Mobile Robot Navigation using a Vision based Approach

by

Mehmet Serdar Güzel

2012

The thesis is submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy

School of Mechanical and Systems Engineering

Newcastle University

United Kingdom

This is dedicated to my grandmother and son.

ABSTRACT

This study addresses the issue of vision based mobile robot navigation in a partially cluttered indoor environment using a mapless navigation strategy. The work focuses on two key problems, namely vision based obstacle avoidance and vision based reactive navigation strategy.

The estimation of optical flow plays a key role in vision based obstacle avoidance problems, however the current view is that this technique is too sensitive to noise and distortion under real conditions. Accordingly, practical applications in real time robotics remain scarce. This dissertation presents a novel methodology for vision based obstacle avoidance, using a hybrid architecture. This integrates an appearance-based obstacle detection method into an optical flow architecture based upon a behavioural control strategy that includes a new arbitration module. This enhances the overall performance of conventional optical flow based navigation systems, enabling a robot to successfully move around without experiencing collisions.

Behaviour based approaches have become the dominant methodologies for designing control strategies for robot navigation. Two different behaviour based navigation architectures have been proposed for the second problem, using monocular vision as the primary sensor and equipped with a 2-D range finder. Both utilize an accelerated version of the Scale Invariant Feature Transform (SIFT) algorithm. The first architecture employs a qualitative-based control algorithm to steer the robot towards a goal whilst avoiding obstacles, whereas the second employs an intelligent control framework. This allows the components of soft computing to be integrated into the proposed SIFT-based navigation architecture, conserving the same set of behaviours and system structure of the previously defined architecture. The intelligent framework incorporates a novel distance estimation technique using the scale parameters obtained from the SIFT algorithm. The technique employs scale parameters and a corresponding zooming factor as inputs to train a neural network which results in the determination of physical distance. Furthermore a fuzzy controller is designed and integrated into this framework so as to estimate linear velocity, and a neural network based solution is adopted to estimate the steering direction of the robot. As a result, this intelligent

approach allows the robot to successfully complete its task in a smooth and robust manner without experiencing collision.

MS Robotics Studio software was used to simulate the systems, and a modified Pioneer 3-DX mobile robot was used for real-time implementation. Several realistic scenarios were developed and comprehensive experiments conducted to evaluate the performance of the proposed navigation systems.

KEY WORDS: Mobile robot navigation using vision, Mapless navigation, Mobile robot architecture, Distance estimation, Vision for obstacle avoidance, Scale Invariant Feature Transforms, Intelligent framework.

ACKNOWLEDGEMENTS

It is a pleasure to thank many people who have made this thesis possible. First of all, I'd like to thank to my Ph.D. supervisor Dr Robert Bicker for all of his ideas, encouragement, and excellent writing tips. He has always been eager to help me through the toughest challenges during my research. I also would like to thank my second supervisor Dr John Hedley for his support.

I sincerely thank my colleagues who have offered encouragement along this long way, especially all members of the robotics lab, and my old mate Yaskil for programming tips.

I would like to express my deep gratitude to my parents for their endless support, encouragement and financial assistance.

Finally, heartfelt thanks are due to my wife Eda for all she has done over the past number of years. Her love, kindness and support make life as pleasurable as it is.

LIST OF CONTENTS

ABSTRACT ___ *iii*

ACKNOWLEDGEMENTS ___ *v*

LIST OF CONTENTS ___ *vii*

CHAPTER 1 INTRODUCTION ___ *1*

1.1 Overview ___ **1**

1.2 Aims and Objectives ___ **2**

1.3 Hypothesis ___ **3**

1.4 Contributions ___ **3**

1.5 Overview of the Thesis ___ **4**

CHAPTER 2 LITERATURE REVIEW ___ *6*

2.1 History and Categories of Mobile Robots ___ **6**

2.1.1 Trends in mobile robots ___ 7

2.1.2 Categories of mobile robots ___ 8

2.2 Mobile Robot Control Architecture ___ **9**

2.2.1 Reactive/Behaviour based architecture ___ 10

2.2.1.1 Subsumption architecture ___ 11

2.2.1.2 Motor schema ___ 12

2.2.2 Hybrid architecture ___ 13

2.3 Mobile Robot Navigation ___ **14**

2.4 Vision Based Mobile Robot Navigation ___ **15**

2.4.1 Map-based navigation ___ 17

2.4.2 Map-building-based navigation ___ 19

2.4.3 Mapless navigation ___ 20

2.4.3.1 Optical flow techniques for navigation ___ 20

2.4.3.2 Appearance-based methods ___ 22

2.4.3.3 Object recognition ___ 26

2.4.3.4 Navigation techniques based on feature tracking ___ 26

2.5 Obstacle Avoidance Systems based on Qualitative Information ___ **29**

2.6 Scale Invariant Feature Transform (SIFT) ___ **31**

2.7 Sensor Theory and Vision Based Sensors ___ **32**

2.7.1 Active ranging sensors ___ 32

2.7.2 Vision based sensors ___ 33

2.8 Mobile Robot Software 34
2.8.1 Player architecture 34
2.8.2 Microsoft Robotics Studio (MSRS) 36
2.9 Software for Computer Vision 36
2.10 Soft Computing 38
2.10.1 Fuzzy logic 38
2.10.1.1 Fuzzification 40
2.10.1.2 Fuzzy rule-base 43
2.10.1.3 Fuzzy inference 44
2.10.1.4 Defuzzification 46
2.10.2 Neural Network 48
2.10.2.1 Background to Artificial Neural Network 49
2.10.2.2 Topologies of artificial neural networks 51
2.10.2.3 Learning using neural networks 53
2.10.2.4 Back-propagation algorithm 54
2.11 Summary 56
CHAPTER 3 VISION BASED OBSTACLE AVOIDANCE *57*
3.1 Vision Based Obstacle Avoidance Techniques 57
3.2 Optical Flow for Obstacle Avoidance 58
3.2.1 Horn-Schunk method for obstacle avoidance 60
3.2.1.1 Estimating the partial derivatives 62
3.2.1.2 Minimization 63
3.2.1.3 Multi-scale optical flow estimation 65
3.2.2 Applying optical flow for obstacle avoidance 67
3.2.2.1 FOE and TTC calculation 68
3.2.2.2 Behavioural module and implementation algorithm 70
3.2.3 Evaluation of flow vectors for mobile robot navigation 72
3.3 Appearance Based Methods for Obstacle Avoidances 75
3.3.1 Template matching 78
3.3.2 Implementation of obstacle avoidance technique using appearance based approach 79
3.4 Integration of Appearance Based Method with Optical Flow Architecture 82
3.5 Modelling and Simulation using Microsoft Robotics Studio 90
3.6 Summary 100
CHAPTER 4 VISION BASED MOBILE ROBOT NAVIGATION USING SIFT *102*
4.1 Local Features 102
4.2 Scale-Invariant Feature Transform 105
4.2.1 Scale-space extreme detection 107
4.2.2 Keypoint localization and edge elimination 110
4.2.3 Orientation assignment 113
4.2.4 Keypoint descriptor 114
4.2.5 SIFT matching 115
4.3 Evaluation of the SIFT Algorithm 118

4.4 Navigation via SIFT based on Monocular Vision 119

4.5 Design of a Reactive Architecture using Subsumption Architecture 126

4.5.1 Goal seeking 128

4.5.2 Approach 129

4.5.3 Wander 129

4.5.4 Obstacle avoidance 130

4.5.5 Completed 133

4.6 Prediction of the heading angle 134

4.7 Modelling and Simulation using Microsoft Robotics Studio 135

4.8 Summary 145

CHAPTER 5 INTELLIGENT NAVIGATION USING SIFT *147*

5.1 Design of an Intelligent Framework for Vision Based Mobile Robot Navigation 148

5.2 Robust Estimation of Heading Direction of a Mobile Robot using ANN and Linear Regression 151

5.2.1 Conventional method for camera calibration 151

5.2.2 The proposed algorithm to estimate heading direction 152

5.2.2.1 Assessment of scale-invariant features 154

5.2.2.2 The estimation of heading direction using ANN 156

5.2.2.3 Linear regression technique for calibration 160

5.3 Scale Parameters for Distance Estimation Based on ANN 161

5.4 ANN based Approach for Obstacle Avoidance 169

5.5 Estimation of Global Linear Velocity using Fuzzy Logic 175

5.5.1 Design of membership functions for the linear velocity controller 176

5.5.2 Defining the defuzzification method 179

5.6 Design of behaviours based on Subsumption Architecture 180

5.7 Modelling and Simulation using Microsoft Robotics Studio 182

5.8 Summary 189

CHAPTER 6 ROBOT CONFIGURATION AND SOFTWARE DESIGN *190*

6.1 IWARD Project 190

6.2 IWARD Pioneer Robot (Pioneer 1) 191

6.2.1 Robot sensors and peripheral device design 194

6.2.1.1 AXIS-213 pan/tilt/zoom camera 197

6.3 Software Design 198

6.3.1 Software tools and libraries used in this project 201

6.3.2 Performance analysis of Fast SIFT library 209

6.4 Calibration analysis of the sensors for the INUS technique 211

6.5 Summary 214

CHAPTER 7 IMPLEMENTATION AND EVALUTION OF PROPOSED NAVIGATION

SYSTEMS 216

7.1 Experimental Procedures 216

7.1.1 Performance evaluation 217

7.2 Experimental Design and Results of the Hybrid Vision Based Obstacle Avoidance System 222

7.2.1 Definition of scenarios 223

7.2.2 Navigation test results 223

7.2.3 Comparison and evaluation of methods 230

7.3 Experimental Design and Results of SIFT based Navigation Systems 233

7.3.1 Experimental Implementation 236

7.3.1.1 Preliminary test results 236

7.3.1.2 Complex test results 244

7.3.2 Performance Analysis 261

7.4 Summary 265

CHAPTER 8 CONCLUSIONS AND FUTURE WORK 267

8.1 Conclusions 267

8.2 Summary of Achievements 272

8.3 Recommendations for Future Work 273

8.4 Publications 275

REFERENCES 277

APPENDIX A: Optical flow vectors with high resolution 290

APPENDIX B: Specification of Corobot mobile robot 292

APPENDIX C: Wander behaviour 293

APPENDIX D: Back-propagation algorithm 294

APPENDIX E: Camera calibration with conventional method 295

APPENDIX F: Specification of Pioneer 3-DX 298

APPENDIX G: Platform for the AXIS-213 camera 300

APPENDIX H: Specification of URG-04LX laser range finder 301

APPENDIX I: Specification of LinITX 8.4" Touch-Screen 302

APPENDIX J: Specification of AXIS 213 camera 303

APPENDIX K: Evaluations of goals via SIFT algorithm 304

APPENDIX L: Goal tracking example via calibrated camera 309

APPENDIX M: The training results for the simulated camera 311

APPENDIX N: Control outputs of output scenarios 313

CHAPTER 1

INTRODUCTION

The popularity of autonomous mobile robots has been rapidly increasing due to emerging areas of application. New markets for these types of robotics systems include room cleaning, tourist guidance, and entertainment applications. However existing applications of autonomous systems have one problem in common, which is navigation. Mobile robot navigation entails solutions to different problems, including planning, localisation, and obstacle avoidance. If the working environment is not already known or may vary over time, such as in households or offices, the navigation problem becomes far more difficult.

