class of machine learning techniques in which the ‘teacher’ is replaced by an evaluative signal (or signals) derived from the environment. Hence, it is eminently suitable for on-line robot learning applications in which continuous interaction with the environment is all-important. Furthermore, RL is fully automatic, doing away with the need for intervention of an expert (the ‘teacher’) to provide a suitable training dataset. The evaluative (or reinforcement) signals do not specify what the correct answer should be, since this is unknown. Rather, they specify whether the system output is right or wrong, as well as (possibly) providing a scalar indication of the degree of correctness. This evaluative signal is often couched in terms of a reward or a punishment, for being right or wrong respectively. Because there is no explicit provision of a correct answer by a 'teacher', we take RL to be distinct from supervised learning—although some researchers do treat it as supervised "because the network *does*, after all, get some feedback from the environment" [18, p188].

In our work, we have used both supervised and reinforcement learning to produce neurofuzzy controllers for the gripper. We have also used a hybrid of the two. Hence, we have a total of three learning systems: off-line supervised learning, reinforcement on-line learning and a hybrid of supervised and reinforcement on-line learning.

5.4.1 Supervised Learning

In supervised learning, knowledge of the problem environment is represented by a set of input-output examples—the training dataset. The task of the learning algorithm is to adjust the system parameters in response to the given inputs so as to reproduce the desired output [15, p63]. To achieve high system performance, the training data must be consistent and complete. This is an obvious shortcoming of the supervised learning approach for our problem. In the real environment, we will not be able to anticipate all conditions which will be met, so it is not possible to guarantee completeness.

To collect the training data, a simple C program was written to control the gripper manually using several keys of the PC keyboard. Six keys were used for a fixed applied motor voltage (i.e., -5, 0, 1.5, 2.7, 3.9, 5V) and four to increase/decrease the applied voltage by 0.1V or 0.25V. The operator (the first author), using personnel judgment and the current object status (e.g., gripper or not, crushed, etc.), manually increased/decreased the applied voltage so the gripper could grasp the object properly (i.e. fast, stable, with minimum finger force and without allowing the object to fall). An extra (stop) key was included to allow the operator to pause and rethink the action to be taken to be taken as well as to check the object status. The data collected during the stopped time was not used for training because they are not actually part of the control action.

Several trials were carried out to obtain several collections of training data. Every trial started with the gripper fingers completely opened. The object used was an empty can with uneven surface and variable weight. The weight was varied using cans with different mass (i.e., 50, 125 and 210 grams). During the manual control, several disturbances of different magnitude were applied on the object to induce slip. Each experiment was finished once the gripper was gripping the object properly. In total, 15 collections were selected to be used as training data because very good manual control was achieved. Some other collections were eliminated because the operator failed to control the gripper properly (i.e., the object fell or was crushed).

The parameters of the network are updated under the combined influence of the available training examples, and the error signal, $e_y(n)$ which is defined as the difference between the desired response, $\hat{y}(n)$ and the actual response, *y(n)* at iteration *n*,

$$e_j(n) = \hat{y}_j(n) - y_j(n)$$

This error signal, $e_j(n)$ is used to apply a sequence of corrective adjustments to the weights. This sequence is carried out until a stopping criterion is met. The commonest and best-understood approach for off-line supervised learning is the gradient descent method: back-propagation [19]. Also, its computational cost is lower than the other methods (i.e., Newton, conjugate gradient) which leads to a faster convergence[20]. The weight correction, $\Delta w_{ji}(n)$ applied to the weights connecting neuron *i* to neuron *j* is defined by the delta rule:

$$\Delta w_{ji}(n) = \eta \delta j(n) x_j(n) + m \Delta w_{ji}(n-1)$$

where η is the learning-rate parameter, $\delta_j(n)$ is the local gradient, *m* is the momentum constant, and $x_j(n)$ the input signal of neuron j in the defuzzification layer. In this work, η and m are both set to 0.5. Network training is considered converged when the average squared error (between desired and actual output voltages) is less than or equal to 0.2V^2.

5.4.2 Reinforcement Learning

Reinforcement learning (RL) is the natural framework for the solution of the on-line learning problem [21]. It encompasses a broad range of techniques with the common feature that a goal-oriented system adapts its on-line behaviour in such a way as to maximise some 'reward' signal derived from its environment. Because the reward reflects the system's interaction with its environment, learning can be unsupervised, without manual intervention to indicate correct versus erroneous behaviour. Negative rewards, i.e., 'punishments', can also be conveniently incorporated into the paradigm. In view of its generality and on-line nature, RL has found wide application in robotics and autonomous system studies (e.g., [22-24]).

In many formulations of RL, the system is supposed to decide the best action to select based on its current state. When this step is repeated, the problem is known as a Markov decision process [25]. A finite Markov decision process (MDP) is defined by the state and action sets, and by the one-step dynamics of the environment. The components of an MDP are:

- a set of states *S*;
- a set of actions *A*;
- a reinforcement function $R : S \times A \rightarrow \Re$; and
- a state transition function $T : S \times A \rightarrow \prod(S)$ where $\prod(S)$ is a probability distribution over the set *S*.

At each decision point, the controller has to select an action based on the environmental information (i.e., state). After performing the action a_i the system receives an immediate reinforcement, r_i. The action taken affects the subsequent state, s_{i+1}. The probability distribution of the reinforcement r_t and the subsequent state s_{i+1}.depends only on the starting state s_{i+1} and the taken action a_i. The objective of the system is to maximise the sum of discounted rewards [26, 27]. The reinforcement at time *t* is given by:

$$r_i = \sum_{i=0}^{\infty} \gamma^i r_{i+1}$$

where $0 \le \gamma \le 1$ is a discount factor that determines the relative contributions of delayed rewards. If $\gamma = 0$ only the immediate reward is considered. Long-running systems that learn from delayed rewards generally achieve the best state-action [28].

The dynamics of a finite MDP can be summarised in a high-level form, namely an MDP transition graph, which has to be designed ‘by hand’ to suit the system’s intended purposes. Figure 5.4 illustrates the transition graph for the gripper system. For our problem, the state set is defined as:

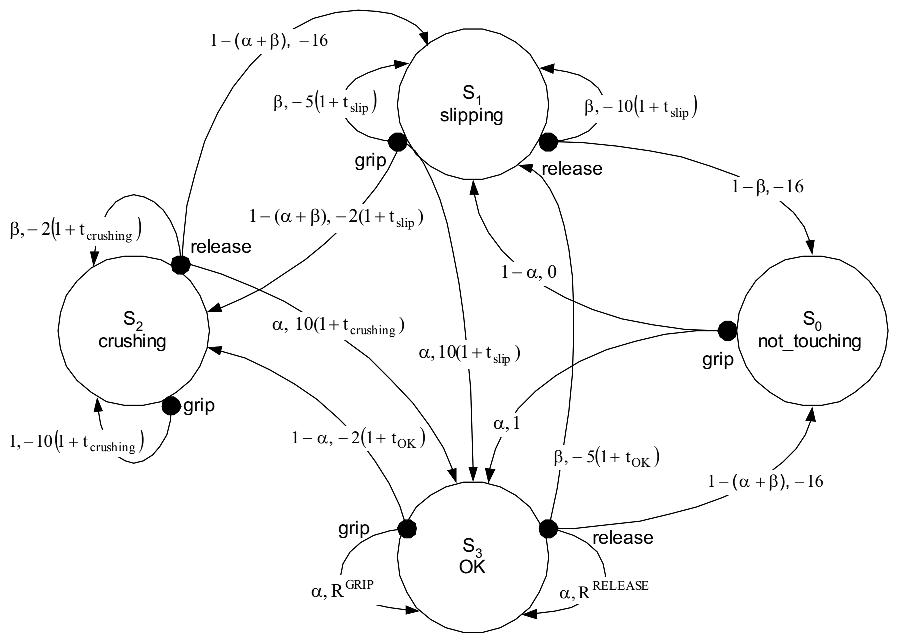


Fig. 5.4 Transition graph specifying the dynamics of the finite MDP. The large open circles denote state nodes and the small filled circles denote action nodes. Arcs are labelled with an ordered pair consisting of a transition probability and a reward. Punishments are implemented as negative rewards. In this work, we simplify the MDP by making all transition probabilities functions of just two parameters, α and β

$$S = \{S_0, S_1, S_2, S_3\}$$

$$= \{\texttt{not\_touching, slipping, crushing, OK}\}$$

and the system decisions (actions) are given by the action set:

A = {grip, release}

There is a state node (a large open circle labelled by the name of the state) for each possible state, and an action node (a small solid circle labelled by the action name) for each action associated with those states. Each arc is labelled with an ordered pair consisting of the transition probability of moving from state S to state S' with associated action A, and the expected reward for that transition, $R_{SS'}$.

Two transition-probability parameters, α and β, are used. Note that at least 2 parameters are required because the maximum outdegree of an action node is 3, and 1 degree of freedom is fixed since the transition probabilities must sum to 1. Accordingly, α is set relatively high, as it denotes a transition probability into a desired state (i.e., `OK`), while β is set relatively low. In many cases, but not all, β indicates a transition probability into a undesirable state (i.e., `slipping` or `crushing`). In some cases, the transition probability into these states will be $1-(\alpha+\beta)$, so a further requirement is that $\alpha+\beta<1$ to ensure that these transition probabilities do not vanish. In this work, α and β have been set by trial and error to 0.85 and 0.12, respectively.

Rewards and punishments are either fixed, or made to depend upon the time spent in the originating state for that transition. This is to increase the rewards and punishments if the system keeps in a *good* or *bad* state-action, respectively. Rewards are attached to transitions into the `OK` state. Punishments are attached to transitions into the undesirable (`slipping`, `crushing` and `not_touching`) states. Time-dependent rewards have the effect of increasing the reward for moving to the OK state according to the length of time spent in an undesirable state, and conversely for punishments. Both time-dependent and fixed rewards/punishments have been set empirically. Transitions with a -16 punishment are included in the MDP specification for completeness; it is envisaged that they would never occur in the trained network. When the object is being satisfactorily gripped, i.e., the system remains in the state `OK` for t_{OK}, the rewards for the two associated actions grip and release are:

$$R_{S_3S_3}^{GRIP}=R_{S_3S_3}^{RELEASE}=c\frac{2500-F}{2500}(1+t_{OK})$$

where t_{OK} is the time without both slipping and crushing, F is the applied force and c is a constant empirically set equal to 20 in this work. With this scheme, the rate of increase of the reward decreases as the applied force approaches its upper limit of 2500, corresponding physically to 2500 mN.

There are three fundamental classes of methods that tackle the reinforcement learning problem: dynamic programming (DP), Monte Carlo methods and temporal difference (TD) learning. TD methods are based on learning the difference(s) between temporally-successive predictions. As TD learning combines characteristics of dynamic programming (i.e., policy evaluation, value iteration) and Monte Carlo methods, it can approximate the value function using update estimates from other learned estimates, like those found from dynamic programming. Also, like Monte Carlo methods, TD methods can learn from experience following a policy without the dynamics of the environment. Accordingly, TD has some advantages over the other methods [21]. The most important ones are: TD methods do not require a model of the environment, rewards and next-state probability distribution while DP methods do; TD methods can be implemented in an on-line fashion while Monte Carlo methods can not.

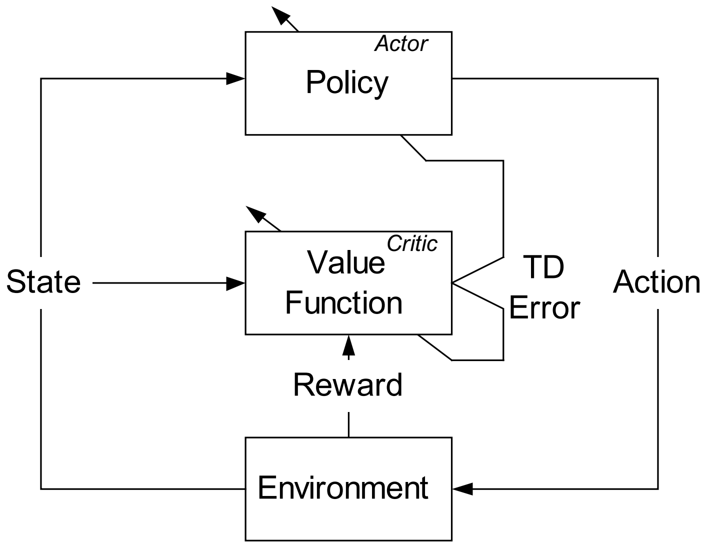


Fig. 5.5 Block diagram of the actor-critic reinforcement learning system

In this work, we have used an *actor-critic* framework. Actor-critic methods are temporal difference learning methods [29]. To solve the prediction problem, TD methods compute the state-value function using the experience of following a policy (i.e., the mapping between states and actions). At the next time step, based on the obtained reward or punishment, the estimated value function for a given policy is updated. Actor-critic methods have two separate memories in order to represent the policy independently of the value function. This leads to the following most important advantages of actor-critic methods over other TD techniques: minimal computation required to select actions and its ability to learn an explicitly stochastic policy, allowing the optimal probability of selecting appropriate actions to be learned [21, p153]. Figure 5.5 shows the architecture of the actor-critic method. The policy structure is the *actor* and the estimated value function is the *critic*. The former selects actions and the latter criticises those actions. This criticism has the form of an ‘internal’ (TD-error) signal to drive the learning process.

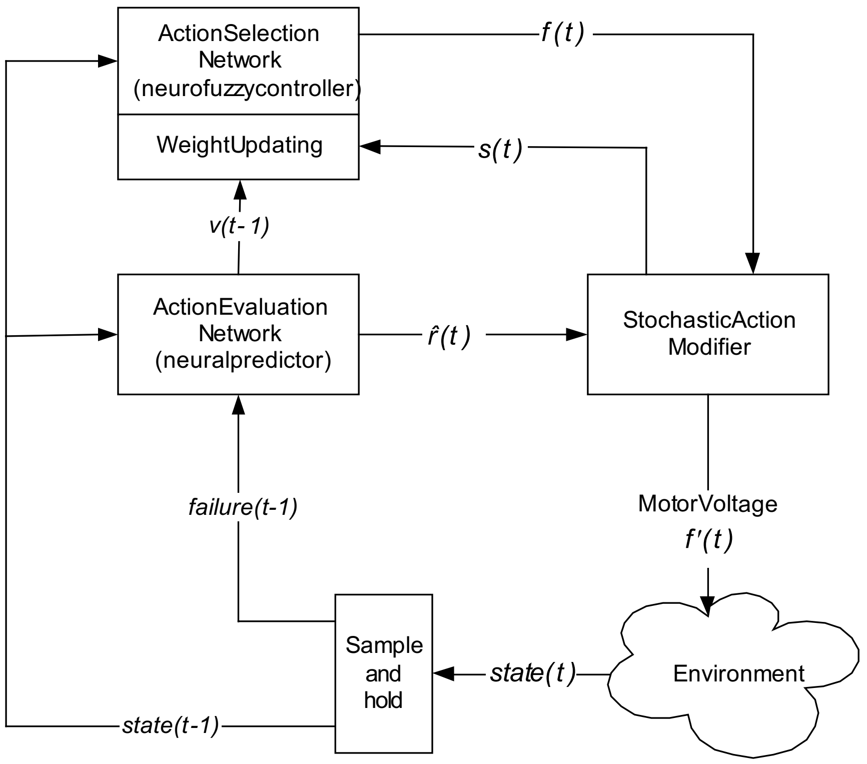

ActionSelection Network (neurofuzzycontroller)
WeightUpdating
f(t)
s(t)
v(t-1)
ActionEvaluation Network (neuralpredictor)
r̂(t)
StochasticAction Modifier
failure(t-1)
MotorVoltage
f'(t)
Sample and hold
state(t)
Environment
state(t-1)

Fig. 5.6 Block diagram of GARIC architecture which combines actor-critic and neurofuzzy methods

The GARIC (Generalised Approximate Reasoning-based Intelligent Control) architecture of [30] is used here as the basis for the developed controller. Figure 5.6 shows the block diagram of this architecture which has been widely used in intelligent control because of the facility it gives to implement a neurofuzzy controller in an actor-critic framework. GARIC consists of a fuzzy controller, the Action Selection Network (ASN), which operates as the actor, and a neural network, the Action Evaluation Network (AEN), which criticises the actions made by the ASN. Outputs of these two networks feed into the Stochastic Action Modifier (SAM) which solves stochastically the exploration-exploitation dilemma: Neither exploration nor exploitation can be pursued exclusively. Exploiting experience, a reward can be obtained but perhaps missing a possibly larger reward. Exploring the space of actions, a better action may be found but with the risk of failing and, consequently, getting a punishment.

Action Selection Network

The Action Selection Network is the fuzzy controller that performs all the control operations. The fuzzy controller is expanded into a neurofuzzy controller so its parameters can be updated according to the signal received from the Action Evaluation Network. The employment of a (neuro)fuzzy network as critic gives transparency to the system. That is, the behaviour of the physical system is described by a set of easily-interpreted linguistic rules. These can then be used to understand and validate the process under control, or to modify it.

The ASN output is $f(t)$, which determines the 'provisional' voltage to be applied to the gripper motor. By 'provisional', we mean that this value is subject to random modification by the Stochastic Action Module as described immediately below. The updating of the ASN modifiable parameters (i.e., the rule confidence vector of connection weights between rule and defuzzification layers) is in the direction which increases the future reward $v(t)$, predicted by the AEN. Hence, to effect gradient descent optimisation, the weight update should be proportional to $\frac{\partial v(t)}{\partial w_{ij}(t)}$, giving:

$$\Delta w_{ij}(n) = \eta \sum_{k=0}^{n} m^{n-k} \hat{r}(k) s(k) \frac{\partial v(k)}{\partial w_{ij}(n)} \tag{1}$$

where k is an index that runs from the initial time 0 up to the current time n, [15], $\hat{r}(t)$ is the 'internal' reinforcement signal, $s(t)$ is an output of the SAM, and η and m are the learning and momentum constants which were empirically set to 0.75. Equation (1) is similar to equation (29) of [30], except that they modify the antecedent and consequent membership functions whereas here we are updating the rule confidence vector.

Action Evaluation Network

The Action Evaluation Network (AEN), or critic, is a neural predictor. Its structure is shown in detail in Figure 5.7. The AEN indicates the current state 'goodness', mapping the input state vector to the reward signal from the environment, $r(t)$. This mapping produces a scalar score, which is a prediction of future reinforcements, $v(t)$. This is combined with $r(t)$ to generate an 'internal' reinforcement signal, $\hat{r}(t)$:

$$\hat{r}(t) = \begin{cases} 0 & \textit{start state} \\ -r(t) - v(t-1) & \textit{failure state} \\ -r(t) + \gamma v(t) - v(t-1) & \textit{otherwise} \end{cases} \quad (2)$$

where γ , set here to 0.47, is a discount rate used to control the balance between long-term and short term consequences of the system's action [15, p606]. Here, 'start' sate means the sate encountered on power-up, and 'failure' state indicates that the system has to restarted, i.e. the gripper has to grip the object anew because it is either crushing it or dropped it. The system detects that is has dropped the object when the force signal changes from a positive value to zero. Unfortunately, however, crushing is not so easy to detect automatically. We hope to develop methods for this in the future. In the meantime, to allow progress to be made, we simplify the problem for the failure situation of crushing: The operator indicates to the system that it has crushed the object.

This network fine-tunes the rule confidence vector. Its input variables are the normalised measurements of the slip rate, the applied force to the object and the applied motor voltage. The hidden layer activation function is sigmoidal. Using a gradient descent algorithm, the formulae for updating the AEN weights are:

$$a_{ij}(t) = a_{ij}(t-1) + \beta_1 \hat{r}(t) y_j(t-1)(1 - y_j(t-1)) sgn(c_j(t-1)) x_i(t-1)$$

$$b_i(t) = b_i(t-1) + \beta_2 \hat{r}(t) x_i(t-1)$$

$$c_{j(t)} = c_j(t-1) + \beta_2 \hat{r}(t) y_j(t-1)$$

$$\textit{for } i = 1, 2, 3 \textit{ and } j = 1.....5$$

where β_1 and β_2 were set empirically to 0.68 and 0.45 respectively. The signum function, sgn(), takes the value 1 when its argument is positive and 0 otherwise.

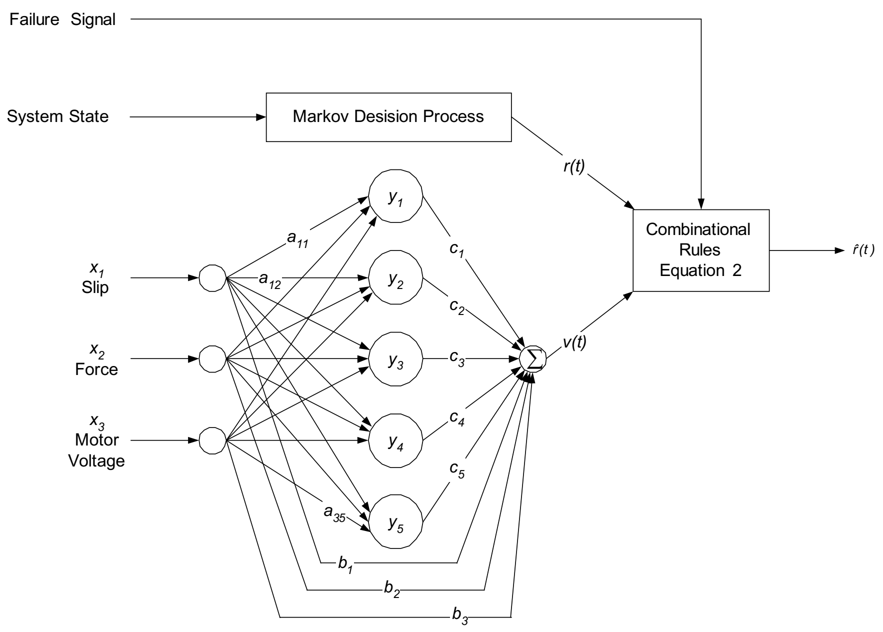


Fig. 5.7. The Action Evaluation Network consists of a neural network predicting future reward, $v(t)$, which is combined with the reward signal, $r(t)$, from the environment to produce an 'internal' reward signal, $\hat{r}(t)$, as described by equation (2)

Stochastic Action Modifier

The Stochastic Action Modifier gives a stochastic deviation to the output of the ASN, so the system can have a better exploration of the state space and a better generalisation ability [16]. The numerical amount of deviation, which is used as a learning factor for ASN, is given by,

$$s(t) = \frac{y^{*\prime}(t) - y^{*}(t)}{e^{-\hat{r}(t-1)}}$$

where instead of using $y^{*}(t)$ as output, the control action applied to the system is $y^{*\prime}(t)$ a Gaussian random variable with mean $y^{*}(t)$ and standard deviation $e^{-\hat{r}(t-1)}$ This leads to the action having a large deviation when the last taken action was *bad*, and conversely when the action was *good.*

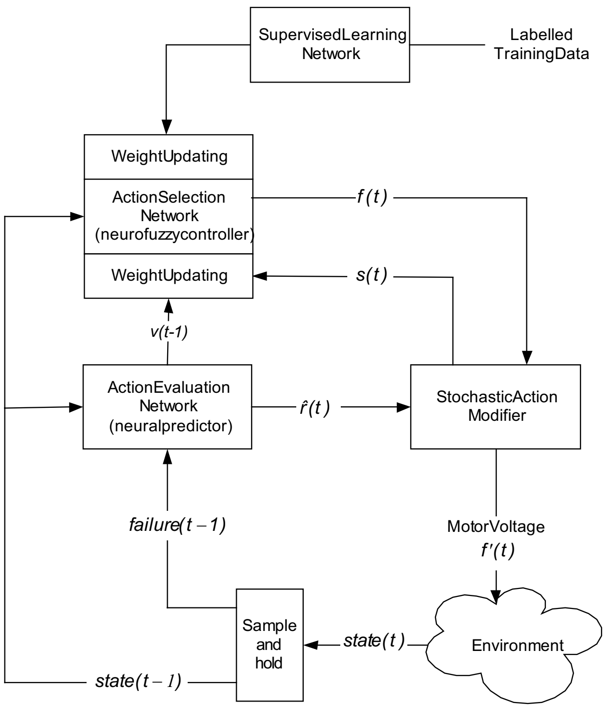


Fig. 5.8 Block diagram of the hybrid supervised/reinforcement system in which a Supervised Learning Network (SLN), trained on pre-labelled data, is added to the basic GARIC architecture

5.4.3 Hybrid Learning

Looking to have a faster adaptation to environmental changes, we have implemented a hybrid learning approach which uses both supervised and reinforcement learning. The combination of these two training algorithms allows the system to have a faster adaptation [16]. The hybrid approach has not only the characteristic of self-adaptation but the ability to make best use of knowledge (i.e., pre-labelled training data) should they exist. The proposed hybrid algorithm is also based on the GARIC architecture. An extra neurofuzzy block, the supervised learning network (SLN), is added to the original structure (Figure 5.8). The SLN is a neurofuzzy controller which is trained in non-real time with (supervised) back-propagation. When new training data are available, the SLN is retrained without stopping the system execution; then it sends a parameter updating signal to the action selection network. The ASN parameters can now be updated if appropriate.

As new training data become available during system operation (see below), the SLN loads the rule-weight vector from the ASN and starts its (re)training, which continues until the stop criterion is reached (average error less than or equal to $0.2V^2$, see Section 5.4.1). The information loaded (i.e, rule confidence vector) from the ASN is utilised as *a priori* knowledge by the SLN. Once the SLN training has finished, the new rule weight vector is sent back to the ASN. Elements of the confidence vector (i.e., weights) are transferred from the SLN to the ASN only if the difference between them is lower than or equal to 5%:

$$\begin{aligned} &if\,(w_i^{ASN} > 0.95 w_i^{SLN}) \wedge (w_i^{ASN} < 1.05 w_i^{SLN}) \\ &\qquad\qquad then\; w_i^{ASN} \leftarrow 0.95 w_i^{SLN} \quad \forall_i \end{aligned} \tag{3}$$

where i counts over all corresponding ASN and SLN weights.

Neurofuzzy techniques do not require a mathematical model of the system under control. The major disadvantage of the lack of this model is the impossibility to derive a stability criterion. Consequently, the use of a 5% threshold as in equation (3) was proposed as an attempt to minimise the risk of system instability. This allows the hybrid system to 'ignore' prelabelled data if they were inconsistent with current-encountered conditions (given by the AEN). The value of 5% was set empirically, although the system was not especially sensitive to this value. For instance, during a series of tests with the value set to 10%, the system still maintained correct operation.

5.5 Results with Real Gripper

To validate the performance of the various learning systems, various experiments have been undertaken to compare the resulting controllers used in conjunction with the simple, low-cost, two-finger end effector (Section 5.2.1). The information provided by the force and slip sensors forms the inputs to the neurofuzzy controller, and the output is the applied motor voltage. Inputs are normalised to the range [0, 1].

Experiments were carried out with a range of weights placed in one of the metal cans (Figure 5.2). Hence, the weight of the object was different from that utilised in collecting the labelled training data (when the cans were empty). This is intended to test the ability of neurofuzzy control to

maintain correct operation robustly in the face of conditions not previously encountered. In addition, information concerning the object to be gripped and the end effector itself were never given to the control system.
To recap, three experimental conditions were studied:

i. off-line supervised learning with back-propagation training;
ii. on-line reinforcement learning;
iii. hybrid of supervised and reinforcement learning.

In (i), we learn 'from scratch' by back-propagation using the neurofuzzy network depicted in Figure 5.9. The linguistic variables used for the term sets are simply value magnitude components: Zero (Z), Very Small (VS), Small (S), Medium (M) and Large (L) for the fuzzy set slip while for the applied force they are Z, S, M and L. The output fuzzy set (motor voltage) has the set members Negative Very Small (NVS), Z, Very Small (VS), S, M, L, Very Large (VL) and Very Very Large (VVL). This set has more members so as to have a smoother output. In (ii), reinforcement learning is seeded with the rule base obtained in (i), to see if RL can improve back-propagation. The ASN of the GARIC architecture is a neurofuzzy network with structure as in Figure. 5.9. In (iii), RL is again seeded with the rule base from (i), and when RL discovers a 'good' action, this is added to the training set for background supervised learning. Specifically, when t_{ok} reaches 3 seconds, it is assumed that gripping has been successful; and input-output data recorded over this interval are concatenated onto the labelled training set. In this way, we hope to ensure that such good actions do not get 'forgotten' as on-line learning proceeds. Typical rule-base and rule confidences achieved after training are presented in tabular form in Table 5.1. In the table, each rule has three confidence values corresponding to conditions (i), (ii) and (iii) above. We choose to show typical results because the precise findings depend on things like the initial start points for the weights [31], the action of the Stochastic Action Modifier in the reinforcement and hybrid learning systems, the precise weights in the metal can, and the length of time that the system runs for. Nonetheless, in spite of these complications, some useful generalisations can be drawn.

One of the virtues of neurofuzzy systems is that the learned rules are transparent so that it should be fairly obvious to the reader what these mean and how they effect control of the object. For example, if the slip is large and the fingertip force is small, it means that we are in danger of dropping the object and the force must be increased rapidly by making the motor voltage very large. As can be seen in the table, this particular rule has a high confidence for all three learning strategies (0.9, 0.8 and 0.8 for (i), (ii) and (iii) respectively). Network transparency allows the user to verify the rule base and it permits us to seed learning with prior knowledge about good actions. This seeding accelerates the learning process [16].

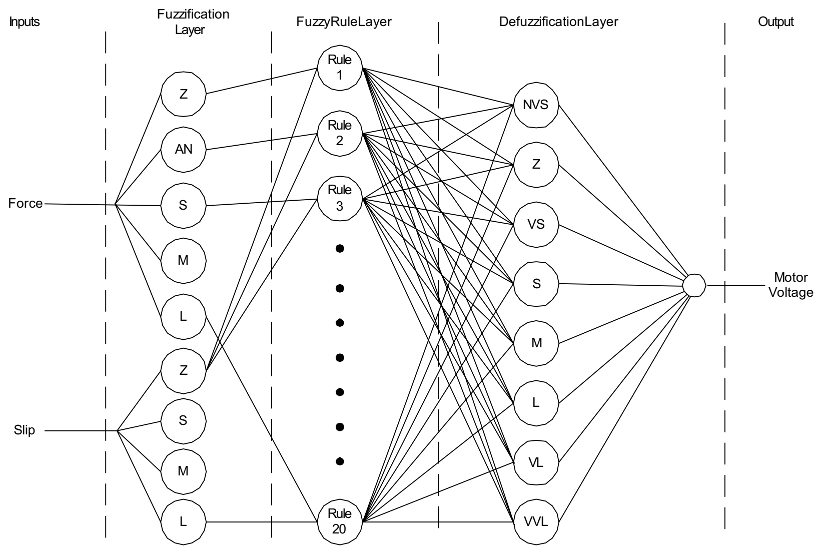


Fig. 5.9 Structure of the neurofuzzy network used to control the gripper. Connections between the fuzzification layer and the rule layer have fixed (unity) weight. Connections between the rule layer and the defuzzification layer have their weights adjusted during training

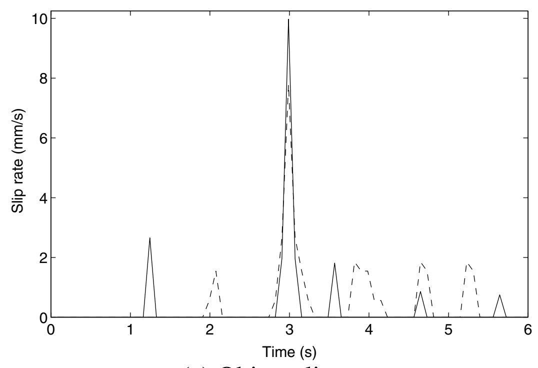
Table 5.1 Typical rule-base and rule confidences obtained after training. Rule confidences are shown in brackets in the following order: (i) weights after off-line supervised training; (ii) weights found from on-line reinforcement learning while interacting with the environment; and (iii) weights found from hybrid of supervised and reinforcement learning

Voltage		Fingertip force			
		Z	S	M	L
Slip	Z	L (0.0, 0.1, 0.0) VL(0.1, 0.6, 0.05) VVL (0.9, 0.3, 0.95)	S (0.05, 0.1, 0.0) M (0.1, 0.4, 0.5) L (0.8, 0.5, 0.5) VL (0.05, 0.0, 0.0)	NVS (0.2, 0.4, 0.3) Z (0.8, 0.6, 0.7)	NVS (0.9, 0.8, 0.8) Z (0.1, 0.2, 0.2)
	VS	L (0.2, 0.2, 0.0) VL (0.7, 0.8, 0.6) VVL (0.1, 0.0, 0.4)	S (0.3, 0.2, 0.2) M (0.6, 0.6, 0.7) L (0.1, 0.2, 0.1)	Z (0.1, 0.2, 0.0) VS (0.9, 0.5, 0.6) S (0.0, 0.3, 0.4)	NVS (0.0, 0.2, 0.3) Z (0.75, 0.7, 0.6) VS (0.25, 0.1, 0.2)
	S	M (0.2, 0.1, 0.2) L (0.8, 0.6, 0.4) VL (0.0, 0.3, 0.4)	M (0.25, 0.3, 0.2) L (0.65, 0.7, 0.7) VL (0.1, 0.0, 0.1)	S (0.4, 0.3, 0.4) M (0.6, 0.7, 0.6)	VS (0.4, 0.5, 0.4) S (0.6, 0.5, 0.6)
	M	L (0.08, 0.1, 0.2) VL (0.9, 0.7, 0.4) VVL (0.02, 0.2, 0.4)	L (0.2, 0.3, 0.2) VL (0.8, 0.7, 0.8)	M (0.3, 0.4, 0.2) L (0.7, 0.6, 0.6) VL (0.0, 0.0, 0.2)	S (0.3, 0.4, 0.1) M (0.7, 0.6, 0.7) L (0.0, 0.0, 0.2)
	L	VL (0.1, 0.3, 0.0) VVL (0.9, 0.7, 1.0)	L (0.1, 0.2, 0.2) VL (0.9, 0.8, 0.8)	L (0.8, 0.7, 0.6) VL (0.2, 0.3, 0.4)	S (0.0, 0.1, 0.0) M (0.9, 0.8, 0.85) L (0.1, 0.1, 0.15)

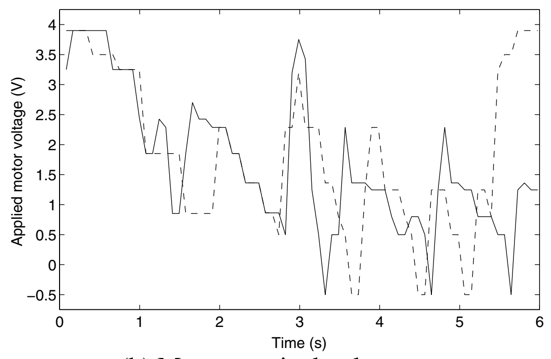
To answer the question of which system is the best, the three learning methods were tested under two conditions: normal (i.e., same conditions as they were trained for) and environmental change (i.e., simulated sensor failure). The first condition evaluates the systems' learning speed while the second one tests their robustness to unanticipated operating conditions. Performances were investigated by manually introducing several disturbances of various intensities acting on the object to induce slip. For all the tests, the experimenter must attempt to reproduce the same pattern of manual disturbance inducing slip at different times so that different conditions can be compared. This is clearly not possible to do precisely. (It was aided by using an audible beep from the computer to prompt the investigator and to act as a timing reference.) To allow easy comparison of these slightly different experimental conditions, we have aligned plots on the major induced disturbance, somewhat arbitrarily fixed at 3 s.

The solid line of Figure 5.10 shows typical performance of the supervised learning system under normal conditions; the dashed line shows operation when a sensor failure is introduced at about 5.5 s. The system learned how to perform under normal conditions but when there is a change in the environment, it is unable to adapt to this change unless retrained with new data which include the change.

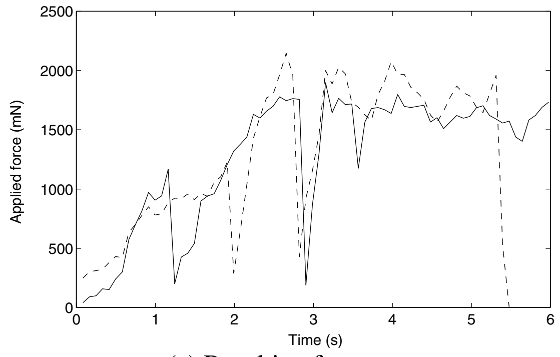
Figure 5.11 shows the performance of the system trained with reinforcement learning during the first interaction (solid) and fifth interaction (dashed) after the simulated sensor failure. To simulate continuous on-line learning but in a way which allows comparison of results as training proceeds, we broke each complete RL trial into a series of 'interactions'. After each such interaction, lasting approximately 6 s, the rule base and rule confidence vector obtained were then used as the start point for reinforcement learning for the next interaction. (Note that the first interaction *after* a sensor failure is actually the second interaction in real terms.) Simulated sensor failure were introduced at approximately 5.5 s during the (absolute) first interaction. As can be seen, during the first interaction following a failure, the object dropped just before 6 s. There is a rapid fall off of resultant force (Figure 5.11(b)) while the control action (end effector motor voltage) saturates (Figure 5.11(c)). The control action is ineffective because the object is no longer present, having been dropped. By the fifth interaction after a failure, however, an appropriate control strategy has been learned. Effective force is applied to the object using a moderate motor voltage. The controller learns that it is not applying as much force as it 'thinks'. This result demonstrates the effectiveness of on-line reinforcement learning, as the system is able to perform a successful grip in response to an environmental change and manually-induced slip.



(a) Object slip

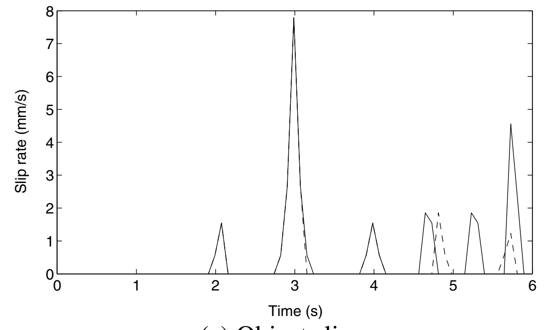


(b) Motor terminal voltage

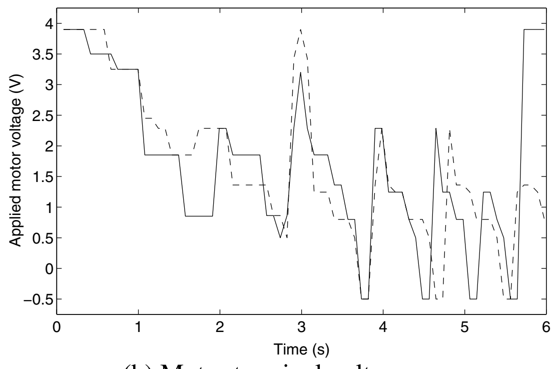


(c) Resulting force

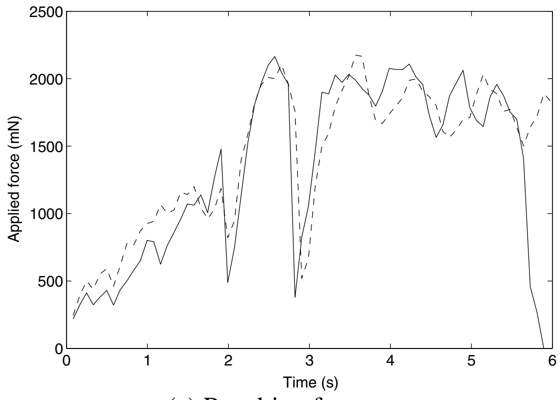
Fig. 5.10 Typical performance with supervised learning under normal conditions (*solid line*) and with sensor failure at about 5.5s. (**a**) slip initially induced by manual displacement of the object; (**b**) control action (applied motor voltage); (**c**) resulting force applied to the object. Note that the manually induced slip is not precisely the same in the two cases because it was not possible for the experimenter to reproduce this exactly



(a) Object slip

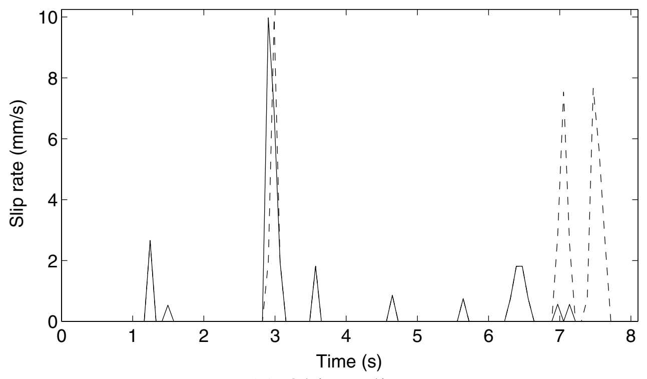


(b) Motor terminal voltage

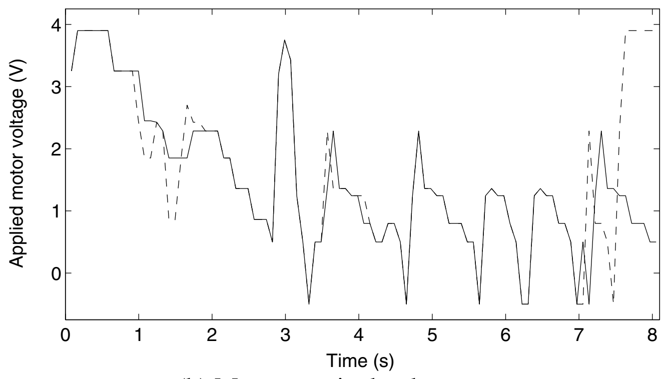


(c) Resulting force

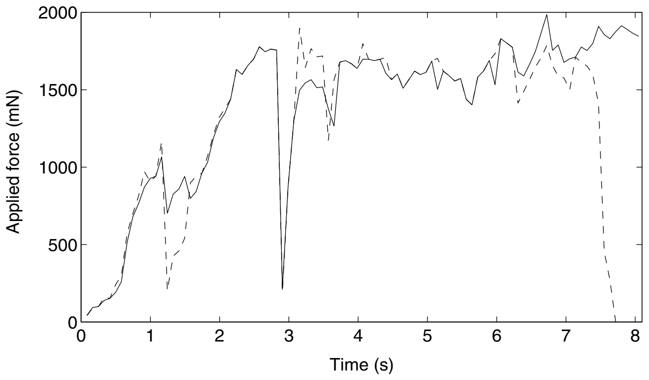
Fig. 5.11. Typical performance with reinforcement learning during the first interaction (*solid line*) and the fifth interaction (dashed line) after sensor failure: (**a**) slip initially induced by manual displacement of the object; (**b**) control action (applied motor voltage); (**c**) resulting force applied to the object



(a) Object slip



(b) Motor terminal voltage



(c) Applied force

Fig. 5.12 Comparison of typical results of hybrid learning (*solid line*) and supervised learning (dashed line) during the first interaction after a sensor failure: (**a**) slip initially induced by manual displacement of the object; (**b**) control action (applied motor voltage); (**c**) resulting force applied to the object

Figure 5.12 shows the performance of the hybrid trained system during the first interaction after a failure (solid line) and compares it with the performance of the system trained with supervised learning (dashed line). Note that the latter result is identical to that shown by the full line in Figure 5.10. It is clear that the hybrid trained system is able to adapt itself to this disturbance where the supervised trained system is unable to adapt and fails, dropping the object.

The important conclusions drawn from this work on the real gripper are as follows. For the system to have on-line adaptation to unanticipated conditions, its training has to be unsupervised. (For our purposes, we count reinforcement learning as unsupervised.) The use of *a priori* knowledge to seed the initial rules helps to achieve quicker neurofuzzy learning. The use of knowledge about good control actions, gained during system operation, can also improve on-line learning. For all these reasons, a hybrid of unsupervised and reinforcement learning should be superior to the other methods. This superiority is obvious when the hybrid is compared against off-line supervised learning.

5.6 Simulation of Gripper and Six Degree of Freedom Robot

Thus far, the gripper studied has been very simple, with a two-input, one-output control action and a single degree of freedom. We wished to consider more complex and practical setups, such as when the gripper is mounted on a full six degree of freedom robot and has more sensor capabilities (e.g., accelerometer). A particular reason for this is that neurofuzzy systems are known to be subject to the well-known *curse of dimensionality* [32, 33] whereby required system resources grow exponentially with problem size (e.g., the number of sensor inputs). To avoid the considerable cost of studying these issues with a real robot, this part of the work was done by software simulation.

A simulation of a 6 DOF robot was developed to have the effects of the robot movements and orientation on the gripping process of the end effector and to avoid the considerable cost of building the full manipulator. The experiments reported here were undertaken under two conditions: external forces acting on the object (with the end effector stationary), and vertical end effector acceleration.

Four approaches are evaluated for the gripper controller with the presence of end effect or acceleration:

i. traditional approach without accelerometer;
ii. traditional approach with accelerometer;
iii. approach with accelerometer and hierarchical modelling;
iv. hierarchical approach with acceleration control.

These are described in the following sections. As the situation studied is virtual, we do not have any labelled data suitable for supervised training. Hence, the four approaches are trained using reinforcement learning. The Markov decision process is the only component which remains identical for all the approaches. The action selection network and the action evaluation network are modified to reflect the new input.

5.6.1 Approach without Acceleration Feedback

Figure 5.13 shows the high-level structure of the neurofuzzy controller used in the previous section (Figure 5.9). This controller is the simplest of all the approaches discussed here: It has only information of the object slip rate and the force applied to the object, so it 'sees' the end effector acceleration as any other external disturbance.

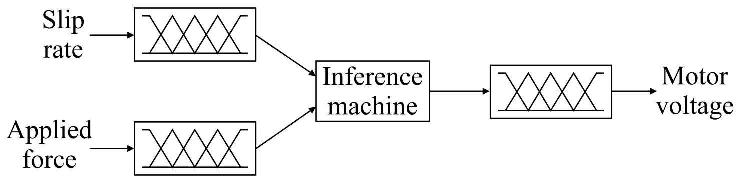


Fig. 5.13 High-level structure of the neurofuzzy controller used in conjunction with the real (two-input) gripper

We now wish to add a new input: the end effector vertical acceleration (i.e., in the *z*-direction). This has the memberships Negative Large (NL), Negative Small (NS), Z, S and L. The density of this fuzzy set is medium [8, p108] so it should be possible to avoid having an excessively complex rule base. For the current conditions, the total number of combinations in the antecedent part is 100 and the possible number of rules is P = 700, according to $P = \prod_{i=1}^{3} N_i$ (see caption of Figure 5.3). Because of the addition of the extra input, a different Action Evaluation Network is required, as shown in Figure 5.14. Again, the input state vector is normalised so the inputs lie in the range [0,1].

The rule base and confidences obtained after training the neurofuzzy controller without accelerometer for 20 minutes, after which time learning had stabilised, are shown in Table 5.2. The dashed line of shows a typical performance of this neurofuzzy controller. While the end effector was stationary, an external force of 10N was applied to the object at 3 seconds with an other external force of -10N being applied to the object as 5 seconds as described in Section 5.2.2. Both external forces induce slip of about the same intensity but with opposite directions. The system is able to grasp the object properly despite the induced disturbances. After 6 seconds, the end effector was subjected to a particular pattern of vertical accelerations as shown in Figure 15(d). The disturbances are standard for testing all four controllers. As the system does not have acceleration feedback, it sees acceleration as any other external disturbance, like a force on the object. Although, the system manages to keep the object grasped, the continual presence of acceleration had made the object slip considerably.

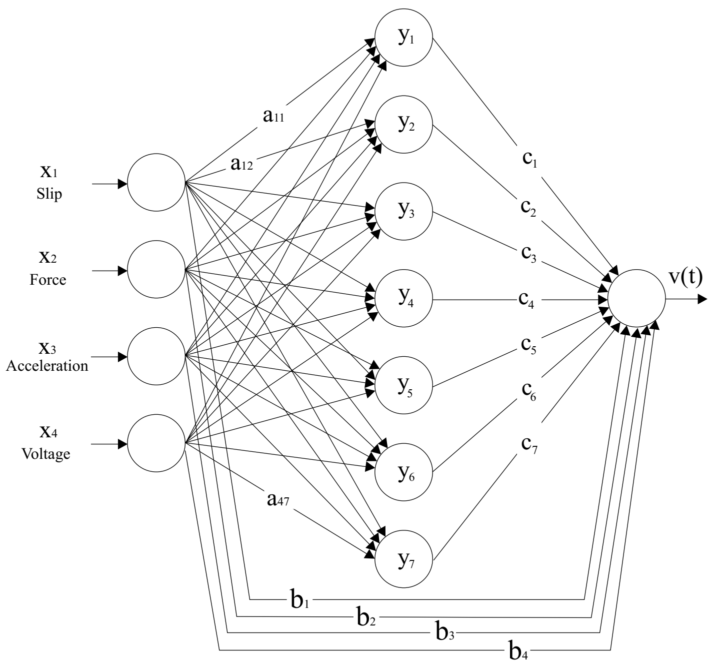


Fig. 5.14 Action evaluation network for the three-input neurofuzzy controller

Table 5.2 Rule-base and rule confidences (in brackets) found after reinforcement learning for the controller without acceleration feedback

Voltage		Fingertip force			
		Z	S	M	L

Slip	Z	VL(0.4)
		VVL(0.6)
			M (0.4)
			L (0.6)
				NVS (0.4)
				Z (0.6)
					NVS (0.8)
					Z (0.2)

	AN	L (0.2) VL (0.8)	S (0.25) M (0.5) L (0.25)	Z (0.3) VS (0.6) S (0.1)	Z (0.8) VS (0.2)
	S	M (0.2) L (0.6) VL (0.2)	M (0.3) L (0.7)	S (0.4) M (0.6)	VS (0.5) S (0.5)
	M	L (0.1) VL (0.8) VVL(0.1)	L (0.3) VL (0.7)	M (0.3) L (0.6) VL (0.1)	S (0.4) M (0.6)
	L	VL (0.2) VVL(0.8)	L (0.1) VL (0.9)	L (0.7) VL (0.3)	M (0.8) L (0.2)

5.6.2 Approach with Accelerometer

The controller described in Section 5.6.1 cannot distinguish the end effector vertical acceleration from any external disturbance acting on the object. If the controller had knowledge of the acceleration such as would provided by an accelerometer, it might be able to react in advance to that disturbance. Accordingly, in this section, a controller which uses acceleration information is developed. The proposed controller is shown in Figure 5.15. This is the traditional approach: It integrates all the inputs into one single fuzzy machine.

For neurofuzzy controllers with more than two inputs, to express the obtained rule base in tabular form, the rule base has to be separated into several tables. The minimum number of tables required is equal to the number of memberships of the smallest fuzzy set. The smallest fuzzy set is the one which has the least number of memberships. Another option (for the three-input case) is to put the rule base into a single table with several rule confidences, each one corresponding to a fuzzy set of the third fuzzy variable. A problem with this approach is that there may be many rules with zero confidence. Table 5.3 shows the obtained rule base after training for 38 minutes, after which time learning had stabilised. Each rule has four confidences corresponding to (i) applied force is Zero; (ii) applied force is Small; (iii) applied force is Medium; and (iv) applied force is Large.

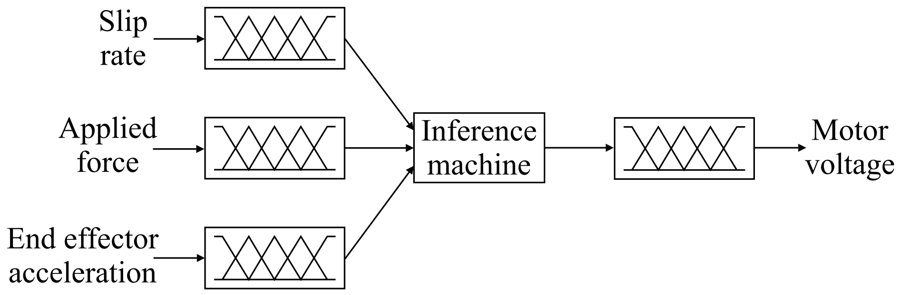
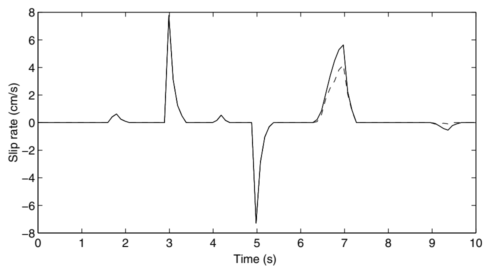
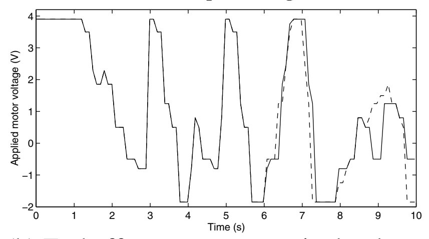


Fig. 5.15. Traditional approach for a neurofuzzy controller with three inputs

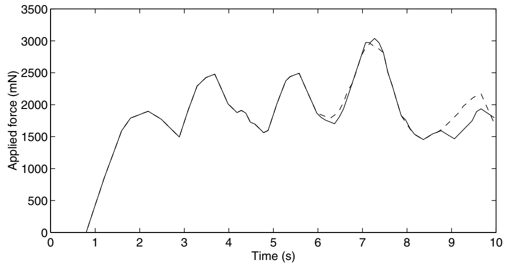
The solid lines of Figure 5.16 show typical performance of the system without acceleration feedback, whereas the dashed lines depict the situation with such feedback. Again, the standard pattern of disturbances is applied: an external force of approximately 10 While the end effector was stationary, an external force of 10N was applied to the object at 3 seconds with an other external force of -10N being applied to the object as 5 seconds with the end effector stationary. These external forces induce slip of about the same intensity but with opposite directions. The system is able to grasp the object properly despite the induced disturbances. After 6 seconds, the end effector was subjected to a particular pattern of vertical accelerations as shown in Figure 5.16(d). The neurofuzzy controller with acceleration feedback increase the motor terminal voltage and so the applied force when the end effector starts accelerating, and does so earlier than the system without such feedback (Figures 5.16(b) and 5.16(c)). This reduces the extent of the slippage, as shown in the latter part of Figure 5.16(a). The system prevents almost perfectly the object slippage due to negative acceleration: Only the positive acceleration is able to induce significant slip.



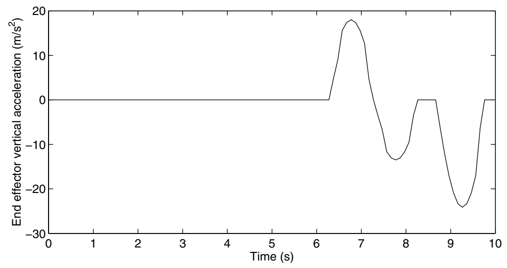
(a) Object slip



(b) End effector motor terminal voltage



(c) Applied force



(d) Vertical acceleration]

Fig. 5.16 Simulated results for the system without information of the end effector vertical acceleration (*solid*) and the system with (*dashed*): (**a**) object slip behaviour; (**b**) control action (applied motor voltage); (**c**) resulting force applied to the object; (**d**) end effector vertical acceleration

Table 5.3 Typical rule-base and rule confidences obtained after training. Rule confidences are shown in brackets in the following order: end effector vertical acceleration is (i) *NL* (Negative Large), (ii) *NS* (Negative Small), (iii) *Z* (Zero), (iv) *S* (Small), (v) *L* (Large).

Voltage		Applied force			
		Z	S	M	L
Slip	Z	VL (0.3, 0.4, 0.4, 0.3, 0.25) VVL (0.7, 0.6, 0.6, 0.7, 0.75)	M (0.1, 0.3, 0.4, 0.3, 0.3) L (0.9, 0.7, 0.6, 0.7, 0.5) VL (0.0, 0.0, 0.0, 0.0, 0.2)	NVS (0.3, 0.4, 0.4, 0.3, 0.1) Z (0.7, 0.6, 0.6, 0.7, 0.9)	NVS (0.7, 0.8, 0.8, 0.7, 0.5) Z (0.3, 0.2, 0.2, 0.3, 0.5)
	AN	L (0.1, 0.2, 0.2, 0.15, 0.1) VL (0.9, 0.8, 0.8, 0.85, 0.9)	S (0.1, 0.2, 0.25, 0.1, 0.0) M (0.4, 0.4, 0.5, 0.5, 0.4) L (0.5, 0.4, 0.25, 0.4, 0.6)	Z (0.1, 0.2, 0.3, 0.2, 0.1) VS (0.8, 0.7, 0.6, 0.7, 0.7) S (0.1, 0.1, 0.1, 0.1, 0.2)	Z (0.6, 0.8, 0.8, 0.6, 0.5) VS (0.3, 0.1, 0.2, 0.4, 0.4) S (0.1, 0.1, 0.0, 0.0, 0.1)
	S	M (0.1, 0.1, 0.2, 0.1, 0.0) L (0.6, 0.7, 0.6, 0.7, 0.7) VL (0.3, 0.2, 0.2, 0.2, 0.3)	M (0.1, 0.2, 0.3, 0.3, 0.1) L (0.8, 0.7, 0.7, 0.6, 0.7) VL (0.1, 0.0, 0.0, 0.1, 0.2)	S (0.2, 0.3, 0.4, 0.2, 0.1) M (0.8, 0.7, 0.6, 0.8, 0.7) L (0.0, 0.0, 0.0, 0.0, 0.2)	VS (0.3, 0.4, 0.5, 0.4, 0.2) S (0.6, 0.6, 0.5, 0.5, 0.7) M (0.1, 0.0, 0.0, 0.1, 0.1)
	M	L (0.0, 0.1, 0.1, 0.0, 0.0) VL (0.8, 0.75, 0.8, 0.9, 0.7) VVL (0.2, 0.15, 0.1, 0.1, 0.3)	L (0.1, 0.2, 0.3, 0.1, 0.0) VL (0.9, 0.8, 0.7, 0.9, 0.9) VVL (0.0, 0.0, 0.0, 0.0, 0.1)	M (0.1, 0.2, 0.3, 0.2, 0.1) L (0.8, 0.7, 0.6, 0.7, 0.7) VL (0.1, 0.1, 0.1, 0.1, 0.2)	S (0.3, 0.3, 0.4, 0.3, 0.1) M (0.6, 0.7, 0.6, 0.6, 0.7) L (0.1, 0.0, 0.0, 0.1, 0.2)
	L	VL (0.15, 0.2, 0.2, 0.2, 0.1) VVL (0.85, 0.8, 0.8, 0.8, 0.9)	L (0.0, 0.0, 0.1, 0.0, 0.0) VL (0.8, 0.9, 0.9, 0.8, 0.6) VVL (0.2, 0.1, 0.0, 0.2, 0.4)	M (0.0, 0.0, 0.0, 0.0, 0.2) L (0.9, 0.8, 0.7, 0.9, 0.8) VL (0.1, 0.2, 0.3, 0.1, 0.0)	M (0.3, 0.5, 0.8, 0.4, 0.2) L (0.7, 0.5, 0.2, 0.6, 0.8)

Comparing the performances of the system with and without acceleration feedback, we conclude the following. When there is no end effector acceleration, both systems perform similarly. In the presence of end effector acceleration, the system with acceleration feedback is able to eliminate or reduce the slippage. However, this improvement has come at the price of having now 700 possible rules whereas before there were only 140 possible rules. So, there is a trade-off between simplicity of the system and a better performance. Nevertheless, this application involving three inputs is still considered a low-dimensional problem [8, p108]; the 700 possible rules demand modest memory and processing time. Accordingly, the mechanical response is not affected by undue processing delay.

5.6.3 Approach with Accelerometer and Hierarchical Modelling

Hierarchical control divides a problem into several simpler subproblems: High dimensional complex systems are divided into several low dimensional subsystems. Hence, this is an attractive technique to identify parsimonious neurofuzzy models [34-37].

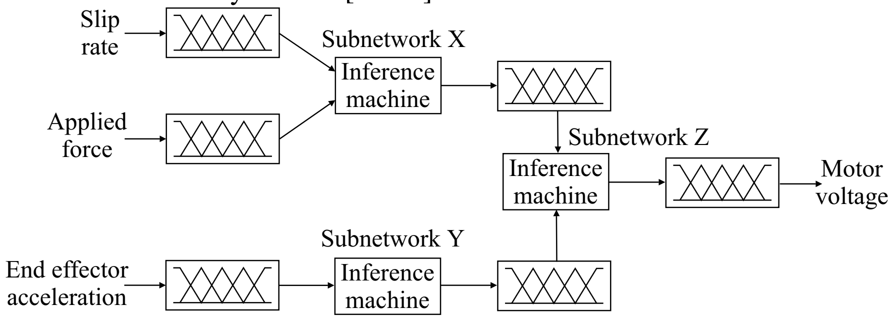


Fig. 5.17 Traditional hierarchical model for the neurofuzzy controller with three inputs.

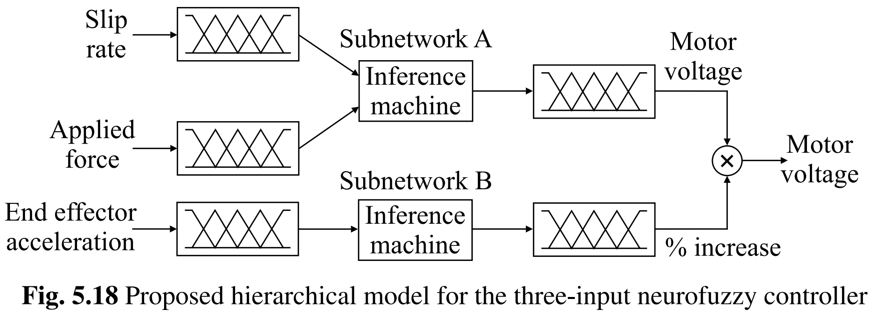


Fig. 5.18 Proposed hierarchical model for the three-input neurofuzzy controller

In the previous section, we saw how the addition of one input to the neurofuzzy controller results in a bigger and more complex rule base. Figure 5.17 shows a neurofuzzy hierarchical structure commonly used to overcome the curse of dimensionality, adapted for the control of our gripper with acceleration feedback. The outputs of the subnetworks *X* and *Y* form the inputs of the subnetwork *Z*. With this approach, the addition of an input variable increases linearly the number of rules. However, the overall network training is difficult as the outputs are complex nonlinear functions of the weights [37, 38]. Consequently, the idea of multiplying the outputs of the subnetworks to generate the overall network output is used here, see Figure 5.18. This design is based on previous results, which have shown that the gripper controller has to increase the motor voltage when the acceleration increases.
In a neurofuzzy hierarchical structure, the rule base increases linearly, so the density of the end effector acceleration fuzzy set can be finer. Consequently, this fuzzy set has now seven memberships: NL, Negative Medium (NM), NS, Z, S, M and L, and the (new) subnetwork *B* output set (i.e., percentage increase in motor voltage) has the memberships Z, S, M and L. The total possible number of rules of the entire network is equal to 188. Accordingly, there has been a considerable reduction of the rule base in comparison with the approach of Section 5.6.2.

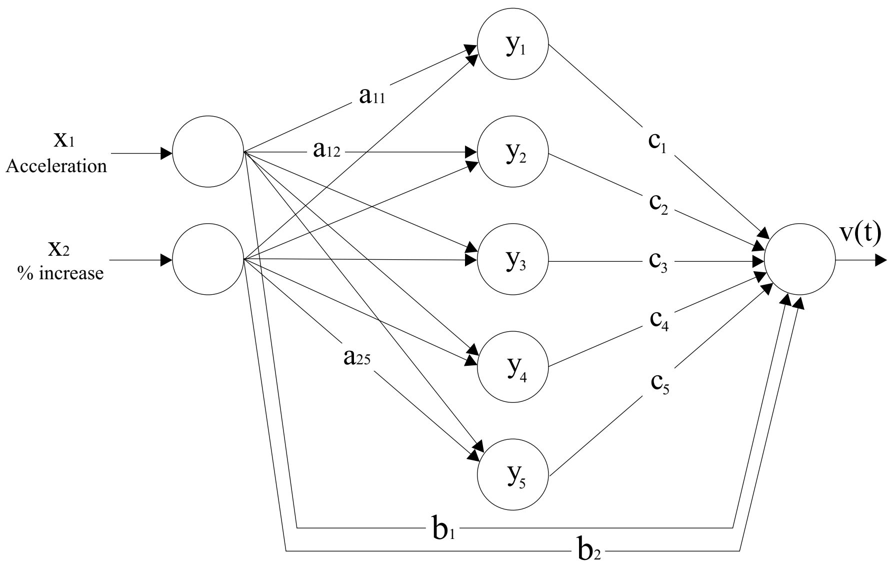


Fig. 5.19 Action Evaluation Network for the neurofuzzy subnetwork *B*

Table 5.4 Rule-base and rule confidences (in brackets) found after reinforcement learning for the neurofuzzy subnetwork *A*

Voltage		Fingertip force			
		Z	S	M	L

Slip	Z	VL(0.4)
		VVL(0.6)
			M (0.5)
			L (0.5)
				NVS (0.4)
				Z (0.6)
					NVS (0.75)
					Z (0.25)

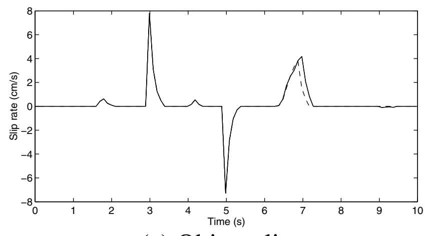
	AN	L (0.2) VL (0.8)	S (0.3) M (0.6) L (0.1)	Z (0.4) VS (0.6)	NVS (0.1) Z (0.8) VS (0.1)
	S	M (0.1) L (0.6) VL (0.3)	M (0.3) L (0.6) VL (0.1)	S (0.5) M (0.5)	VS (0.5) S (0.5)
	M	L (0.1) VL (0.8) VVL(0.1)	L (0.3) VL (0.7)	M (0.4) L (0.6)	S (0.3) M (0.6) L (0.1)
	L	VL (0.1) VVL(0.9)	L (0.1) VL (0.9)	L (0.6) VL (0.4)	S (0.1) M (0.8) L (0.1)

Table 5.5 Rule-base and rule confidences (in brackets) found after reinforcement learning for the neurofuzzy subnetwork *B*

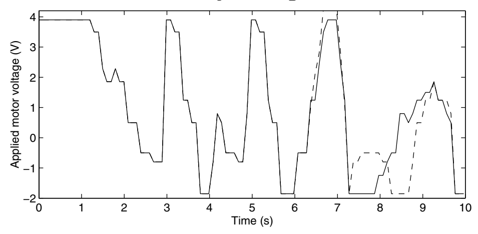
		End effector vertical acceleration						
		NL	NM	NS	Z	S	M	L
% increase	Z	0.0	0.05	0.2	0.95	0.1	0.0	0.0
	S	0.0	0.25	0.7	0.05	0.8	0.25	0.0
	M	0.2	0.6	0.1	0.0	0.1	0.6	0.1
	L	0.8	0.1	0.0	0.0	0.0	0.15	0.9

The training of subnetworks *A* and *B* is identical to the training of the previous neurofuzzy systems, but subnetwork *B* has a different Action Evaluation Network, as shown in Figure 5.19. In the neurofuzzy hierarchical controller, each subnetwork has an independent rule base. Tables 5.4 and 5.5 show the rule bases obtained after 30 minutes of training, for the subnetworks *A* and *B*, respectively. The two subnetworks were trained simultaneously.

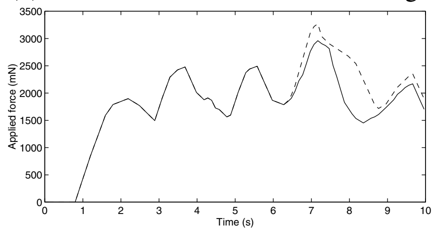
In, Figure 5.20 the dashed line shows typical performance of the neurofuzzy hierarchical controller compared with that of the controller described in Section 5.6.2. Again, with the end effector stationary, two external



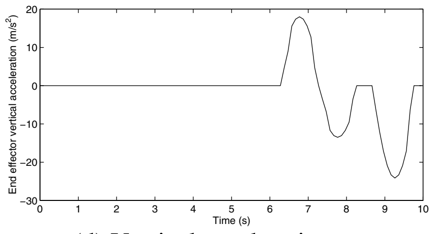
(a) Object slip



(b) End effector motor terminal voltage



(c) Applied force



(d) Vertical acceleration

Fig. 5.20 Simulated results for the system with information about the end effector vertical acceleration (*solid*) and the neurofuzzy hierarchical controller with end effector acceleration feedback (*dashed*): (**a**) object slip behaviour; (**b**) control action (applied motor voltage); (**c**) resulting force applied to the object; (**d**) end effector vertical acceleration

forces are applied to the object to induce slip: 10 N at 3 seconds and -10 at 5 seconds. The system is capable of performing a stable grip despite these disturbances, After 6 seconds, the end effector is subjected to the same patern of acceleration as before. Again, this controller is able to mange excellently the negative acceleration, and reduce slippage for the positive acceleration. The neurofuzzy hierarchical system was not only able to reduce the number of rules but also give a slight improvement in the system performance.

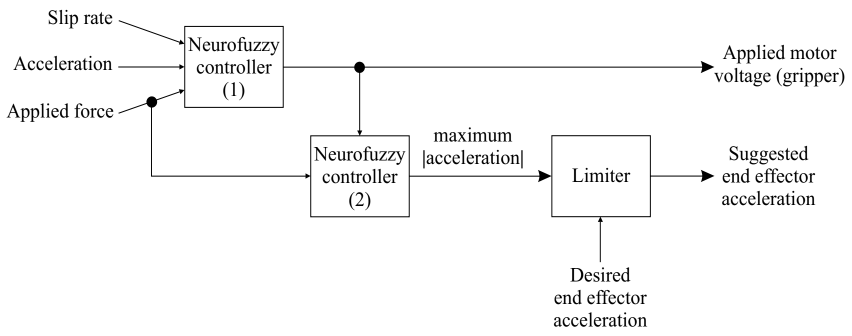

Slip rate
Acceleration
Applied force
Neurofuzzy controller (1)
Applied motor voltage (gripper)
Neurofuzzy controller (2)
maximum |acceleration|
Limiter
Suggested end effector acceleration
Desired end effector acceleration

Fig. 5.21 Hierarchical framework to improve the performance of the end effector, controlling its maximum acceleration

5.6.4. Hierarchical Approach with Acceleration Control

As end effector acceleration can induce slippage, it is worth considering controlling the end effector acceleration. Object slippage can be reduced if more force is applied to the object. However, there is a limit to the force that the gripper can apply because:

- the object can be crushed if large force is applied;
- the object can slip if the gripper cannot apply more force because of mechanical limits.

As result, there is also a limit in the end effector acceleration to ensure a stable gripping. Accordingly, a hierarchical framework to limit the maximum end effector acceleration is proposed, as shown in Figure 5.21. This design features two neurofuzzy controllers. Neurofuzzy controller (1) is the neurofuzzy hierarchical controller described in Section 5.6.3. Neurofuzzy controller (2) is responsible for calculating the maximum end effector acceleration that the gripper can exert yet still guarantee stable gripping.

The output set of Neurofuzzy controller (2) is the maximum absolute acceleration $\left(|a_{max}|\right)$. The absolute value of the maximum acceleration is used to have a more transparent network. It has four memberships: Z, S, M and L. The output of the limiter, i.e., the suggested end effector acceleration, a_s as, is given by:

$$a = min(|a_{max}|, \quad a_d\,)sgn(\,a_d\,)$$

where a_d is the desired end effector vertical acceleration. This value comes from another level of the robot controller, for instance, path or task planning. Figure 5.22 shows the AEN of Neurofuzzy controller (2).

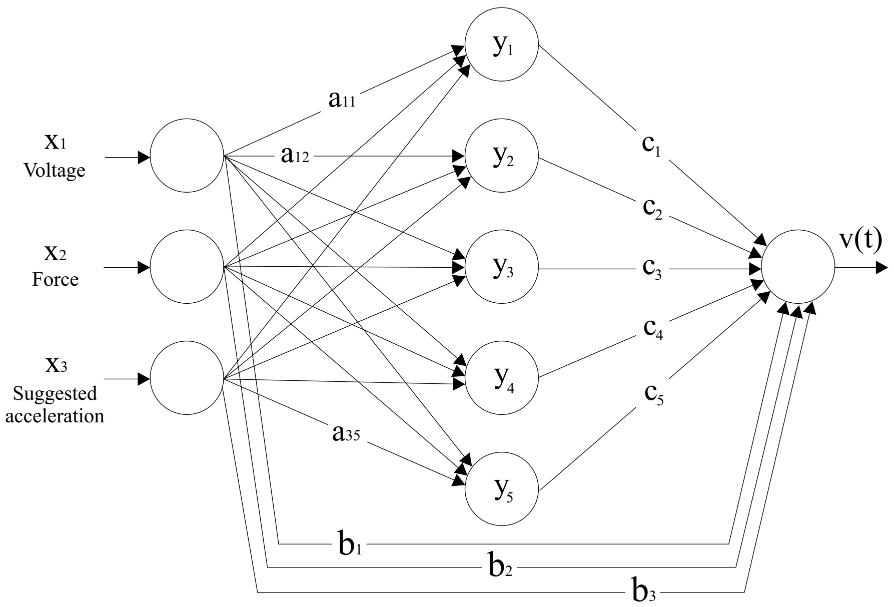
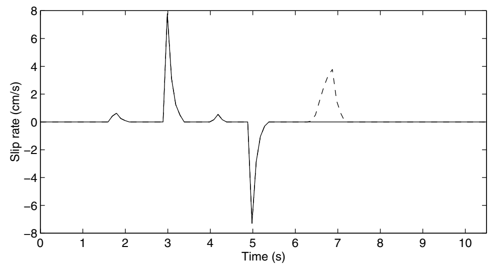


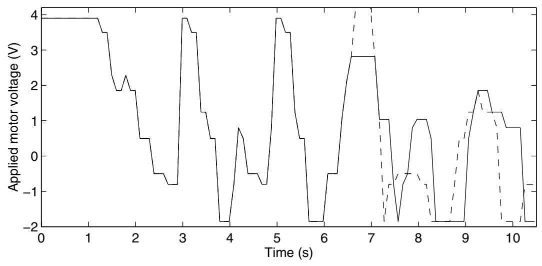
Fig. 5.22. Action evaluation network for Neurofuzzy controller (2) shown in Figure 5.21

Table 5.6 Rule-base and rule confidences (in brackets) found after reinforcement learning for the Neurofuzzy controller(2) in Figure 5.21.