To overcome such problems, the system needs to use sensory data to extract representations of the environment and/or estimate the position of goals. Data is interpreted by the robot's control system so as to fulfil the navigation task using an appropriate strategy. However, the development of a satisfactory control algorithm to allow autonomous mobile robots to navigate safely in these environments is still an open research problem. Vision is capable of supplying the robot with detailed information from its environment. It is essential for the design of mobile robots to progress in the directions of increased robustness and reduced costs. Mapless strategies and methodologies developed so far resemble human behaviour more than other approaches, and have become applicable to any indoor environment consisting of corridors and doorways which can be accessed by a mobile robot platform.

1.1 Overview

The focus of this thesis is the study of vision based mobile robot navigation in an indoor environment using a mapless strategy. It addresses different aspects, including visual feature tracking for navigation and navigation based on artificial intelligence

techniques, obstacle avoidance, localisation and control architecture. Three different navigation systems are proposed based on a mapless strategy. In the first system a hybrid vision based obstacle avoidance architecture is proposed which combines conventional optical flow and appearance-based methods, based upon a behavioural strategy. The proposed architecture navigates the robot using monocular vision in a partially cluttered indoor environment where the robot may encounter unknown obstacles that prevent it from moving forward safely, and thus it requires the capability to detect objects and avoid collisions.

In the second system, a vision based behavioural architecture is proposed to overcome the mapless navigation problem. The architecture comprises several modules, which facilitate the robot's navigation and ensure that it maintains a safe distance from obstacles while finding goals from its current position. The highest level of the architecture is based on extracting and tracking scale invariant features. The third system is also based on a behavioural architecture, but unlike the second system, it draws its inspiration from various disciplines in providing an intelligent solution. This allows the mobile robot to successfully avoid obstacles whilst maintaining its progress to its final goal.

1.2 Aims and Objectives

The overall goal of this research is to design and develop systems which can be used for mapless navigation problems in indoor environments. The research focuses on two important aspects of vision based mapless navigation. The first aim addresses vision based obstacle avoidance using monocular vision, and the second concerns the design of a robust and safe navigation system for mobile robots using vision as a preliminary sensor which incorporates goal-based navigation, and collision avoidance.

In order to satisfy the above goal and associated aims, the following objectives have been identified.

- To undertake an in-depth critical review of available reactive vision based obstacle avoidance and navigation algorithms, including classification based on localization techniques, mobile robot control architectures and the relevant soft computing techniques.
- To design a hybrid vision based obstacle avoidance system, integrating a conventional appearance-based obstacle detection method into an optical flow based navigation architecture.
- To develop a behaviour based navigation control algorithm using scale invariant features so as to navigate a mobile robot via a feature tracking approach.
- To apply soft computing techniques to provide on improved navigation system.
- To assess the performance of the developed navigation systems using both simulation and physical experiments.

1.3 Hypotheses

Two hypotheses are tested in this research. The first is that "it is possible to develop a vision based obstacle method, using a single monocular vision camera as the only sensor to allow a mobile robot to navigate safely". The second hypothesis is that "it is possible to develop vision based mapless navigation using a behaviour based framework to allow a robot to safely complete its tasks in a robust and smooth manner".

1.4 Contributions

A robust and novel vision based obstacle avoidance algorithm has been developed and implemented to enable safer indoor navigation. This combines a conventional appearance-based obstacle detection method and an optical flow based navigation system into a hybrid architecture.

A feature based navigation technique using the accelerated version of the SIFT algorithm has been developed and implemented to navigate a mobile robot towards its goal. This technique is integrated into a reactive behavioural architecture which

coordinates individual behaviours to allow the robot to complete its tasks in a partially cluttered environment, and artificial intelligence techniques have been incorporated into the developed navigation framework in order to improve the navigation performance. A feed-forward neural network was developed to compute the steering parameter according to the estimated error in the image space. In addition, a fuzzy inference system has been proposed and integrated into the proposed intelligent navigation framework to adjust the second control parameter, linear velocity. A third contribution of this thesis is a novel distance estimation method using monocular vision systems which employ scale parameters from the SIFT algorithm to train a feed-forward neural network. The output of the network generates the physical distance in meters.

Finally, a rigorous series of experiments have been conducted to demonstrate the functionality of the proposed navigation systems using realistic scenarios to evaluate the robot's performance.

1.5 Overview of the Thesis

Chapter 1 provides an overview of the work and sets out the aim and objectives of the study. Chapter 2 is a review of the relevant literature, and of the background work that forms the foundations of this thesis. The development of a novel vision based obstacle avoidance architecture which integrates a high performance appearance based obstacle detection method with conventional optical flow based navigation architecture is addressed in Chapter 3.

Chapter 4 addresses the SIFT (Scale Invariant Feature Transformation) algorithm and its adaption to a monocular vision based navigation strategy together with a set of developed robot behaviours. The fundamentals of fuzzy logic theory and artificial neural networks, including a brief description of their main components are outlined in Chapter 5. This chapter primarily focuses on applying soft computing techniques to enhance the performance of SIFT based behavioural architectures. A description of the physical robot and software designs for the systems are presented in Chapter 6, whereas

Chapter 7 focuses on the implementation and evaluation of the proposed navigation systems. An analysis of the test results is also presented. Finally, the achievements of the study are summarised in Chapter 8 in which recommendations for further work are presented.

CHAPTER 2

LITERATURE REVIEW

The previous chapter has described the motivations behind the present work and given an introduction to the dissertation. This chapter addresses the existing state of knowledge related to vision based mobile robots, including their background and history, current trends, control architectures, navigation and mapless navigation, and the software involved. This literature review not only discusses studies relevant to vision based mobile robot systems but also critically evaluates the methodologies which have been developed that directly affect such systems.

2.1 History and Categories of Mobile Robots

A robot is defined as a programmable, self controlled device consisting of electronic, electrical and mechanical units. More generally, it is a machine which is able to function in place of a living agent. Mobile robots have a long history. Shakey, the world's first mobile robot, was developed in the late 1960s at SRI's Artificial Intelligence Centre (Stanford Research Institute) [Nilsson, 1984]. Not surprisingly, it has had a substantial legacy and influence on present day artificial intelligence and robotics. Shakey was equipped with sensors and driven by a problem solving program called 'STRIPS', and used algorithms for perception, world modelling, and actuation. Low-level action routines took care of simple moving, turning and route planning tasks. The high-level program could make and execute plans to achieve goals.

Another example of an early robot is CART, illustrated in Figure 2.1, which was developed at Stanford University in 1977 by Hans Moravec as part of his doctoral thesis [Hellström, 2011]. However CART was very slow, not because it was slow-moving by design, but because it was 'slow-thinking'. The main reason for this was the difficulty of processing vision data using slow computer processors. Another example is Rover,

developed at Carnegie Mellon University (CMU), in the early 1980s. Rover was also designed and constructed by Hans Moravec [Moravec, 1983] and used both a camera and ultrasonic sensors. Although more advanced in structure than CART, its thinking and acting were still very slow. The following sections detail more recent trends and categories of mobile robots.

2.1.1 Trends in mobile robots

In the last decade the main developments in the area of robotics have come through technological breakthroughs in the areas of computing telecommunications, software, and electronic devices. These technologies have facilitated improvements in intelligent sensors, actuators, and planning and decision making units which have significantly increased the capabilities of mobile robots. The latest trend in robotic intelligence is toward imitating life, for instance in evolutionary robots and emotional control robots. Another area of technological challenge for the next decade is the development of microrobots and nanorobots for medical applications. On top of this, a paramount challenge will be to find an appropriate balance between human assisted systems and fully autonomous systems, and to integrate technological capabilities with social expectations and requirements.

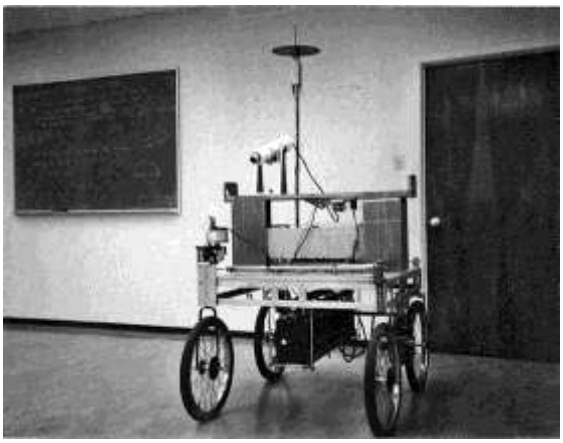


Figure 2.1: Stanford cart robot from 1977,
[Hellström, 2011]

2.1.2 Categories of mobile robots

Mobile robots are able to move from place to place under their own power. Mobility gives robots much greater flexibility to perform new, complex and exciting tasks. Mobile robots can be classified into one of three types depending on the environment in which they are designed to operate. The first category is robots that work in water, including surface and sub-sea robots, and an example of automated underwater vehicles (AUVs) is illustrated in Figure 2.2 (a). The second category is airborne robots, which employ engines and thrusters to move around. Unmanned aerial vehicles (UAV) can be remotely controlled or fly autonomously based on dynamic autorotation systems, and are currently used in a number of military tasks and in small but growing numbers of civil applications. These include fire fighting when a human observer would be at risk or the police observation of civil disturbances and crime scenes. Figure 2.2 (b) displays an autonomous UAV. The third and most common one type of mobile robot is those that move on a solid surface, as shown in Figure 2.2 (c). The wheel has been by far the most popular locomotion mechanism in mobile robotics, and it is able to complete very good efficiency with relatively simple mechanical implementation.