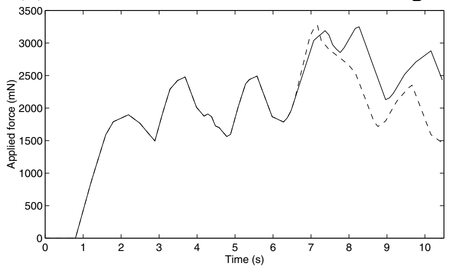
max-min		Applied motor voltage							
		NVS	Z	VS	S	M	L	VL	VVL
Fingertip force	Z	Z (1.0)	Z (0.9) S (0.1)	Z (0.85) S (0.15)	Z (0.8) S (0.2	Z (0.7) S (0.3)	Z (0.6) S (0.4)	Z (0.6) S (0.4)	Z (0.7) S (0.3)
	S	L (0.9) M (0.1)	L (0.9) M (0.1)	L (0.85) M (0.15)	L (0.8) M (0.2)	L (0.8) M (0.2)	L (0.7) M (0.3)	L (0.5) M (0.5)	L (0.4) M (0.6)
	M	L (0.6) M (0.4)	L (0.5) M (0.5)	L (0.4) M (0.5) S (0.1)	L (0.2) M (0.6) S (0.2)	L (0.1) M (0.6) S (0.3)	M (0.5) S (0.5)	M (0.1) S (0.8) Z (0.1)	S (0.6) Z (0.4)
	L	Z (0.2) S (0.8)	Z (0.4) S (0.6)	Z (0.6) S (0.4)	Z (0.7) S (0.3)	Z (0.8) S (0.2)	Z (0.8) S (0.2)	Z (0.95) S (0.05)	Z (1.0)



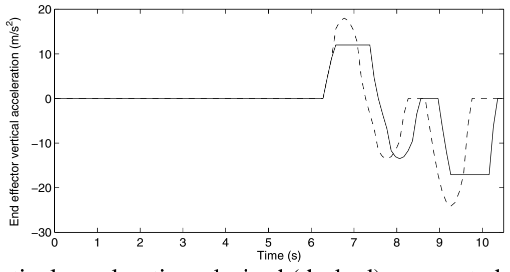
(a) Object slip



(b) End effector motor terminal voltage



(c) Applied force



(d) Vertical acceleration: desired (dashed), suggested (solid)

Fig. 5.23 Simulated results for the hierarchical controller with (*solid*) and without (*dashed*) end effect or acceleration limitation: (**a**) object slip behaviour; (**b**) control action (applied motor voltage); (**c**) resulting force appliedtotheobject; (**d**) end effect or vertical acceleration

Both neurofuzzy controllers share the same Markov decision process. Neurofuzzy controller (1) is trained first; once its training has been completed, Neurofuzzy controller (2) is trained. If the two neurofuzzy controllers were trained simultaneously, it would be impossible to determine which controller was responsible for either failure or success. Table 5.6 shows the Neurofuzzy controller (2) rule base and rule confidences found after 45 minutes of reinforcement learning. The rule base of Neurofuzzy controller (1) corresponds to that of the neurofuzzy hierarchical controller (Tables 5.4 and 5.5). Figure 5.23 shows typical performance with and without acceleration limitation, subjected to the the standard pattern of disturbances. As illustrated by the dashed line in Figure 5.23(a), the hierarchical end effector controller was able to eliminate the object slippage due to end effector vertical acceleration.

5.7 Conclusions

Robotic grippers are commonly used to restrain and manipulate a variety of objects under a wide range of conditions. Accordingly, robotic end effectors are required to be capable of considerable gripping dexterity within an unstructured environment. The variety of objects and working conditions that might be met in practice make it impossible to foresee all the situations the system might encounter. Therefore, robotic systems need to be capable of dynamic adaptation to external disturbances, changing circumstances and unforeseen situations. Ideally, the system should learn its adaptive behaviour on-line, through interaction with its environment, without the supervision of a 'teacher'. This becomes vital when labelled training data are not available. However, when such training data are available, they can profitably be used to provide seed weights for on-line learning. Reinforcement learning is an effective way to train the neurofuzzy system on-line, facilitating adaptation to unexpected conditions. Its use led to successful control in situations where the supervised learning system failed.

Looking to have a faster adaptation to environmental changes, we have implemented a hybrid learning approach which uses both supervised and reinforcement learning. The combination of these two training algorithms allows the system to make good use of effective control actions discovered during operation. These can be considered as 'labelled' data since a record of inputs and outputs is continually kept over a preceding (three-second) period. The supervised learning scheme ran 'in the background' and was only allowed to overwrite the results of reinforcement learning if the corresponding neurofuzzy network weights were not too dissimilar. In this way, the hybrid system can 'ignore' inconsistent (labelled) data. In addition, stability can be maintained (although not guaranteed) in the absence of a precise mathematical model of the system. The performance of the hybrid system was surprisingly good, recovering from simulated sensor failure much faster than the 'pure' reinforcement learning system.

In practice, all robot applications require end effector movement, so there will be end effector acceleration, which can induce slippage. Consequently, it is vital to validate the gripper controller under that condition. A controller without any information of the end effector acceleration treats this disturbance as any other (i.e., external force acting on the object). Although this controller is able to respond adequately over short time periods, the continuous presence of acceleration could lead to the object slipping from the gripper. We therefore developed an approach using end effector acceleration feedback, integrating this extra input in the same fuzzy machine. This controller is indeed able to discover that in the existence of acceleration, it has to apply more force. Thus, slippage is reduced. However, this improvement is at the cost of increasing the rule base, so reducing network transparency. At this point, the dilemma between model simplicity and model accuracy is evident. Hence, a parsimonious neurofuzzy hierarchical controller was proposed. With this approach, the number of rules was smaller than required by the traditional neurofuzzy controller with acceleration feedback. The parsimonious controller had an excellent performance, allowing only slight slippage.

As with any large external force acting on the object, high accelerations of the effector will lead to slippage of the grasped object. The control technique proposed to solve this problem is again hierarchical. The addition of a controller in charge of limiting the end effector acceleration improved noticeably the overall performance. However, such limitation strategies may have other unintended effects on the general performance of the robot. For instance, if the robot is path following, imposing a limit on acceleration places restrictions on the robot capabilities.

In conclusion, it is apparent that when the system has knowledge of a disturbance (i.e., acceleration), it is able to take a better action than when it is ignorant of such a disturbance. In the same way, when the system is able to modify the disturbance, it can help the controller to perform properly. Finally, the curse of dimensionality affects not only the system transparency but also its learning speed and effectiveness. If the number of inputs increases, the system complexity increases too. A more complex system means that there are more possible neurons, weights and rules.

Consequently, the learning problem is made harder. These drawbacks can be reduced using parsimonious adaptive/product models.

References

1. Bicchi, A., V. Kumar, and IEEE, Robotic Grasping and Contact: A Review, in International Conference on Robotics & Automation. 2000: San Fransico. p. 348-53.
2. Dubey, V.N., Sensing and Control Within a Robotic End Effector, in Department of Electrical Engineering. 1997, University of Southampton: Southampton, UK.
3. Chappell, P.H., M.M. Fateh, and R.M. Crowder, Kinematic Control of a Three-Fingered and Fully Adaptive End-Effector Using a Jacobian Matrix. Mechatronics, 2001. **11**(3): p. 355-368.
4. Dollar, A. and R. Howe, Towards Grasping in Unstructured environments: Optimization of Frasper Compliance and Configuration, in IEEE/RSJ International Conference on Intelligent Robots and Systems. 2003: Las Vegas, Navada. p. 3410-6.
5. Nefti, S., et al., *Intelligent Adaptive Mobile Robot Navigation.* Journal of Intelligent and Robotic Systems, 2001. **30**(4): p. 311-329.
6. Brown, M. and C.J. Harris, *Neurofuzzy Adaptive Modelling and Control.* 1994, Hemel Hempstead, UK: Prentice Hall.
7. Kecman, V., *Learning and Soft Computing*. 2001, Cambridge, MA: MIT Press/Bradford Books.
8. Harris, C.J., X. Hong, and Q. Gan, Adaptive Modelling, Estimation and Fusion from Data: A Neurofuzzy Approach. 2002, Berlin and Heidelberg, Germany: Springer-Verlag.
9. Crowder, R.M., Sensors: Touch, Force, and Torque, in Handbook of Industrial Automation, R.L. Shell and E.L. Hall, Editors. 2000, Marcel Dekker: New York, NY. p. 377-392.
10. Domínguez-López, J.A., Intelligent Neurofuzzy Control in Robotic Manipulators, PhD in School of Electronics and Computer Science. May 2004, University of Southampton: Southampton.
11. Brown, M. and C. Harris, Neurofuzzy Adaptive Modelling and Control. 1994, Hertfordshire, UK: Prentice Hall International.
12. Kecman, V., *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. 2001, Boston, MA: MIT Press.
13. Jantzen, J., *Neurofuzzy Modelling*. 1998, Department of Automation, Technical University of Denmark,: Lyngby, Denmark.
14. Driankov, D., H. Hellendoorn, and M. Reinfrank, *An Introduction to Fuzzy Control*. 1993, London, UK: Springer-Verlag.
15. Haykin, S., Neural Networks A Comprehensive Foundation. 1999, Upper Saddle River, NJ: Prentice-Hall.

16. Domínguez-López, J.A., et al. Hybrid neurofuzzy online learning for optimal grasping. in Proceedings of IEEE International Conference on Machine Learning and Cybernetics. 2003. Xi'an, China.
17. Kasabov, N., *Evolving Fuzzy Neural Networks for Supervised/Unsupervised Online Knowledge-Based Learning*. IEEE Transactions on Systems, Man, and Cybernetics, 2001. **31**(6): p. 902-918.
18. Hertz, J., A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*. 1991, Reading, MA: Addison-Wesley.
19. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, Learning representations by back-propagating errors. Nature, 1986. **323**(9): p. 533-536.
20. Chen, O.T.-C. and B.J. Sheu, Optimization Schemes for Neural Network Training, in IEEE International Conference on Neural Networks. 1994: Orlando, FL. p. 817-822.
21. Sutton, R.S. and A.G. Barto, Reinforcement Learning: An Introduction. 2000, Cambridge, MA: MIT Press.
22. Sutton, R.S., Reinforcement learning architectures for animats, in From Animals to Animats: Proceedings of the 1st International Conference on Simulation of Adaptive Behavior. 1991, Bradford Books/MIT Press: Cambridge, MA. p. 288-296.
23. Kaelbling, L.P., *Foundations of learning in autonomous systems.* Robotics and Autonomous Systems, 1991. **8**: p. 131-144.
24. Watkins, C.J.C.H. and P. Dayan, *Q-learning*. Machine Learning, 1992. **8**: p. 279-292.
25. Singh, S., P. Norving, and D. Cohn, A Tutorial Survey of Reinforcement Learning. Sadhana, 1994. **19**(6): p. 851-889.
26. Watkins, C.J.C.H., Automatic Learning of Efficient Behaviour, in Proceedings of First IEE International Conference on Artificial Neural Networks. 1989: London, UK. p. 395-398.
27. Sutton, R., S., A.G. Barto, and R.J. Williams, *Reinforcement Learning is Direct Adaptive Optimal Control.* IEEE Control Systems Magazine, 1992. **12** (2): p. 19-22.
28. Kaelbling, L.P., M.L. Littman, and A.W. Moore, *Reinforcement Learning: A Survey.* Journal of Artificial Intelligence Research, 1996. **4**: p. 237-285.
29. Sutton, R.S., Learning to predict by the methods of temporal differences. Machine Learning, 1988. **3**(1): p. 9-44.
30. Berenji, H. and P. Khedkar, Learning and Tuning Fuzzy Logic Controllers Through Reinforcements. IEEE Transactions on Neural Networks, 1992. **3** (5): p. 724-740.
31. Kolan, J. and J.B. Pollack, *Back propergation is sensitive to initial conditions.* Complex Systems, 1990. **4**(3): p. 269-280.
32. Bellman, R., *Adaptive Control Processes: A Guided Tour*. 1961, Princeton, NJ: Princeton University Press.
33. Bishop, C.M., Neural Networks for Pattern Recognition. 1995, Oxford, UK: Oxford University Press.

34. Bhanu, B., N. Thune, and M. Thune, CAOS: A Hierarchical Robot Control System, in Proceedings of the IEEE International Conference on Robotics and Automation. 1987: Raleigh, NC. p. 1603-1608.
35. Raju, G.V.S. and J. Zhou;, *Adaptive Hierarchical Fuzzy Controller.* IEEE Transactions on Systems, Man and Cybernetics, 1993. **23**(4): p. 973-980.
36. Mills, D.J., C.J. Harris, University of Southampton Technical Report, Neuro-fuzzy modelling and control of a six degree of freedom AUV. 1995.
37. Bossley, K.M., Neurofuzzy modelling approaches in system identification. 1997.
38. Brown, M. and C.J. Harris, *A Perspective and Critique of Adaptive Neuro-fuzzy Systems used in Modelling and Control Applications.* International Journal of Neural Systems, 1995. **6**(2): p. 197-220.

6 Voronoi-Based Outdoor Traversable Region Modelling

Cristina Castejón[1], Dolores Blanco[2], Beatriz L. Boada[1] and Luis Moreno[2]

1. Mechanical Dept., University of Carlos III, Leganés Madrid, Spain.
{castejon,bboada}@ing.uc3m.es

2. Automation and System Division, University of Carlos III, Leganés Madrid, Spain.
{dblanco,moreno}@ing.uc3m.es

6.1 Introduction

Nowadays, the interest in mobile robotics working in outdoor environments is increasing due to the new applications focussed on aiding humans. To collaborate with humans in transportation, or evolve in dangerous or unknown places it is necessary to develop new skills [20]. To give the robot the autonomy skill, knowledge about the environment, where the robot is going to evolve, to manipulate objects, to navigate without collision or to plan trajectories, is needed. The environment modelling is one of the main tasks to develop in mobile robotics. Path planning, localization and control tasks, need a model which represents the environment around the robot.

To model outdoor environments for navigation, a methodology must be developed. The model obtained with the methodology must be useful for navigation, and it must be computed in real-time. The model will be adapted to the terrain type and to the robot's characteristics.

Previous to the model building, the development of a suitable sensor system is very important for the robot's autonomy. While the robot is moving, the information about the environment is being acquired by the sensor system. In each perception, the information sensed is processed to obtain useful information for the decisional systems. The robot's autonomy will depend on the information richness and the data interpretation. The information perceived by the sensor system is used for localization, manipulation or path-planning. In general, to model environments, those sensors that obtain depth information such as the scanner laser, sonars or digital cameras will be of interest. As it is explained in [18], when an outdoor environment has not regular properties (as indoor has), the perception system must provide a generic characterization of the obstacles in the environment. To avoid the obstacles when the robot evolves, more information than in an indoor environment is needed. Therefore, the use of three-dimensional information sensors is of great importance. Two different methodologies are found in literature, those based on scanner laser [9,36,26,24] and stereovision [37,28,29,12].

6.1.1 Modelling

The data perceived by the sensor system are transformed in models to represent the environment. The environment modelling allows the robot, besides knowing its location, to identify the different alternatives to accomplish its mission with success. There are different techniques developed to model the environment.

- The most basic representation is the space discretization in cells which provides a flexible obstacles representation [51]. This representation is easy to manipulate and the discretization facilitates sorted and specific information for the robot tasks. For the discretization construction, the sensor information is transformed using techniques such as occupancy grids [19], digital elevation maps (as in figure 6.1) [26] or spatial representations such as octrees, quadtrees [33], etc.

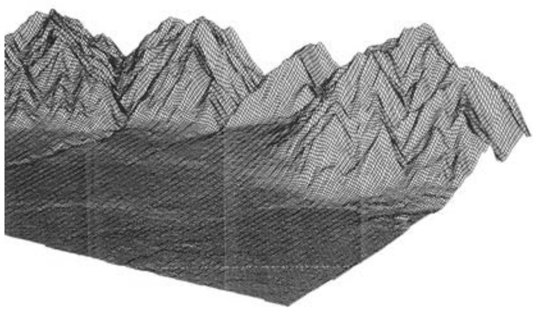


Fig. 6.1. Digital Elevation Map

- Geometrical representation approximates the measurements by mathematical functions, easy to manipulate by a computer. The geometrical model is used to represent objects and compute its attributes. This type of models is of interest for manipulation, virtual spaces construction for tele-operation and geometrical localization. There are different techniques to build geometrical maps, such as super-quadrics or hyper-quadrics, functional models (polinomial, splines, etc.) or deformable surfaces [31]. Geometrical models generate accurate maps, close to the real environment, but these representations are not used in maps for outdoor navigation due to its high time computing and the bulky models which generate.

- In topological models, the numerical information is replaced by symbolic data, and the map is represented as a graph. Topological models have been proved to be useful in navigation along corridors and in topological localization. Topological model applications in outdoor environments have not been found, despite of its effectiveness in large environments [43], [40]. It allows path-planning in a fast way and it reduces the amount of data and the computing time. But the graph representation needs repetitive elements (such as doors, walls, corners, etc.) [7], that in outdoor environments do not exist. This technique is interesting to model large indoor environments where a great deal of common objects exists. The objects allow the robot to localize and to evolve. But for outdoor environments, a-priori unknown, these models are not successful.

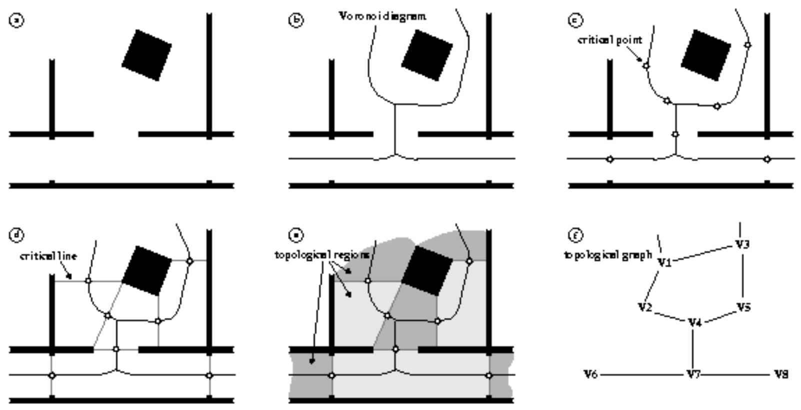


Fig. 6.2. Topological maps extraction [49]. (**a**) Cell map, (**b**) Voronoi Diagram, (**c**) critical points, (**d**) critical lines, (**e**) topological regions, and (**f**) topological graph

- Hybrid representations are a good alternative to the methods described above. This method combines different modelling techniques to take advantage of each method and to solve the disadvantages. Currently, it is a good choice for navigation map modelling. Some authors work with topo-geometrical models [4], where they carry out a terrain discretization, using a sensor based model. Thrun in [49] builds a topological graph based on an occupancy grid. Betgé-Brezetz in [2] describes, in a topological way, the locations, starting with the geometrical information. The author builds planes to represent the flat and super-quadrics for the obstacles representation. With digital image techniques and border extraction, the objects are separated from the environment to obtain the topological model. This description is limited to easy environments with circular-form obstacles on the terrain. Other authors build topological models from the geometrical information provided by the sensor system. Simhon and Dudeck, which propose in [46] a global hybrid topological model, built from local maps called *Islands of reliability*, where each local map has quantitative information about the environment, based on the metric information from the sonar. The same sensor is used by Choset et al. in [15], which describes the sensor-based exploration with Voronoi diagrams. The system builds, incrementally, a connected graph with points that belong to the Voronoi edges. This technique is used by Blanco et al. in [6] with a laser scanner.

In next sections a technique for modelling traversable regions will be developed. The model is a topo-geometric type and can represent large outdoor environments. The model will be represented as a Voronoi Diagram based on the sensor information provided by a three-dimensional scanner laser. The environment is divided in regions that can be crossed by the robot or not. The regions that are crossable by the robot will be called *Traversable regions*, and the traversability characteristic will be defined based on the robot and the terrain characteristics.

6.1.2 Traversability

The traversability attribute represents the terrain suitability to navigate. Several authors study this attribute. The traversability concept was introduced by Langer in [16], and it highlights the region capacity to be crossable or not by a robot.

Most of the works, related to outdoor navigation, try to divide the perceived terrain in regions with different characteristics (in a segmentation process). Langer in [29] performs a terrain segmentation and generates a list of crossable regions for the robot, with depth images provided by a stereovision system and the sorted information with a Digital Elevation Map (DEM). The classification allows the robot in [16] to navigate over roads and highways, we mean, easy environments, partially structured. For planetary environments, Seraji in [45] introduces the concept of *traversability index*. The index expresses the suitability of a terrain to be crossed, based on physical properties such as the slope and the roughness. Gennery in [22] uses this concept and a set or fuzzy rules to classify the terrain in difficulty degrees.

A great deal of authors work with two base parameters: the terrain slope and the roughness degree. To obtain the slope, algorithms have been already developed. For example the general slope calculation using the horizon line obtained in three-dimensional coordinates, with a stereovision system and interpolating only one plane, as Howard el al. present in [25], or the use of neural networks with a complex training, based on the presentation of different terrain patterns to obtain a numerical slope [1].

The last two works presented before, have been developed specifically for planetary type environments, we mean, flat terrain with defined form obstacles (basically rocks over the flat), different to unknown terrestrial environments. Other authors compute the slope in each point. The number of operations is higher but it is not necessary to perform interpolations or to pre-process the sensor information. Among these methods we highlight the interpolation of four connected points performed by Nashashibi in [39], and the use of vision masks in a neighbourhood of a point [3]. The last method has been chosen to evaluate the slope in each point.

On the other hand, the roughness is treated as a measurement of the surface variation. Most of the authors evaluate the roughness as the height variation. For example Nashashibi in [39] and Langer in [29] evaluate the discontinuities with the gradient calculation and the sorted data provided by a telemeter. Other authors calculate the dispersion points with respect to an interpolated plane [22]. For Seraji [23], the roughness depends on the number of rocks in the environment and the parameter is computed using artificial vision techniques and fuzzy rules.

In this work, the Traversability Numerical Model is built in a low level. The terrain is divided in crossable and non crossable areas for the robot. It must be taken into consideration that the traversability characteristic not only depends on the terrain properties, but it also depends on the robot physical restrictions and the robot task.

6.2 The Modelling System: General Structure

The philosophy of the modelling system proposed in this chapter, and in [11], is to divide the general task in three abstraction levels (as shown in figure 6.3). The idea is to choose a method which is able to build a conceptual model that allows representing the terrain difficulty for robot navigation. The levels are described as follows:

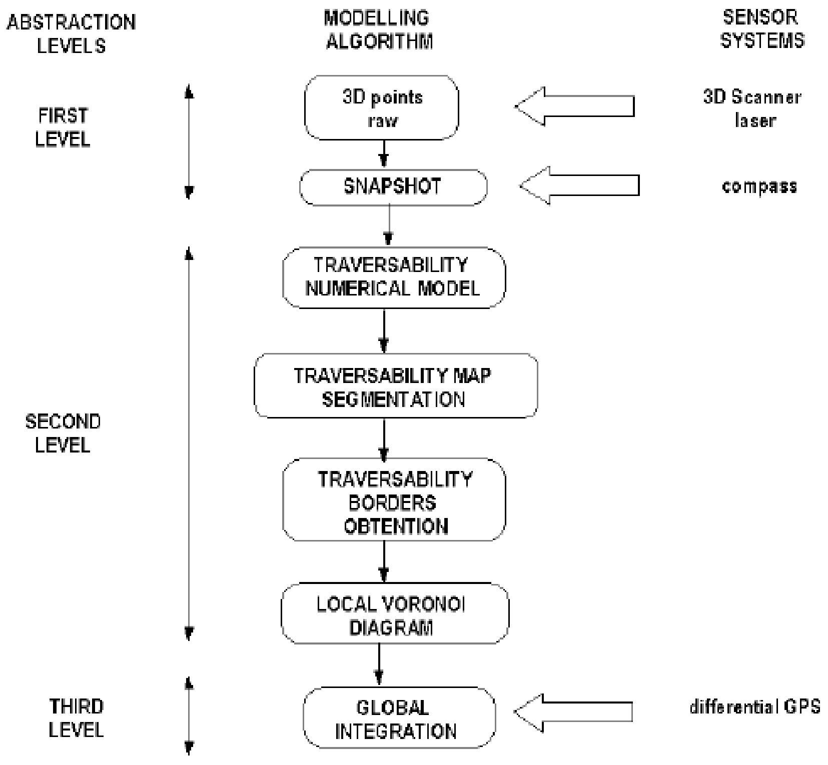


Fig. 6.3. Modelling system scheme

1 In a first level, the three-dimensional information is captured by the sensor system, (the experimental tests have been done with a scanner laser), and it is computed with the aid of a compass in a snapshot representation.

2 In a second level, the model is enhanced with the traversability information. The model obtained will be called Traversability Numerical Model (TNM), which will be defined in section 6.3. The TNM divides the 3D environment in regions that can be crossable or not crossable by the robot. Inside this level, the 3D measurements are projected in a XY plane and are discretized in a digital elevation map to obtain a region representation. Besides, the borders of the non traversable re-

gions are extracted. This information will be useful for the local and global Traversable Region modelling technique chosen.

3 At last, in the third level, and thanks to the global information provided by a Global Positioning System (GPS), a global integration with the previous local models obtained in the successive perceptions, while the robot is moving, will be done.

6.3 Traversability Analysis

In outdoor navigation is interesting to estimate some parameters which define the terrain difficulty to be crossed. Path-planning in large and outdoor environments is a complex task, because there are a lot of parameters which define the traversability, for example:

- The task the robot is going to perform over the environment. Manipulation, planetary exploration, etc., need different models and information. In outdoor navigation, the robot task is to build a sensor based model to evolve in a large outdoor environment, a priori unknown. The model chosen is of topo-geometric type.
- The experimental platform characteristics, its locomotion system and physical restrictions affect in future decisions-taking. In the outdoor navigation case, *the locomotion system*, which shows the maximum height the robot can go through in its movement, and *the robot size*, that allows to determine the minimum elevation the robot can go under (the 3D free space), are the most important robot characteristics (but others can be considered).
- The terrain characteristics, which determine the terrain difficulty to be crossed. Most authors focused their research in two basic parameters to specify the terrain characteristics, the terrain slope and the roughness degree. The two parameters not only are used in robotics, but also in the topography field, and specifically in Geographical Information Systems (GIS) but in another level and with different sensor systems. Currently, two basic characteristics are used to define the traversable zone for the robot and for an outdoor environment that can be non-structured. These parameters are the *slope* and the *roughness*.

Based on this, and taking into account the general modelling structure, the Traversability Numerical Model (TNM) is defined as the 3D snapshot model enhanced with the traversability characteristic. The 3D points, perceived by the sensor system, will be considered as traversable or non traversable by the robot. It must be considered reminded that the traversability characteristic strongly depends on the terrain and robot features.

To obtain the TNM, four different methodologies can be presented:

1 Elevation analysis.
2 Slope analysis.
3 Slope and roughness analysis with:
   (a) elevation gradient analysis,
   (b) slope gradient analysis.

6.3.1 Elevation Analysis

The elevation analysis is an easy and classic method, which allows determining free 3D spaces. It consists of considering as non traversable points those that have an elevation or z coordinate between thresholds T_{min} and T_{max} obtained by the following reasoning:

- T_{min} depends on the robot locomotion system and gives the higher obstacle value the robot can go through.
- T_{max} depends on the maximum robot height and it is used to detect aerial obstacles (such as aerial beams or door-hinges)

In the mathematical expression for this analysis, we name NTR the Non-Traversable-Region and it is defined as:

Definition 1. Given a 3D points image $I = \{p_i(x_i, y_i, z_i)\}$ we define the set NTR (Non Traversable Region) as the sub-set of I, such as it fulfills the following traversability property :

$$NTR^{test1} = \{\, p_i(x_i, y_i, z_i) \;/\; T_{min} < z_i < T_{max} \,\} \tag{6.1}$$

Defining as TR the Traversable Region, it will be fulfilled in all cases that:

$$NTR \bigcup TR = I \tag{6.2}$$

$$NTR \bigcap TR = \varnothing \tag{6.3}$$

We mean, the NTR is defined as the set of points p_i, such as the elevation z_i is placed between borders, which will depend on the robot's physical characteristics [10]. In figure 6.4 the NTR for a generic outdoor robot is presented.

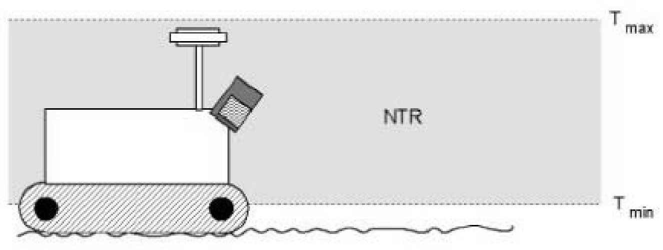


Fig. 6.4. NTR definition

The consideration of this parameter detects, exclusively those aerial or terrain obstacles which can collide with the robot, such as: aerial beams or door-hinges.

For example, in figure 6.5 a partially structured environment with aerial obstacles is shown. The environment presents a low roughness degree and accessible for the robot. In figure 6.6 the TNM obtained with the *test 1* evaluation is presented. In dark colour, points belonged to the NTR are presented, and in light colour, those belonged to the TR. The aerial obstacle is evaluated as belonged to the TR, because the aerial beam is place higher than the robot height. Besides, the terrain has a low roughness degree, so this test, in this case, provides a suitable result.

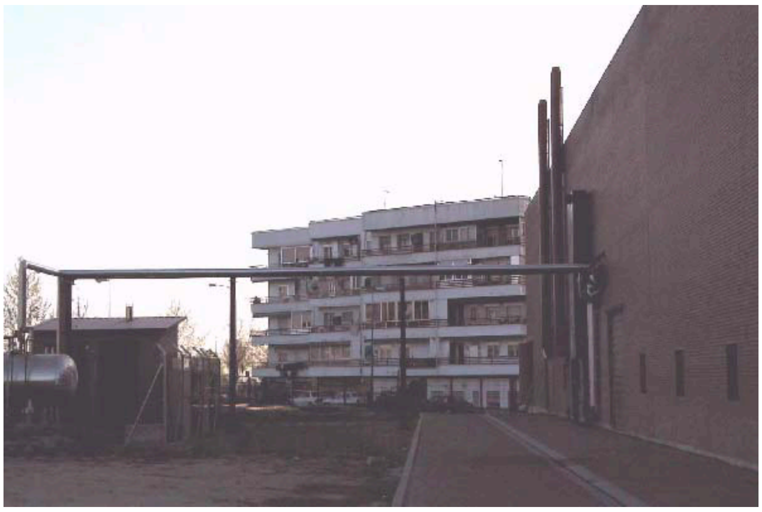


Fig. 6.5. Environment with aerial obstacles. Real image

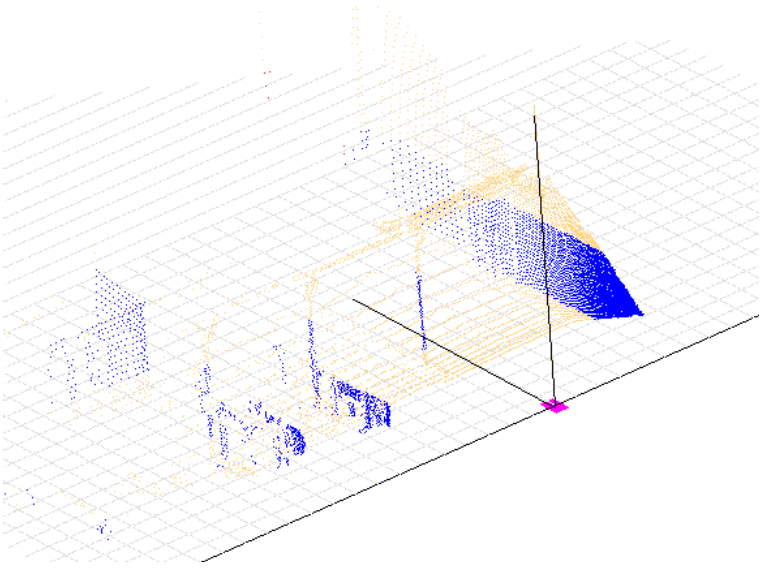


Fig. 6.6. Environment with aerial obstacles. TNM

Nevertheless the elevation analysis is restricted to generally flat environments, with certain degree of roughness, but without slope. Sloped terrain will be considered by the robot as obstacles. So it will be necessary in these cases to do a slope analysis.

6.3.2 Slope Analysis

The surface inclination or slope can be defined as the existing angle between the surface normal vector $\overrightarrow{N}$ and the vector $\overline{W_\pi}$, perpendicular to the horizontal surface, as we represent in figure 6.7.

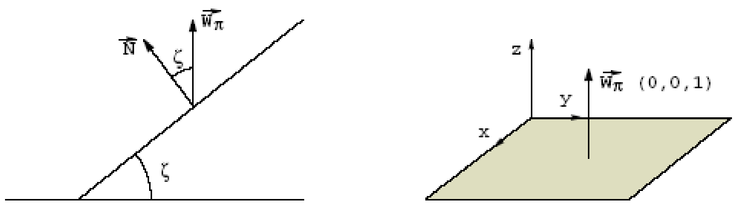


Fig. 6.7. Terrain slope definition

Lots of methods can be found in literature to obtain the terrain slope: in each point [39] or interpolating planes and calculating the normal vector in each one [25]. Betgé-Brezetz in [2] calculates the normal vector in each 3D point. The number of required operations is greater than in previous cases, but it is easier to obtain because it is not necessary to apply complicated interpolation methods or pre-processing the information.

The algorithm used to calculate the slope in each point developed by Betgé-Brezetz in [3] consists of applying, over a sorted data image, a Sobel filter in the horizontal and vertical directions, to obtain the tangents in each point along the two scanning directions, defined in figure 6.8 as θ and ϕ scans.

Each 3D point (P) is represented in a Cartesian reference system:

$$P = \begin{pmatrix} x(\rho,\theta,\phi) \\ y(\rho,\theta,\phi) \\ z(\rho,\theta,\phi) \end{pmatrix} \tag{6.4}$$

The slope in each point is calculated based on the previous normal vector calculation, represented in figure 6.8 as $\overline{N_P}$.

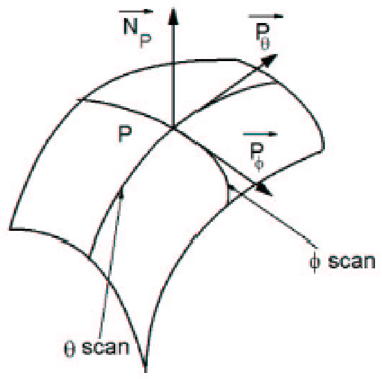


Fig. 6.8. Normal in the point P estimation

We define the lines ϕ and θ as the horizontal and vertical scans respectively. And the 3D points image is defined as $I(\theta,\phi)$, such as the value in each pixel is the depth ρ. In figure 6.8, the normal vector is observed to be the normalized cross product of the tangent vectors to the lines ϕ and θ (called P_ϕ and P_θ). We mean:

$$\overrightarrow{N_P}(\theta,\phi) = \frac{\overrightarrow{P_\theta} \wedge \overrightarrow{P_\phi}}{\left\|\overrightarrow{P_\theta} \wedge \overrightarrow{P_\phi}\right\|} \tag{6.5}$$

with

$$\overline{P_\theta} = \begin{pmatrix} \dfrac{\partial x}{\partial \theta} \\ \dfrac{\partial y}{\partial \theta} \\ \dfrac{\partial z}{\partial \theta} \end{pmatrix}; \quad \overline{P_\phi} = \begin{pmatrix} \dfrac{\partial x}{\partial \phi} \\ \dfrac{\partial y}{\partial \phi} \\ \dfrac{\partial z}{\partial \phi} \end{pmatrix} \tag{6.6}$$

The surface orientation depends on the cross product sign; the orientation is chosen such as $\overline{N_P}$ goes always to the sensor direction.

Mathematically, the tangent vector is obtained when the curve in the point is derived. If a discretization is done and we apply digital image processing techniques, the continuous derivatives that appear in P_θ and P_ϕ are calculated by multiplying the mask and adding the elements.

As can be seen in figure 6.7, the slope is the angle between the normal vector in a point, and the vector perpendicular to the XY, situated in the robot base. The angle that defines the slope in the point P is calculated in the following equation:

$$\vec{N} \cdot \overline{W_\pi} = \left|\vec{N}\right| \cdot \left|\overline{W_\pi}\right| \cdot \cos\xi \tag{6.7}$$

where,

$$\cos\xi = \frac{\vec{N} \cdot \overline{W_\pi}}{\left|\vec{N}\right| \cdot \left|\overline{W_\pi}\right|} \tag{6.8}$$

and then,

$$\xi = \arccos\frac{\vec{N} \cdot \overline{W_\pi}}{\left|\vec{N}\right| \cdot \left|\overline{W_\pi}\right|} \tag{6.9}$$

This method gives the possibility to calculate the normal vector in each 3D point in a fast and easy way. It is suitable in unknown outdoor environments and non-structured, because the operations are performed in a local neighbourhood. Besides, the algorithm can be parallelized. Furthermore, despite the slope values are not exact, due to the filter applied; its accuracy is enough for solving the navigation problem.

In order to the robot physical restrictions, basically the robot mass and its locomotion system, a maximum and minimum slope, that the robot can go up and go down, exist. With the robot information, the traversable slope analysis is performed as follows:

Definition 2. Given a three-dimensional points image $I = \{p_i(x_i, y_i, z_i)\}$ the Non Traversable Region (NTR) is defined as the subset of 3D points $p_i \in I$, such as the slope in each point ξ_i fulfil the following property:

$$NTR^{test2} = \left\{ p_i(x_i, y_i, z_i) \; / \; \xi_{min} > \xi_i \quad AND \quad \xi_i < \xi_{max} \right\} \tag{6.10}$$

We mean, the slope analysis considers as non traversable areas (NTR) those regions where the slope in each 3D point exceeds two thresholds.

In figure 6.9 a semi-structured environment is presented. As the most relevant characteristics the big differences in the slopes, drawn in figure 6.9, and the high positive slope, the tree on the left hand side present, can be highlighted.