(a)

(b)

(c)

Figure 2.2: Classification of mobile robots, (a) MQ-9 Reaper [Wikipedia, 2011]**,** (b) underwater robot [Rutherford, 2009], (c) wheeled mobile robot [Active Media, 2010]

A number of potential markets are slowly emerging for mobile robotic systems. Entertainment applications and household or office assistants are the primary targets in this area of development. These types of robots are designed to move around within an often highly unstructured and unpredictable environment. Existing and future applications for these types of autonomous systems have two key problems in common: control architecture and navigation [Althaus, 2003]. These issues are detailed in following sections

2.2 Mobile Robot Control Architecture

Mobile robot control architecture involves the process of taking in information about the environment through the robot's sensors, processing it as necessary in order to make decisions about how to act, and the execution of action in the environment. Traditional (deliberative) control architectures are derived from traditional artificial intelligence (AI) paradigms, in which a central planner fuses all sensors readings, builds a world model, plans the next action, and finally steers the robot. Figure 2.3 depicts such architecture.

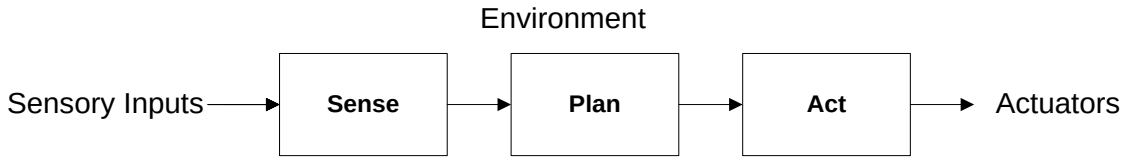


Figure 2.3: Traditional *sense–plan-act* architecture

Early robots such as Shakey [Nilsson, 1984] adopted this type of architecture, which in essence attempted to overcome environmental uncertainty by creating a world model. The deliberation refers to thinking hard, and is defined as thoughtfulness in decision and action [Nattharith, 2010]. The control system is generally organised using a functional

decomposition of the decision making process, consisting of several modules for sensory processing, modelling and planning, value judgement, and execution [Brooks, 1986]. Such functional decomposition allows complex operations to be performed, but implies strong sequential independencies between the decision-making modules. This architecture is able to work successfully in a structured environment. For instance, if there is sufficient time to generate a plan and the world model is accurate, this approach allows the robot to produce the best action for a given situation. However, such architectures tend to fail in an unstructured environment or even in a loosely structured environment due to their inability to adapt to the environment. In addition such approaches are limited in their usefulness due to a lack of real time reactivity, and may entirely fail if any single part fails. Therefore a purely deliberate architecture is no longer used for the majority of physical mobile robots working in the complex and dynamically changing real world environments [Peng, 2004; Nattharith, 2010].

2.2.1 Reactive/Behaviour based architecture

In the late 1980s the concept of behaviour-based robotics was introduced in the MIT AI lab [Brooks, 1986]. According to this paradigm, basic behaviours, which involve motor reactions to sensory stimuli, are the building blocks of more complex behaviours. This concept abandons the idea of a central planner that has comprehensive knowledge of the system [Peng, 2004]. This approach was inspired by the biological notion of stimulus-response, so that it does not rely on the types of complex reasoning processes utilised in a deliberate architecture.

The information is processed in parallel rather than sequentially. Basically, sensory data is distributed to individual reactive modules. Each of these performs a specific task such as avoiding obstacles or identifying goals. The best known system for behaviour based control is the *subsumption* architecture, introduced by Rodney Brooks in 1985, [Brooks, 1986], which is detailed in the following section.

2.2.1.1 Subsumption architecture

Brook believed that a robot must be fundamentally reactive [Brooks, 1986]. In a subsumption architecture, the behaviour based approach entails the horizontal decomposition of planning into a collection of concurrent layers; each connected to its own sensory inputs. A set of behaviours defines the control system. Behaviours are implemented as real time processes that take inputs from sensors or other behaviours and send output commands to effectors or other behaviours. The controller is essentially a distributed network of concurrently executing behaviours. An example of the given architecture is illustrated in Figure 2.4. A subsumption architecture consists of a set of complete robot control systems. Each of these is able to achieve a particular level of competence. The conventional subsumption architecture design proposed by Brooks [1986] defines eight layers of competence which are labelled from 0 to 7.

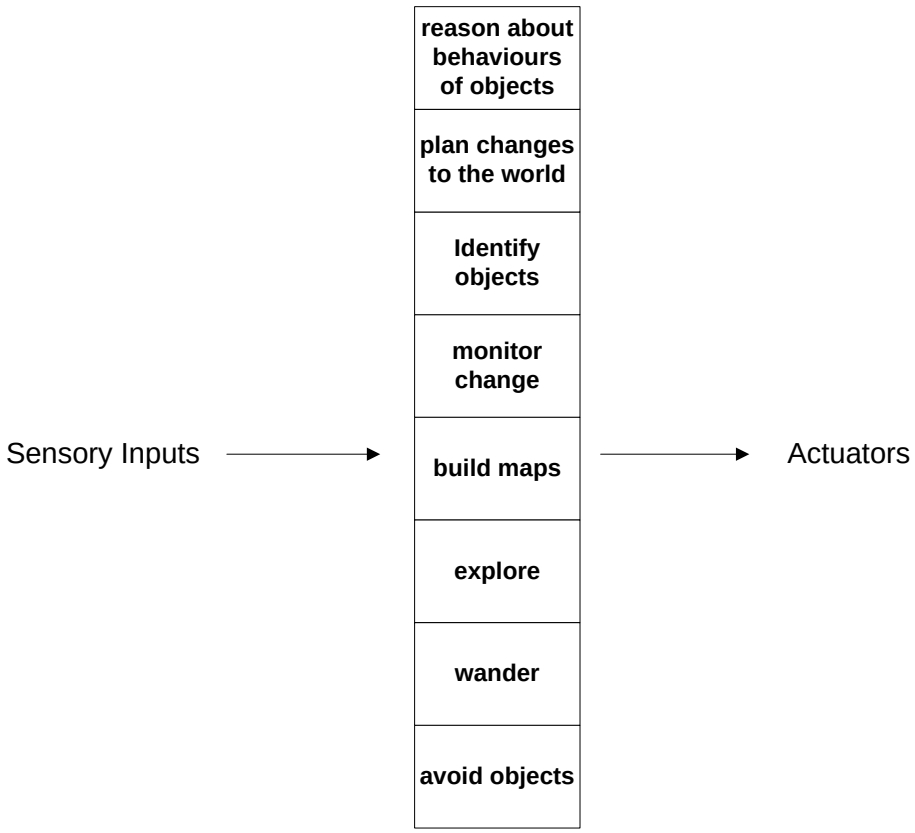


Figure 2.4: Subsumption based robot control architecture [Brooks, 1986]

However only layers 0 to 2 have been implemented on a robot [Toal et al., 1995] . Layer 0 provides the capability of avoiding obstacles, while Layer 1 allows the robot to wander around aimlessly. Layer 2 endows the robot with the ability to explore the world by using its sensors and head towards observed locations. Layers 3 to 7 entails more complex behaviour, such as the ability to map the environment, formulate plans about it, and reason about the state of the world. The team guided by Brooks conducted some initial investigations into how such behaviours can be implemented in a robot, but it is still not clear how successful these would be in the long term [Toal et al., 1995].

Several robots have been designed based on subsumption architecture. For instance: Allen - the first subsumption based robot [Brooks, 1986], Tom and Jerry – two small toy cars equipped with infrared proximity sensors [Brooks, 1990]; and *Toto* – focussed on map construction for a subsumption based robot [Mataric, 1992]. A more recent implementation of this architecture was designed to run a mobile robot in rough terrain, using an intelligent visual landmark-recognition and fuzzy based obstacle avoidance [Li and Yang, 2003]. Alternatively, a behavioural based architecture to find and trace a chemical plume using subsumption architecture has been implemented for AUVs [Wei et al., 2006]. On the other hand, since this architecture can execute only one task at a time, the robot will sooner or later experience a situation where the correct action should be established using a combination of behaviours. For example, when a robot avoids an obstacle while moving towards its goal, the subsumption architectures would invoke the avoiding behaviour as a priority ahead of the goto behaviour. The robot may avoid the obstacle successfully, but the robot may avoid it in a manner that directs it away from its goal. That is because it is not able to consider multiple behaviours which may significantly reduce performance [Hoffmann, 2003].

2.2.1.2 Motor schema

Another important example of reactive based architectures is the motor schema proposed by Arkin [1987], as illustrated in Figure 2.5. Motor schemas are proposed as a basic unit of behaviour specification for the navigation of a mobile robot. They generate response vectors based on the outputs of the perceptual schemas. The schema has a

fusion mechanism used to combine the response vectors generated in a manner similar to the Potential Field Method [Khatib, 1985]. According to this method, a goal is represented by an attractive force while obstacles are represented by repulsive forces. The summation of these force vectors is treated as the coordinated action for the robot to take to complete a particular task. However, the architecture has certain drawbacks. The most common is the local minima problem in which attractive and repulsive forces cancel each other out. Thus the overall sum is null and the robot cannot move from its current location. Various alternative solutions have been proposed to overcome this problem [Nattharith, 2010]. However, another problem with this architecture is that the action executed is, in essence, one that no behaviour has generated. For instance, consider a robot with an obstacle ahead. Assuming that two different behaviours generate outputs for avoiding that obstacle, one trying to avoid it to the right and the other one trying to avoid it to the left, then the sum of the vector would direct the robot straight ahead at the obstacle [Peng, 2004].

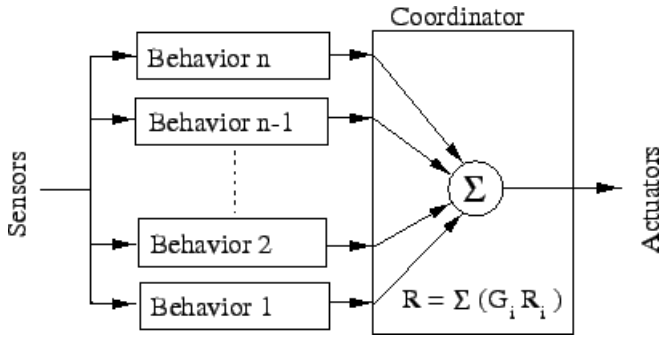


Figure 2.5: Motor schema based robot control architecture [Arkin, 1987]

2.2.2 Hybrid architecture

While it has been widely demonstrated that behaviour based architectures effectively produce robust performance in dynamic and complex environments, they are not always the best choice for some tasks. Sometimes the task to be performed needs the robot to undertake some degree of deliberation and maintain a model of the environment.

However, behaviour based architectures avoid this deliberation and modelling. Additionally, as mentioned previously, purely deliberative architectures are also not the best choice for tasks in complex environments. Thus, a compromise between these two completely opposite views must be reached. Hybrid architectures are composed of two parts, the first for deliberation and the other is the reaction part. The deliberation part allows the modelling of the world and creating plans, while the reactive part on the other hand is responsible for executing plans and quickly reacting to any unpredicted situation that may arise. Hybrid architectures are essentially structured in three layers. Because of the ability to combine the advantages of both deliberate and behaviour based systems, this approach has become important in designing mobile robot systems, and is considered to offer an appropriate solution for further development. For instance, Nattharith [2010] implemented a hybrid based architecture which is based upon the motor schema described above.

2.3 Mobile Robot Navigation

Navigation is one of the key and most challenging issues for mobile robots, it involves practically every aspects of mobile robots, including sensing, acting, planning and hardware architecture etc. It is essentially the process of determining and maintaining a course or trajectory to a reach a goal. Many robotic navigation algorithms found in the literature explicitly try to answer the questions ‘Where is the robot in the coordinate system?’ and ‘Where is the goal in the same coordinate system?’ The localization problem is the one of the most fundamental problems for a mobile robot with autonomous capabilities. While navigating, the robot has to solve the so-called data association problem which determines which landmarks it is currently sensing given its current sensor data and the landmark descriptions provided.

After this the robot employs the egocentric bearing and/or range of the landmarks identified to determine its position. On the other hand, conventional (deliberate) mobile robot architectures are rather purposeful, such as in planning to reach a particular location. They are in essence required to answer some or all of questions, shown below:

Where am I going? This is directly related to a human or mission planner.

What is the best way there? The key issue here is to determine the optimal path to reach a goal which has been assigned by the planner.

Where I have been? The key issue now is map making, in which robot creates maps or updates their common maps when navigating in an unknown environment.

Where am I? This is also called the localization problem. In order to conduct path planning and map making a robot must know its position in the environment.

Since the focus of this thesis is the study of vision based mobile robot navigation in indoor environments, the following section reviews and classifies the developments in the area of vision for mobile robot navigation.

2.4 Vision Based Mobile Robot Navigation

Vision based indoor mobile robot navigation has been studied for decades, and is one of the most powerful and popular sensing method used for autonomous navigation. Compared with other on-board sensing techniques, vision based approaches to navigation continue to demand attention from the mobile robot research community, due to their ability to provide detailed information about the environment which may not be available using combinations of other types of sensors. The great strides achieved in the area of vision based navigation systems are significant, however there is still a long ways to go. Two important survey papers have been published which reviews various aspects of the progress made so far in vision for mobile robot navigation [DeSouza and Kak, 2002; Bonin-Font et al., 2008]. This section discusses their classification of vision based navigation systems and attempts to reveal the appropriate state of the art for indoor environments.

Mobile robot navigation in a structured or unstructured environment requires the integration of various functionalities, ranging from the navigation control to mission

management while encompassing the modelling of the perceived environment and the planning of trajectories and strategies of motion. Among these functionalities, localisation, which is the robot's capacity to constantly estimate its own position, is very significant. Indeed, knowledge of robot position is essential to the correction of trajectories and the execution of planned tasks [Chaari et al., 2008]. Thereby, DeSouza and Kak [2002] classified vision based approaches into three groups depending on the localization methods used, namely: Map-Based, Map-Building-Based, Mapless approaches [DeSouza and Kak, 2002]. Bonin-Font et al. [2008] followed the same procedure and employed the same classification criteria to subdivide existing methods [Bonin-Font et al., 2008]. A corresponding schema, summarizing state of the art of vision based navigation, is illustrated in Figure 2.6 [DeSouza and Kak, 2002; Bonin-Font et al., 2008].

.

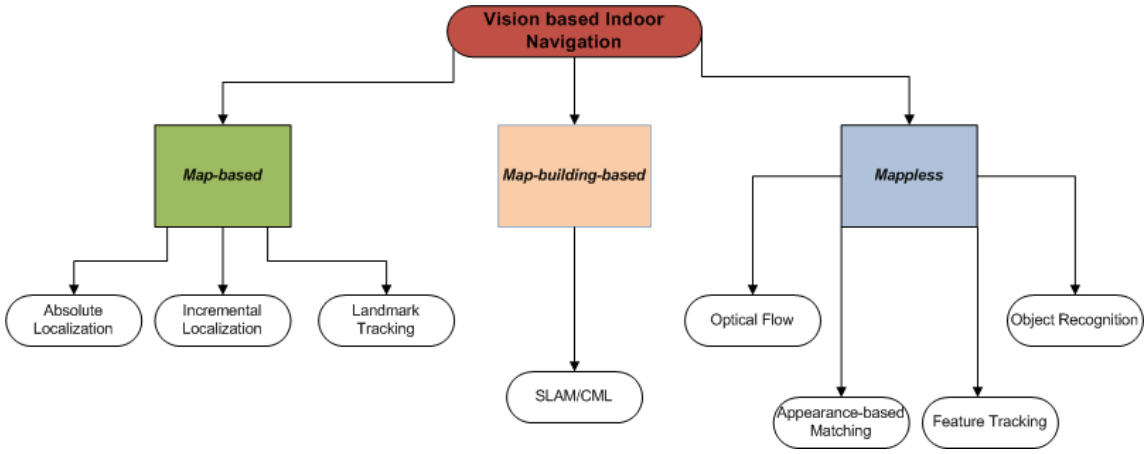


Figure 2.6: Vision based indoor mobile robot navigation techniques

To understand these concepts, a simple analogy was described [Guerrero et al, 2005], and a similar analogy including a daily life scenario is described next. In the scenario it is assumed that a visiting researcher is in Newcastle city centre and needs to return to the Daysh Building (Newcastle University,Claremont Road) where his office is located. There are various methods he can follow to reach the office. First, he could have memorized the number of steps walked from the university and he could return trying to count the same number of steps. This would be dead-reckoning navigation. He could also buy a map of the environment in order to reach the goal, as in map-based

navigation. However, this solution entails that somebody has previously named the streets and have drawn the map.

Alternatively, he could also draw his own map using map-building navigation while exploring the city, but this would cost a lot of time and effort. Finally, he could look around trying to find the Claremont Tower and then try to approach it keeping the top of the tower in his field of view (map-less navigation). The goal is to reach the tower, since he knows that Daysh Building is next to it. Autonomous navigation architectures utilize some of these solutions to track a trajectory towards the required goal. Dead-reckoning navigation is the cheapest method, and essentially includes an odometry system. However, this solution may include many mechanical problems that produce an increasing error which is unacceptable in long term navigation. So, an additional perception system is mandatory. Vision is perhaps the most broadly researched perception system.

2.4.1 Map-based navigation

Many techniques employ metric or topological maps to navigate. Navigation techniques need certain knowledge of the environment, and maps may contain different degrees of detail, varying from a complete CAD model of the environment to a simple graph of interconnections between the elements in the environment. One of the key classification criteria in this approach depends on the type of map. For instance, metric based maps generally favour techniques which produce an optimum according to some measure of 'best', while qualitative methods such as topological maps seem content to produce a route with identifiable landmarks or gateways [Bonin-Font et al., 2008].

The main idea behind map-based navigation is essentially to provide the robot with a sequence of landmarks expected to be found during navigation, and the task of the vision system is then to search for and recognize the landmarks observed in an image. When the landmarks are recognized, the robot can employ the map to estimate its own position (self-localization) by matching the observation (image) against the expectation

(landmark) description in the database. The steps of vision-based localization can be divided into four steps [Bonin-Font et al., 2008].

Acquire sensory information: Acquiring images.
Detect landmarks: Extracting edges, smoothing, filtering, and segmenting regions.
Matching: Identifying landmarks by searching in the database for possible matches according to certain criteria.
Calculate position: Whenever a set of matches is obtained, the system needs to calculate its position as a function of the observed landmarks and their positions in the database.

With Absolute Localization methods, the initial position of the robot is unknown, thus, the navigation system must construct a match between the observations and the expectations, derived purely from the entire database This self localization problem has been solved either using deterministic triangulation [DeSouza and Kak, 2002], or Monte Carlo type localization A detailed implementation of the Monte Carlo localization method to localize a mobile robot without knowledge of its starting location was proposed [Dellaert et al., 1999]. Incremental Localization assumes that, at the beginning of the navigation, the position of the robot is known approximately. In such cases, the localization algorithm basically keeps track of uncertainties in the robot's position as it executes motion commands and, when the uncertainties exceed a limit uses its sensors for a new fix on its position. The FINALE system is a good example of being able to achieve incremental localization using a geometrical representation of space and a statistical model of uncertainty in the location of the robot [Kosaka and Kak, 1992].

The final method is Landmark Tracking in which landmark tracking algorithms determine the position of the robot, detect landmarks on the camera image and track them in the successive scenes. Landmarks can be artificial or natural. In both cases the robot needs to recognize the landmarks in order to be able to track them. Artificial landmark were first introduced by Kabuka and Arenas [Bonin-Font et al., 2008]. An example of a natural landmark tracking-based navigation system is proposed by Hashima et al. [1997]. The technique selects landmarks, uses correlation techniques to

track them, computes their 3D position using stereo vision information, and selects new landmarks so as to keep on moving towards the goal point.

2.4.2 Map-building-based navigation

Sometimes modelling an environment could be difficult particularly if one also has to provide metrical information. An alternative navigation strategy, the map-building-based approach, has been used in both autonomous and semi-autonomous systems that entails searching the environment and building a representation of it. One of the earliest attempts at a map-building technique was carried out by the Stanford CART Robot equipped with a camera (see Figure 2.1). Subsequently, an Interest operator algorithm was improved to detect 3D coordinates of the images [Thorpe, 1984]. The system basically demonstrated the 3D coordinates of the objects, which were stored on a grid having 2m cells. The map was updated at each iteration; and obstacles were also shown in the map. But the most important problem with the whole system was performance, which took five hours to go 20 metres. Visual navigation studies, employing map-building-based strategies, from the late 1990s to the present have focused on two methodologies, namely: Simultaneous Localization and Mapping (SLAM) or Concurrent Mapping and Localization (CML). These principally propose solutions to automatically overcome the problem of the exploration and mapping of any unknown environment, which essentially entails three simultaneous tasks comprising navigation, mapping and localization. Vision based SLAM/CML algorithms mainly employ stereo vision as primary sensor.

Se et al. [2001] implemented a vision-based mobile robot localization and mapping system in which the robot was equipped with a stereo system to build a 3D map so as to localize simultaneously in 3D [Se et al., 2001]. The map was represented as a Scale Invariant Feature Transform (SIFT) feature database. It was constantly updated frame by frame and was adaptive to dynamic environments. An alternative and efficient solution to the SLAM problem based on a pair of stereo cameras has also recently been proposed which employs 3D landmarks to localize the robot, as well as constructing an occupancy grid for safe navigation [Sim and Little, 2009].