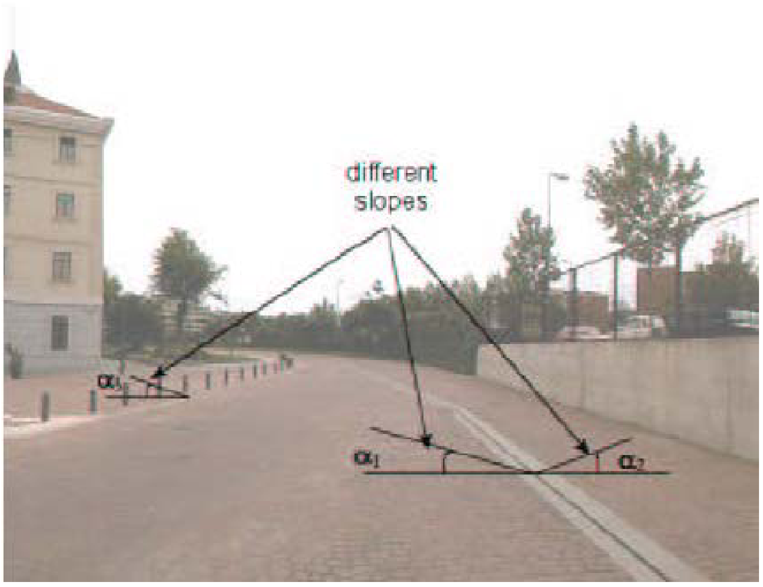


Fig. 6.9. Real environment with different slopes

In figures 6.10 and 6.11, the TNM built with the *test 1* and *test 2* respectively is presented. In dark colour, those points labelled as non traversable are represented, and in light colour, those belonging to the traversable regions.

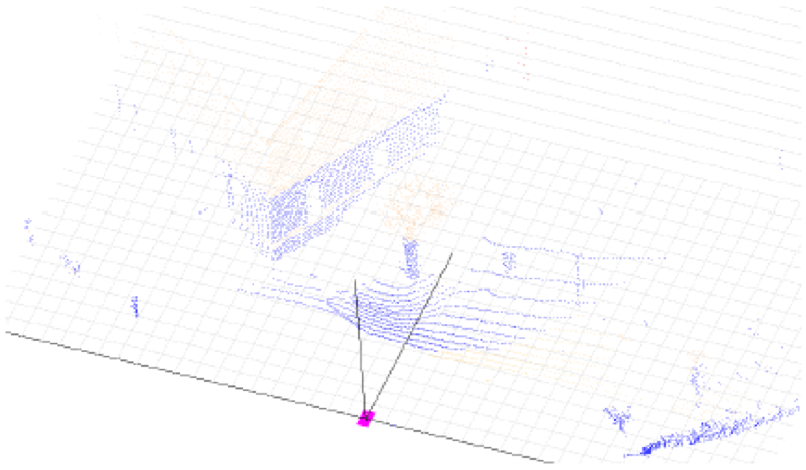


Fig. 6.10. TNM. Elevation analysis (test 1)

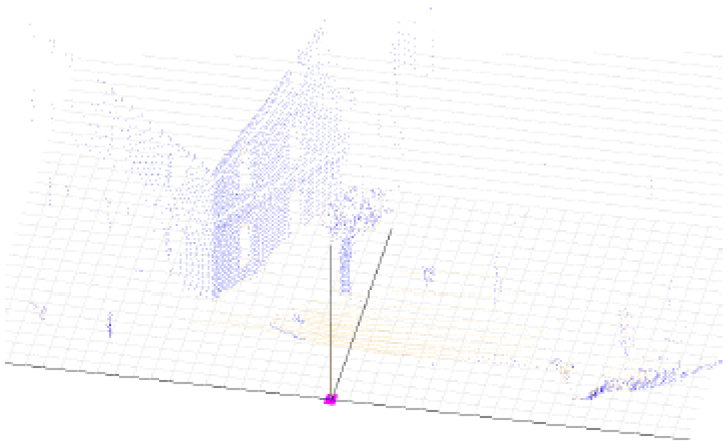


Fig. 6.11. TNM. Slope analysis (test 2)

In these results can be appreciated that the first analysis is not good enough for terrains with slopes. The model constructor identifies as non crossable the positive slope, such as the zones marked with α_1 and α_3 slopes in figure 6.9. Nevertheless, the robot can go through it, as can be demonstrated in the second analysis, where a slope computing has been carried out.

The elevation analysis is useful in flat terrains with different slopes, but this is not enough to define a natural terrain and the method presents some problems with surfaces such as steps (non traversable by a robot but considered as traversable by this method), or little rocks (easy to cross for the robot locomotion system, but considered as obstacle by the method).

For example, in figure 6.12 a complex environment, due to the number of obstacles, is shown. And in figure 6.13 the TNM with test 2 is presented. It can be seen, that the bank has not been detected, due to its horizontal slope.

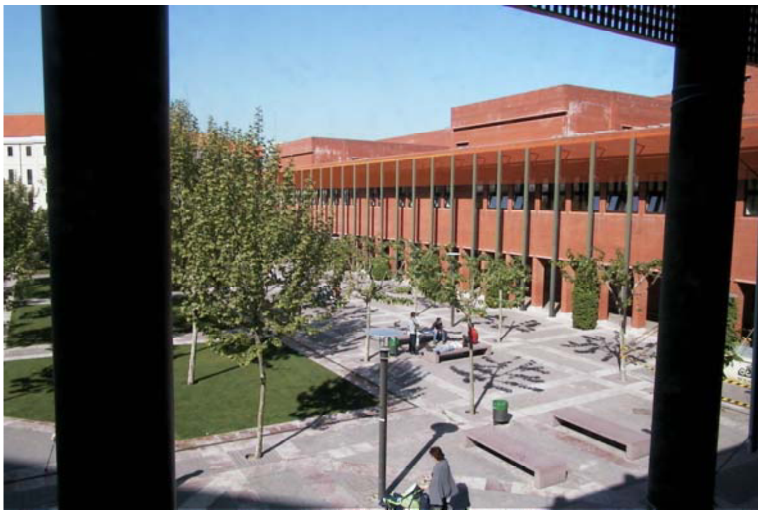


Fig. 6.12. Real complex environment

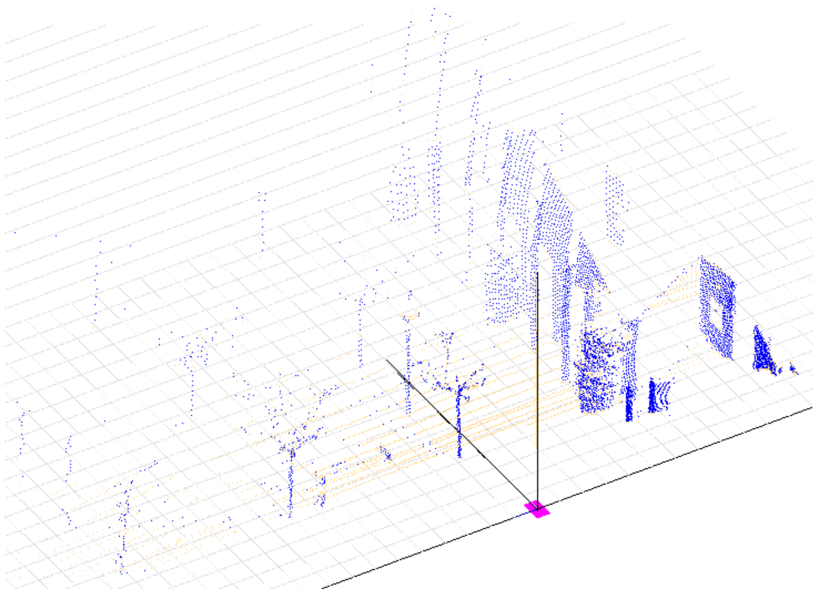


Fig. 6.13. TNM. slope analysis (test 2)

In that case it will be necessary to study another parameter to determine the traversability. In following approximations a terrain roughness analysis is considered as a complement to the current slope analysis. In figure 6.14 a TNM with roughness study is presented to compare with figure 6.13. In this case the obstacle has been detected.

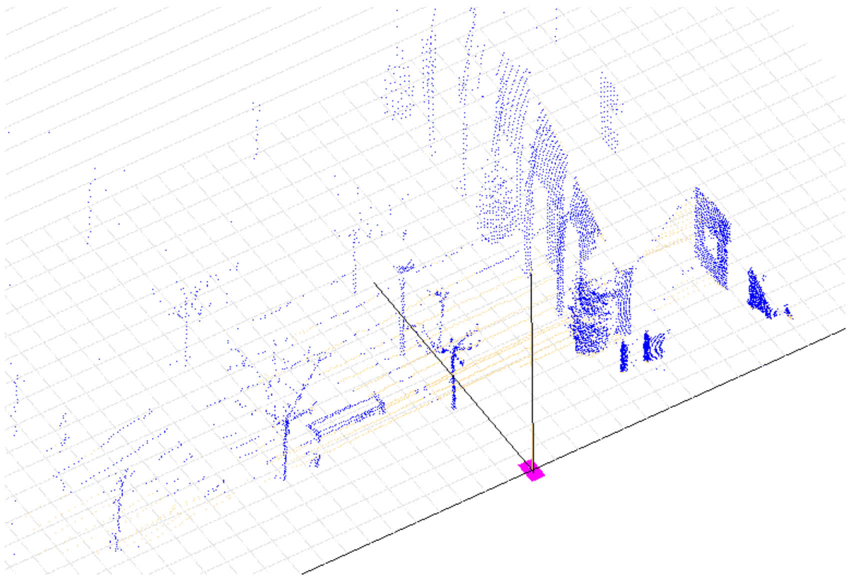


Fig. 6.14. TNM with a roughness analysis

6.3.3 Slope and Roughness Analysis

Most of the definitions found in literature about roughness make reference to irregularity measurements in the terrain surface, and usually it refers to elevation irregularities. Classical techniques to calculate this measurement are: the gradient elevation analysis [39,29] and the dispersion points calculation with respect to the plane [22]. In the topography field, the roughness measurement has been studied in depth. Gadelmawla et al. present a large amount of all possible parameters, which represent the characteristic [21]. Two methodologies for the roughness study are presented: first, a classical method, transforming the information in a digital image and second, a new study based on a topographical parameter.

Elevation Gradient Analysis

The elevations are considered, in this case, in a local level. For each point, the local height gradient is calculated in a neighbourhood, and the result is considered as the roughness parameter. Big differences in the gradient, mean differences in the terrain surface or that an obstacle exists.

Based on the sorted information provided by a scanner laser, the elevation in each point can be treated as an image, and the gradient is calculated with fast and classical vision techniques. The digital image is defined as a two-dimensional matrix I with the elevation information in each element. To obtain the gradient with vision techniques, we apply the digital mask presented in equation 6.11:

$$\begin{pmatrix} 0 & 0 & 0 \\ -1 & 5 & -1 \\ -1 & -1 & -1 \end{pmatrix} \qquad \begin{matrix} - & - & - \\ \otimes & \square & \otimes \\ \otimes & \otimes & \otimes \end{matrix} \tag{6.11}$$

where, only the information between the point and its right and left hand side neighbour, and their neighbours in the previous vertical scan are considered (meant in the measurement sense).

To calculate the TNM two thresholds are considered, which determine, in a roughness study point of view, if the robot can cross the areas. These thresholds depend on the locomotion system, and they are defined as T_{max} and T_{min}, with the same values than those considered in section 6.3.1. Meaning, little differences in elevation between points mean that the robot can go over the object. Big differences elevations means that the robot can go through them (for example in case of aerial obstacles).

So: r_i is defined as the roughness parameter in a point $p_i(x_i, y_i, z_i)$ calculated when the mask presented in equation 6.11 is applied. NTR^{test3} is considered as the subset of I, such as the following traversability property is fulfilled:

$$NTR^{test3} = NTR^{test2}\ AND\ \{p_i(x_i, y_i, z_i) / r_i > T_{min}\ AND\ r_i < T_{max}\} \tag{6.12}$$

with this approach, discontinuities in obstacles can be detected, and aerial obstacles, which are not real obstacles, can be separated.

Slope Gradient Analysis

This method calculates the roughness degree with a non classical method in the robotics field, but widely used in the topography field. The method is based on the normal vector deviation in each point, with the calculus of a statistic called *spherical variance* [35].

In this case, we take advantage of the previous normal vector calculation to evaluate the variation inside a local region.

The spherical variance is obtained from the orientation variation of the normal vector in each point. The study uses the following reasoning:

- *In a uniform terrain (low roughness), the normal vectors in a surface will be approximately parallel and, for this reason, they will present a low dispersion (see figure 6.15 left).*
- *On the other hand, in an uneven terrain, the different changes in the orientation normal vectors will present great dispersion (see figure 6.15 right).*

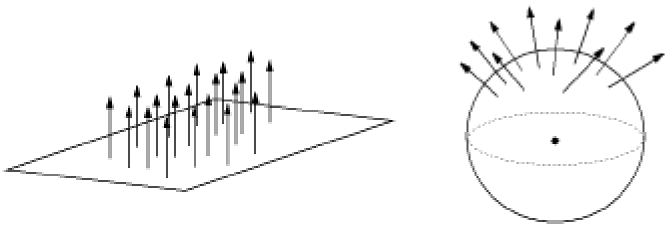


Fig. 6.15. Spherical variance analysis

Definition 3. ***Spherical Variance***

Given a set of vectors $\{\overline{N_i}\}$, corresponding to the normal vectors in a neighbourhood, inside the perceived space, the spherical variance ω_i is defined as the complementary to the normalized mean vectors module.

$$\omega_i = 1 - \frac{R_i}{n} \tag{6.13}$$

Where ω_i is a measure of the vectors dispersion. The vectors are defined by their module and direction, in the 3D space. The method to obtain the parameter is detailed below:

1. Given a set of n normal vectors to a surface, defined by their three components $\overline{N_i} = (x_i, y_i, z_i)$, the module of the sum vector R is calculated in equation 6.14:

$$R = \sqrt{\left(\sum_{i=0}^{n} x_i\right)^2 + \left(\sum_{i=0}^{n} y_i\right)^2 + \left(\sum_{i=0}^{n} z_i\right)^2} \tag{6.14}$$

2. The mean value is normalized by dividing between the number of data n, in this way the result value is between the range $[0,1]$:

$$\frac{R}{n} \in [0,1] \tag{6.15}$$

3. Finally, the complementary of the result is calculated to give sense to the statistic in equation 6.13.

Therefore, the values are standardized and they are distributed in a theoretical range between 0 and 1. When $\omega = 1$ there exists a maximum dispersion that can be considered as the maximum roughness degree, and when $\omega = 0$, a full alignment exists and the terrain will be completely flat.

The spherical variance has not been used before in the robotics field. It is of great interest to consider the normal vector variation as a roughness measurement, instead of the elevation gradient as different authors do [39,16]. A comparative study between the spherical variance and gradient elevation techniques can be seen in [8]. The advantage of our method over the traditional ones can be found, above all, in the 3D points sensed by the scanner laser, where the density information decreases with the distance. An example of this can be clearly found in those environments with negative slope [9], where the scanner does not sense too much information.

Therefore, the 3D traversability model is defined, in equation 6.16, as follows:

$$NTR^{test4} = NTR^{test2} \ AND \ \left\{ p_i(x,y,z) \in TR^{slope} / \omega_i > \omega_{th} \right\} \tag{6.16}$$

The terrain will be considered as non traversable, due to its roughness, when the spherical variance in a point overcomes a threshold (ω_{th}) depending on the robot's locomotion system.

The image 6.16 represents a rough terrain. The robot is able to go through the rough terrain but it is not able to go through the obstacles found on the left hand side of the image.

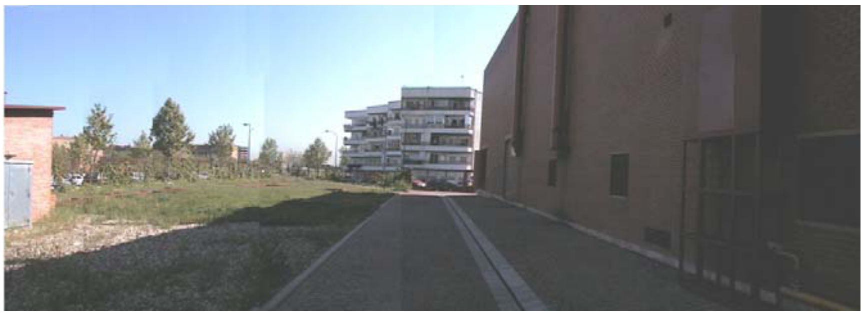


Fig. 6.16. Real environment. roughness terrain

In figures 6.17, 6.18, 6.19 and 6.20 the TNM top view, performed with the four tests are shown.

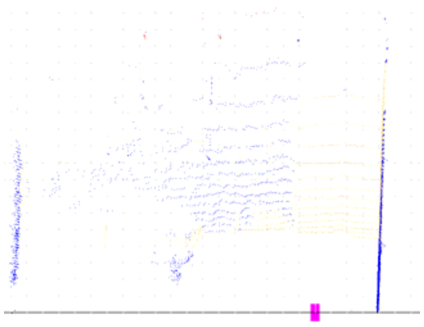


Fig. 6.17. TNM test 1, top view

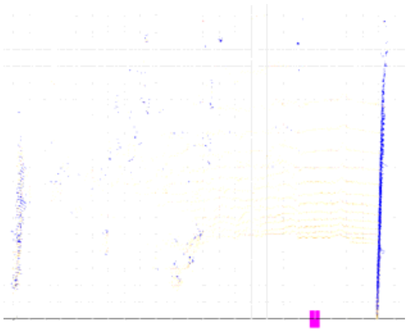


Fig. 6.18. TNM test 2, top view

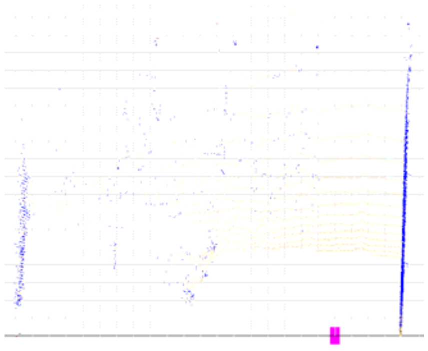


Fig. 6.19. TNM test 1, top view

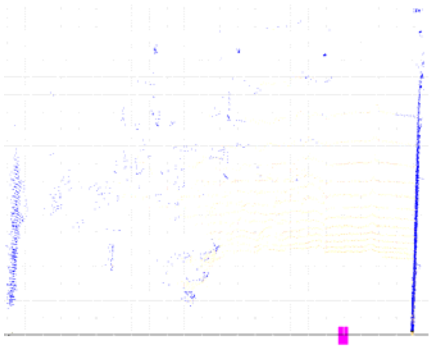


Fig. 6.20. TNM test 1, top view

These results cannot be appreciated well if we don't represent the Traversable Region Models above, where in red the non traversable regions for each test are represented in figures 6.21, 6.22, 6.23 and 6.24.

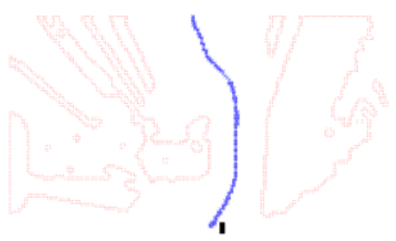


Fig. 6.21. Traversable regions model, test 1

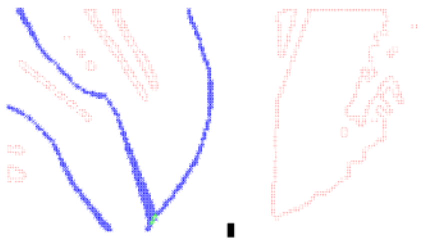


Fig. 6.22. Traversable regions model, test 2

In this figures it can be highlight the different results obtained. Above all, the test 1 results (figure 6.21) where there are obstacles in traversable regions. Best results are obtained in the two last figures (6.23 and 6.24).

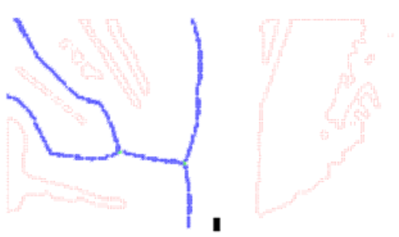


Fig. 6.23. Traversable regions model, test 3

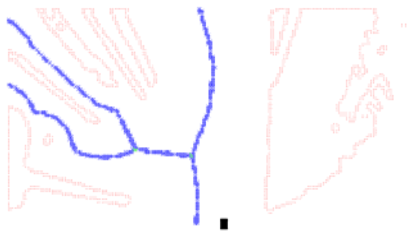


Fig. 6.24. Traversable regions model, test 4

To compare all the methodologies, another experiment is carried out in a terrain with a negative slope, perfectly traversable by the robot (the descent to a garage in figure 25 right hand side), in this case, we can prove, the non viability of the first analysis in outdoor environments.

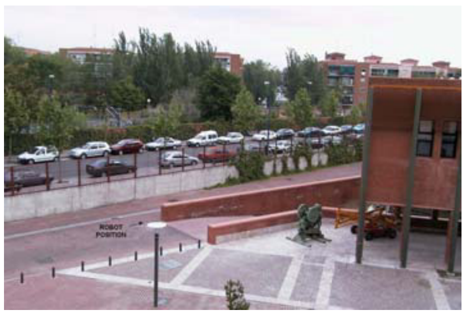


Fig. 6.25. Real environment, negative slope

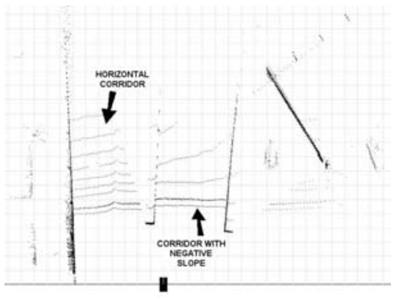


Fig. 6.26. Snapshot, top view

This environment is one of the worst cases because of the sensor system used. As can be seen in figure 6.26, where the top view snapshot representation is shown, the number of scans in the negative slope area is less than the left hand side scans, where a horizontal terrain is. In these cases, the data density is low and furthermore, the model built is less reliable.

The TNMs obtained in this environment are shown in figures 6.27, 6.28, 6.29 and 6.30. In figure 6.27 can clearly be seen how a simple elevation analysis is not able to distinguish a traversable zone. The rest of the tests (2, 3 and 4) are improved with the slope analysis.

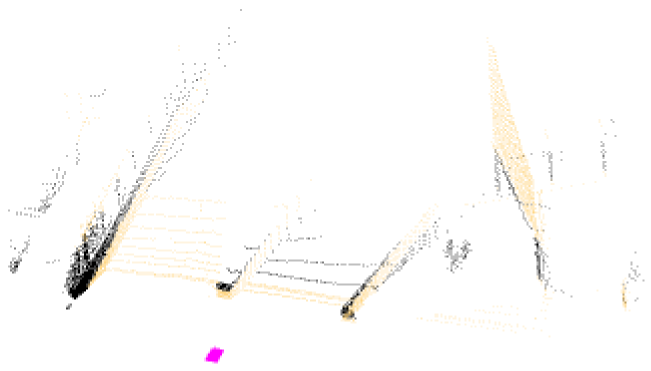


Fig. 6.27. TNM. Elevation analysis (test 1)

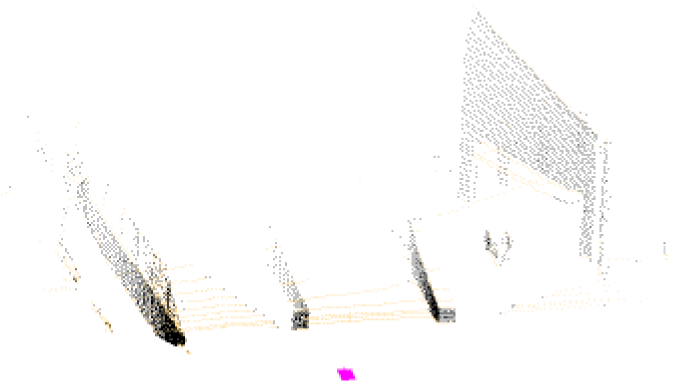


Fig. 6.28. TNM. Slope analysis (test 2)

In figure 6.29 the sensor problem is shown. Despite of detecting the descent as a traversable slope, the roughness analysis detects it as a non traversable area. This is due to the fact that increasing the distance between vertical scans, which obtains a difference in elevation higher than the established threshold.

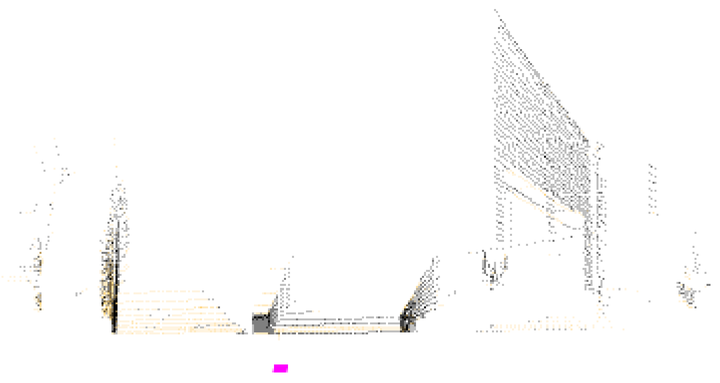


Fig. 6.29. TNM. Elevation gradient analysis (test 3)

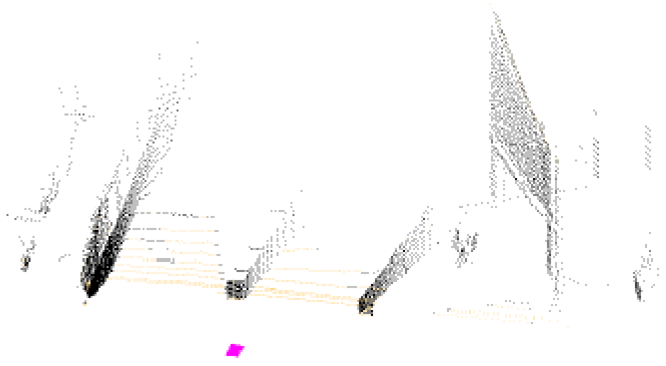


Fig. 6.30. TNM. Slope gradient analysis (test 4)

6.4 Traversable Region Model

When the 3D model is defined (as the one presented in figures 6.31 and 6.32), the free space can be extracted to build a traversable region model for the robot navigation and path-planning. For regions representation, a Voronoi Diagram (VD) technique has been chosen.

The TNM provides geometrical information that can be useful for object modelling or virtual models construction. Besides, the bulky information provided by the sensor system makes the model difficult to manipulate because of the time computing cost. Previous to obtain the traversable region model, 3D information must be reduced and simplified. The information needed for the model constructor is the two-dimensional coordinates of the borders that separate the TA from the NTA. To achieve this goal, digital image processing techniques has been chosen.

In this case, the image is defined as a two-dimensional matrix where each pixel is defined by its traversability characteristic. The 3D image is, in this way, transformed in a binary image of traversable and non traversable regions. The image defines the robot free space.

To achieve the model is necessary to follow these steps:

1. Starting from the TNM, a Digital Elevation Map is obtained by dividing the workspace in cells, and storing in each of them the elevation characteristic (see figure 6.33). This characteristic is defined based on the previously calculated traversability characteristic in each point

and the calculus of the number of obstacles presented in the cell. To obtain the number of objects the *chain map algorithm* is applied.

2. With the DEM, only those cells that the sensor system perceives can be labelled. Then, there are some cells in the discretized environment without information, because the sensor maximum range has been overload or because another cell is occluding it. For this reason, the *visibility map* is built (see figure 6.34). To obtain the information in all the cells, an elevation and the traversability estimation is assigned to the occluded cell. The estimation is performed with the extrapolation of the three-dimensional segment that joins the sensor with the visible cell. The classical digital vision techniques have been implemented to obtain the two maps presented before. These techniques have been chosen for its simplicity and robustness. The DEM allows sorting the 3D information, obtained in the TNM, in cells projected over the XY plane and in each cell only the useful information needed to represent the environment is stored. Its structure, in the matrix form, allows the use of artificial vision algorithms.

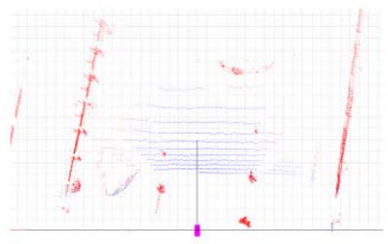


Fig. 6.31. TNM. Top view

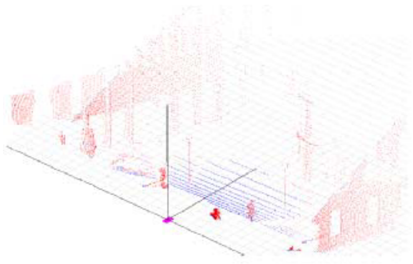


Fig. 6.32. TNM. Side view

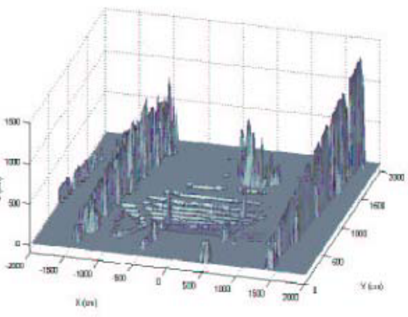


Fig. 6.33. DEM

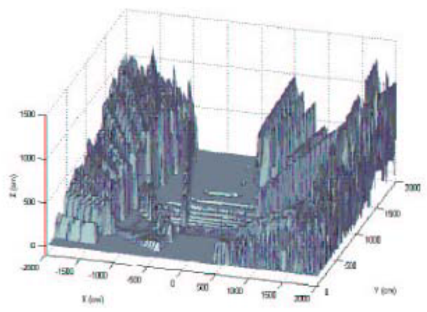


Fig. 6.34. Visibility map

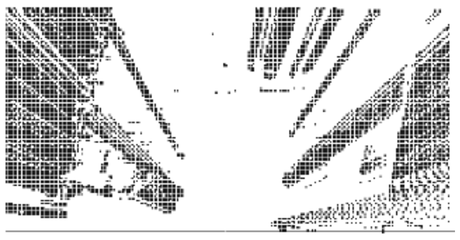


Fig. 6.35. Binary image.

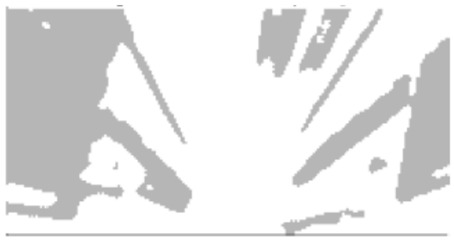


Fig. 6.36. Morphological operation. Closing.

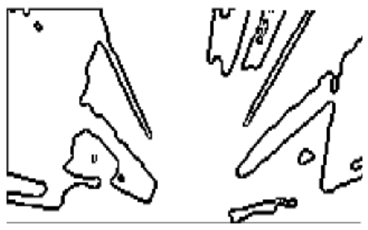


Fig. 6.37. Edge detection

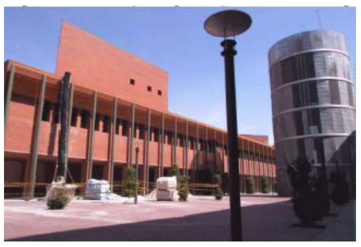


Fig. 6.38. Real image

3. With the workspace divided in cells with the traversability information, the image is transformed in a binary matrix (see figure 6.35), that represents the traversable regions (0) or not traversable regions (1). The image is filtered with the classical morphological operations (closing operation) to smooth the shapes without loosing the geometrical information (see figure 6.36). This pre-processing technique enhances the subsequent edge detection. The boundaries that separate the traversable regions from the non traversable ones are needed to build the traversable region model (TRM). In this case different filters have been tested giving good result the Prewitt, Robert and Isotropic filters. These filters are developed as matrix with dimension 3×3 and a convolution is performed with the binary image (the result is presented in figure 6.37).

In previous steps the information needed to represent the TRM is obtained.

6.4.1 Traversable Region Modelling Algorithm

There exist many methods to obtain a model useful to navigation and motion planning. Among classical methods proposed for two-dimensional space, a roadmap approach has been selected. The roadmap represents the robot's free space in a one-dimensional representation. There are various types of roadmaps. Specifically, to obtain a model from the traversable regions segmentation a Voronoi Diagram (VD) technique has been chosen. This approach is used for the following reasons:

- The model is built based on the sensor information, its construction is fast and the model can be used in real time.
- The VD, by definition, gives safe trajectories that maximize the clearance between the robot and the obstacles. The obtained model is very useful for navigation.
- The environment model is a simplification of the robot's free space or represents a free space scheme. It does not imply memory storing problems.

The Voronoi Diagram is a classical technique that splits up the space into non uniform regions. The more intuitive and simple definition for the Voronoi construction is: given a finite set of distinct, isolated points in the space, all locations in that space are associated with the closest member of the point set [42].

Voronoi Diagrams

The Voronoi Diagram (VD) divides the free space in a set of different regions called *Voronoi regions*; each of one is formed by the set of locations that is placed closer to an object than the rest [13]. This basic concept has been applied to solve problems concerning a variety of disciplines; the robotic is one of them. The Voronoi Diagram can be seen as a type of roadmap approach. This roadmap algorithm is used generally for the topological map construction because it uses regions segmentation.

The use of this technique in the robotic field started at the beginning of the 80's in two-dimensional path planning [41,32,30]. In those approaches a complete knowledge of the workspace's geometry was available; the VD was constructed from a priori established model of the space with polygonal obstacles. Then in the 90's, with the sensor information incorporation, the VD has been used in robot navigation and planning task. In [44], an incremental method based on sensor data is introduced to construct the VD of a terrain whose model is not a priori know, this algorithm is still limited to polygonal objects. The technique presented in [13] to incrementally construct the *hierarchical generalized Voronoi graph* uses local sensor information, without restrictions about to obstacle's shape.

There are different methods to build VD from sensor data. For example, the approximation by points suggested in [48], where each object belonged to the workspace has a defined shape and is replaced by a set of representative points. In [34], objects are represented by set of points that are the measurements from a laser scanner. The use of artificial vision techniques is presented in [47], the objects are replaced as pixels and a skeletization of the free space is performed to obtain the Voronoi Edges. The approaches shown in [13] and [52], are based on the incremental construction of Voronoi Diagrams, moving the robot along the estimated Voronoi edges. Distances to objects in the environments are calculated from the current robot position, in the first case objects are represented by points and by segments, in the second. Blanco in [6] represents the discretization of the free space in cell as an approximation to a digital image to build Local Voronoi Diagrams (LVD). This is the technique chosen to obtain the TRM, but with modifications in the information treatment because the two-dimensional information provided by the laser in [6] is shorted and only the free space is needed to be discretized. In this work all the workspace is discretized because we are starting from 3D information. In the majority of previous works, the robot is supposed to be equipped with a two-dimensional sensor (scanner laser) with a 180º field of vision in horizontal scans, as also in [34].

We take the advantage of the digital image obtained in the previous steps to work with the algorithm proposed, where the input is an image with the borders between the traversable and non traversable zones. The visibility model information determines which cells belong to the free space and which not.

6.4.2 Basic Concepts

VD is based on a simple concept. Given a set of *generators*, [6] different and isolated, in a continuous space, every location in this space is associated to the closest generator in the set. The result is a space partition in a set of regions mutually exclusive except for the limits that we call Voronoi regions. All the locations assigned to each member forms the Voronoi region associated with that generator. The boundaries of the Voronoi regions are the locations assigned to two or more generators.

According to the geometric feature of the generators, ordinary and generalized Voronoi diagrams have been defined.

Definition 4. ***Ordinary Voronoi Diagram***

For a set of n points, $P = \{p_1, p_2, ..., p_n\}$, with $2 \le n < \infty$, in the bi-dimensional Euclidean space, the Voronoi region associated to one point $V(p_i)$ is defined as all those points of bi-dimensional space having p_i as the nearest point in the set P. Thus:

$$V(p_i) = \{p \mid p \in \mathbb{R}^2, d_E(p, p_i) < d_E(p, p_j), j \ne i\} \tag{6.17}$$

Where $d_E(p, p_i)$ denotes the Euclidean distance between the points p and p_i, and the sequence give as:

$$V = \{ V(p_1), V(p_2), \cdots, V(p_n)\} \tag{6.18}$$

will be the VD generated by the set P.

The VD generated by a set of points in the Euclidean space divides the space in connected regions that have only one point closest under any metric (normally the Euclidean distance) [14].

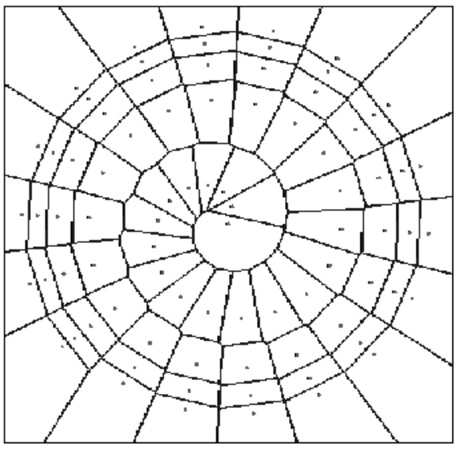


Fig. 6.39. Ordinary Voronoi Diagram

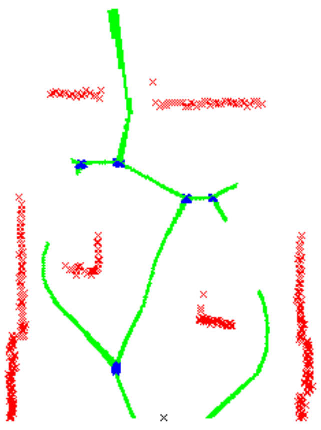


Fig. 6.40. Voronoi Diagram for set of points

The previous abstract definition of the ordinary VD has been generalized to facilitate practical applications. Some of those generalizations have considered different assignment rules that Euclidean distance. Another type of generalization is based on the generators, considering a generator may be a point, a set of points, a line, an area, etc. The specific generalized

Voronoi diagram referred in this work is the VD generated for sets of points, where a generator is defined as a set of points. In this case, the region segmentation based on the minimum Euclidean distance computation between a location and a set of points is called Generalized Voronoi Diagram (GVD).