Other map-building based navigation techniques are those that impose a human-guided training stage. In such solutions, a human operator guides the robot through an unknown environment. During this process, the robot records images with a stereo camera and constructs the 3D map incrementally. After the map is built, the robot tracks extracted features and computes the optimum path [Kidono et al., 2002].

2.4.3 Mapless navigation

This section discusses a representative selection of mainly reactive visual based navigation techniques in which navigation is performed without any prior description of the environment. Mapless navigation is scarcely new compared with the previously defined solutions, but new projects using vision systems have been developed in several directions in the last few years. In the systems surveyed in this section, no maps are ever created. The robots can navigate by observing and extracting relevant information about the landmarks in the environment. These elements can be objects such as desks, boxes, doorways, and so on. The mapless visual navigation techniques discussed here are classified in accordance with the main vision technique or types of clues used during the navigation which are optical flow, appearance based, and object recognition navigation techniques based on feature tracking (see Figure 2.6) [DeSouza and Kak, 2002].

2.4.3.1 Optical flow techniques for navigation

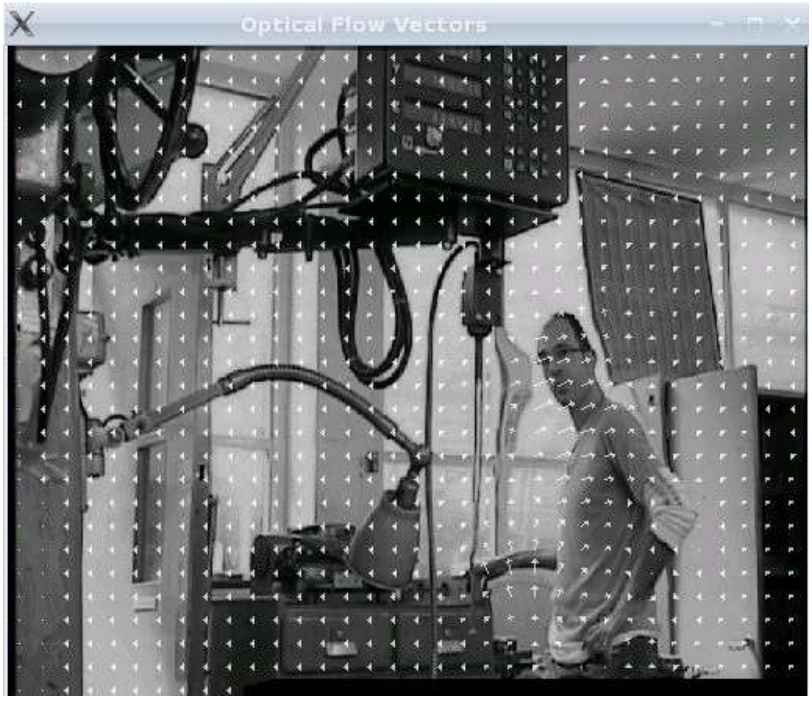
Optical flow is defined as the motion of all the surface elements from the visual world. When a person moves through the world, the objects and surfaces within the visual environment flow around this person. The human visual system can determine that person current direction of travel from the movement of these surfaces. Optical flow can be defined as the apparent motion of features in a sequence of images, as shown in Figure 2.7. It is believed that when the insect is in relative motion with respect to the environment. Accuracy and the range of operation can be altered by changing the relative speed. For instance, features such as “time-to-contact” (depending on speed) are

more relevant than distance when it is necessary to avoid an obstacle [DeSouza and Kak, 2002].

Santos-Victor et al. [1993] developed an optical flow based navigation system imitating the visual behavior of bees, called *robee*, and was equipped with a stereo vision system, mimicking the centring reflex behaviour used by a bee to navigate safely. The robot localizes itself using the difference between the velocity of the image seen with the left eye and the velocity seen in the right eye. If the difference is close to the zero, the robot keeps moving forward. However, if the velocities are different, the robot moves toward the side whose image changes with a lower velocity. Several successful navigation systems have recently been inspired from by this centring reflex, and implemented to navigate a mobile robot through an unknown indoor environment. For instance, a mobile robot platform utilizing a binocular vision system to estimate optical flow in some way emulates corridor following behaviour to navigate [Bernardino and Santos-Victor, 1998]. Duchon et al. [1994] implemented a monocular vision based navigation system based on optical flow algorithms and action modes (behaviours). Simulation and real experiments revealed that the system was capable of navigating in a maze whilst successfully avoiding obstacles [Duchon et al., 1994].

Furthermore, optical flow based control algorithms based on behaviours have recently been implemented to evaluate their performance in real time applications [Souhila and Karim, 2007; Guzel and Bicker, 2010]. Further explanations and definitions of optical flow based reactive navigation algorithms are discussed and detailed in Chapter 3.

(a) **(b)** **(c)**



(d)

Figure 2.7: Optical Flow calculation and motion estimation, (a) source image, (b) destination image, (c) motion estimation, (d) estimated flow vectors

2.4.3.2 Appearance-based methods

Appearance-based methods fundamentally rely on the idea of memorizing the working environment. The main idea is to store images or templates of the environment and associate these images with commands that will steer the robot to its final destination. These methods mainly consist of two procedures. The first one is the training phase in which images or prominent features of the environment are stored as model templates.

The models are labeled with certain localization information and/or with an associated control steering command. Secondly, in the navigation stage, the robot has to recognize the environment and self-localize in it by matching the current on-line image with the stored templates. The main problems with the method are to find an appropriate algorithm for the representation of the environment, and to define the on-line matching criteria [DeSouza and Kak, 2002; Bonin-Font et al., 2008]. These methods can be classified into two groups as follows:

Model-based approaches: Pre-defined object models are utilized for feature recognition and self-localization in cluttered environments.

View-based approaches: The self-localization is performed using simple image matching algorithms.

Gaussier et al. [1997] developed an appearance-based approach using neural networks to map perception into action. The robot, in essence, merges visual information and their azimuths to build up a representation of its location which is used to estimate the best movement to reach the goal [Gaussier et al., 1997].

The view-sequenced route representation technique was proposed by Matsumoto et al [1996]. This technique primarily focuses on route construction using a sequence of images and a template matching algorithm to guide robot navigation. Captured images are used to form a sequence of images. Each image in the sequence is associated with the motions required to move to a corresponding destination. This approach basically introduces the concept of visual memory [Matsumoto et al., 1996] , as illustrated in Figure 2.8.

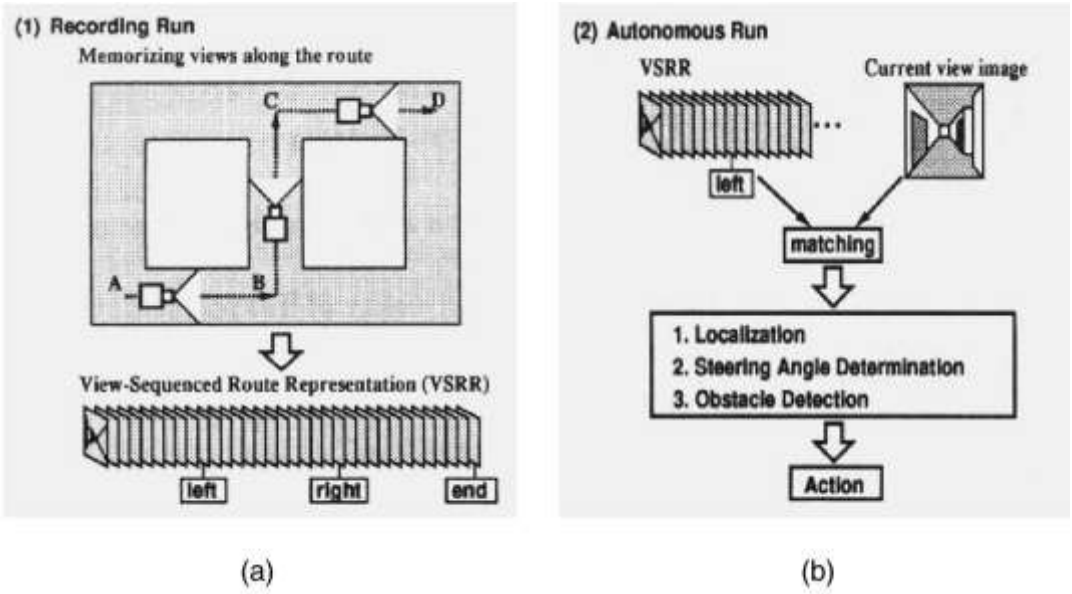


Figure 2.8: The view-sequenced route representation [Matsumoto et al., 1996]

Multi-dimensional histograms provided by the statistical analysis of images are an alternative method to guide mobile robots in appearance based strategies. Statistical data, including that related to colour, edge density and texture, are utilized to build a multi-dimensional histogram database. The recognition of the environment during the navigation stage is achieved by matching the multi-dimensional histogram of the current image with the multi-dimensional histogram of the stored templates. This technique consumes less computational resources than when using correlation algorithms [Chao et al., 2003].

Recent groundbreaking research has proposed an entirely qualitative method in which feature points are automatically detected and tracked throughout the image sequence. In the teaching phase the (KLT) feature tracker computes the displacement and minimizes the sum of the squared differences between consecutive image frames. The feature coordinates in the replay phase are compared with those computed previously in the teaching phase in order to estimate the steering commands for the robot. Experimental results revealed that the capability of autonomous navigation in both indoor and outdoor environments was successful with the proposed method [Zhichao and Birchfield, 2006].

An important concept in visual based mobile robot navigation is the idea of visual homing, inspired by insect behaviour. Insects are able to return to important places in their environment by storing an image of the surroundings while at the goal, and later computing a home direction from a match between this snapshot image and the currently perceived image. For instance, an agent employing a visual homing algorithm captures an image I_S (snapshot) at the home location $S = (x_S, y_S)$. It then attempts to return to this location from a nearby position $C = (x_C, y_C)$. It compares the current image I_C with the snapshot and infers the direction and/or distance to the location of the goal from the disparity between these images. It is considered that these aspects of insect behaviour can be a basis for the development of robust navigation algorithms for mobile robots. Visual homing is an appearance based navigation strategy whose homing algorithms are based on image based holistic methods using disparities between whole images to compute homing vectors.

Image warping is a popular method which is considered to be one of the most reliable visual homing methods for indoor use in this category. It involves calculating the set of all changes in pose (position and orientation) between the I_S and I_C. Warping methods distort the I_C as if the agent would move according to certain movement parameters. The space of possible movement parameters is then searched for the parameter combination leading to the warped image that is as similar as possible to the stored I_S. In order to achieve this, each warped I_C is compared with I_S using a pixel-by-pixel correlation measure. The current home vectors are determined based on the strongest similarity between those images [Szenher, 2008]. Arena et al. [2007] have proposed a new and simple visual homing algorithm employing the root mean square (RMS) difference and exclusive or (XOR) functions to compare I_S and I_C, where the home position is a recharging station. They demonstrated that it is possible to implement homing algorithms which allow a robot, fitted with a panoramic camera return to a reference position from any starting point in an area. A detailed review of corresponding image-based (holistic) visual homing methods has been conducted by Szenher [2008].

2.4.3.3 Object recognition

For the appearance-based approaches previously mentioned, the robot is only able to access few sequences of images that help it to reach its final destination, or uses predefined images of target goals that it can use to track and pursue. An alternative method has been proposed which essentially employs a symbolic navigation approach instead of memorizing the environment [Kim and Nevatia, 1998; Kim and Nevatia, 1999]. In this case, the robot utilizes symbolic commands such as "go to the desk in front of you" or "go to the main exit". For instance, a command such as "go to the desk in front" informs the robot that the landmark is the desk and the path points straight ahead. The robot builds a map called an "s-map" which is a 2D grid that stores the projections of the observed landmarks as they are recognized. Once the target landmark such as the desk is recognized and its location is projected into the s-map, the robot plots a path using a GPS-like path planner and dead reckoning to approach the target [DeSouza and Kak, 2002].

2.4.3.4 Navigation techniques based on feature tracking

Techniques for tracking moving elements such as corners, lines, object outlines or specific regions in a video sequence have become robust enough to be useful for navigation. Feature-based approaches determine the trajectory and motion of the robot by tracking and finding relative changes in the position of extracted features. This category can also include feature-based visual homing strategies.

Feature-based methods fundamentally segment snapshot and current images into landmarks and background. They then attempt to pair each landmark in the snapshot image with a landmark in the current image, which is called the correspondence problem. To operate successfully, feature-based navigation algorithms must extract the same features from I_S and I_C (the feature-extraction problem). Each feature extracted from I_S must then be paired with a feature from I_C (the correspondence problem). The feature extraction and correspondence problems are difficult to solve in cluttered

environments in real-time, since the appearance of landmark changes with viewpoint [Szenher, 2008].

One of the earliest studies regarding feature tracking systems was conducted by Harrell et al. [1989]. They introduced a fruit tracking system employing the size and position of a valid fruit's regions in colour images, to control the motion of a fruit-picking robot. Trahanias et al. [1997] implemented a robotic system able to extract landmarks automatically in indoor environments, using a selective search for landmark patterns which relies both on the workspace and the distinctiveness of the objects in the environment. For recognition purposes, a viewing transformation has been developed that transforms a stored pattern according to the current (new) position of the observer. This facilitates accurate recognition, and has been demonstrated by experimental results in an indoor environment [Trahanias et al., 1997]. A KLT tracker based homing schema was rooted in the extraction of very low-level sensory information, namely the bearing angles of corners. This was implemented on a robotic platform to evaluate the results [Argyros et al., 2001].

On the other hand, in most cases, feature tracking-based navigation algorithms do not provide an obstacle avoidance module, which must therefore be implemented by other means depending on the problem. For instance, Hao and Yang [2003] proposed a behavioural based navigation architecture for mobile robots, that utilized a robust visual landmark-recognition system based on genetic algorithms to guide the robot, which used a fuzzy based obstacle avoidance system and ultrasonic range finder [Hao and Yang, 2003].

It can be assumed that any two images of the same planar surface in space are related by a homographies. This concept has many practical applications, including mobile robot navigation. In a recent study, Guerrero et al [2005] introduced a method based on homographies computed between current images and images taken in a previous teaching phase with a monocular vision system. The vertical lines (features) were used to estimate the homographies, which are automatically extracted and matched. From

that, a complete homography motion, with rotation and translation up to a scale factor can be computed [Guerrero et al., 2005].

The scale invariant feature transform (SIFT) method is a milestone among techniques to detect the features of images or relevant points, and nowadays has become a method commonly used in landmark detection applications [Lowe, 2004]. The SIFT algorithm, introduced in this Chapter and detailed in Chapter 4, extracts features that are invariant to image scaling, rotation, and illumination. During the robot navigation process, invariant features which have been detected are then observed from different points of view, angles, and distances and under different illumination conditions, and thus become highly appropriate landmarks to be tracked for navigation. Pons et al. [2007] employed the SIFT algorithm for feature-based homing, and are utilized to recover the misalignment of orientation between the current and goal positions. Finally, a home vector between these two positions is calculated using the SIFT matches as a correspondence field [Pons et al., 2007].

Visual servoing is another important concept which can be included in this category, and is defined as the capability to employ visual information to control the pose of the robot's end-effectors relative to a target object or a set of target features. The task can also be defined for mobile robots, where it becomes the control of the vehicle's pose with respect to specific landmarks. Thus, Szenher [2008] defined the feature based visual homing algorithms as a type of image-based visual servoing There are two main approaches for visual servoing systems namely: PBVS (position based Visual Servoing) and IBVS (image based visual servoing) [Hutchinson et al., 1996]. PBVS algorithms solve the trajectory problem in workspace; however, in IBVS, the control commands are deduced directly from image features. Figure 2.9 provides the architectures of both approaches.

Kim and Oh [2007] have proposed an intelligent mobile robot navigation architecture comprising both of these servoing methods to guide a mobile robot. The IBVS module estimates the motion planning directly from the image space so as to keep the target object always in the field of view. As well as this, the PBVS module is employed to

conduct an image-to-workspace transform to plan an optimal pose trajectory directly in the Cartesian space. The proposed fuzzy control architecture is considered to integrate these two types of visual servoing through a warning signal indicating that the target may escape the field of view. In addition, a neural network module is integrated into the architecture for the prediction of the target position for a robust timely tracking of the object [Kim and Oh, 2007].

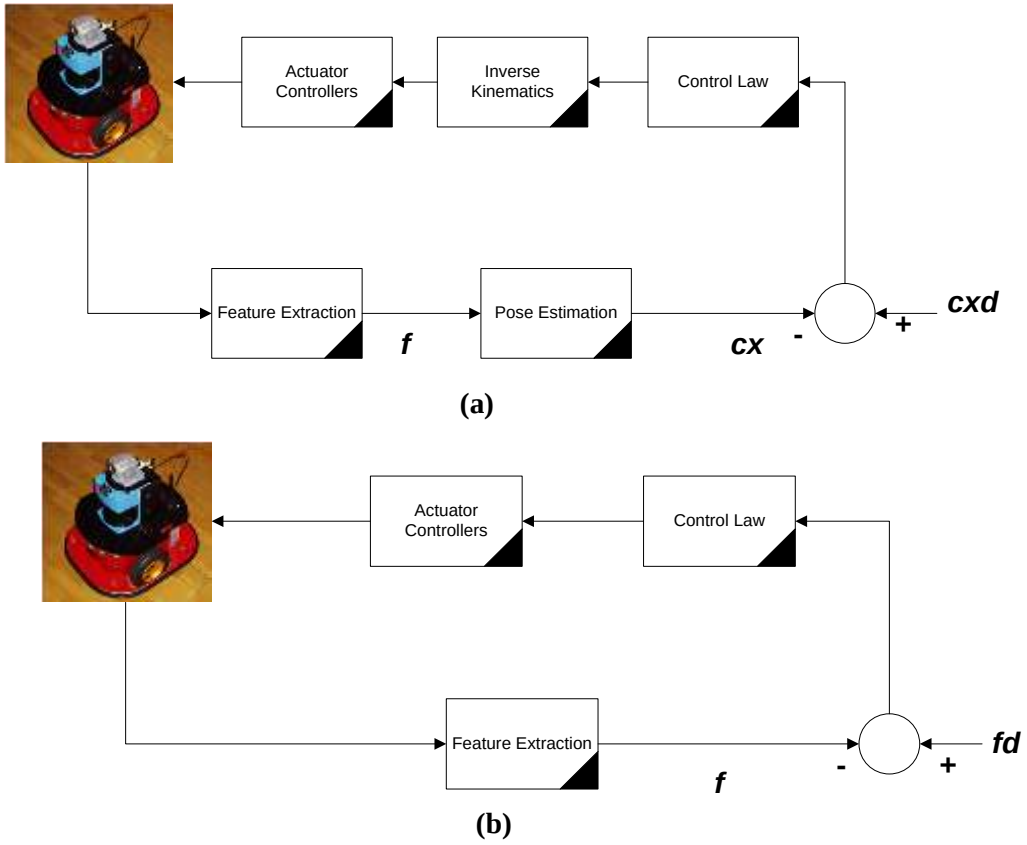


Figure 2.9: PBVS and IBVS architectures, (a) PBVS, (b) IBVS, inspired by [Hutchinson et al., 1996]

2.5 Obstacle Avoidance Systems based on Qualitative Information

Obstacle avoidance techniques classified essentially entail extraction of qualitative image characteristics and their interpretation [Bonin-Font et al., 2008]. They are basically defined as sensor-based obstacle avoidance systems which process every item of online sensor data to estimate free and occupied space. These methods are considered

useful in avoiding having to compute accurate numerical data such as distance and position coordinates.

Lorigo et al. [1997] proposed a low resolution vision-based obstacle avoidance architecture consisting of three dependent vision modules for obstacle detection. These modules were associated with edges, RGB (red, green, blue) colours and HSV (hue, saturation, value) information. The data from these modules was analyzed by a fourth module so as to simultaneously generate motion commands [Lorigo et al., 1997].