Definition 5. ***Generalized Voronoi Diagram***

Given $G = \{g_1, g_2, \cdots, g_n\}$ a collection of n point series in the plane which do not overlap

$$g_i \subset \mathbb{R}^2, \quad i = 1, 2, \cdots, n \tag{6.19}$$

$$g_i \bigcap g_j = \varnothing, \quad i \neq j \tag{6.20}$$

for every point $p \in \mathbb{R}^2$, the minimum Euclidean distance from p to another point belonging to the series g_i is called $\mathbf{d_E}(p, g_i)$.

$$\mathbf{d_E}(p, g_i) = \min(d_E(p, p_i), \forall p_i \in g_i) \tag{6.21}$$

The Voronoi region will be defined as:

$$V(g_i) = \{p \mid p \in R^2, \mathbf{d_E}(p, g_i) < \mathbf{d_E}(p, g_j), \; j \neq i\} \tag{6.22}$$

and the given sequence:

$$V = \{V(g_1), V(g_2), \cdots, V(g_n)\} \tag{6.23}$$

will be the generalized Voronoi diagram generated by G.

From now on, the GVD term is used when the generators are series of points instead of isolated points. Those series of points will be defined as *generator group*.

The algorithm we present in this work is based on the GVD and is implemented in a sensor based way because it is defined in terms of a metric function ($\mathbf{d_E}(p, g_i)$), that measure the Euclidean distance to the closest object (g_i), represented as a set of points supplied by a sensor system.

6.4.3 TRM Algorithm

Supposedly the robot is modelled as a point operating in a subset belonging to the two-dimensional Euclidean space (in our particular case, although this is true with n-dimension too). The space W which is called Workspace, is obstacle populated $\{C_1, C_2, ..., C_n\}$ that will be considered as a close set. The set of points where the robot can manoeuvre freely will be called free space and is defined in [5] as:

$$FS = \left\{ W \setminus \bigcup_{i=1}^{i=n} C_i \right\} \tag{6.24}$$

The workspace W is represented as a two-dimensional binary image $B(i, j)$, where each position (i, j) has assigned a field value (0 or 1), that indicates if a pixel belong to a generator (field 0) o not (field 1). For each point belonged to the free space $(i, j) \in FS$ is, at least, one point closest to the occupied space $\overline{FS}$ that will be called *base point* [50].

The LVD is obtained from the distances to the generated points which belong to the objects' borders. To obtain the local model, the algorithm is executed in the three steps presented in next paragraphs.

1. Clustering

Data representing borders between traversable and non-traversable regions are grouped in clusters. The cluster determination, in the three-dimensional information case, is not trivial. In other works, the sorted two-dimensional information, from a scanner laser, is easily separated in clusters, using the distances between successive scanned points [6], [27]. In this approach, the 3D information cannot be treated as other authors do, and a labelling technique is used to obtain the clusters. The kernel applied is a circumference. The radius is the robot's size. If there is a distance greater than the robot's size between two points which belong to the non-traversable region border, then the robot can cross between them, and the points will be considered as belonging to different clusters.

In figure 6.41 the data grouping of the real environment presented in 38 is presented. The result of this step is three generators called A, B and C.

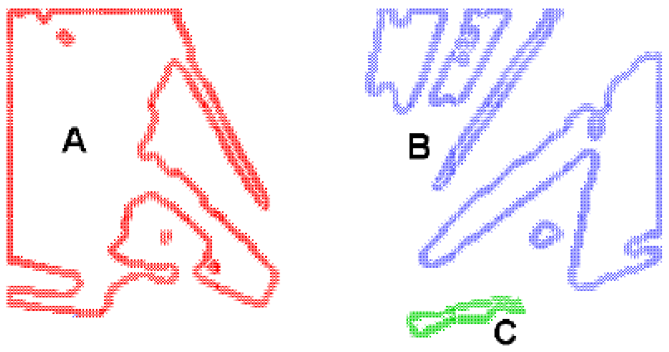


Fig. 6.41. Data grouping to obtain generators

2. Distance to the Generator Elements Computing

For the VD construction, the assignment rule is the Euclidean distance to generators. Generators are the clusters. For each free space cell $(i, j) \in FS$, we mean whose cells where the robot can evolve, the distance between the cell and each point, which belongs to each clusters, is calculated. The minimum Euclidean distance between the cell and the points to one cluster will be considered to assign the cell to the corresponding *Voronoi region*.

3. LVD Algorithm

The label cell is evaluated, based on the minimum distance, as follows (see figure 6.42):

- If the distance to an A cluster is less than the rest, the cell is evaluated as belonging to the *Voronoi region* associated to the cluster A.
- If there are two equidistant clusters in a cell (for example A and B), then the cell is labelled as *Voronoi edge*.
- For bigger equidistant it will be labelled as *Voronoi node*.

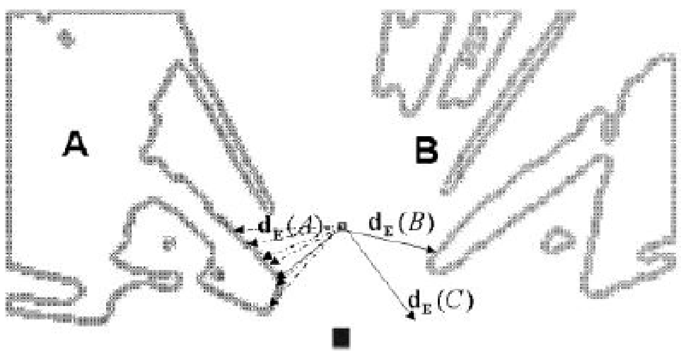


Fig. 6.42. Label cell evaluation

Error Caused by Discretizing

The labelled of each cell belonged to the FS is carried out computing the distance between the centre of the cells. Nevertheless, the real measurement can be located in any position inside the cell. This must be taken into account when the distance to the generators is calculated. The maximum discretized error is calculated, based on figure 6.43:

$$d_{\max} = \sqrt{(x_2' - x_1')^2 + (y_2' - y_1')^2} \tag{6.25}$$

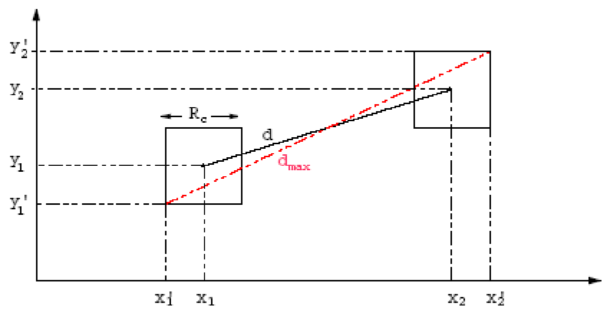


Fig. 6.43. Maximum distance between two cells

In the same figure 43 can be seen that:

$$x_1' = x_1 - \frac{R_c}{2}$$
$$x_2' = x_2 + \frac{R_c}{2}$$
$$y_1' = y_1 - \frac{R_c}{2}$$
$$y_2' = y_2 + \frac{R_c}{2} \tag{6.26}$$

and replacing 6.26 in 6.25 we obtain:

$$d_{\max} = \sqrt{(x_2 - x_1 + R_c)^2 + (y_2 - y_1 + R_c)^2} \tag{6.27}$$

Therefore, $d_{\max}$ is the maximum distance that we must consider when the labelling step is performed. We mean, the maximum error E when two cells are evaluated is:

$$E = d_{\max} - d \tag{6.28}$$

And the LVD generation step is modified as follows:
For each cell (i, j) belonging to the free space:

1. The distance to all the points belonged to each generator is computed.
2. The minimum distance to each generator is obtained. We consider, for example in figure 6.42, the distance difference between two generator A and B as $E_c(A,B) = \mathbf{d_E}(g_A, p) - \mathbf{d_E}(g_B, p)$, where p is the centre of the cell (i, j) with coordinates (x, y), then:

– If $E_c(A,B) > 2 \times E$ for all $A \neq B$, the cell is considered as belonged to the Voronoi region associated to the object A.
– If $E_c(A,B) < 2 \times E$ and $E_c(H,B) > 2 \times E$ for all $H \neq A \neq B$, the cell is considered as Voronoi edge between two objects A and B.
– For bigger equidistant the cell will be considered as Voronoi node.

In figure 6.44 the result of the DVL computing from figure 6.41 is shown. In the figure the Voronoi edges and nodes are highlighted.

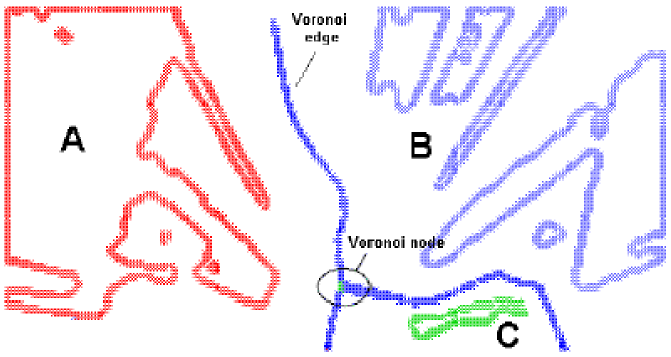


Fig. 6.44. DVL representation

6.4.4 Cell Size Influence

A priori, the cell size has influence on time computing and on the model accuracy. To test the influence, two cell sizes (suitable for the environment, the robot size and the robot speed) have been chosen. The cell sizes are 20 cm and 50 cm. The experimental results have been carried out in an environment presented in figure 6.45 and the TNM in figures 6.46 and 6.47.

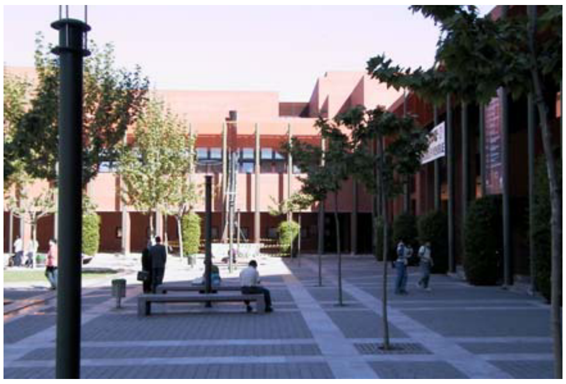


Fig. 6.45. Environment with high obstacles density

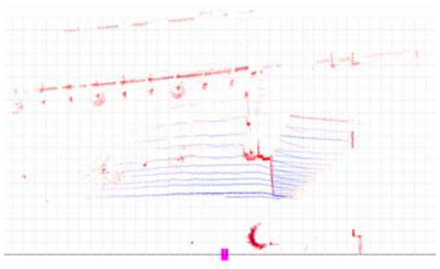


Fig. 6.46. TNM. Top view

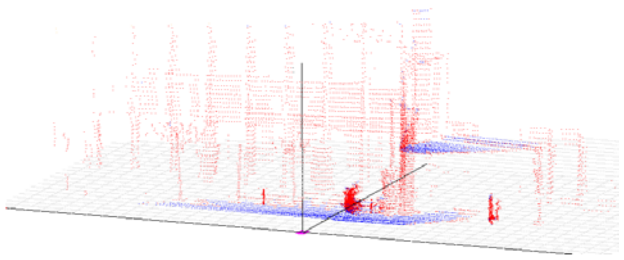


Fig. 6.47. TNM. Side view

The LVD obtained are presented in figures 6.50 where a 20 cm cell resolution is implemented, and 6.51 with a 50 cm resolution. And in figures 6.48 and 6.49 the visibility maps are presented to compare the cell size.

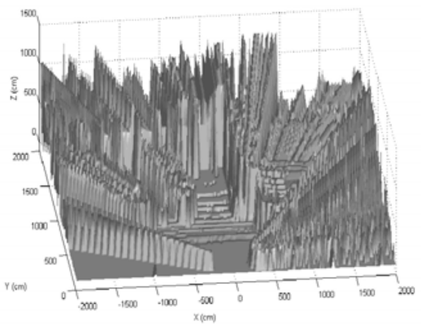


Fig. 6.48. Visibility Map. Cell size of 20 cm

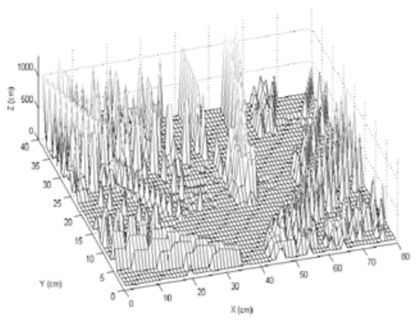


Fig. 6.49. Visibility Map. Cell size of 50 cm

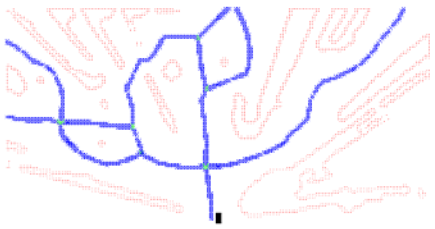


Fig. 6.50. DVL for 20 cm cell size

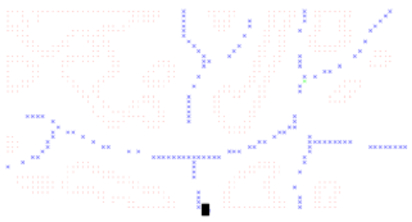


Fig. 6.51. DVL for 50 cm cell size

Apparently, the graphical result does not change, we mean, the LVD form has not been loosen, only the number of points have decreased. In 50 cm cell size the number of nodes decreases and, above all, we notice that the difference between the two resolutions increases with the distance to the sensor system. Of course, the processing time in steps with digital image processing algorithms, is reduced because of the increases in cell size. The image changes the size from 100×200 to 40×80 (see table 6.1).

Tab. 6.1. Different cell size time computing

	t for $R_c = 20$ cm in ms	t for $R_c = 50$ cm in ms
DEM	273.82	706.73
VM	116.72	42.72
CLOS.	127.90	14.18
EDG. d	77.15	19.14
CLUS	(8 clusters)1941.22	(9 clusters) 11.84
LVD	(15 nodes) 2261.88	(1 nodes) 99.34
TOTAL	4798.69	893.22

In table 6.1 the time computing for 50 cm cell size is reduced to the 20% of the time for the 20 cm cell size.

6.5 Incremental Model Construction

In order to explore a large unknown environment, the robot needs to build a global model, to know where it is at each instant, and where it can go. The model can be introduced a priori in the robot or it can be built while the robot is travelling, fussing local models. The model merge consist of assembling a set of data, obtained in different acquisitions in a unique model [17]. In this article, an incremental global map is built merging the different LVD while the robot is moving.

To obtain an incremental model, based on the robot successive perceptions, the following steps must be carried out [38]:

1. Environment information sensing: The robot, stopped, uses its external sensors (in this case a 3D scanner laser and a compass) to acquire all the needed information and build a local model (LVD).
2. The local model (LVD), useful for navigation, is transformed in a global coordinates system, with a differential GPS, and the local model is integrated with the global current model.
3. The robot moves to the next position, by path-planning or guiding, and goes back to the step 1.

The exact global model construction, based on the successive perceptions, is in general difficult to obtain due to the sensor uncertainty. Besides, in the majority of the cases, it is not interesting not only for the imprecision, but also the CPU time computing and the high memory storing makes the model not interesting for real-time objectives.

For the integration, a transformation in the global Cartesian system is needed. A Global Positioning System (GPS) is used to perform the transformation for each LVD point (x_i, y_i). The algorithm is the following:

For the first local map (transformed into global reference system):

1. To obtain a seed point $P(x_{CM}, y_{CM}) \in LVD$, and to set it as Centre of Mass (CM).
2. Calculate the Euclidean distance between all the points $P_i(x_i, y_i) \in LVD$ and the CM. If the distance is less than an error value called $r_{\max}$, the point P_i is included in the list belonging to the CM and a new CM is calculated. If the distance is greater than $r_{\max}$, then a new list is formed and it is set as another CM.

For the rest of the LVD maps (transformed into global reference system):

1. For each point $P_i \in LVD$ we try to find the CM in the global map to which P_i belongs (the Euclidean distance between P_i and CM is less than $r_{\max}$).
2. If we do not find it a new list is formed with $CM = P_i$.

Equation 6.29 calculates the CM for a set of N points. The Euclidean distance is calculated in equation 6.30 to know if a point belongs to the CM.

$$CM = \left(\frac{\sum_1^N x_i}{N}, \frac{\sum_1^N y_i}{N} \right) \tag{6.29}$$

$$r = \sqrt{(x_i - x_{CM})^2 + (y_i - y_{CM})^2} \tag{6.30}$$

The $r_{\max}$ value must be chosen considering the map cell resolution, and it always depends on the robot's velocity movement.

In figures 6.52, 6.53 and 6.54 the steps performed to build the global map are presented.

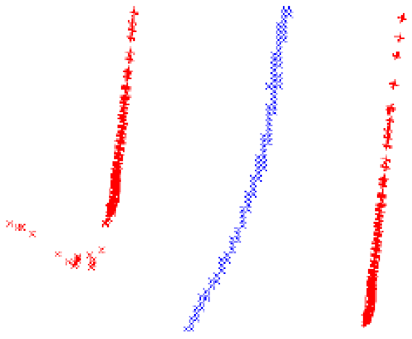


Fig. 6.52. LVD

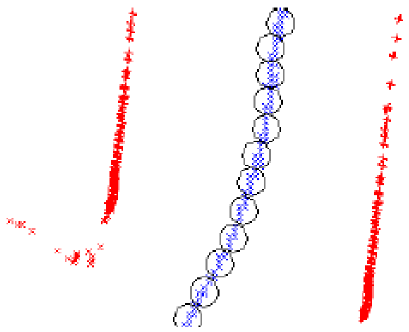


Fig. 6.53. Data map clustering

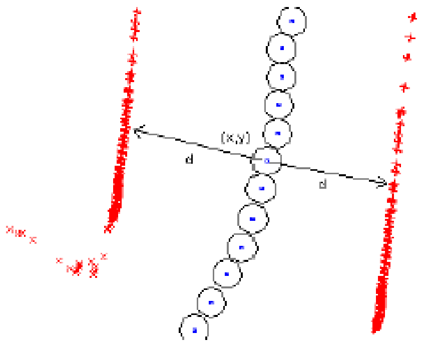


Fig. 6.54. Incremental Voronoi map

In next figures (first, see figure 6.55), an exploration task for an outdoor robot is presented. The robot senses the environment and builds the LVD (figure 6.56), then it moves three meters ahead and repeats the process eleven times, building the incremental model (in figures 6.57 to 6.66).

In the global model built, the good results can be appreciated thanks to the additional information provided by the GPS that supply global measures. The local model integration is easy and allows building topo-geometrical models in real time.

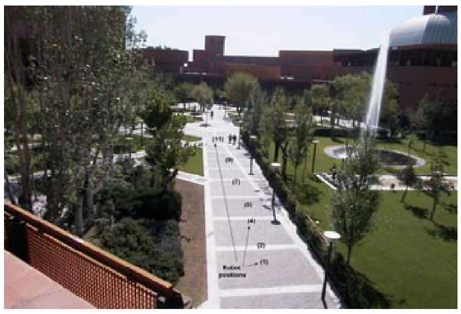


Fig. 6.55. Real environment

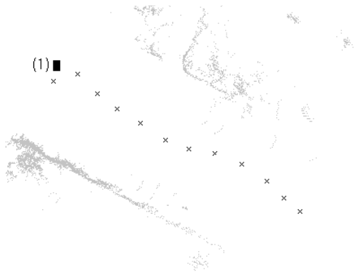


Fig. 6.56. Robot exploration sequence. Position 1

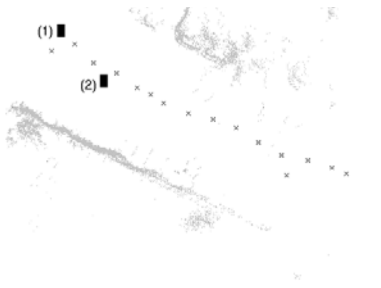


Fig. 6.57. Robot exploration sequence. Position 2

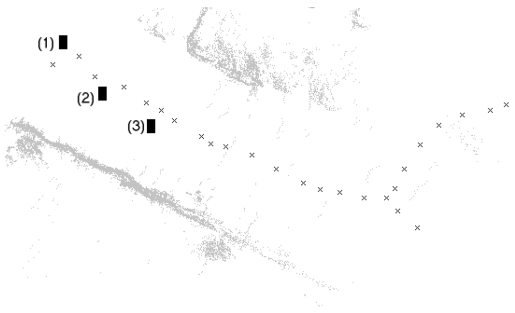


Fig. 6.58. Robot exploration sequence. Position 3

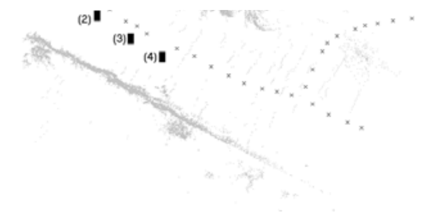


Fig. 6.59. Robot exploration sequence. Position 4

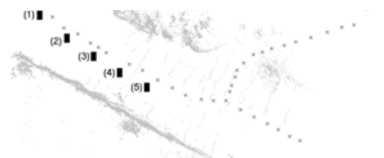


Fig. 6.60. Robot exploration sequence. Position 5

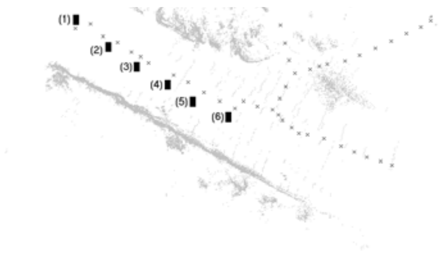


Fig. 6.61. Robot exploration sequence. Position 6

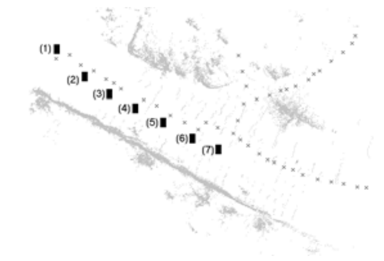


Fig. 6.62. Robot exploration sequence. Position 7

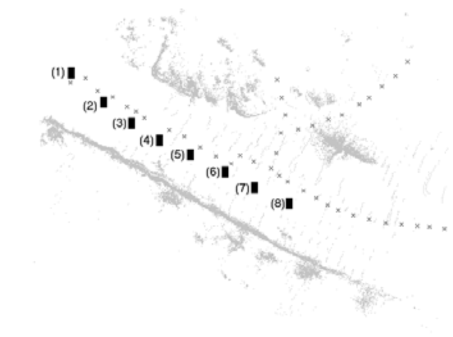


Fig. 6.63. Robot exploration sequence. Position 8

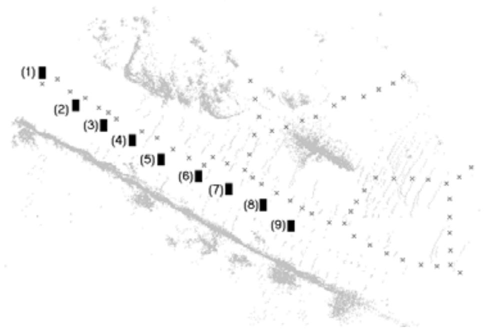


Fig. 6.64. Robot exploration sequence. Position 9

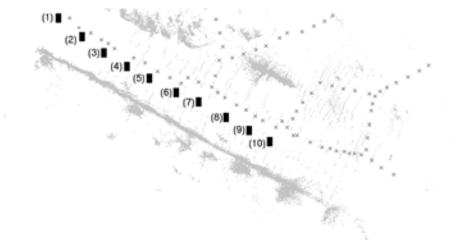


Fig. 6.65. Robot exploration sequence. Position 10

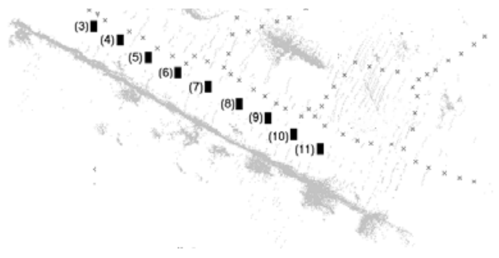


Fig. 6.66. Robot exploration sequence. Position 11

During the sequence, we can see that the robot acquires new local models and regions that in previous maps were considered as non traversable or not visible, change to traversable regions with the increase of new data and the different robot point of view. With this algorithm, the global model can be built off-line, based on different local models or in real time, as the robot is travelling in autonomous or guiding mode.

6.6 Conclusions

A new methodology to model outdoor environments, based on three-dimensional information and a topographical analysis, has been done. The model can be built in real time while the robot is moving and it is possible to carry out local models integration in order to build an environment's knowledge data base in a global map, using a GPS system onboard the robot as work [9] presents. The computing time is low, taking into account the cell size used (20 cm), that is, the minimum size considered for an outdoor environment and for a long size robot. The computing time decreases in 80% for a 50 cm cell size.

As conclusions based on the experimental results presented in this paper, we can highlight the followings:

- The 3D scanner laser is a good choice, to obtain information to model environments with a certain degree of complexity. It senses with precision the terrain surface, and obtains a 3D dense map.
- Nevertheless, the sensor system presents problems in negative sloped terrains, where the information density is fewer when the distance between the measurements and the sensor increases. This is because of the scanner laser non linearity, when a vertical scan with constant increment is done. Different solutions can be set out to solve this problem, such as: the local model size reduction or the sensor placing (in a

more elevated position) and the use of a non constant increment in the vertical scans, to obtain more information in the slope area.

- The cell size used in the model discretization has influence on time computing. For a same size environment, the increase in the resolution cell will decrease the number of cells presents on the map and then, all the digital image algorithms will reduce the process time. Based on the experimental results, we have conclude that, for the robot size and the environment type we are going to work, a good size cell will be between 20 and 50 cm.
- The DVL process time, increases when the number of free space cells increases and when the number of points belonged to the generators increases.

6.7 Acknowledgements

The authors gratefully acknowledge the funds provided by the Spanish Government through the CICYT Project TAP 1997-0296 and the DPYT project DPI 2000-0425. Further, we thank Angela Nombela for her assistance.

References

1 Howard A. and Seraji H., *Vision-based terrain characterization and traversability assessment*, Journal of Robotic Systems 18 (2001), no. 10, 577–587.

2 S. Betgé-Brezetz, R. Chatila, and M. Devi, *Natural scene understanding for mobile robot navigation*, IEEE International Conference on Robotics and Automation, vol. 1, 1994, pp. 730–6.

3 Stéphane Betgé-Brezetz, *Modélisation incrémentale et localisation pour la navigation d'un robot mobile autonome en environnement naturel*, Ph.D. thesis, Laboratoir d'analyse et d'architecture des systèmes du CNRS. Université Paul Sabatier de Touluse, février 1996.

4 D. Blanco, B.L. Boada, C. Castejón, C. Balaguer, and L.E. Moreno, *Sensor-based path planning for a mobile manipulator guided by the humans*, The 11th international conference on Advanced Robotics, ICAR 2003, vol. 1, 2003.

5 D. Blanco, B.L. Boada, and L. Moreno, *Localization by voronoi diagrams correlation*, IEEE International Conference on Robotics and Automation, vol. 4, 2001, pp. 4232–7.

6 D. Blanco, B.L. Boada, L. Moreno, and M.A. Salichs, *Local mapping from on-line laser voronoi extraction*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000.

7 B.L.Boada, D. Blanco, and L. Moreno, *Symbolic place recognition in voronoi-based maps by using hidden markov models*, Journal of Intelligent and Robotic Systems (2004), to be published.

8 C. Castejón, D. Blanco, B.L. Boada, and L. Moreno, *Traversability analysis technics in outdoor environments: a comparative study*, The 11th international conference on Advanced Robotics, ICAR 2003.

9 C. Castejón, B. L. Boada, and L. Moreno, *Topographical analysis for voronoi-based modelling*, The 28th Annual Conference of the IEEE Industrial Electronics Society, 2002.

10 C. Castejón, L. Moreno, and M.A. Salichs, *Traversability modelling in 3d environments*, 3rd International Conference on Field and Service Robotics FSR2001 (Finland), June 2001.

11 Cristina Castejón, *Modelado de zonas cruzables en entornos exteriores para robots móviles*, Ph.D. thesis, Universidad Carlos III de Madrid, Julio 2002.

12 Kuang-Hsiung Chen and Wen-Hsiang Tsai, *Vision-based obstacle detection and avoidance for autonomous land vehicle navigation in outdoor roads*, Automation in construction 10 (2000), 1–25.

13 H. Choset, *Sensor based motion planning: The hierarchical generalizerd voronoi graph*, Ph.D. thesis, California Institute of Technology, Pasadena, California, March 1996.

14 H. Choset, I. Konukseven, and A. Rizzi, *Sensor based planning: a control law for generating the generalized voronoi graph*, 8th International Conference on Advanced Robotics. (ICAR'97) (New York, NY, USA), 1997, pp. 333–338.

15 H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, *Sensor-based exploration: Incremental construction of the hierarchical generalized voronoi graph*, International Journal of Robotics Research 19 (2000), no. 2, 126–148.

16 Langer D., Rosenblatt J.K., and Hebert M., *A behavior-based system for off-road navigation*, IEEE Trans. Robotics and Automation 10 (1994), no. 6, 776–782.

17 A.J. Davison and N. Kita, *Sequential localisation and map-building for real time computer vision and robotics*, Robotics and Autonomous Systems 36 (2001), 171–183.

18 Guilherme N. DeSouza and Avinash C. Kak, *Vision for mobile robot navigation: a survey*, IEEE Transactions on patter analysis and machine 24 (2002), no. 2, 237–267.

19 A. Elfes, *Occupancy grids: a stochastic spatial representation for active robot perception*, Proceedings of the sixth conference on unvertaintu in AI, Morgan Kaufmann Publishers, Inc, 1990.

20 V. Fernández, C. Balaguer, D. Blanco, and M.A. Salichs, *Active Human-Mobile Manipulator Cooperation Through Intention Recognition*, Proceedings of the IEEE International Conference on Robotics and Automation (Seoul, Korea), 2001, pp. 2668–2673.

21 E.S. Gadelmawla, M.M. Koura, T.M.A. Maksoud, I.M. Elewa, and H.H. Soliman, *Roughness parameters*, Journal of Materials Processing Technology 123 (2002), no. 1, 133–45.

22 D.B. Gennery, *Traversabilty analysis and path planing for a planetary rover*, Autonomous Robots 6 (1999), 131–146.

23 Seraji H., *Fuzzy traversability index: A new concept for terrain-based navigation*, Journal of Robotic Systems 17 (2000), no. 2, 75–91.

24 D. Hähnel, W. Burgard, and S. Thrun, *Learning compact 3d models of indoor and outdoor environments with a mobile robot*, Robotics and Autonomous Systems 44 (2002), no. 1, 15–27.

25 A. Howard and H. Seraji, *Real-time assesment of terrain traversability for autonomous rover navigation*, Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000.

26 In S. Kweon and T. Kanade, *High resolution terrain map from multiple sensor data*, IEEE International Workshop on intelligent robots and systems, 1990.

27 Y.D. Kwon and J.S. Lee, *A stochastic map building method for mobile robot using 2-d laser range finder.*, Autonomous Robots 7 (1999), 187–200.

28 S. Lacroix, I.K. Jung, and A. Mallet, *Digital elevation map building from low altitude stereo imagery*, Robotics and Autonomous systems 41 (2002), 119–127.

29 D. Langer, J.K. Rosenblatt, and M. Hebert, *An integrated system for autonomous off-road navigation*, IEEE International Conference on Robotics and Automation, vol. 1, 1994, pp. 414–19.

30 Jean-Claude Latombe, *Robot motion planning*, Kluwer academic publishers, Boston/Dordrecht/London, 1991.

31 Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun, *Using EM to learn 3D models of indoor environments with mobile robots*, International Conference on Machine learning, June 2001.

32 T. Lozano-Pérez and M.A. Wesley, *An algorithm for planning collision-free paths among polyhedral obstacles*, Comunications of the ACM 22 (1979), no. 10, 560–570.

33 M. Macri, Suvranu De, and M. S. Shepard, *Hierarchical tree-based discretization for the method of finite spheres*, Computers & Structures 81 (2003), 789–803.

34 R. Mahkovic and T. Slivnik, *Generalized local voronoi diagram of visible region*.

35 K.V. Mardia and P.E. Jupp, *Directional statistics*, Wiley Series in Probability and Statistics, 1999.

36 K. Mayora, I. Moreno, and G. Obieta, *Perception system for navigation in a 3d outdoor environment*, (1998).

37 R. Murrieta, C. Parra, and M. Devy, *Visual navigation in natural environments: from range and color data to a landmark-based model*, Autonomous robots 13 (2002), 143–168.

38 K. Nagatani and H. Choset, *Toward robust sensor based exploration by constructing reduced generalized voronoi graph*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999, pp. 1678–1698.

39 F. Nashashibi, M. Devy, and P. Fillatreau, *Indoor scene terrain modeling using multiple range images for autonomous mobile robots*, IEEE International Conference on Robotics And Automation, vol. 1, 1992, pp. 40–6.

40 U. Nehmzow and C. Owen, *Robot navigation in the real world: experiments with manchester's fortytwo in unmodified, large environments*, Robotics and Autonomous Systems 33 (2000), 223–242.

41 C. Ó'Dúnlaing and C. K. Yap, *A "retraction" method for planning the motion of a disk*, Algorithmica 6 (1985), no. 53, 104–111.

42 A. Okabe, B. Boots, and K. Sugihara, *Spatial tessellations concepts and applications of voronoi diagrams*, John Wiley and Sons, 1992.

43 P. Ranganathan, J. B. Hayet, M. Devy, S. Hutchinson, and F. Lerasle, *Topological navigation and qualitative localization for indoor environment using multi-sensory perception*, Robotics and Autonomous systems 41 (2002), 137–144.

44 N.S.V. Rao, N. Stoltzfus, and S.S. Iyengar, *A "retraction" method for learned navigation in unknown terrains for a circular robot.*, IEEE Transactions on Robotics and Automation 7 (1991), no. 5, 699–707.

45 H. Seraji, *Traversability index: a new concept for planetary rovers*, Proceedings of the 1999 IEEE International Conference on Robotics & Automation, 1999.

46 S. Simhon and G. Dudeck, *A global topological map formed by local metric maps*, International Conference on Intelligent robots and Systems (Victoria, Canada), 1998.

47 N. Sudha, S. Nandi, and K. Sridharan, *A parallel algorithm to construct voronoi diagram and its VLSI architecture*, Proceedings of the 1999 IEEE Conference on Robotics and Automation, 1999, pp. 1683–1688.

48 K. Sugihara, *Approximation of generalized voronoi diagrams by ordinary voronoi diagrams*, CVGIP: Graphical Models and Image Processing 55 (1993), no. 6, 522–531.

49 S. Thrun, *Learning metric-topological maps for indoor mobile robot navigation*, Artificial Intelligence 99 (1998), no. 1, 21–71.

50 S. Thrun and A. Bücken, *Integrating grid-based and topological maps for mobile robot navigation*, Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, vol. 2, 1996, pp. 944–50.

51 A. Yahja, S. Singh, and A. Stentz, *An efficient on-line path planner for outdoor mobile robots*, Robotics and autonomous systems 32 (2000), 129–143.

52 D. Van Zwynsvoorde, T. Simeon, and R. Alami, *Incremental topological modeling using local voronoi-like graphs*, EEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, 2000, pp. 897–902.

7 Using Visual Features for Building and Localizing within Topological Maps of Indoor Environments

Paul E. Rybski[1], Franziska Zacharias[2], Maria Gini[1], Nikolaos Papanikolopoulos[1]

1. University of Minnesota, Minneapolis, MN 55455, USA
{rybski,gini,npapas}@cs.umn.edu

2. Universität Karlsruhe, Germany
FranziskaZacharias@gmx.de

Abstract

This paper addresses the problem of localization and map construction by a mobile robot in an indoor environment using only visual sensor information. Instead of trying to build high-fidelity geometric maps, we focus on constructing topological maps because they is less sensitive to poor odometry estimates and position errors. We propose a method for incrementally building topological maps for a robot which uses a panoramic camera to obtain images at various locations along its path and uses the features it tracks in the images to update the its position and the map structure. The method is very general and does not require the environment to have uniquely distinctive features. We analyze feature-based localization strategies and present experimental results in an indoor environment.

7.1 Introduction

We are interested in building maps of indoor environments using small robots that have limited sensing. Since the robot must physically carry any sensors that it will use, laser range finders or stereo camera systems cannot be used. Cameras with omnidirectional lenses are better suited in terms of size, but do not provide the same amount of information about the environment. In addition, small robots typically have extremely poor odometry. Slight differences in the speeds of the wheels and small debris or irregularities on the ground will degrade the performance of any dead-reckoning position estimate and make accurate localization or mapping very difficult.

Any method for map construction must take into account the large amount of error in the robot's sensing and odometric capabilities. We propose the construction of a topological map as a graph where each node represents a location the robot visited and took a sensor reading of its surroundings. Initially, the map will contain a node for each sensor snapshot that the robot acquires. Thus, if the robot has traversed the same location more than once, there will be multiple nodes in the map for a single location. These nodes will have to be identified and combined in order to generate a map which correctly matches the topology of the environment.

In this paper we present a method for building such topological maps using monocular panoramic images of the robot's surroundings as sensor data. We take a purely qualitative approach to landmarks by which a location "signature" is used to match robot poses. In this approach, landmarks correspond to sensor readings taken at various (x, y) positions along the robot's path. The specifics of the sensor modality are not important as long as the derived signature can be compared against another sensor signature to determine whether the robot has visited that location before.

For the specific implementation of this algorithm in this paper, we use two different kinds of information extracted from camera images as features. The first kind of features are extracted using the *Kanade-Lucas-Tomasi* (KLT) feature tracking algorithm [18, 27] that automatically extracts and matches visual features from the images. The second kind make use of 3D color histograms. Specific details of the features are described later in Section 7.4.1.

Section 7.3 describes the proposed method, explaining how to model the map as a physics-based mass and spring system. Linear distances between each of the nodes are represented as linear springs while rotational differences between nodes are represented as torsional springs. The spring constants capture the certainty in the odometry estimates. Stiff springs represent high measurement certainty while loose springs represent low certainty. To identify nodes that correspond to the same physical location, we use Markov localization [8] to determine the probability of the robot's position at each timestep. When a pair of nodes in the map is merged, the graph finds a stable energy configuration so that each of the local displacements between the nodes is maintained properly. As individual nodes are merged, the structure of the map changes and the relative distances and headings between the nodes are affected.

In Section 7.4 we report experimental results obtained with a mobile robot in an indoor office environment and we measure the quality of the results in image-based localization and mapping experiments.

7.2 Related Work

Physics-based models that involve spring dynamics have been used quite effectively to find minimum energy states [6, 10]. The work most similar to ours is by Andrew Howard *et al.* [11] where spring models are used to localize mobile robots equipped with laser range finders. All of the landmarks used in their work are unique, and precise distances to objects are identified using the range finders. In contrast, we only assume we have bearing readings to landmarks and that the landmarks may not be distinguishable. Other maximum-likelihood based methods such as Konolige [14], Folkesson and Christensen [7], and Lu and Milios [17] describe how to minimize an energy function when registering laser scan matches. Our work differs from this in that we linearize the energy function. While this simplification may not generate an optimal solution, the method is not likely to be affected by local minima in the energy space during relaxation.

Sim and Dudek [22] describe a visual localization and mapping algorithm which uses visual features to estimate the sensor readings from novel positions in the environment. In practice, our vision system could be replaced by any other kind of boolean sensor modality which can report whether the robot has re-visited a location.

In [29], a map is learned ahead of time by representing each image by its principal components extracted with Principal Component Analysis (PCA). Brian Pinette [19] described an image-based homing scheme for navigating a robot using panoramic images. Kröse *et al.* [15] and Artač [3] built a probabilistic model for appearance-based robot localization using features obtained by PCA. In [28], a series of images from an omnicamera is used to construct a topological map of an environment. Kuipers [16] learns to recognize places by clustering sensory images obtained with a laser range finder, associating them with distinctive states, disambiguating distinctive states in the topological map, and learning a direct association from sensory data to distinctive states. A color "signature" of the environment is calculated using color histograms. Color information, which is provided by most standard cameras, is receiving increasing attention. Swain and Ballard [24] address the problem of identifying and locating an object represented by color histograms in an image. Cornelissen *et al.* [4] apply these methods to indoor robot localization and use color histograms to model predefined landmarks.

We use the KLT algorithm to identify and track features. Lucas and Kanade [18] proposed a registration algorithm that makes it possible to find the best match between two images. Tomasi and Kanade [27] proposed a feature selection rule which is optimal for the associated tracker under pure translation between subsequent images. We use an implementation of these feature selection and tracking algorithms to detect features in the environment [13]. Similarly, Hagen [25] has described a method by which a local appearance model based on KLT features were combined with a local homing technique to generate a pose-free mapping system. This method differs from ours in that we are primarily interested in recovering the robot's pose from its environmental exploration.

7.3 Localization and Map Construction

We are interested in constructing a spatial representation from a set of observations that is topologically consistent with the positions in the environment where those observations were made. The goal is to reduce the number of nodes in the map such that only one node exists for each location the robot visited and where it took an image.

More formally, let D be the set of all unique locations (d_i) the robot visited. Let S be the set of all sensor readings that are obtained by the robot at those positions. Each $s_i^t \in S$ represents a single sensor reading taken at a particular location d_i at time t. If the robot never traveled to the same location twice, then $|D| = |S|$ (the cardinality of the sets is the same). However, if the robot visits a particular location d_i more than once, then $|D| < |S|$ because multiple sensor readings ($s_i^{t_m}$,$s_i^{t_n}$,...) were taken at that location. The problem then is to determine from the sensor readings and the sense of self-motion which locations in D are the same. Once identified, these locations are merged in order to create a more accurate map.

When using small, resource-limited robots, there are several assumptions about the hardware and the environment that must be made. First, we assume that the robot will operate in an indoor environment where it only has to keep track of its 2D position and orientation. This is primarily a time-saving assumption which is valid because (for the most part) very small robots can only be used on flat surfaces. Second, we assume that the robot is capable of sensing the bearings of landmarks around it. This is a valid assumption even for small robots because the cameras and omnidirectional mirrors can be made quite small [5]. Third, we assume that the robot has no initial map of its environment and that we make no assumptions on the mechanism by which it explores its environment (it might be randomly wandering in an autonomous fashion, or it might be completely teleoperated) [23]. As the robot moves, it keeps track of its rotational and translational displacements. Finally, we assume that the robot moves in a simplified "radial" [9] fashion where pure rotations are followed by straight-line translations. This is not an accurate representation of the robot's motion because the robot will encounter rotational motion while translating, however in practice we have found that we can discount this for small linear motions.

7.3.1 Spring-Based Modeling of Robot Motion

Following each motion, a reading from the robots sensors is obtained. This sequence of motions and sensor observations can be represented as a graph where each node initially has at most two edges attached to it, forming a single chain (or a tree with no branches). The edges represent the translational and the rotational displacement. This can be visualized using the analogy of a physics-based model consisting of masses and springs. In this model, translational displacements in the robot's position can be represented as linear springs and rotational displacements can be represented as torsional springs. The uncertainty in the robot's positional measurements can be represented as the spring constants. For example, if the robot were equipped with high precision odometry sensors, the stiffness in the springs would be very high.

By representing the locations as masses and the distances between those locations as springs, a formulation for how well the model corresponds to the data can be expressed as the potential energy of the system. The *Maximum-Likelihood Estimate* (MLE) of the set of all sensor readings S given the model of the environment M can be expressed as $P(S|M) = \prod_{s \in S} P(s|M)$.

By taking the negative log likelihood of the measurements, the problem goes from trying to maximize a function to minimizing one. Additionally, by expressing the allowable compressions of the spring as a normal probability distribution (i.e., the probability is maximized when the spring is at its resting state), the log likelihood of the analytical expression for a Gaussian distribution is the same as the potential energy equation for a spring, or $-log(P(s|M)) = \frac{1}{2}(e - \hat{e})^2 k$.

In this formulation, e is the current elongation of the spring, $\hat{e}$ is the relaxation length of the spring and k is the spring constant. In order to minimize the energy in the system, direct numerical simulation based on the equations of motion can be employed. Figure 1 shows a simple example of how the linear and torsional springs are used to represent the difference between the current model and the robot's sensor measurements.

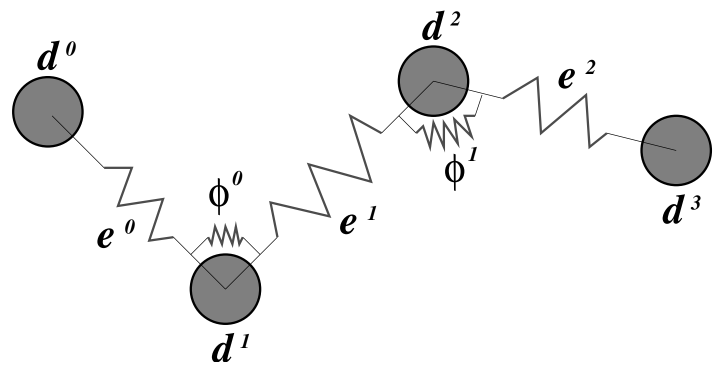


Fig. 7.1. Examples of relative poses of the robot connected by linear and torsional springs. Locations of sensor readings, lengths of linear robot translation, and angles of robot rotation are represented as d^i, e^j, and ϕ^k, respectively.

When the sensor readings of two nodes are similar enough to be classified as a single node, the algorithm will attempt to merge them into a single location. This merge will increase the complexity of the graph by increasing the number of edges attached to each node. This merge will also apply additional tension to all of the other springs, and the structure will converge to a new equilibrium point.

If the landmarks observed at each location are unique, such as in the work of Howard *et al.* [11], then the task of matching two nodes which represent the same locations is fairly straightforward. However, in real world situations and environments, this is extremely unlikely to occur. Without pre-marking the environment and/or without extremely good *a priori* information, a robot cannot assume to be able to uniquely identify each location. This requires the robot to use additional means for determining its most likely location given its current sensor readings and knowledge of its past explorations encoded in the topological map.

7.3.2 Linear vs. Torsional Springs

Since the linear and torsional springs are separate, their error measurements must be considered individually. The importance of the two kinds of springs should also be considered separately. Several simulation experiments were performed to analyze the relative importance of the linear and torsional spring strengths. A set of simple three-node paths were generated such that the robot returned to the starting point after tracing out a regular polygon. The linear and rotational odometry estimates were corrupted by Gaussian random noise with variance ranging from 0 to 1.0. The constants for the linear and torsional springs were set to be the inverse of the noise. Thus, in these experiments, the assumption was made that the robots had a good estimate for the amount of error in both cases.

Figure 2 illustrates the process with two different variances. In this figure, the initial true path of the robot is described as a regular polygon where the first and last node close the polygon. The odometric estimates are corrupted by Gaussian noise. The first and last nodes are attached (merged) and the whole spring model is allowed to relax. Finally, a transformation is found which minimizes the total distance between the corresponding points in each dataset. This removes errors based on global misalignments and only illustrates the relative errors in displacement between the points in space. As can be seen, the distortion of 0.7 variance Gaussian noise in both linear and torsional springs produces a relaxed path that is very different from the true path and thus has a very low sum of squared difference match.

The results for the three-node experiment can be seen in Figure 3(a). A similar experiment was run for four- and five-node paths. The resulting curves are extremely similar to the shown three-node path. The results indi-

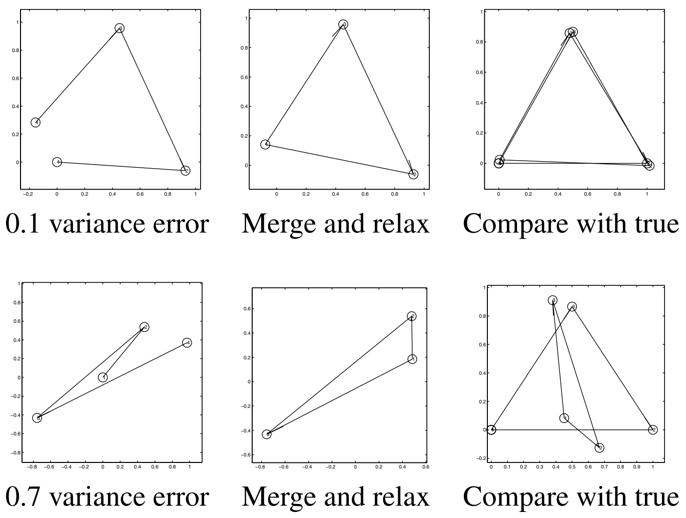


Fig. 7.2. Linear vs torsional constant comparison experiment. A three-node circular path (triangle) has its linear and rotational components corrupted by noise. The start and endpoints are merged (as they are the same location) and the model is allowed to relax. Two sample variances, 0.1 and 0.7, are shown

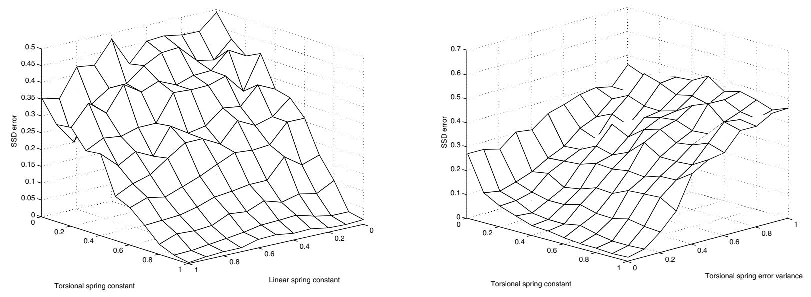
cate that the torsional spring constant is far more important than the linear spring constant. As long as the torsional spring constant is strong (and thus has a correspondingly low error estimate), the linear spring constant can be very weak (with a correspondingly high error estimate), and the final model will still converge to a shape that is very similar to the original path.

7.3.3 Torsional Constants vs. Error

The relative strengths of the spring constants must reflect the certainty of the robot's sensor measurements. The more certain the robot is of its sensor readings, the stronger the spring constants should be. This adds rigidity to the structure of the map and limits the possible distortions and displacements that could occur.

If the torsional error estimates are very high, then it does not matter how strong the spring constants are. Very large rotational errors introduce too much distortion into the map to be corrected by correspondingly strong spring constants. Thus, it is vital that the robot's rotational estimate errors be low.

Figure 3(b) illustrates the results from this experiment. As can be seen, a good error estimate for the torsional results is absolutely critical. The



(a) Linear vs torsional spring const. (b) Torsional spring const. vs err.

Fig. 7.3. Simulation study of the effects of spring constants on the accuracy of the estimated relative node positions. (a) Results for the three-node linear vs torsional spring constant experiment. (b) Results for the three-node torsional spring constant vs torsional error experiment

error estimate completely dominates the accuracy of the final relaxed model, regardless of the strength of the spring.

An interesting conclusion from these experiments is that linear odometry estimates are not nearly as important as rotational odometry estimates. Unfortunately, this is where the majority of the errors in robot odometry propagation estimates occur. Methods for augmenting the robot's odometric estimates such as with visual odometry tracking or with a compass, such as in [6], would thus greatly assist in estimating the robot's position.

7.3.4 Sensor and Motion Models

The robot's sensor model can be described as $P\left(s^t|L^t, M\right)$. This is an expression for the probability that at time t, the robot's sensors obtain the reading s^t assuming that the estimate for the robot's position is L^t. We represent the probability distribution over all possible robot poses through a non-parametric method called Parzen windows (a similar approach is used by [15]). Parzen windows are typically used to generate probability densities over continuous spaces, in this instance, we use the technique to generate a probability mass over the the space of likely robot poses. Following the definition of conditional probabilities, the equation for the sensor model can be described as

$$\begin{aligned} P\left(s^t | L^t, M\right) &= \frac{P\left(s^t, L^t, M\right)}{P\left(L^t, M\right)} \\ &= \frac{\frac{1}{N}\sum_{n=1}^{N} g_s\left(s^t - s_n^t\right) g_d\left(d^t - d_n^t\right)}{\frac{1}{N}\sum_{n=1}^{N} g_d(d^t - d_n^t)} \end{aligned}$$

where g_s and g_d are Gaussian kernels. The value $\left(s^t - s_n^t\right)$ represents the difference between two sensor snapshots and is described in Section 7.4.1 below. The value $\left(d^t - d_n^t\right)$ represents the shortest path between two nodes.

Similarly, the robot's motion model can be expressed as$P\left(L^{(t+1)} | s^{(t)}, L^{(t)}\right)$, which represents the probability that the robot is in location $L^{(t+1)}$ at time $t + 1$ given that its odometry registered reading $s^{(t)}$ after moving from location $L^{(t)}$ at time t. This is represented as

$$P\left(L^{(t+1)} | s^{(t)}, L^{(t)}\right) = g_e(e - \hat{e}) g_\phi(\phi - \hat{\phi})$$

where e and ϕ represent the linear and torsional components of the robot's motion in the current map and $\hat{e}$ and $\hat{\phi}$ represent the originally measured values.

7.3.5 Map Construction

The sequence of observations that is generated by the robot's exploration represents a map whose initial topology is a chain where each node only connects to at most two other nodes. To construct a more representative topology, the localization algorithm must identify nodes that represent the same location in space, i.e. where the robot has closed a cycle in its path. Markov localization will compute, for each timestep, a distribution which shows the probability of the robot's position across all nodes at a particular time. Traditionally, Markov localization cannot handle the "kidnapped robot" problem because a robot localizing itself is essentially tracking incremental changes in its own position. In order to recognize when two nodes are the same, the robot must acknowledge the possibility of being in two different locations in the map at once so that the nodes can be joined. To handle this situation, the robot must solve the localization problem starting with a uniform distribution over all possible starting positions in the graph. Thus, the robot must solves the complete Markov localization problem from an unknown starting pose. This way, the robot is able to identify the multi-modal case, assuming that its path had enough similarity over the parts where the robot crossed its own path. This localization algorithm must be run every time the robot attempts to find nodes that are the same location. Fortunately, the relative sparseness of a topological map as compared to a grid-based map (which is traditionally used for Markov localization), keeps the computational complexity at a minimum.

After the Markov localization step, the robot now has a probability distribution over all possible poses for each timestep. In cases where the probability distribution is multi-modal, or where it is nearly equally likely that the robot was in more than one node at a time, there exists a good chance that those nodes are actually a single node that the robot has visited multiple times. The hypothesis with the highest probability of match from all of the timesteps is selected and those nodes are merged. Merging nodes distorts the model and increases the potential energy of the system. The system then attempts to relax to a new state of minimum energy. If this new state's potential energy value is too high, then the likelihood that the hypothesis was correct is very low and must be discarded. Additionally, merges that are incorrect will affect the certainty of the the localization probability distribution after a Markov localization step. This can be observed by an increase in entropy $H(X) = -\sum_{i=1}^{n} p(x_i) log(p(x_i))$ of the probability distribution over the robot's pose in the topology. An increase in entropy can also be used as an indicator that the merge was incorrect.

This process runs through several iterations until it converges on the most topologically-consistent map of the environment. This iterative process is similar in spirit to the algorithm proposed by Thrun *et al.* [26]. Since this algorithm relies on local search to find nodes to merge, there is no guarantee that the map constructed from this algorithm will be optimal. As the robot continues to move around, more information about the environment will be gathered and can be used to get a more accurate estimate of the robot's position.

7.4 Real-World Validation

In order to determine the effectiveness of the proposed method for image based localization and map construction, two separate experiments were performed in the office environment shown in Figure 4. The first was a localization-only experiment where the KLT algorithm was used in two different ways, termed *feature matching* and *feature tracking*, in addition to a third method based on a 3D color (RGB) histogram feature extraction. The second experiment combined the KLT algorithm with the spring system to test the ability of the MLE algorithm to converge to a topologically-consistent map.

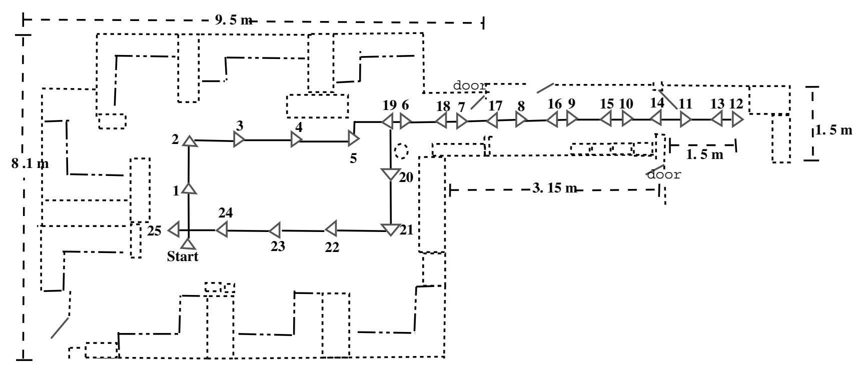


Fig. 7.4. Map of the office environment where our tests were conducted. The nodes of the robot's training path are shown with triangles

7.4.1 Extraction of Visual Features

Three different methods for extracting features from the images were tried: (1) KLT feature matching, (2) KLT feature tracking, and (3) 3D color histogram feature extraction.

1. In the *feature matching* approach, features are selected in each histogram normalized image using the KLT algorithm. The **undirected Hausdorff metric H(A,B)** [12] is used to compute the difference between the two sets. Since this metric is sensitive to outliers, we used the generalized undirected Hausdorff metric and looked for the k-th best match (rather than just the overall best match), where k was set to 12. This is defined as

$$H(A,B) = \max_{kth}(h(A,B), h(B,A)) \tag{1}$$

$$h(A,B) = \max_{a \in A} \min_{b \in B} \parallel a_i - b_j \parallel \tag{2}$$

   where $A = \{a_1, a_2, ..., a_m\}$ and $B = \{b_1, b_2, ..., b_n\}$ are two feature sets. Each feature corresponds to a 7x7 pixel window (the size of which was recommended in [27]) and $\parallel a_i - b_j \parallel$ corresponds to the sum of the differences of the pixel intensities.

   To take into account the possibility that two images might correspond to the same location but differ in rotation, the test image was rotated to eight different angles to find the best match.

2. In the *feature tracking* approach, KLT features are selected from each of the images and are tracked from one image to the next taking into account a small amount of translation between the two positions where the images were taken. The degree of match is the number of features successfully tracked from one image to the next.

3. In the *3D color histogram feature extraction* method, features representing interesting color information in the image are extracted. Colors that are very sparse in the image are considered to be interesting since they carry more unique information about features. We have derived the following index for windows of pixels in an image:

   $$value(w) = \sum_i h(i) * (1 - P(i)) \tag{3}$$

   where i is a color value, h is the histogram bin of window w for color i and $P(i)$ is the probability that color i is observed in the image. We approximate $P(i)$ by the actual distribution of colors in the image normalized to the range [0,1]. Thus the higher the value of a window w the more valuable we assume the feature to be.

   After finding interesting features, we extracted a feature set from an image at the current position and compared it to the feature sets for positions of our topological map using the Hausdorff metric. To measure the distance between single histograms, $\|a - b\|$ in Equation 2, we take the histogram intersection index

   $$intersection\ (h_k(i), h_j(i)) = \sum_i min(h_k(i), h_j(i)) \tag{4}$$

   We then localize to that map position for which the feature set is closest in the above sense to the one for the current position. To enhance the performance of the color histogram approach, we have implemented an adaptation of the data-driven color reduction algorithms presented in [2].

Each of the approaches has different advantages and disadvantages. Extracting features using the KLT algorithm but not accounting for the translation of the feature from one image to the next has the advantage of being faster and requiring less memory than using the associated tracker. However, it is less precise due to the fact that there is no model for how the features move in the images. The KLT tracker required 10.22 s per position estimation compared to 360 ms for the KLT matcher on a 1.6 GHz Pentium 4 with 512 MB RAM. The color histogram method required 1.3 s. Feature extraction required 330 ms for KLT and 1.25 s for the color histogram.

7.4.2 Determining the Number of Features to Track

Determining the number of features to track is very important since this choice can represent a major tradeoff in accuracy and performance. Typically, as the number of features increases, the accuracy of the localization algorithm will increase at the expense of computation time.

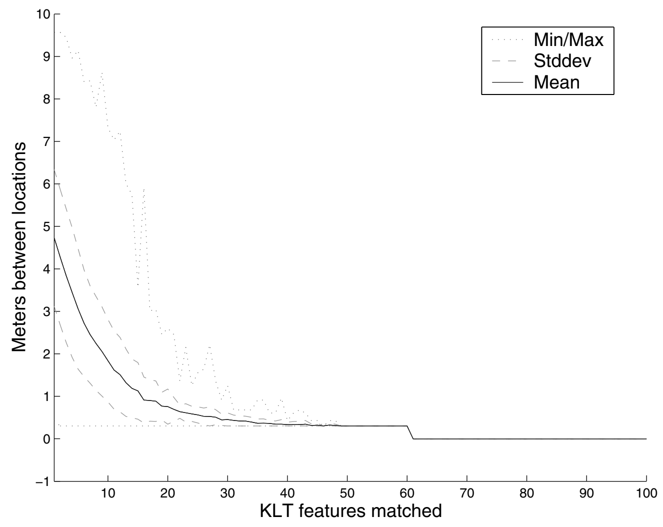


Fig. 7.5. Comparison of the number of features tracked vs. the Euclidean distance between locations where the features were obtained

To evaluate how many features to track, we obtained a set of 320 images taken at 0.3 m intervals in the office environment used for the robotics experiments. Figure 5 shows a plot of the Euclidean distance estimate between each pair of locations as a function of the number of features that the KLT algorithm can track between the respective images. As can be seen, until the number of features tracked drops to between 40-50, the likelihood that the two images are within 0.5 m of each other is extremely high. With fewer features, it becomes extremely hard to tell whether a location is the same or not. In this graph, there were no values of matched features of 60 and higher. A match of 100 features would indicate that the the robot was in exactly the same location. From this experiment, it was determined that operating over a set of 15 features was an adequate trade off between performance and accuracy since all three of the algorithms performed at an acceptable speed with this number of features.

This metric can also be used by the exploration algorithm to determine when to take a new image. When the robot takes an image as a landmark, it would attempt to track the features in subsequent images while simultaneously moving to a new location. Once the number of tracked image features drops below the above threshold, a new landmark image can be stored.

7.4.3 Image-Based Localization Experiments

A set 26 of panoramic images were obtained in the office environment. The room has a checkerboard floor while the corridor has a floor of uniformly-colored tiles. The dotted lines show the outline of the office and the furniture within it while the solid lines show the path along which the images were taken.

Images were taken at 1.07 m increments by a panoramic camera mounted on the back of a Pioneer 2 [1] mobile robot. Images have 640x480 pixels and are unwrapped into images of 816x155 pixels. This set of images was used to construct the topological map shown previously in Figure 4 and serves as the reference set.

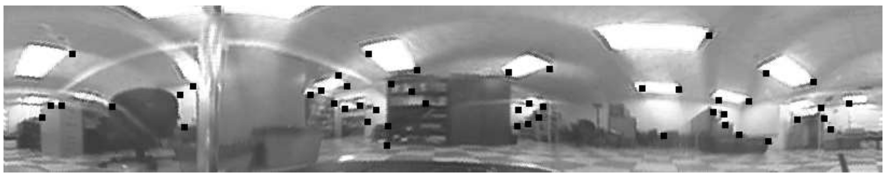


Fig. 7.6. The 50 best features selected with the KLT feature matcher on a panoramic image. In our experiments, only the 15 best features were used

The KLT feature matcher was used to extract features from the panoramic images. 15 features were selected from each image, since, as described in Section 7.4.2, this offered the best compromise between performance and accuracy. Figure 6 shows a set of features obtained by applying the feature matcher to a panoramic image. As can be seen, features corresponding to corners and prominent edges are selected.

Two sets of test images were acquired along the paths shown in Figure 7 and 8. Triangles show the positions of the original test set of images. Circled arrows show the positions of the images taken for the test sets. The images in the first test set were mostly taken along the original path from which the training set was obtained. The images in the second set were taken in a zig-zag pattern that moved mostly perpendicular to the path of the training set. These two sets were used to test the ability to localize the robot in the previously constructed topological map using only the visual information.

Table 1 illustrates the performance of the three vision algorithms on the two different sets of data. The average distance error is the average Eu-

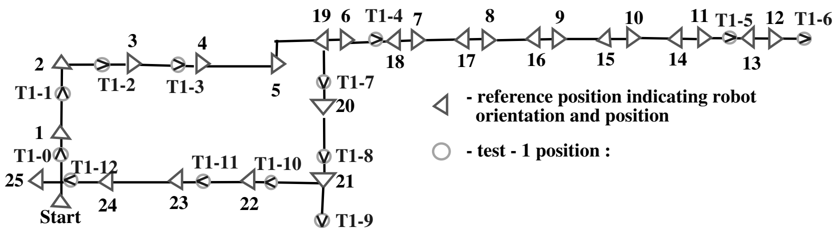

19 6 T1-4 7 8 9 10 11 T1-5 12 T1-6
3 4
2
18 17 16 15 14 13
T1-1 T1-2 T1-3 5
T1-7
20
1
T1-0 T1-12 T1-11 T1-10 T1-8
25 21
24 23 22
Start
T1-9
- reference position indicating robot orientation and position
- test - 1 position :

Fig. 7.7. Paths in the environment where test 1 was conducted. The training positions are labeled with triangles, the test positions with circled wedges. The heading of the robot at each node is shown by the direction of the triangle or wedge

Table 1: Average errors for the tests

	Test set 1			Test set 2		
Error metric	Feature matcher	Feature tracker	Color Hist.	Feature matcher	Feature tracker	Color Hist.
Avg. distance in meters	1.58	0.51	3	2.97	1.34	2.9
Multiple pos. estimates	0	1	1	0	6	2

clidean distance between the correct position and the reported position. The second metric is the number of position matches that reported multiple possible positions of the robot with equal certainty (this is caused by perceptual aliasing). The correct position to be attributed to a test position is assumed to be the nearest position (by Euclidean distance) of the reference path. When multiple position estimates are available, the worst possible position is used. The reason that the tracker and color histogram algorithms had multiple position estimates when the matcher did not was due to the scale difference in the error metrics. The feature matcher compared the difference in pixel image intensity which could range between $[0 - 12495]$ while the tracker and color histogram matcher had far fewer possible values.

As can be seen from the results, the static KLT feature matching algorithm was worst at finding the best match between an image in the training

Images for test set 2 were taken on a zigzag path across the training path.

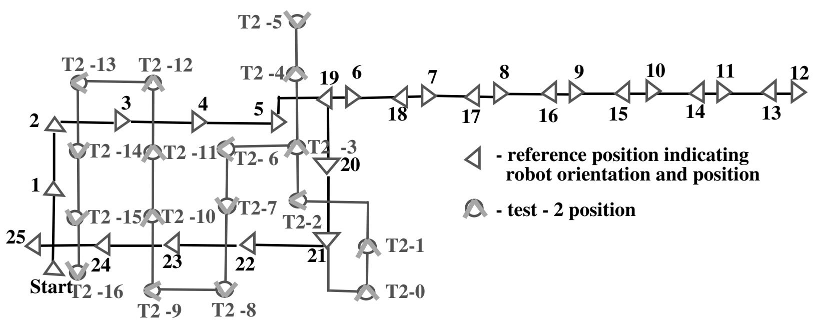


Fig. 7.8. Paths in the environment where test2 was conducted. The training positions are labeled with triangles, the test positions with circled wedges. The heading of the robot at each node is shown by the direction of the triangle or wedge

set and an image in the test set. When the training and test images were nearly identical (taken from virtually the same location in space), the static feature matcher was very good at finding the correct match. However, as the spatial difference between the images increased, the resulting match rapidly degraded. The feature tracking algorithm did a much better job of matching images in the test set to the training set. This algorithm was also much better at handling changes in feature position caused by the motion of the robot since it takes into account the translational motion of the features in the image. Unfortunately, the KLT feature tracking algorithm is much more complex in terms of computing time and memory/storage requirements.

7.4.4 Mapping Experiment

The set of training images taken in the previous experiments was used to test the MLE map construction algorithm. Noisy odometry estimates were assigned to each of the paths between images in the training set. The KLT feature tracking algorithm was used to compare features in pairs of images and only the training set of images was used. This corresponds to the case where a robot explores an unknown environment. As the robot explores, it attempts to find the most likely structure by merging nodes from its map which appear to correspond to the same sensor data.

Figure 9 illustrates the process of how the algorithm works. The original data reflects the errors in the odometric readings of the robot. In Step 1, Markov localization identifies a high probability of the robot's position in nodes at timestep 6 and 19. These two are merged and the spring model is allowed to relax. In Step 2, Markov localization is run again on the map and nodes 11 and 14 are merged. By this point, the map has obtained a shape that better matches the topology of the environment. Each possible merge candidate is evaluated by how the merge affects the entropy of the pose distribution. Bad merges will create inconsistent topological structures and have a tendency to increase the robot's pose entropy. This means that it is less sure of its position in the environment.

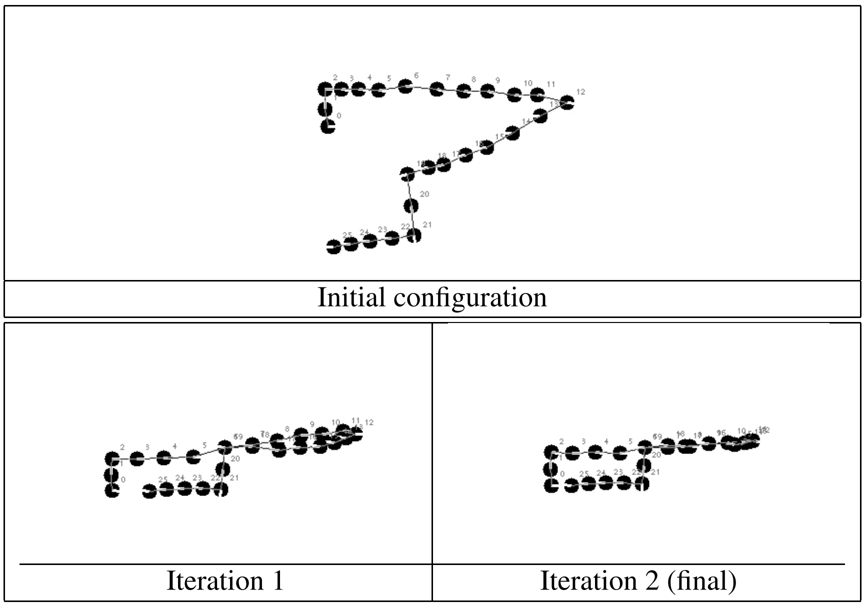


Fig. 7.9. Several iterations of the convergence algorithm. Circled nodes are to be merged in the next iteration. Only the accepted node merge candidates are shown in this example. Node merge candidates that increased the entropy of the pose distribution (and thus were rejected) are not shown. After iteration 2, all other node merge pairs were rejected

7.5 Conclusions and Future Work

Several different sensor approaches were tried for image based localization. Feature tracking was found to be better than simple feature matching since the positions of the features move in non-linear fashions around the image as the robot moves around its environment. The tradeoff is that the feature tracking algorithm is much slower to operate than the simple feature comparison algorithm.

Feature tracking was also better than the color histogram feature extraction. The main reason for the poor performance of the histogram method is the lack of distinctive colors in the environment where the experiments were conducted, which was exacerbated by the poor quality of the images. The tarnishing of the mirror introduced reflection stripes in some of the images. The reflections were very bright, similar to ceiling lights, causing confusion. Additional tests done with higher quality images have shown improvement, but the method is still not as reliable as the KLT tracker. The KLT tracker is too slow for real time performance, while both the KLT matcher and the color histogram have a chance to become real-time with additional optimization.

The KLT-based approaches operate on the intensity of pixels found in grayscale images. The features found with this method tend to differ greatly from the features found with the color-based histogram method. An interesting direction for future work would be to determine how to find good features that exhibit good qualities for both the KLT and color histogram methods. By using a combination of intensity and color information for the features, we would expect that the individual features would be even more readily distinguishable from the image background and thus easier to track overall.

The mapping algorithm has been found to be very sensitive to certain parameters. The spring and dampening constants used by the spring convergence step must be selected carefully to ensure convergence. To address this, other methods have been examined, include weighted least squares [21], and the Kalman filter [20]. Another parameter that could affect the performance of the localization algorithm are the widths of the Gaussian distributions used in the Parzen windows. Empirical studies are being done to determine good values for these parameters.

The entropy of the pose distribution is used as a method for tracking the progress of the algorithm. The specific thresholds for determining when a distribution's entropy is too high are empirically determined but more work needs to be done to fully make this a robust empirical metric. Finally, we do not "reset" the relaxation constants after the merge as the total energy after all merges have been completed represents how much error exists in the robot's odometry. Should we decide to "undo" a merge later on, we would want the map to retain the original information so that new merges could be handled. Future work will consider methods by which old merges can be undone in lieu of better information.

7.6 Acknowledgements

The authors would like to thank Andrew Howard at the University of Southern California for his insight into MLE formalisms, Jean-François Lett and Osama Masoud at the University of Minnesota for their help with the KLT tracker software, and Josep Porta at the University of Amsterdam for his thoughtful comments on our manuscript.

This material is based in part upon work supported by the National Science Foundation through grants #EIA-0224363 and #EIA-0324864, Microsoft Inc., and the Defense Advanced Research Projects Agency, MTO, Distributed Robotics.

References

[1] ActivMedia Robotics, LLC, 44 Concord Street, Peterborough, NH, 03458. *Pioneer 2 Operation Manual v6*, 2000.

[2] Kevin Amaratunga. A fast wavelet algorithm for the reduction of color information. Technical report, Department of Civil and Environmental Engineering, MIT, 1997.

[3] Matej Artač, Matjaz Jogan, and Ales Leonardis. Mobile robot localization using an incremental eigenspace model. In *Int'l Conference on Robotics and Automation*, University of Ljubljana, Slovenia, May 2002.

[4] Lode A. Cornelissen and Frans C.A. Groen. Automatic color landmark detection and retrieval for robot navigation. In Maria gini, editor, *Intelligent Autonomous Systems 7*. IOS Press, 2002.

[5] Steven Derrien and Kurt Konolige. Approximating a single viewpoint in panoramic imaging devices. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 3932–3939, 2000.

[6] Tom Duckett, Stephen Marsland, and Jonathan Shapiro. Learning globally consistent maps by relaxation. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, volume 4, pages 3841–3846, 2000.

[7] John Folkesson and Henrik Christensen. Graphical slam: A self-correcting map. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 383–390, 2004.

[8] Dieter Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*. PhD thesis, Institute of Computer Science III, University of Bonn, Germany, December 1998.

[9] Robert Grabowski, Luis E. Navarro-Serment, Chris J. J. Paredis, and Pradeep Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308, 2000.