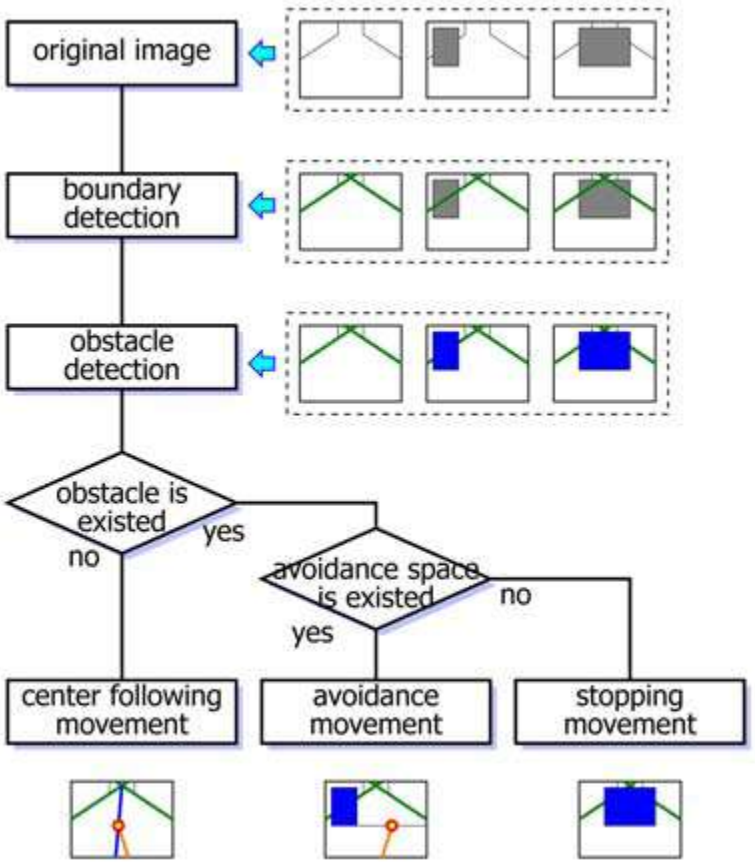


Figure 2.10: Flowchart of proposed algorithm [Saitoh et al., 2009]

Ulrich and Nourbakhsh [2000] proposed a similar vision based obstacle avoidance strategy based on monocular vision. The strategy essentially involves assigning each pixel as either obstacle or ground according to its colour appearance [Ulrich and Nourbakhsh, 2000]. Saitoh et al. [2009] integrated this obstacle avoidance technique into a centre followed based mobile robot navigation architecture. The system does not

need prior knowledge of the environment and employs a low cost monocular vision camera as the only sensor needed to navigate the robot safely. The robot has a basic navigation strategy so that it moves towards the centre of the corridor until it encounters an unexpected obstacle. When any obstacle is detected, the robot attempts to avoid it or stops depending on the size of the obstacle. If the robot manages to pass the obstacle successfully, it then localizes itself toward the centre. The system is also able to detect boundaries using the Hough transform [Saitoh et al., 2009]. The flowchart of the given obstacle avoidance system is illustrated in Figure 2.10.

ROBOCUP competition has become quite popular and has attracted the attention of many researchers in recent years. The detection of an opponent robot and the ball are two challenging tasks which must be solved efficiently. Fasola and Veloso [2006] proposed using image colour segmentation techniques for object detection, and gray-scale image processing to detect the opponent robots [Fasola and Veloso, 2006].

The qualitative based obstacle avoidance systems, which are also called appearance-based obstacle avoidance technique [Ulrich and Nourbakhsh, 2000; Guzel and Bicker, 2011] are described in more detail in the following chapter.

2.6 Scale Invariant Feature Transform (SIFT)

A local feature is basically defined as an image pattern which differs from its immediate neighbourhood. It is usually associated with a change in an image property or several properties simultaneously. The image properties most commonly considered are intensity, colour, and texture. A prominent survey of the local feature detectors classifies and evaluates them comprehensively [Tuytelaars and Mikolajczyk 2008], defining feature detection as methods that aim at computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not. The scale invariant feature transform (SIFT), proposed by Lowe [2004] is a milestone in local feature detection. It allows the robust identification of objects among clutter and under partial occlusion. SIFT features are demonstrably invariant to translation, scaling and rotation in images, and are also highly

distinguishable from one another and relatively invariant to changes in illumination. These properties render them suitable for the purposes of mobile robot navigation. In recent years, studies regarding the performance analysis of local feature detectors have been carried out. One such evaluation concluded that SIFT-based descriptors are the most robust and distinctive, and are therefore best suited for feature matching [Mikolajczyk and Schmid, 2005]. However, the most recent feature descriptor, called SURF was not evaluated in this study. It is faster than the conventional SIFT algorithm and has been claimed by its originators to be more robust [Bay et al., 2006]. Both SIFT and SURF algorithms are open source and can be employed freely for non-commercial projects freely, however, SURF is not as flexible as SIFT in terms of platform dependency. Besides this it requires specific hardware configurations to run the applications.

2.7 Sensor Theory and Vision Based Sensors

Sensor technology has advanced considerably in the last decade, and many low cost sensor systems are available that can easily be deployed on robots. Sensors can be basically classified into two groups based on their interaction with the environment, as either passive (P) or active (A). Passive sensors employ energy that is naturally presents in the environment to obtain information. Computer vision is considered a typical example of a passive sensor. Active sensors, on the other hand, involve the emission of energy by the sensor into its environment, some of which is then reflected back in some manner to the robot. The laser range finder is one of the most common active sensor modalities used on mobile robots.

2.7.1 Active ranging sensors

The laser range finder is an active sensor which is able to measure distance, and is an important device for obstacle avoidance. It measures distance using time of flight (TOF) parameters. The laser range finder emits a coherent beam with approximately 0.5 degree spread, and it is difficult for it to be influenced by the environment. However, it is a

high cost sensor and detects object in a plane, which implies that if the object is just above or below the height at which the laser is positioned, it will not be detected.

An alternative and low cost active sensor is the ultrasonic sensor. This emits a high frequency chirp which reflects off a nearby surface and is returned in a measureable time, which is then used to estimate distance. These sensors essentially emit a beam that receives echoes from a region of approximately 30 degrees wide from its source. They are mainly used for obstacle detection at short range; however they are vulnerable to noise due do the environmental conditions. Infrared sensors are another example of distance measurement which can be used for obstacle avoidance; however their major limitation is their relatively low accuracy.

2.7.2 Vision based sensors

Vision provides the most comprehensive information to mobile robots. However, due to its complexity and sensitivity to factors such as lighting it is sometimes difficult to use effectively. There are various architectures for vision based sensors. One of these is stereo vision which is mainly used to extract range data. This is a promising sensing method that uses two or more cameras placed in different positions, capturing images which are then analysed to detect the objects. However, this architecture has several important drawbacks. Firstly processing costs can become excessive for relevant architectures, in terms of both software and hardware. A more fundamental problem is to estimate how the robot knows that it is looking at the same point in both images. This is called the correspondence problem [Murphy, 2000]. Omni-directional vision is another popular sensing technique widely used by researchers. An omni-directional camera has a 360-degree field of view in the horizontal plane, or with a visual field that covers approximately the entire sphere. However, it is a specialist camera and should be mounted on top of the robot in order to take all round view. This causes limitations in terms of appearance, and the detection of obstacle region or walls is difficult [Saitoh et al., 2009]. Another important disadvantage of omni-directional cameras is loss of resolution in comparison with standard images. PTZ (Pan-Tilt-Zoom) vision refers to

mechanically operated cameras, which are considered one of the most useful sensors [Jae Kyu et al., 2011]. The user typically has the ability to control the pan to the left and right, tilt up and down and the zoom of the camera with a joy stick or some other devices. The main advantage is they allow the operator to track objects or respond to a threat and follow it much more closely. They can zoom in and capture key information that can be used to help in loss prevention. However the limitation is that it is only able to record where the camera is pointed and focused, and if the camera is pointed away from where an incident happens it could potentially miss the event entirely.

2.8 Mobile Robot Software

Robot software is the coded commands that instruct a robot what tasks to perform and is used to control its actions. Robot programming is a challenging task, and thus several software systems have been proposed to facilitate programming as well as being deployed on widely distributed robotics platforms. Several software packages used in mobile robots have been developed and a comparison of those packages was carried out by Nattharith [2010].

The most popular of this types of software is Player/Stage which has been utilised by many research groups around the world. Player has become the preferred simulator in the mobile robot community because it does not place any constraints on how client programs should be written. Furthermore it can be used to interface with 2D and 3D robot simulators such as Stage and Gazebo. In addition, it provides several tools to display sensor output graphically. The following section details the structure of Player. Microsoft Robotics Developer Studio (MRDS), which provides a powerful simulation environment, is then introduced.

2.8.1 Player architecture

Player is a free and very popular software program which is able to control several robotics platforms. Its client/server model allows robot control programs to be written in

any programming language that can run on any computer with a network connection to the robot. Player supports multiple concurrent client connections to devices, creating new possibilities for distributed and collaborative sensing and control. Gerkey et al. [2001] states that: "Player is a network server interface to a collection of sensors and actuators, typically constituting a robot. ".

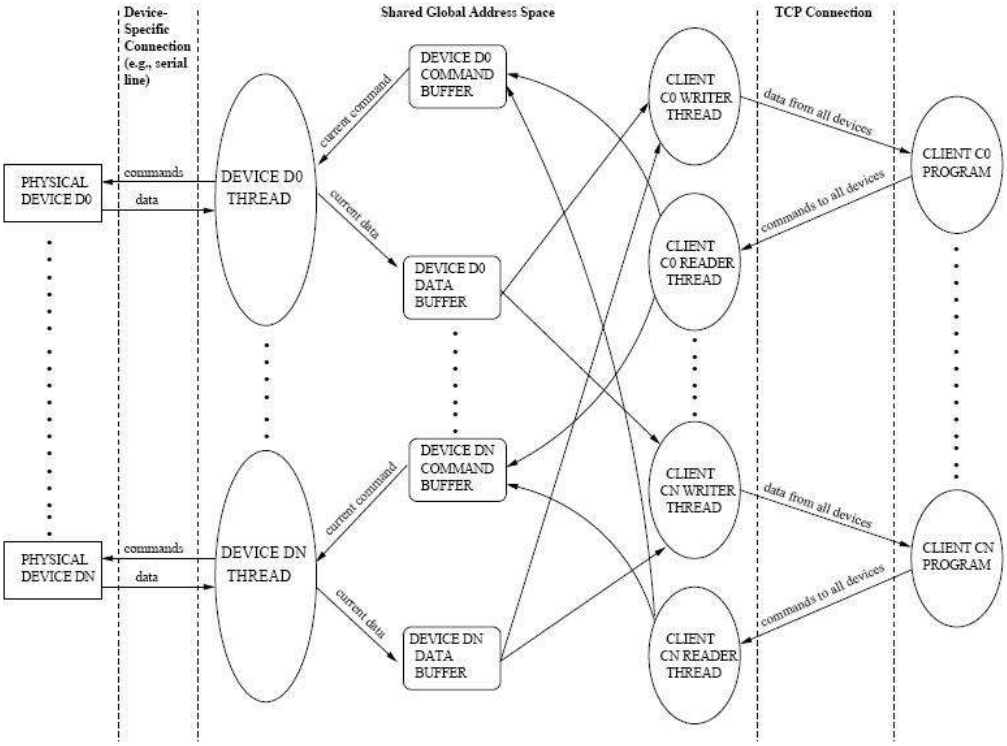


Figure 2.11 Overall system architecture of Player [Gerkey et al., 2001b]

Player is designed as a distributed system and relies on the TCP protocol to handle communications between the client and server layers. The overall system architecture is illustrated in Figure 2.11. It is a language independent platform, as previously mentioned; however, the client programs developed using C++ can take advantage of the object-oriented Player C++ client library. This library employs classes as proxies for local services. For example, an instance of PlayerClient for a single sever proxy is employed to provide a connection with a Player server. Devices are registered by creating instances of the appropriate proxies and initialising them through the established Player Client object. Additionally there are numerous device proxies, such

as the LaserProxy class which acquires scan data for the laser. The Position2DProxy class, on the other hand, is used to obtain data on current position in 2D world with X-Y coordinates and orientation. More details of the attributes of, and methods used by various classes, are presented in [Gerkey et al., 2004].