[10] Verena Vanessa Hafner. Cognitive maps for navigation in open environments. In *Proceedings of the 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 801–808, Venice, Italy, 2000.

[11] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. Localization for mobile robot teams using maximum likelihood estimation. In *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, EPFL Switzerland, September 2002.

[12] Daniel Huttenlocher, Gregory Klanderman, and William Rucklige. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[13] KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker. http://vision.stanford.edu/~birch/klt/.

[14] Kurt Konolige. Large-scale map-making. In *Proc. of the Nat'l Conf. on Artificial Intelligence*, pages 457–463, 2004.

[15] Ben J.A Kröse, Nikos Vlassis, Roland Bunschoten, and Yoichi Motomura. A probabilistic model for appearance-based robot localization. In *Image and Vision Computing*, volume 19, pages 381–391, 2001.

[16] Benjamin Kuipers and Patrick Beeson. Bootstrap learning for place recognition. In *Proc. of the Eighteen Nat'l Conf. on Artificial Intelligence*, 2002.

[17] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[18] Bruce D. Lucas and Takeo. Kanade. An iterative image registration technique with an application to stereo vision. *IJCAI*, pages 674–679, 1981.

[19] Brian Pinette. *Image-based Navigation through Large scale Environments*. PhD thesis, University of Massachusetts, Computer Science Department, 1994.

[20] Paul E. Rybski, Stergios I. Roumeliotis, Maria Gini, and Nikolaos Papanikolopoulos. Appearance-based minimalistic metric slam. In *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2003.

[21] Paul E. Rybski, Stergios I. Roumeliotis, Maria Gini, and Nikolaos Papanikolopoulos. A comparison of maximum likelihood methods for appearance-based minimalistic slam. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2004.

[22] Robert Sim and Gregory Dudek. Learning environmental features for pose estimation. *Image and Vision Computing*, 19(11):733–739, 2001.

[23] Robert Sim, Gregory Dudek, and Nicholas Roy. Online control policy optimization for minimizing map uncertainty during exploration. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 1758 – 1763, April/May 2004.

[24] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 36:31–50, 1991.

[25] Stephen H.G. ten Hagen. Good features to map. In *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 1512–1517, Las Vegas, USA, October 2003.

[26] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.

[27] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, School of Computer Science, Carnegie Mellon University, April 1991.

[28] Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 1023–1029, San Francisco, CA, April 2000.

[29] Niall Winters and José Santos-Victor. Omni-directional visual navigation. In *Proc. of the 7th Int. Symp. on Intelligent Robotic Systems (SIRS 99)*, pages 109–118, Coimbra, Portugal, July 1999.

8 Intelligent Precision Motion Control

Kok Kiong Tan[1], Sunan Huang[1], Ser Yong Lim[2], Wei Lin[2]

1. Department of Electrical and Computer Engineering, National University of Singapore, Singapore
   {eletankk,elehsn}@nus.edu.sg
2. Singapore Institute of Manufacturing, 71 Nanyang Drive
   Singapore 638075
   {sylim,wlin}@SIMTech.a-star.edu.sg

8.1 Introduction

Precision engineering [3] has been steadily gathering momentum over the last century in terms of research, development, and application to product innovation. The driving force in this development appears to arise from requirements for much higher performance of products, higher reliability, longer life, lower cost, and miniaturization. In the new millenium, ultra precision manufacture is poised to progress further and it is expected to enter the nanometer scale regime (nanotechnology). Increasing packing density on integrated circuits and sustained breakthrough in minimum feature dimensions on semiconductor set the pace in the electronics industry. Emerging technologies such as MEMS (Micro-Electro-Mechanical Systems), otherwise known as MicroSystems Technology (MST) in Europe expand further the scope of miniaturisation and integration of electrical and mechanical components.

One enabling technology which has made these and more modern applications possible is the advance and development in precision mechanisms and motion control. An increasing number of the precision motion systems today, general purpose or application specific, are based on the use of DC permanent magnet linear motors (PMLM) [1] for the main reason that among the electric motor drives available, the PMLMs are probably the most naturally akin to applications involving high speed and high precision motion control. The increasingly widespread industrial applications of PMLMs in various semiconductor processes, precision metrology and miniature system assembly are self-evident testimonies of the effectiveness of PMLMs in addressing the high requirements associated with these application areas. The main benefits of a PMLM include the high force density achievable, low thermal losses and, most importantly, the high precision and accuracy associated with the simplicity in mechanical structure. Unlike rotary machines, linear motors require no indirect coupling mechanisms as in gear boxes, chains and screws coupling. This greatly reduces the effects of contact-type nonlinearities and disturbances such as backlash and frictional forces, especially when they are used with hydrostatic, aerostatic or magnetic bearings. However, the advantages of using mechanical transmission are also consequently lost, such as the inherent ability to reduce the effects of model uncertainties and external disturbances. An adequate reduction of these effects, either through a proper physical design or via the control system, is of paramount importance in order to achieve the end objectives of high-speed and high precision motion control.

There are several important challenges to the precision motion control system. First, the measurement system must be capable of yielding a very fine resolution in position measurements. Today, laser interferometers can readily yield a measurement resolution of down to one nanometer. Where cost is a concern, a high grade analog optical encoder in conjunction with an efficient interpolator can be used to provide sub-micrometer resolution measurements [4]. In the latter case, interpolation factors of up to 4096 times have been reported. This will effectively yield a resolution in the nanometer regime, given the fine scales manufacturing tolerance currently achievable. However, one should be cautious of interpolation errors associated with limited wordlength A/D operations, and imperfect analog encoder waveform with mean, phase offsets, noise as well as non-sinusoidal waveform distortion. The interested readers may refer to [15] for more details on these aspects and possible remedial measures.

Secondly, the control electronics must have a sufficient bandwidth to cope with the high encoder count frequency associated with high speed motion on one hand, and a sufficiently high sampling frequency to circumvent anti-aliasing pits when motion is at a very low speed. Consequent of these requirements, the control algorithms must also be efficient enough to be executed within each time sample, and yet possess sufficient capacity to provide precision motion tracking and rapid disturbance suppression. This calls for a good weighted selection of efficient control components to address not only the specific dynamics of the servo system in point, but also exogenous disturbances arising from the application, including load changes, and drives-induced electro-magnetic interference.

Thirdly, the geometrical imperfections of the mechanical system should be adequately accounted for in the control system, if absolute positioning accuracy is crucial to the application concerned [7]. A 3D cartesian machine, for example, has 21 possible sources of geometrical errors (linear, angular, straightness, orthogonality errors from the 3 axes combined). Yet, many control engineers may evaluate positional accuracy solely with respect to encoder measurements, assuming ideal geometrical properties of the mechanical system. This assumption can lead to drastic and undesirable consequences when a high absolute positioning accuracy of the end object (e.g. machine tool) is required, since a very small tracking error with respect to encoder counts can be magnified many times over, when verified and calibrated in terms of absolute accuracy using a laser interferometer. These errors, arising from geometrical imperfections, can be calibrated and compensated for, if they are repeatable. The present common mode to deal with this problem is to build a look-up table model of the geometrical errors. The table maps an encoder reported position into the actual absolute position, and it can thus be used as the basis for geometrical offset compensation.

In high precision motion control applications, vibrations induced from the mechanical system should be minimised as far as possible. Ideally, this calls for a highly rigid mechanical design, active damping and stable support structures. In the control system, this issue is commonly addressed by having in place a notch filter which will terminate the transmission of frequencies which will cause a resonance. Environmental issues, inducing effects which result in a change in machine parameters, should be carefully considered too. These should include stability in local and ambient temperature, humidity, air-flow, air-particle, even possibly uniformity of lighting.

This chapter is concerned with the development of an integrated precision motion control system on an open-architecture and rapid prototyping platform. It will attempt to address the abovementioned challenges. The various selected control components, which constitutes the final overall strategy, will be elaborated in terms of their purposes and designs. The theoretical aspects associated with these components can be found in the respective literature and work of the authors which will be highlighted to the readers in due course. Detailed implementation aspects, including the hardware architecture, software development platform and user interface design, are given to provide a general reference of the key issues which should be addressed in the design of a precision motion control system.

8.2 Overall Control Strategy

The overall control structure is shown in Figure 8.1.

In what follows, the purpose and design of each component depicted in Figure 8.1 will be elaborated. Since the design of several of these components will be based on a model of the PMLM, the initial part of this section will attempt to provide a concise system description of PMLM-based servo systems.

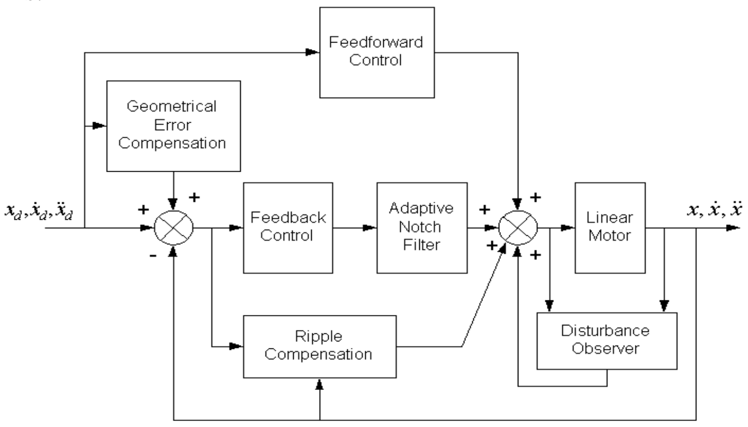


Fig. 8.1. Overall structure of control system

8.2.1 PMLM System Description

The type of motor predominantly addressed in this chapter is a DC permanent magnet linear motor (PMLM). The dynamics of the PMLM can be viewed as comprising of two components: a dominantly linear model, and an uncertain and nonlinear remnant which nonetheless must be considered in the design of the control system if high precision motion control is to be efficiently realised.

In the dominant linear model, the mechanical and electrical dynamics of a PMLM can be expressed as follows:

$$M\ddot{x} + D\dot{x} + F_{load} = F_m ,$$

$$K_e\dot{x} + L_a\frac{dI_a}{dt} + R_aI_a = u ,$$

$$F_m = , K_tI_a ,$$

where x denotes position; M, D, F_m, F_{load} denote the mechanical parameters: inertia, viscosity constant, generated force and load force respectively;

u, I_a, R_a, L_a denote the electrical parameters: input DC voltage, armature current, armature resistance and armature inductance respectively; K_t denotes an electrical-mechanical energy conversion constant; K_e is the back EMF constant of the motor. It should be noticed that F_{load} also includes some bounded disturbances, such as connecting cables, vibration due to external sources and machine configurations and self-excited vibration.

Since the electrical time constant is typically much smaller than the mechanical one, the delay due to electrical transient response may be ignored, giving the following simplified model:

$$\dot{x} = -\frac{K_1}{M}\dot{x} + \frac{K_2}{M}u - \frac{1}{M}F_{load}, \tag{8.1}$$

where

$$K_1 = \frac{K_e K_t + R_a D}{R_a}, K_2 = \frac{K_t}{R_a}.$$

Clearly, this is a second-order linear dynamical model.

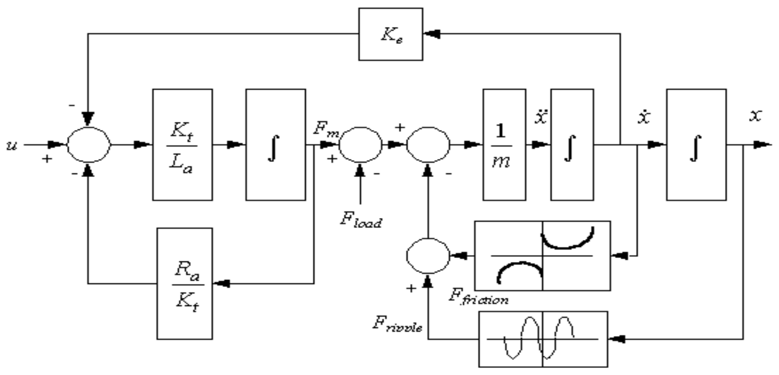


Fig. 8.2. Model of PMLM

The dominant linear model has not included extraneous nonlinear effects which may be present in the physical structure. Among them, the two prominent nonlinear effects associated with PMLM are due to ripple and frictional forces, arising from the magnetic structure of PMLM and other physical imperfections. Figure 8.2 depicts a block diagram model of the motor, including explicitly the various exogenous disturbance signals present.

8.2.1.1 Force Ripples

The thrust force transmitted to the translator of a PMLM is generated by a sequence of attracting and repelling forces between the poles of the permanent magnets when a current is applied to the coils of the translator. In addition to the thrust force, parasitic ripple forces are also generated in a PMLM due to the magnetic structure of PMLM. This ripple force exists in almost all variations of PMLM (flat, tubular, moving-magnet etc.), as long as a ferromagnetic core is used for the windings.

The two primary components of the force ripple are the cogging (or detent) force and the reluctance force. The cogging force arises as a result of the mutual attraction between the magnets and iron cores of the translator. This force exists even in the absence of any winding current and it exhibits a periodic relationship with respect to the position of the translator relative to the magnets. Cogging manifests itself by the tendency of the translator to align in a number of preferred positions regardless of excitation states. There are two potential causes of the periodic cogging force in PMLMs, resulting from the slotting and the finite length of iron-core translator. The reluctance force is due to the variation of the self-inductance of the windings with respect to the relative position between the translator and the magnets. Thus, the reluctance force also has a periodic relationship with the translator-magnet position.

Collectively, the cogging and reluctant force constitute the overall force ripple phenomenon. Even when the PMLM is not powered, force ripples are clearly existent when the translator is moved along the guideway. There are discrete points where minimum/maximum resistance is experienced. At lower velocity, the effects are more fully evident due to the lower momentum available to overcome the magnetic resistance.

Due to the direct-drive principle behind the operation of a linear motor, the force ripple has significant effects on the position accuracy achievable and it may also cause oscillations and yield stability problems, particularly at low velocities or with a light load (low momentum). The ripple periodicity has a fixed relationship with respect to position, but the amplitude can vary with velocity.

A first order model for the force ripple can be described as a position periodic sinusoidal type signal:

$$F_{ripple}(x) = A(x)\sin(\omega x + \phi) \tag{8.2}$$

Higher harmonics of the ripple may be included in higher order models.

8.2.1.2 Friction

Friction is inevitably present in nearly all moving mechanisms, and it is one major obstacle to achieving precise motion control. Several characteristic properties of friction have been observed, which can be broken down into two categories: static and dynamic. The static characteristics of friction, including the stiction friction, the kinetic force, the viscous force, and the Stribeck effect, are functions of steady state velocity. The dynamic phenomena include pre-sliding displacement, varying breakaway force, and frictional lag. Many empirical friction models have been developed which attempt to capture specific components of observed friction behaviour, but generally, it is acknowledged that a precise and accurate friction model is difficult to be obtained in an explicit form, especially for the dynamical component. For many purposes, however, the Tustin model has proven to be useful and it has been validated adequately in many successful applications. The Tustin model may be written as:

$$F_{friction} = [F_c + (F_s - F_c)e^{-(|\dot{x}/\dot{x}_s|)^{\delta}} + F_v\,|\,\dot{x}\,|]\mathrm{sgn}(\dot{x}), \tag{8.3}$$

where F_s denotes static friction, F_c denotes the minimum value of Coulomb friction, $\dot{x}_s$ and F_v are lubricant and load parameters, and δ is an additional empirical parameter. Figure 8.3 graphically illustrates this friction model.

Considering these nonlinear effects, the PMLM dynamics may be described by:

$$\ddot{x} = -\frac{K_1}{M}\dot{x} + \frac{K_2}{M}u - \frac{1}{M}(F_{load} + F_{ripple} + F_{friction}) \tag{8.4}$$

The effects of friction can be greatly reduced using high quality bearings such as aerostatic or magnetic bearings.

A common nonlinear function $F_1^*(x,\dot{x})$ may be used to represent the nonlinear dynamical effects due to force ripple, friction and other unaccounted dynamics collectively. The servo system (8.4) can thus be alternatively described by:

$$\ddot{x} = -\frac{K_1}{M}\dot{x} + \frac{K_2}{M}u - \frac{1}{M}F_{load} + F_1^*(x,\dot{x}) \tag{8.5}$$

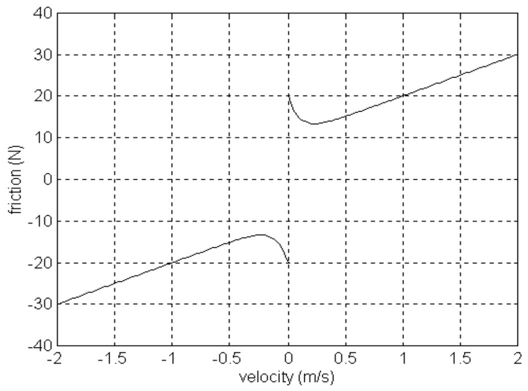


Fig. 8.3. The Tustin friction model

Let

$$\frac{K_2}{M} f(x,\dot{x}) = -\frac{1}{M} F_{load} + F_1^*(x,\dot{x})$$

It follows that

$$\ddot{x} = -\frac{K_1}{M}\dot{x} + \frac{K_2}{M}u - \frac{K_2}{M} f(x,\dot{x}) \tag{8.6}$$

With the tracking error e defined as:

$$e = x_d - x,$$

(8.6) may be expressed as:

$$\ddot{e} = -\frac{K_1}{M}\dot{e} + \frac{K_2}{M}u - \frac{K_2}{M} f(x,\dot{x}) + \frac{K_2}{M}\left(\frac{M}{K_2}\ddot{x}_d + \frac{K_1}{K_2}\dot{x}_d\right) \tag{8.7}$$

Since

$$\frac{d}{dt}\int_0^t e(\tau)dt = e,$$

the system state variables are assigned as $x_1 = \int_0^t e(\tau)dt, x_2 = e$ and $x_3 = \dot{e}$. Denoting $X = [x_1, x_2, x_3]^T$, (8.7) can then be put into the equivalent state space form:

$$\dot{X} = AX + Bu + Bf(x,\dot{x}) + B\left(-\frac{M}{K_2}\ddot{x}_d - \frac{K_1}{K_2}\dot{x}\right) \tag{8.8}$$

with

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\frac{K_1}{M} \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \\ -\frac{K_2}{M} \end{pmatrix}.$$

8.2.2 Feedforward Control

The design of the feedforward control component is straightforward. From (8.8), the term of $B(-\frac{M}{K_2}\ddot{x}_d - \frac{K_1}{K_2}\dot{x}_d)$ may be neutralised using a feed-forward control term in the control signal. The feedforward control is thus designed as:

$$u_{FF}(t) = \frac{M}{K_2}\ddot{x}_d + \frac{K_1}{K_2}\dot{x}_d. \tag{8.9}$$

Clearly, the reference position trajectory must be continuous and twice differentiable, otherwise a pre-compensator to filter the reference signal will be necessary. The only parameters required for the design of the feedforward control are the parameters of the second-order linear model.

Additional feedforward terms may be included for direct compensation of the nonlinear effects, if the appropriate models are available. For example, if a good signal model of the ripple force is available (8.2), then an additional static term in the feedforward control signal $u_{FFx} = \frac{1}{K_2}F_{ripple}(x_d)$ can effectively compensate for the ripple force. In fact, in the proposed overall strategy, an adaptive feedforward control component for ripple compensation (to be elaborated in Section 8.2.4) has been included.

In the same way, a static friction feedforward pre-compensator can be installed if a friction model is available. In [14], an efficient way of friction modelling using relay feedback is proposed where a simple friction model (incorporating coulomb and viscuous friction components) can be obtained automatically.This can be used to construct an additional feed forward signal, basedonly on the reference trajectories. In addition, if the motion control task is essentially repetitive, an iteratively refined addi-tional feedforward signal can further reduce any control-induced tracking error. A possible scheme based on iterative learning control (ILC) can be found in [5], [6]. The basic idea in ILC is to exploit the repetitive nature of the tasks as experience gained to compensate for the poor or incomplete knowledge of the system model and the disturbances. Essentially, the ILC structure includes a feedforward control component which refines the feedforward signal to enhance the performance of the next cycle based on previous cycles. A block diagram of the ILC scheme is depicted in Figure 8.4.

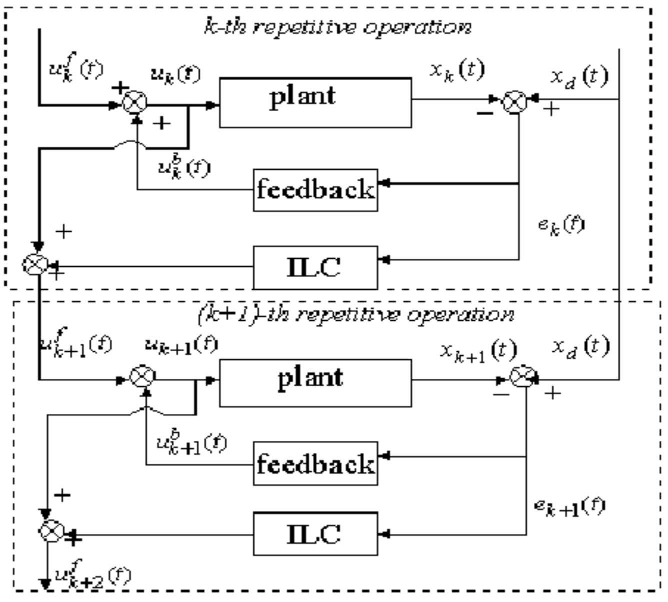


Fig. 8.4. Iterative learning control

Characteristic of all feedforward control schemes, the performance is critically dependent on the accuracy of the model parameters. Therefore, feedforward is usually augmented with suitable feedback control schemes, one of which is given in the next subsection.

8.2.3 PID Feedback Control

In spite of the advances in mathematical control theory over the last fifty years, industrial servo control loops are still essentially based on the three-term PID controller. The main reason is due to the widespread field acceptance of this simple controller which has been effective and reliable in most situations when adequately tuned. More complex advanced controllers have fared less favourably under practical conditions, despite the higher costs associated with implementation and the higher demands in control tuning. It is very difficult for operators unfamiliar with advanced control to adjust the control parameters. Given these uncertainties, there is little surprise that PID controllers continue to be manufactured by the hundred thousands yearly and still increasing. In the composite control system, PID is used as the feedback control term. While the simplicity in a PID structure is appealing, it is also often proclaimed as the reason for poor control performance whenever it occurs. In this design, advanced optimum control theory is applied to tune PID control gains. The PID feedback controller is designed using the Linear Quadratic Regulator (LQR) technique for optimal and robust performance of the nominal system. The feedforward plus feedback configuration is often also referred to as a two-degree-of-freedom (2-DOF) control.

The nominal portion of the system (without uncertainty) is given by:

$$\dot{X}(t) = AX(t) + Bu(t) \tag{8.10}$$

where

$$u = KX = kx_1 + k_{d1}x_2 + k_{d2}x_3 \,. \tag{8.11}$$

This is a PID control structure which utilises a full-state feedback.

The optimal PID control parameters are obtained using the LQR technique that is well known in modern optimal control theory and it has been widely used in many applications. It has a very nice robustness property, i.e., if the process is of single-input and single-output, then the control system has at least a phase margin of 60 degree and a gain margin of infinity. Under mild assumptions, the resultant closed-loop system is always stable. This attractive property appeals to the practitioners. Thus, the LQR theory has received considerable attention since 1950s.

The PID control is given by:

$$u_{PID} = -(r_0 + 1)B^T \Phi X(t) \tag{8.12}$$

where Φ is the positive definite solution of the Riccati equation:

$$A^T\Phi + \Phi A - \Phi BB^T\Phi = -Q \tag{8.13}$$

and $Q = H^T H$ where *H* relates to the states weighting parameters in the usual manner. In general, if (*A,B*) is controllable and stabilizable, the solution of the positive definite Φ always exists. Note that r_0 is independent of Φ and it is introduced to weigh the relative importance between control effort and control errors. Note for this feedback control, the only parameters required are the parameters of the second-order model and a user-specified error weight r_0.

Where other state variables are available (e.g., velocity, acceleration etc.), a full state feedback controller may also be used for the feedback control component. Interested readers may refer to [16] for the implementation of such a scheme on PMLMs. Adaptive and robust control has also been investigated in a previous study as an alternative to the PID feedback control, where the feedback control signal is adaptively refined based on parameter estimates of the nonlinear system model, using prevailing input and output signals. The achievable performance is highly dependent on the adequacy of the model, and the initial parameter estimates. Furthermore, full adaptive control schemes can greatly drain the computational resources available. Interested readers may refer to [8] for more details on adaptive and robust control schemes.

8.2.4 Ripple Compensation

From motion control viewpoints, force ripples are highly undesirable, but yet they are predominantly present in PMLMs. They can be minimised or even eliminated by an alternative design of the motor structure or spatial layout of the magnetic materials such as skewing the magnet, optimising the disposition and width of the magnets etc. These mechanisms often increase the complexity of the motor structure. PMLM, with a slotless configuration is a popular alternative since the cogging force component due to the presence of slots is totally eliminated. Nevertheless, the motor may still exhibit significant cogging force owing to the finite length of the iron-core translator. Finite element analysis confirms that the force produced on either end of the translator is sinusoidal and unidirectional. Since the translator has two edges (leading and trailing edges), it is possible to optimise the magnet length so that the two sinusoidal force waveform of each edge cancel out each other. However, this would again contribute some degree of complexity to the mechanical structure. A more practical approach to eliminate cogging force would be to adopt a sleeve-less or an iron-less design in the core of the windings. However, this approach results in a highly inefficient energy conversion process with a high leakage of magnetic flux due to the absence of material reduction in the core. As a result, the thrust force generated is largely reduced (typically by 30 % or more). This solution is not acceptable for applications where high acceleration is necessary. In addition, iron-core motors, which produce high thrust force, are ideal for accelerating and moving large masses while maintaining stiffness during the machining and processing operations.

In this section, a simple approach will be developed which is based on the use of a dither signal as a “trojan horse" to cancel the effects of force ripples. The construction of dither signal requires knowledge of the characteristics of force ripples which can be obtained from simple step experiments. For greater robustness, real-time feedback of motion variables can be used to adaptively refine the dither signal characteristics.

It is assumed that the force ripple can be equivalently viewed as a response to a virtual input described in the form of a periodic sinusoidal signal:

$$u_{ripple} = A\sin(\omega x + \phi) = a_1 \sin(\omega x) + a_2 \cos(\omega x)\,.$$

The dither signal is thus designed correspondingly to eradicate this virtual force as:

$$u_{AFC} = -\hat{a}_1 \sin(\omega x) - \hat{a}_2 \cos(\omega x). \qquad (8.14)$$

Perfect cancellation will be achieved when

$$\hat{a}_1 = a_1, \hat{a}_2 = a_2\,.$$

Compensation schemes are well-known to be sensitive to modeling errors which inevitably result in significant remnant ripples. An adaptive approach is thus adopted so that $\hat{a}_1$ and $\hat{a}_2$ will be continuously adapted based on desired trajectories and prevailing tracking errors.

Possible update laws for the adaptive parameters will be

$$\dot{\hat{a}}_1(t) = \gamma_1 X^T W B \sin(\omega x), \qquad (8.15)$$

$$\dot{\hat{a}}_2(t) = \gamma_2 X^T W B \cos(\omega x), \qquad (8.16)$$

where W is a positive definite matrix to be defined shortly. In other words, the adaptive update laws (8.15) and (8.16) can be applied as an adjustment mechanism such that $a_1(t)$ and $a_2(t)$ in (8.14) converge to their true values. Interested readers may refer to [13] for full details on this adaptive ripple compensation scheme.

Substituting the feedback and ripple compensation, the closed-loop system without other disturbance is given by

$$\dot{X} = \overline{A}X + B(\tilde{a}_1 \sin(\omega x) + \tilde{a}_2 \cos(\omega x))$$

where $\overline{A} = A + BK$. As shown in the feedback loop, the gain K is designed to ensure the stability of $A+BK$. Thus, the following Lyapunov equation holds

$$\overline{A}^T W + W\overline{A} = -Q, \qquad (8.17)$$

where W is a positive definite matrix and $Q > 0$ has the same meaning as in (8.13). For a stable matrix $\overline{A}$, the solution of W always exists if $Q > 0$. Define the following Lyapunov function

$$V = X^T W X + \frac{1}{\gamma_1}\tilde{a}_1^{\,2} + \frac{1}{\gamma_2}\tilde{a}_2^{\,2}$$

The derivative of the Lyapunov function is given by

$$\dot{V} = X^T(\overline{A}^T W + W\overline{A})X + 2\tilde{a}_1 X^T WB\sin(\omega x) + 2\tilde{a}_2 X^T WB\cos(\omega x)$$
$$-2\tilde{a}_1\dot{\hat{a}}_1 - 2\tilde{a}_2\dot{\hat{a}}_2$$
$$\le -\lambda_{\min}(Q)\|X\|^2 + 2\tilde{a}_1[X^T WB\sin(\omega x) - \dot{\hat{a}}_1]$$
$$+ 2\tilde{a}_2[X^T WB\cos(\omega x) - \dot{\hat{a}}_2].$$

Substituting the adaptive laws (8.15)-(8.16) yields

$$\dot{V} \le -\lambda_{\min}(Q)\|x\|^2 \tag{8.18}$$

Since $\lambda_{\min}(Q) > 0$, it follows that $\dot{V} \le 0$. This implies that $X, \hat{a}_1, \hat{a}_2$ are uniformly bounded with respect to *t*.

Furthermore, with $X, \hat{a}_1, \hat{a}_2$ being bounded, $\dot{X}$ is bounded. Equation (8.18) and the definiteness of V will imply that

$$\lim_{t\to\infty}\int_0^t -\dot{V}(\tau)d\tau = V(0) - \lim_{t\to\infty} V(t) < \infty$$

Applying Barbalat's lemma, it follows that

$$\lim_{t\to\infty}\dot{V}(t) = 0,$$

which by virtue of (8.18), implies

$$\lim_{t\to\infty}\|X\| = 0.$$

This implies that $\lim_{t\to\infty}\|e\| = 0$. In addition, if $\phi = [\sin(\omega x), \cos(\omega x)]^T$ is persistently exciting, it follows that $\lim_{t\to\infty}\hat{a}_1(t) = a_1, \lim_{t\to\infty}\hat{a}_2(t) = a_2$.

8.2.5 Friction Compensation

Friction is another important aspect to be addressed in the control systems of high quality servo mechanisms. With a friction model such as the one given in (8.3), friction compensation schemes can be designed. Unfortunately, such friction models are unknown a *priori* in practice. In this section, an adaptive technique will be described for friction compensation.

Consider the system model

$$\dot{X}(t) = AX(t) + Bu(t) + B[-\frac{F_c}{K_2}\text{sgn}(\dot{x}) - \frac{F_s - F_c}{K_2}e^{-(\dot{x}/\dot{x}_s)^2}\text{sgn}(\dot{x}) - \frac{F_v}{K_2}\dot{x}]$$

Let $\sigma_1 = -\frac{F_c}{K_2}, \sigma_2 = -\frac{F_s - F_c}{K_2}$ and $\sigma_3 = -\frac{F_v}{K_2}$, the model can then be written as

$$\dot{X}(t) = AX(t) + Bu(t) + B[\sigma_1 \operatorname{sgn}(\dot{x}) + \sigma_2 e^{-(\dot{x}/\dot{x}_s)^2} \operatorname{sgn}(\dot{x}) + \sigma_3 \dot{x}]$$

A suitable compensation law is

$$u_{friction} = -\hat{\sigma}_1 \operatorname{sgn}(\dot{x}) - \hat{\sigma}_2 e^{-(\dot{x}/\dot{x}_s)^2} \operatorname{sgn}(\dot{x}) - \hat{\sigma}_3 \dot{x}$$

where $\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3$ are the estimates of the true parameters $\sigma_1, \sigma_2, \sigma_3$, respectively. An adaptive law can be designed so that $\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3$ will converge to actual values as $t \to \infty$. The following update laws can be used

$$\dot{\hat{\sigma}}_1 = \gamma_{\sigma 1} X^T WBsign(\dot{x}),$$

$$\dot{\hat{\sigma}}_2 = \gamma_{\sigma 2} X^T WBe^{-(\dot{x}/\dot{x}_s)^2} sign(\dot{x}),$$

$$\hat{\sigma}_3 = \gamma_{\sigma 3} X^T WB\dot{x},$$

where W is the same as in (8.17), $\gamma_{\sigma 1}, \gamma_{\sigma 2}, \gamma_{\sigma 3}$ are the adaptation factors. A similar proof to the force ripple can be derived to ensure the stability. For uncertain models, robust control schemes can be considered. Interested readers may refer to [9] for details.

8.2.6 Disturbance Observer

The achievable performance of PMLMs is also unavoidably limited by the amount of disturbances present. These disturbances may arise due to load changes, system parameter perturbation owing to prolonged usage, measurement noise and high frequencies generated from the amplifiers (especially when a *Pulse Width Modulated* (PWM) amplifier is used), or inherent nonlinear dynamics such as the force ripples and frictional forces mentioned. Incorporating a higher resolution in the measurement system via the use of high interpolation electronics on the encoder signals can only achieve improvement in positioning accuracy to a limited extent. Thereafter, the amount of disturbances present will ultimately determine the achievable performance. In this subsection, this important issue of disturbance compensation for precision motion control systems will be addressed.

Figure 8.5 shows the block diagram of the ``Disturbance Observer'' part of the proposed control system which uses an estimate of the actual disturbance, deduced from a disturbance observer, to compensate for the disturbances. *x, u, d* and $\hat{d}$ denote the position signal, control signal, actual and estimated disturbance respectively. The disturbance observer, shown demarcated within the dotted box in Figure 8.5, estimates the disturbance based on the output x and the control signal u. $\mathcal{P}$ denotes the actual system. $\mathcal{P}n$ denotes the nominal system which can be generally described by:

$$\mathcal{P}n = \frac{a_0}{s^l(s^{m-l} + a_1 s^{m-l-1} + \ldots + a_{m-l-1}s + a_{m-l})},$$

where $\mathcal{P}n$ is a m-th order delay system and has l poles at the origin. In this chapter, as mentioned, we will use a third order model, i.e., $l=1$, $m=3$.

$$\mathcal{P}n = \frac{a_0}{s\ (s^2 + a_1 s\ + a_2)}.$$

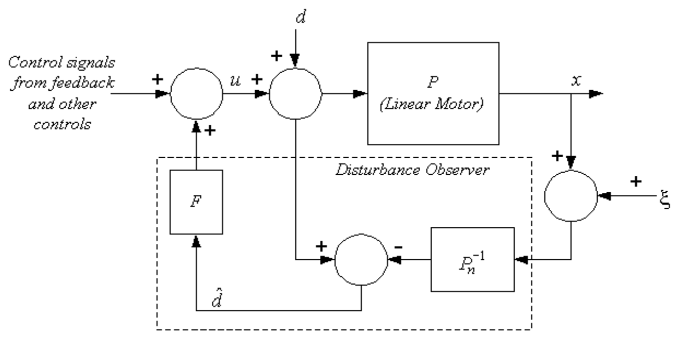


Fig. 8.5. Control system with disturbance observer

The disturbance observer incorporates the inverse of the nominal system, and thus a low pass filter F is required to make the disturbance observer proper and practically realizable. For our choice of a third order model $\mathcal{P}n$, a suitable filter is

$$F(s) = \frac{f_3}{s^3 + f_1 s^2 + f_2 s + f_3}$$

f_1, f_2, f_3 can be adjusted to satisfy a satisfactory compromise between tracking and disturbance rejection.

Interested readers may refer to [11], [18] for full details on the disturbance observer scheme.

8.2.7 Self-Tuning Schemes

Most of the aforementioned control components would require a system/sub-system model. Thus, performance degradation can be expected when model becomes inadequate due to changes in system dynamics with time, following prolonged usage and machine deterioration. Self-tuning schemes which can update the models efficiently are useful and necessary to ensure performance over time. Many high-end controllers appearing in the market now come equipped with auto-tuning and self-tuning features. For the feedback and feedforward control presented, an efficient self-tuning approach can be found in [12]. Using an equivalent relay feedback configuration, both a dominant linear system model and a nonlinear friction model can be obtained.

8.2.8 Vibration Control and Monitoring

Mechanical vibration in machines and equipment can occur due to many factors, such as unbalance inertia in spindles, motors, drives and unstable fluid supplies etc, poor kinematic design resulting in a non-rigid support structure, component failure and/or operations outside prescribed load ratings. The machine vibration signal can be typically characterised as a narrow-band interference signal anywhere in the range from 1 Hz to 500 kHz. To prevent equipment damage from the severe shaking that occurs when machines malfunction or vibrate at resonant frequencies, a filter which terminate signal transmission at these frequencies will be very useful. When the machine is used to perform highly precise positioning functions, undue vibrations can lead to poor repeatability properties, impeding any systematic error compensation effort. This results directly in a loss of precision and accuracy achievable.

8.2.8.1 Adaptive Notch Filter

One approach to eliminate/suppress undesirable narrow-band frequencies can be efficiently accomplished using a notch filter (also known as a band-stop filter). Ideally, the filter highly attenuates a particular frequency component and leaves the others relatively unaffected. Thus, an ideal notch filter has a unity gain at all frequencies and a zero gain at the null frequencies. A single-notch filter is effective in removing a single frequency or a narrow-band interference; a multiple-notch filter is useful for the removal of multiple narrow-bands, necessary in applications requiring harmonics cancellation.

Complete narrow-band disturbance suppression requires an exact adjustment of the filter parameters to align the notches with the resonant frequencies. If the true frequency of the narrow-band interference to be rejected is stable and known *a priori*, a notch filter with fixed null frequency and fixed bandwidth can be used. However, if no information is available *a priori* or when the resonant frequencies drift with time, the fixed notch may not coincide exactly with the desired null frequency if the bandwidth is too narrow (i.e. $\alpha \approx 1$). In this case, an adaptive technique is highly recommended where FFT (Fast Fourier Transform) is used to iteratively derive the signal spectrum (and thus resonants) from the latest n samples of the control signal to update the signal spectrum. Based on the updated spectrum, the filter characteristics can be continuously adjusted for notch alignment. The block diagram of the adaptive notch filter which has been developed, with its adjusting mechanism, is shown in Figure 8.6.

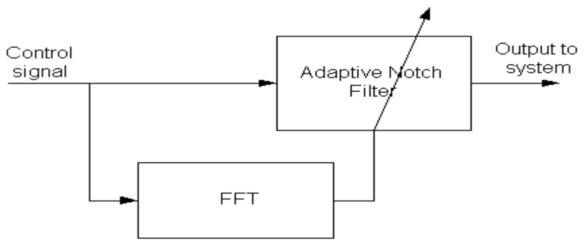


Fig. 8.6. Adaptive notch filter

Interested readers may refer to [2], [17] for full details on the derivation and other aspects of the adaptive notch filter.

8.2.8.2 Real Time Monitoring Device

A notch filter is a dynamical element. It will inevitably change and complicate the original system dynamics. Apart from the targeted signal frequencies to be eliminated, the filter affects also other signal frequencies. This may cause a deterioration in control performance. An alternative towards vibration monitoring can be to use a separate monitoring entity working in real-time to continuously derive and analyse vibration signals.

The main idea behind this approach is to construct a vibration signature based on pattern recognition of ``acceptable" or ``healthy" vibration patterns. The device is expected to enter an initial learning mode, to yield a set of vibration signatures based on which the monitoring modes will operate. In the monitoring mode, with the machine under normal closed-loop control, the analyser only uses a naturally occurring vibration signal to deduce the condition of the machine. No test excitation is deliberately added to the input signal of the machine. More than one criterion may be used in the evaluation of the condition of the machine, and in which case, a fusion approach would generate a combined output (machine condition) based on the multiple inputs.

A possible block diagram of the monitoring device is shown in Figure 8.7. It consists of an accelerometer, which is mounted on the machine to be monitored. The accelerometer measures a multi-frequency vibration signal and transmits it to an intelligent DSP module, after performing appropriate signal conditioning. This module can be a standalone device, or one integrated to a *Personal Computer* (PC) host. The vibration analysis algorithm is downloaded to this DSP module. With this algorithm, it can establish as to whether the condition of the machine is within a pre-determined acceptable threshold. If the condition is determined to be poor, the DSP module will trigger an alarm to the operator, or automatically activate a corrective action (e.g., change the operating conditions of the machine, modify the parameters of the controller or shut down the machine).

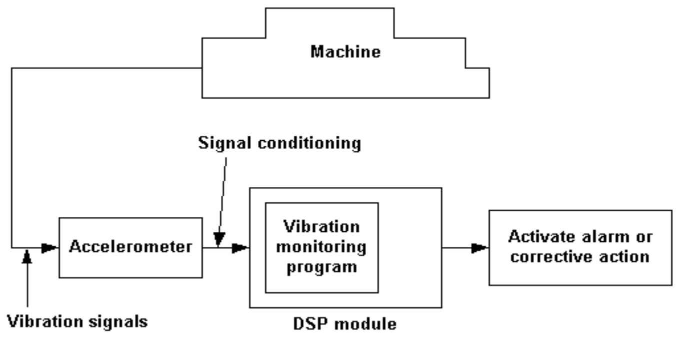


Fig. 8.7. Schematic of vibration monitoring device

The construction of the real-time vibration analyser is inexpensive and requires only commercially available, low cost components. The installation can be hassle-free, as the accelerometer is able to gather vibration signals, independent of the machine's own control system. Thus, there is no need to disrupt the operation of the machine. For more details on the development of the device as well as possible fusion techniques used in the analysis, the readers may refer to [17].

8.2.9 Geometrical Error Compensation

In automated positioning machines such as Co-ordinate Measuring Machines (CMMs) and machine tools, the relative position errors between the end-effector of the machine and the workpiece directly affect the quality of the final product or the process concerned. These positioning inaccuracies arise from various sources, including static/quasi-static sources such as geometrical errors from the structural elements, tooling and fixturing errors, thermally induced and load induced errors, and also dynamic ones due to the kinematics of the machine. These errors may be generally classified under two main categories:

- systematic errors which are completely repeatible and reproducible, and
- apparently random errors which vary under apparently similar operating conditions.

Although a complete elimination of machine errors is physically unachievable, these errors may be reduced to a level which is adequate for the particular application of the machine with a sufficiently high investment in machine design and construction. It is widely reckoned that for an increase in the precision requirements, the corresponding increase in cost will be far steeper. Thus, rather than relying solely on the precision design and construction of the machine which is expensive, this performance-cost dilemma set the motivation for a corrective approach instead in the form of an appropriate error compensation in the machine control to achieve comparable machine precision at a much reduced cost.

The basis of all soft compensation approaches is a geometrical error model which relates the positioning error to the measured position of a focused point. In this section, these relations will be given for commonly encountered configurations of positioning stages.

8.2.9.1 Single Axis Stage

For this simple stage, the linear motion along one axis and the linear error resulting along this axis arising (e.g., inherent encoder calibration errors) are of interest. Consider the single axis stage as shown in Figure 8.8, moving along the *X* direction. When the focused point (*P*) translates from the origin *O* to a nominal distance *OP*, it follows that

$$\overrightarrow{OP} = x + \delta x,$$

where x is the desired position and δx represents the linear error along x axis. The geometrical error along the x is therefore given by

$$\Delta x = \delta x .$$

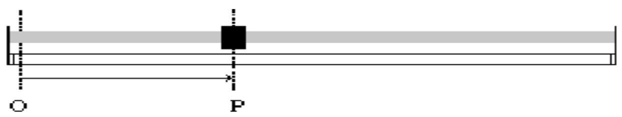


Fig. 8.8. Single axis stage

8.2.9.2 Dual Axis (Gantry) stage

In some installations, one axis is controlled by two drives and two feedback control systems, e.g., and H-type gantry stage. In this instance, the second axis brings about another source of linear displacement error. It is necessary to do calibration for this additional error source.

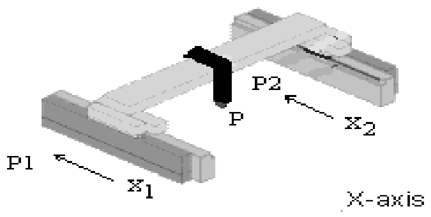


Fig. 8.9. Dual axis stage

Consider a dual axis stage as shown in Figure 8.9, moving a focus point along an axis which is still X in this example. The focus point moves from the origin O_1 to O_1P, while the other axis moves from the origin O_2 to O_2P. It follows that

$$\overrightarrow{O_1P} = \overrightarrow{O_1P_1} + \overrightarrow{P_1P},$$

and

$$\overrightarrow{O_2P} = \overrightarrow{O_2P_2} + \overrightarrow{P_2P},$$

for the x_1, x_2 axis respectively. Thus,

$$\overrightarrow{O_1P} = x + \delta(x_1) + \delta(x_1, x_2),$$
$$\overrightarrow{O_2P} = x + \delta(x_2) + \delta(x_1, x_2),$$

where $\delta(x_1), \delta(x_2)$ are the linear errors of x_1, x_2 axes, respectively, and $\delta(x_1, x_2)$ refers to the cross-coupled linear error arising after each individual axis error is calibrated. A two-phase calibration process may thus be used for such a dual-axis stage. During phase 1, the individual error along either axis ($\delta(x_1), \delta(x_2)$) is first calibrated. Then the axes are individually compensated. After compensation of the individual axis, Phase 2 of calibration will seek to derive the cross-coupled linear error $\delta(x_1, x_2)$ and subsequently compensate for it.

8.2.9.3 General XY stage

A general XY stage with three independent planar systems is shown in Figure 8.10. The planar systems are associated, respectively, with the table (*0, X, Y*), the bridge (O_1, X_1, Y_1), and the carriage (O_2, X_2, Y_2). For conceptual purposes, the measurement systems for the bridge and carriage are shown in Figure 8.10 as being attached to the bridge and X carriage respectively, via small, non-existent connecting rods. It will be assumed that, initially, all three origins coincide and the axes of all three systems are aligned. Thus, when the bridge moves a nominal distance Y, the actual position of the bridge origin O_1, with respect to the table system, is given by the vector:

$$\overrightarrow{OO_1} = \begin{pmatrix} \delta_x(y) \\ y + \delta_x(y) \end{pmatrix}, \tag{8.19}$$

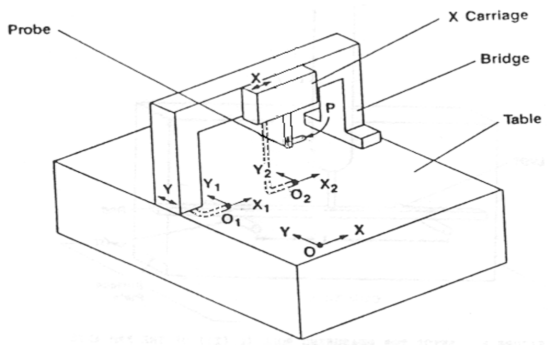


Fig. 8.10. XY stage

At the same time, the bridge coordinate system rotates with respect to the table system due to the angular error motion. This rotation can be expressed by the matrix:

$$R_1 = \begin{pmatrix} 1 & \varepsilon_y \\ -\varepsilon_y & 1 \end{pmatrix}, \tag{8.20}$$

Similarly, when the *X* carriage moves a nominal distance *X*, it follows that

$$\overrightarrow{O_1O_2} = \begin{pmatrix} x + \delta_x(x) \\ \delta_y(x) - \alpha x \end{pmatrix}, R_2 = \begin{pmatrix} 1 & \varepsilon_x \\ -\varepsilon_x & 1 \end{pmatrix} \tag{8.21}$$

$$\overrightarrow{O_2P} = \begin{pmatrix} x_p \\ y_p \end{pmatrix}, \tag{8.22}$$

where *x, y* are the nominal positions; x_p, y_p represent the offsets of the tool tip (Abbe error); $\delta_u(v)$ is the translational error along the u-direction under motion in the v direction; ε_u refers to the rotation along the u axis; and α represents the out-of-squareness error. Therefore, a volumetric error model can be derived with respect to the table system:

$$\overrightarrow{OP} = \overrightarrow{OO_1} + R_1^{-1}\overrightarrow{O_1O_2} + R_1^{-1}R_2^{-1}\overrightarrow{O_2P} \tag{8.23}$$

Substituting (8.19)-(8.22) into (8.23) and noting that $\varepsilon_u\varepsilon_v \approx 0, \varepsilon_u\delta_u(v) \approx 0, \varepsilon_u\alpha \approx 0$ since $\varepsilon_u, \delta_u(v), \alpha$ are very small, the geometrical error compensation along the *x* and *y* directions are respectively:

$$\Delta x = \delta_x(x) + \delta_x(y) - y_p(\varepsilon_x + \varepsilon_y) + x_p$$

$$\Delta y = \delta_y(x) + \delta_y(y) - x_p(\varepsilon_y + \varepsilon_x) + y_p$$

This is a 2D error model. For a more general 3D error model, readers can review in [7, 19] for a detailed presentation. It should be noted that the error sources are all calibrated using only appropriate combinations of linear displacement measurements.

Common to a geometrical error compensator is a model of the machine errors, which is either implicitly or explicitly used in the compensator. A common mode of modelling these errors is via a look-up table which will store the positional offsets. This table will store the overall positional offsets arising from the individual geometrical error components obtained through a typical laser and electronic leveller calibration exercise. For a 3D cartesian machine, there are 21 sources of geometrical errors associated with linear, angular, straightness and squareness errors. Figure 8.11 shows a 2-dimensional look-up mapping along X-Y axes, where the assumed ideal geometrical properties are mapped to the actual ones.

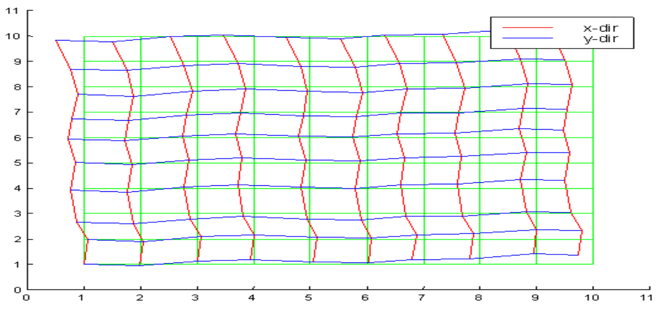


Fig. 8.11. 2-D geometrical error map

The look-up table is built based on points collected and calibrated in the operational working space of the machine. Among the limitations, a look-up table incurs an extensive memory space, is incapable of nonlinear interpolation, and it possesses a rigid structure which is not amenable to consider other factors which may cause the geometrical error model to change, such as ambient temperature and humidity. Dispensing with the look-up table, a radial basis function (RBF) error model based on the calibrated points [7] has been developed to serve as the basis for error compensation. The overall error can then be directly computed from the output of these RBFs based on a geometrical overall model for the machine in point. Figure 8.12 shows the adequacy of using a RBF in the modeling of the linear error along the X axis of an XY table. A multilayer artificial neural networks (ANN), as another possible geometric error model, has also been explored. Details may be found in [7], [10].

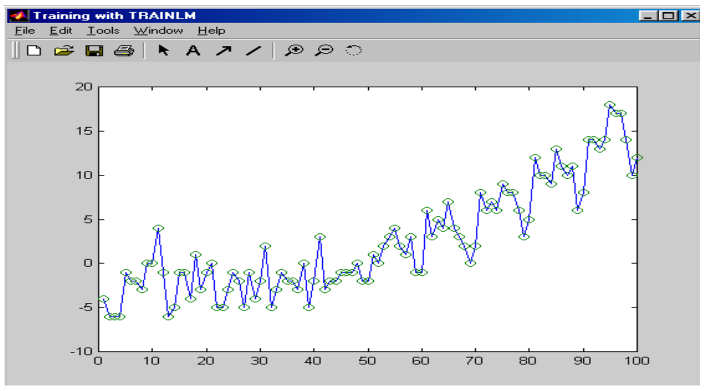


Fig. 8.12. RBF modeling of the linear error

8.3 Implementation

The final implementation of the overall control system is a non-trivial and important process. This development process can be time consuming, leading many to simply settle for off-the-shelves proprietary solutions which may not satisfy all requirements specific to the particular application.

Manually line programming a control system from scratch requires an enormous amount of time and effort to be spent. The high susceptibility in coding errors causes further delay to the development process. Thus, the flexibility, quality, functionality and development time are crucial factors driving the selection of the hardware and software development platform for the control system. In this case, the dSPACE development platform is selected due to three main features and provisions: rapid control prototyping, automatic production code generation, and facilities for hardware-in-the-loop testing.

Rapid control prototyping implies that new and customised control concepts can be directly and quickly developed, and optimised on the real system via the rich set of standard design tools and function blocks available in MATLAB/SIMULINK. Controllers can be directly and graphically designed in the form of functional block diagrams with little or no line programming necessary. Real-time code can be automatically generated from the functional block diagram and implemented on the machine through the automatic production code generation feature provided. The hardware-in-the-loop facilities further allow for a reliable and cost-effective method to perform system tests in a virtual environment. Peripheral components can be replaced by proven working mathematical models, while the actual physical components to be evaluated are inserted systematically into the loop. In addition to savings in time and costs, the modularity and reproducibililty associated with hardware-in-the-loop simulation greatly simplifies the entire development and test process.

In this section, the hardware and software of the system will be described in subsections 8.3.1 and 8.3.2. The user interface for the overall system will also be briefly illustrated in subsection 8.3.3.

8.3.1 Hardware Architecture

The overall system hardware architecture is shown in Figure 8.13. To meet simultaneous high speed and high precision requirements, the control unit is configured with high speed processing modules. A dSPACE DS1004

DSP board is used together with a DS1003 DSP board. The DS1004 DSP board uses a DEC Alpha AXP 21164 processor capable of 600 MHz/1200 MFlops. This board is used to fully concentrate on the computationally intensive tasks associated with control algorithms execution. The DS1003 DSP board uses the TMS320C40 DSP which is capable of 60 MFlops. It can effectively deal with all the I/O tasks because of its high-speed connection to all I/O boards via the Peripheral High-Speed (PHS) Bus.

In addition to the processor boards, a DS2001 board is used which has five parallel high-speed 16 bit A/D channels. The sampling and holding of signals along all channels can be executed simultaneously, with a short sampling time of 5.μs A DS2102 high-resolution D/A board is used to drive the actuators. It has six parallel D/A channels, each with a 16-bit resolution. The typical settling time (full scale) is 1.3-2 μs and output voltage ranges (programmable) of ± 5 V, ± 10 V, or 0-10 V are all supported.

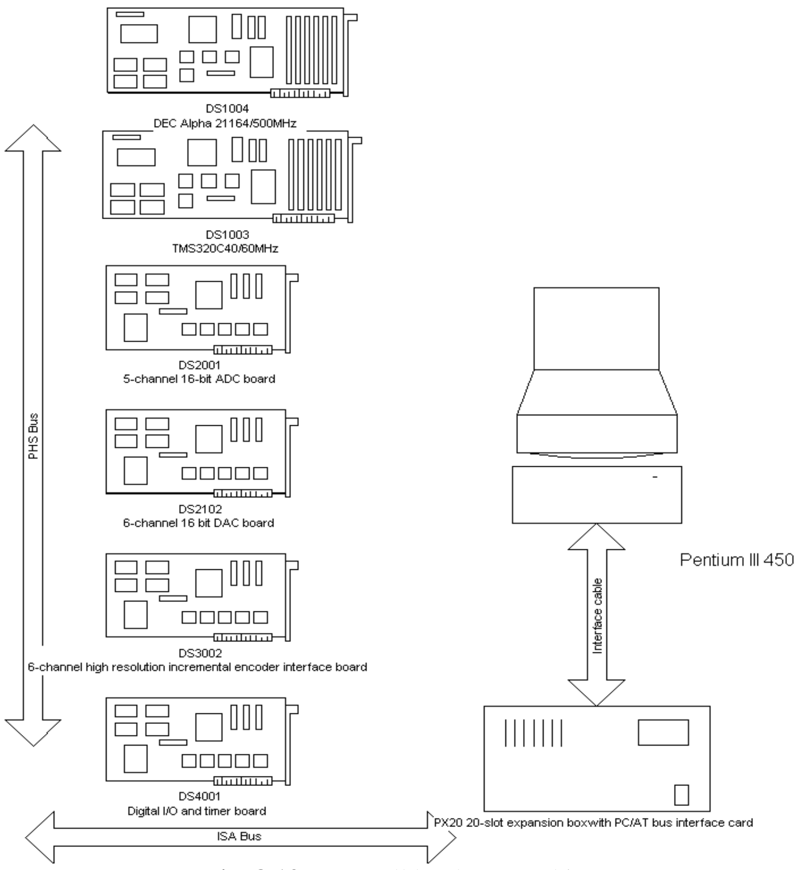

DS1004
DEC Alpha 21164/500MHz
DS1003
TMS320C40/60MHz
DS2001
5-channel 16-bit ADC board
PHS Bus
DS2102
6-channel 16 bit DAC board
DS3002
6-channel high resolution incremental encoder interface board
DS4001
Digital I/O and timer board
ISA Bus
Pentium III 450
Interface cable
PX20 20-slot expansion boxwith PC/AT bus interface card

Fig. 8.13. Overall hardware architecture

To allow fine measurement resolution via analog incremental optical encoders, the DS3002 incremental encoder interface board with a maximum input frequency of 750 kHz is chosen. Sinusoidal encoder signals are captured through six channels in DS3002, converted to 12 bits digital signals and then phase decoded by special highly optimised software functions to extract the relative position from these data. A search block will seek the encoder index lines and updates the corresponding counter when a new index is reported to give an absolute position information. Theoretically, in this way, an interpolation of 4096 can be achieved. This in turns implies that a measurement resolution of less than 1 nm can be achieved if the grating-line pitch is 4 μm. However, one should be cautious of the constraints in terms of interpolation errors associated with limited wordlength A/D operations, and imperfect analog encoder waveform with mean, phase offsets, noise as well as non-sinusoidal waveform distortion. The interested readers may refer to [15] for more details on these aspects and possible remedial measures.

A timer and digital I/O board, DS4001, with 32 in/out channels is used for status checking of limit switches and other safety enhancing digital devices. The 32 in/out channels can be divided into 8-bit groups.

8.3.2 Software Development Platform

The processor boards are well supported by popular software design and simulation tools, including MATLAB and SIMULINK, which offer a rich set of standard and modular design functions for both classical and modern control algorithms. The overall SIMULINK control block diagram customised for a cartesian 3D gantry machine is shown in Figure 8.14. The block diagram can be divided into three parts according to their functions:

- control and automatic tuning,
- geometric error calibration and compensation, and
- safety features, such as emergency stops, limit switches, etc.

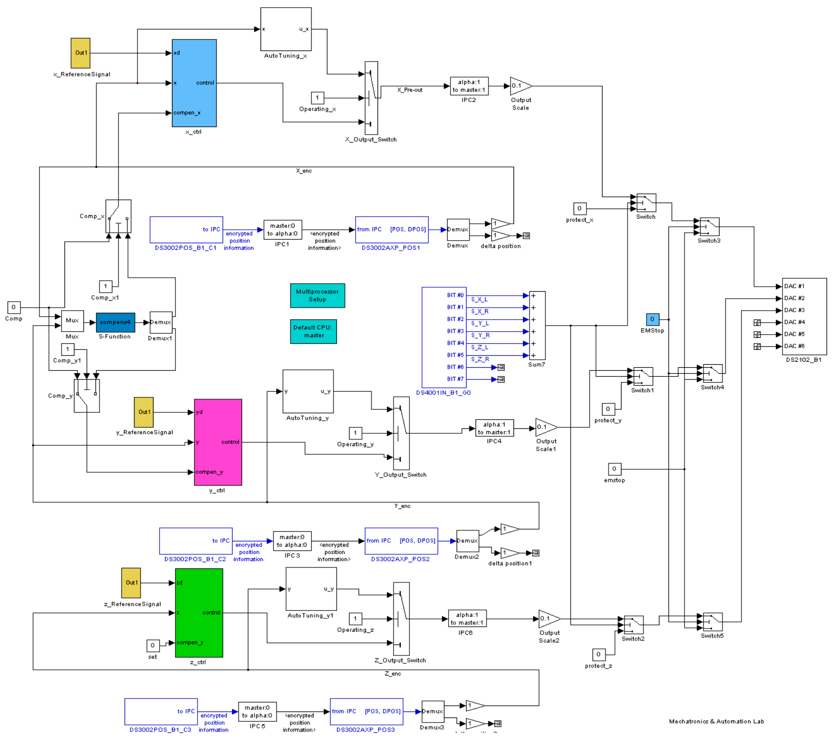


Fig. 8.14. Overall SIMULINK control block diagram

The control algorithms are included in the subsystem **x-ctrl**, **y-ctrl** and **z-ctrl.** Figure 8.15 shows the SIMULINK control block diagram for the x axis. Apart from the PID feedback control which is fixed, the other advanced control schemes are configurable by the operator. An automatic tuning operation mode is also provided for the controllers. The operation modes (control or automatic tuning) can be selected through the switch blocks **X-Output-Switch**, **Y-Output-Switch** and **Z-Output-Switch**.

The geometric error calibration and compensation for the axes are integrated with the controllers via an S-Function interface. These features are enabled through switches **Comp-x** and **Comp-y**, as shown in Figure 8.14.

All the limit switch signals from the three axes are acquired through DS4001 board. These limit switch signals serve as the control input of the three switches shown in Figure 8.14, to nullify the system control signal when the limit switch is activated. An operator emergency stop function is also provided in the overall SIMULINK control block diagram.

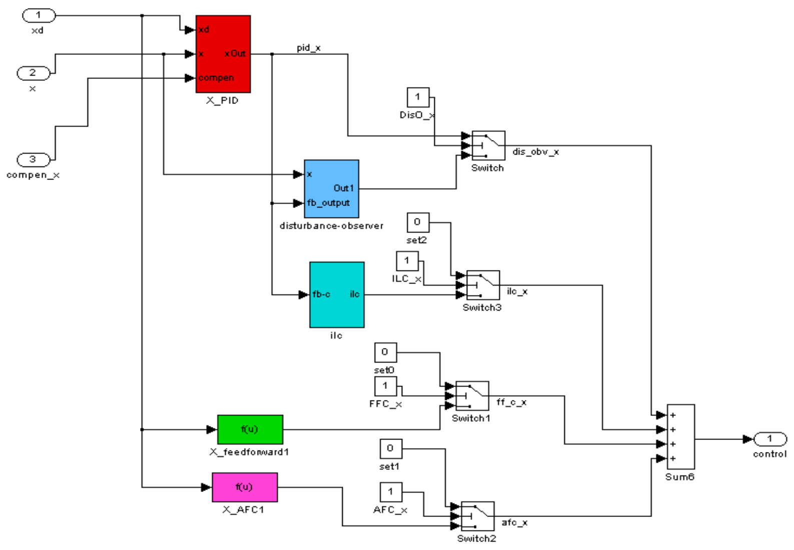


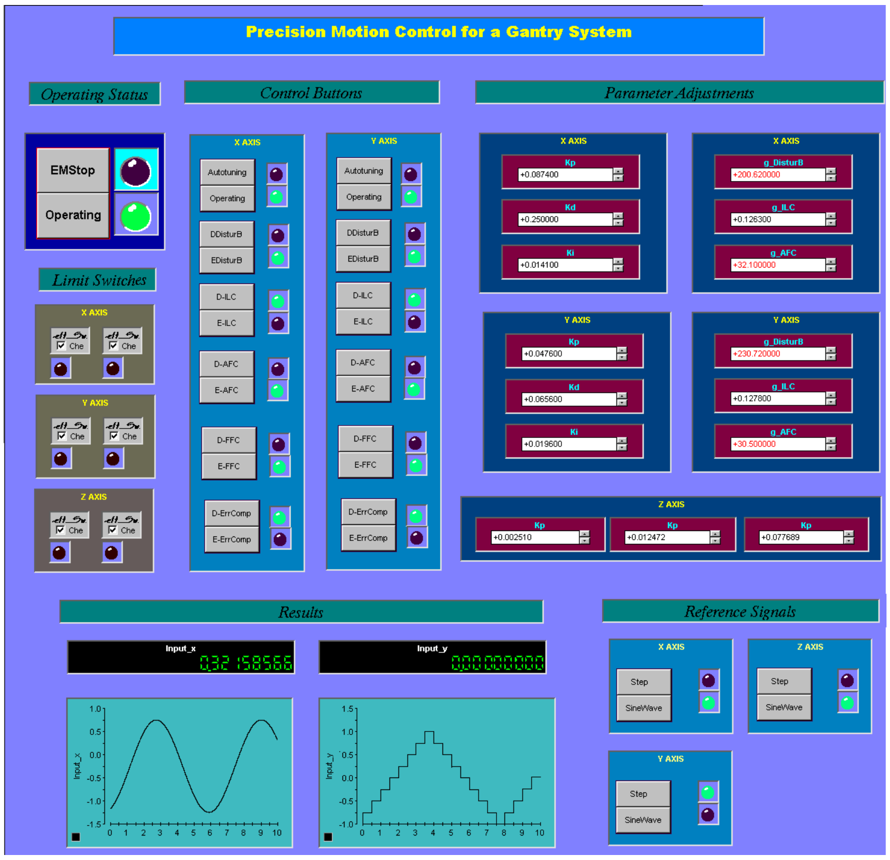
Fig. 8.15. The SIMULINK control block diagram for X axis

A software component, running on MATLAB/SIMULINK is written for the geometrical error compensation. Using this software, an S-function comprising RBF-based error compensation can be automatically produced given the raw data set obtained from the calibration experiments, and simple user inputs on the RBF training requirements. Thus, little prior technical knowledge of RBFs is required of the operator.

Upon a successful automatic code generation from the SIMULINK control block diagram, the controller will run on the dSPACE hardware architecture configured. The user interface, designed using dSPACE CONTROLDESK, allows for user-friendly parameters tuning/changing and data logging during the operations. The control parameters can be changed on-line, while the motion along all axes can be observed simultaneously on the display.

8.3.3 User Interface

The user interface is designed as a virtual instrument panel based on the dSPACE CONTROLDESK instrumentation tool. CONTROLDESK is a comprehensive design environment where designers can intuitively manage, instrument, and automate their experiments and operations. CONTROLDESK is seamlessly integrated within the dSPACE development platform. It can realise real time data acquisition, online parameter-isation and provide an easy access to all model variables without having to interrupt the running operations. The entire user interface design is achieved simply via drag and drop operations from the Instrument Selector provided. This greatly speeds up the design process and helps to avoid standard design pitfalls associated with line programming. Figure 8.16 shows the user interface customised for the gantry motion system.



Parameter Adjustments

X AXIS: Kp +0.087400; Kd +0.250000; Ki +0.014100
X AXIS: g_DisturB +200.620000; g_ILC +0.126300; g_AFC +32.100000
Y AXIS: Kp +0.047600; Kd +0.065600; Ki +0.019600
Y AXIS: g_DisturB +230.720000; g_ILC +0.127800; g_AFC +30.500000
Z AXIS: Kp +0.002510; Kp +0.012472; Kp +0.077689

Results

Input_x 0.32158566
Input_y 0.00000000

Fig. 8.16. User interface

8.4 Results

The precision motion control strategies developed have been applied to and tested on various systems, including ANORAD and Linear Drives (U.K.) servo systems based on PMLMs, as well as other more conventional servo systems. A high level of performance has been achieved in these applications and tests. For illustration purposes for this chapter, an extract of the results from the application of the control system to a Linear Drive direct thrust servo system is provided below. The system uses a 1 μm resolution encoder and it is driven by PWM amplifiers.

The tracking performance, given a sinusoidal type of reference trajectory, is shown in Figure 8.17 with the system under the control of the proposed system. A maximum tracking error of less than 7 μm is achieved. It should be pointed out that this is achieved with the encoder resolution of 1 μm. The controller performs satisfactorily even when a significant load disturbance (50 kg) is deliberately introduced into the system (*Box B* in Figure 8.17). For comparison purposes, *Box A* highlights the performance of the system before the introduction of the load disturbance. The changes in the control signal due to the introduction of disturbance are not reflected in the error signal. In other words, the control system is able to effectively reject the external disturbance and the performance is not significantly affected.

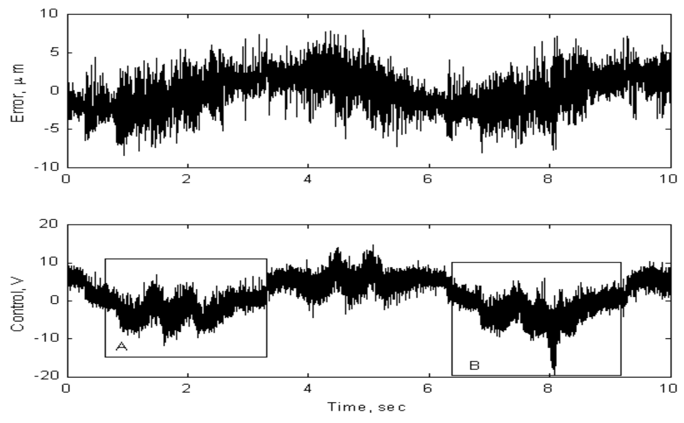


Fig. 8.17. Experimental results with proposed system

The control results achieved with an existing industrial control system are shown in Figure 8.18. The deliberate load disturbance introduced into the system is clearly manifested in the error signal (*Box B* in Figure 8.18).

A comparison between Figure 8.17 and Figure 8.18 shows the significantly better performance achieved using the developed control system. Furthermore, perhaps unnoticed by many, the implicit geometrical error compensator has achieved a four fold reduction in geometrical errors present in the system to help achieve the above results.

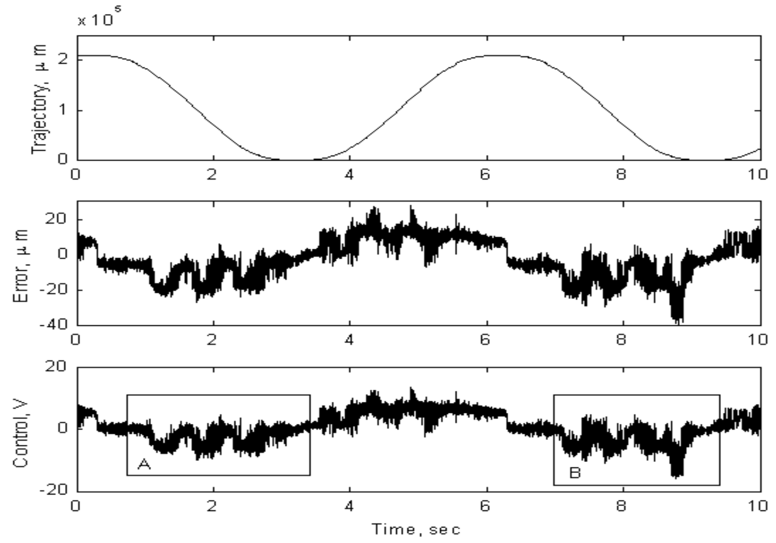


Fig. 8.18. Experimental results with an existing industrial control system (**a**) Desired trajectory (μm); (**b**) Error (μm); (**c**) Control signal (V)

8.5 Conclusions

The chapter has presented the development of an integrated and open-architecture precision motion control system. The control system is generally applicable, but it is developed with a particular focus on direct drive servo systems based on linear motors. The overall control system is comprehensive, comprising of various selected control and instrumentation components, integrated within a dSPACE DS1004 DSP board. These components include a precision composite controller, a disturbance observer, an adaptive notch filter, and a geometrical error compensator. The hardware architecture, software development platform, and the purpose and design of the constituent control components have been duly elaborated on in the chapter.

References

1 Basak A (1996) *Permanent-magnet DC Linear Motors*, Mongraphs in Electrical and Electronic Engineering, Oxford: Clarendon Press.

2 16Bertran E and Montoro G (1998) Adaptive Suppression of Narrow-band Vibrations*, 5th International Workshop on Advanced Motion Control*, 288-292.

3 Evans CJ (1989) *Precision Engineering: An Evolutionary View*, Cranfield Press, Cranfield, UK.

4 Heydemann PLM (1981) Determination and Correction of Quadrature Fringe Measurement Errors in Interferometer, *Applied Optics*, 20, Octoter 1981.

5 Longman RW (1998) Designing Iterative Learning and Repetitive Controllers, In Z. Bien and Xu J.-X. eds, *Iterative Learning Control - Analysis, Design, Integration and Application*, Kluwer Academic Publishers, 107-145.

6 Tan KK, Dou HF, Chen YQ and Lee TH (2001) High Precision Linear Motor Control via Relay Tuning and Iterative Learning based on Zero-Phase Filtering, *IEEE Transactions on Control Systems Technology*, 9: 244-253.

7 Tan KK, Huang SN and Seet HL (2000) Geometrical Error Compensation of precision motion systems using radial basis functions, *IEEE Transactions on Instrumentation and Measurement*, 49: 984-991.

8 Tan KK, Huang SN, Lee TH, Chin SJ and Lim SY (2001) Adaptive Robust Motion Control for Precise Trajectory Tracking Applications, *ISA Transactions*, 40: 17-29.

9 Tan KK, Huang SN and Lee TH (2002) Robust Adaptive Numberical Compensation for Friction and Force Ripple in PMLM, *IEEE Trans.on Magnetics*, 38: 221-228.

10 Tan KK, Huang SN and Lee TH (2004) Geometrical Error Compensation of Precision Motion Systems Using Neural Networks, *Control Engineering Practice: Under Review*.

11 Tan KK, Lee TH, Dou HF and Chin SJ (2000) PWM Modelling and Application to Disturbance Observer-Based Precision Motion Control,*PowerCon 2000, Perth, Australia*, 3:1669-1674.

12 Tan KK, Lee TH, Dou HF and Huang SN (2001) *Precision Motion Control: Design and Implementation*, Springer-Verlag: London.

13 Tan KK, Lee TH, Dou HF and Lim SY (2000) An Adaptive Ripple Suppression/Compensation Apparatus for Permanent Magnet Linear Motors, Technical Report- Department of ECE, National University of Singapore, 2000.

14 Tan KK, Lee TH, Huang SN and Jiang X (2001) Friction Modeling and Adaptive Compensation Using a Relay Feedback Approach, *IEEE Transactions on Industrial Electronics*, 48: 169-176.

15 Tan KK, Lee TH and Zhou HX (2002) New Interpolation Method for Quadrature Encoder Signals, *IEEE Transactions on Instrumentation and Measurement*, 51: 1073-1079.

16 Tan KK, Lim SY, Lee TH and Dou HF (2001) High Precision Control of Linear Actuators Incorporating Acceleration Sensing, *Robotics and Computer-Integrated Manufacturing*, 16: 295-305.

17 Tang KZ, Tan KK, de Silva CW, Lee TH and Chin SJ (2002) Monitoring and Suppression of Vibration in Precision Machines, *Journal of Intelligent and Fuzzy Systems*, 11: 33-52.

18 Yamada K, Komada S, Ishida M and Hori T (1997) Analysis and Classical Control Design of Servo Systems using High Order Distubance Observer, *23rd International Conference on Industrial Electronics, Control and Instrumentation IECON 97*, 1: 4-9.

19 Zhang G, Veale R, Charlton T, Hocken R and Borchardt B (1985) Error Compensation of Coordinate measuring machines, *Annals of the CIRP*, 34: 445-448.