2.8.2 Microsoft Robotics Studio (MSRS)

Microsoft Robotics Developer Studio (MRS) is a robotic programming development environment which was publicly released in December 2006 with the explicit goal of providing an industry software standard for robot control. It provides solutions for concurrency, distribution, abstraction, simulation, and programmer interaction simulators [Jackson, 2007].

The Visual Simulation Environment (VSE) is designed to be used in a variety of advanced scenarios with high demands for visualization and scaling. Furthermore, a beginner with little programming experience can use simulation, and interesting applications can be developed in a game-like environment.

2.9 Software for Computer Vision

Computer vision refers to processing data from any modality which produces an image. The term ‘image’ means a way of representing data in a picture-like format where there is a direct physical correspondence to the scene being imaged. An image implies a multiple reading placed in a two dimensional array in a grid. Every element in the array, called pixels, maps onto a small region of space. The modality of the camera estimates what the image measures, for instance, if a visible light camera is used, the data is subsequently stored at each pixel is the value of the light, such as in colour. Alternatively, if a thermal camera is employed, then the values store data on the heat at the region.

Computer vision on reactive robots (robot vision) is most often achieved using a video camera which is an either IP or CCTV camera. Robot vision is a rapidly developing technology that can increase the productivity and efficiency of all robotic systems. Recently, various different software architectures and frameworks have been developed to acquire and process image data successfully, thus helping researchers to implement both new and conventional algorithms more rapidly and efficiently. There are two popular open source computer vision libraries; namely, OpenCv and Clmg. These libraries are widely used by researchers who need to analyse and process image data for real time applications.

OpenCV is a library of image processing algorithms developed by Intel for use by researchers and professionals alike. The main programming languages which can be used with OpenCV are C and C++. This library supports many image processing and computer vision algorithms, such as those for basic image processing like filtering or edge detection, structural analysis using the Hough transform, template matching, linear algebra routines, and others. Despite its strengths, the implementation of the library may be complex depending on the hardware configuration and third party software required, and serious effort is required to understand some of the basic principles before going on to bigger and better things. The source code and more detailed information have been published in OpenCV [2011].

CIMG (Template image processing toolkit) is an open source image processing and computer vision designed by Tschumperlé et al. [1999]. It can be used across many platforms, including Windows, OS X, and UNIX. It is also highly portable and is stored in a single *.h header* file, which is about 1MB in size. The library itself contains several useful algorithms. The header file *CImg.h* contains all the classes and functions that comprise the library itself. The main advantages of this library are its portable structure which allows it to successfully work in different hardware and software configurations [Tschumperlé et al., 1999].

2.10 Soft Computing

Soft Computing is a collection of techniques including many fields that fall under various categories in Artificial Intelligence. These techniques resemble biological processes more closely than traditional techniques, which are largely based on formal logical systems. It has three main branches including fuzzy logic, neural networks, and genetic algorithms. Navigation is a key issue for mobile robots as previously mentioned; traditional robot control methods rely upon strong mathematical modeling, analysis, and synthesis. However, soft computing techniques provide alternative and simpler solutions to this problem. Two major branches of soft computing used in this research are introduced in the following sections.

2.10.1 Fuzzy logic

Fuzzy logic is essentially a system for dealing with uncertainty, and was developed by Zadeh in the 1960s to characterize types of knowledge that cannot be represented by conventional boolean algebra [Zadeh, 1965]. There are now many variations on the concept of a fuzzy logic that allows objects to take partial membership in vague categories which it achieves through the use of a structure called a fuzzy set. Fuzzy set theory is responsible for representing the elements using grades of possibility called the membership function. These allow the description of the behaviour of systems that are otherwise complex to deal in with mathematical terms. The fuzzy controller is composed of the following four elements: fuzzification, rule-base, fuzzy inference and defuzzification, as shown in Figure 2.12 [Passino and Yurkovich, 1998]. Fuzzy control has been used in a wide variety of applications in engineering, science, business, medicine, psychology, and other fields. Autonomous mobile robots can also employ fuzzy logic for complex control architectures. For instance, Daniel et al. [1999] introduced the design of a fuzzy logic based navigation system for a mobile robot in which the system includes two behaviours: obstacle avoidance and goal seeking. The inputs to the fuzzy controller are the desired direction of motion and sensory data, while the outputs from each behaviour rule are integrated using a command fusion

mechanism, resulting in the smooth motion of the robot [Daniel et al., 1999]. Several techniques for mobile robot using fuzzy logic have been developed, including those proposed by [Hung-Ching and Chih-Ying, 2005], [Kiwon and Nian, 2007] , [Harb et al., 2009].

Hung-Ching and Chih-Ying [2005] introduced a fuzzy logic system on a mobile robot where the steering angle and speed are determined, by two separate fuzzy logic controllers. Additionally, Kiwon and Nian [2007] described similar fuzzy control architecture to guide a mobile robot in which the design of mobile robot navigation architecture was based on the combination of two fuzzy logic controllers acting on 81 different rules. The system was equipped with eight range finder sensors and a GPS sensor, and the outputs of the fuzzy system controlled the speed of two servo motors [Kiwon and Nian, 2007]. Harb et al. [2009] described a navigation architecture employing a fuzzy controller and a neural network to adjust the speed of mobile robots. Additional research in mobile robots utilizing fuzzy based control architectures is described in Ross [2004]. The components of a fuzzy logic controller will be detailed in the following section. Further discussion of fuzzy logic, including its operation and the use of fuzzy control in the area of speed control, is provided in Chapter 5.

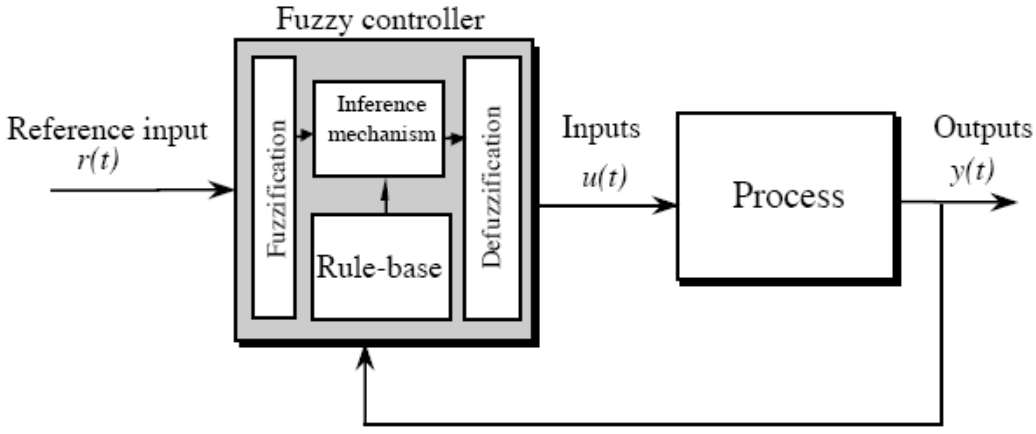


Figure 2.12 Fuzzy controller [Passino and Yurkovich, 1998]

Fuzzy logic lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-

based data acquisition and control systems. It also relatively easy to implement and provides faster and more consistent results than conventional control methods. In this study, a FL based control system based on the Mamdani method is designed to fuse given algorithms. The basic configuration of a fuzzy-logic system is composed of four parts (see Figure 2.12): Fuzzification, Fuzzy rule-base, Inference Mechanism and Deffuzification [Driankov, 1987].

2.10.1.1 Fuzzification

Fuzzification comprises a scale of the transformation of input data from a current process into a normalised domain. This requires the identification of two parts where the first defines the fuzzy variables that correspond to the system input variables. The second part defines the fuzzy sets of the input variables and their representative membership functions, including the range of the data.

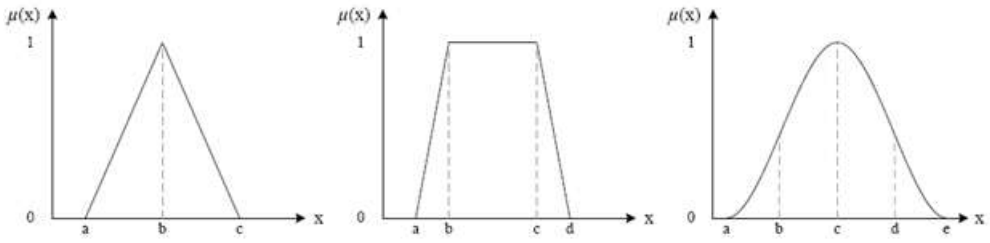


Figure 2.13: Membership function shapes, (a) triangular, (b) trapezoidal, (c) gaussian, [Ross and Hoboken, 2004]

Membership functions may cross each other's boundaries, and may be triangular, trapezoidal or bell shaped, as illustrated in Figure 2.13. The choice of the fuzzy sets is based on expert opinion using natural language terms that describe the fuzzy values. In this study triangular and trapezoid models are utilized to design the membership functions of the input and output values.

The triangular function has three parameters which can be defined as follows:

$$\mu(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \le x < b \\ \frac{c-x}{c-b}, & b \le x \le c \\ 0, & x > c \end{cases}$$

(2.1)

The degree of membership of the Gaussian function (Figure 2.13) depends on two parameters, *c* and σ, which represent the centre and width of the graph respectively and are illustrated as follows:

$$\mu(x) = exp\left[\frac{-(x-c)^2}{2\sigma^2}\right]$$

(2.2)

The trapezoidal function incorporates four parameters which can be represented as:

$$\mu(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \le x < b \\ 1, & b \le x \le c \\ \frac{d-x}{d-c}, & c < x \le d \\ 0, & x > d \end{cases}$$

(2.3)

Fuzzy logic uses intersection, union, and complement operations to represent the standard common operators of AND, OR, and NOT, respectively. The most common method used to calculate intersection and union operations are the Minimum and Maximum functions. For the fuzzy sets *M* and *N*, which are subsets of the universe *X*, the following definitions are proposed to represent the AND, OR, and NOT operators respectively [Ross and Hoboken, 2004] (see Figure 2.14).

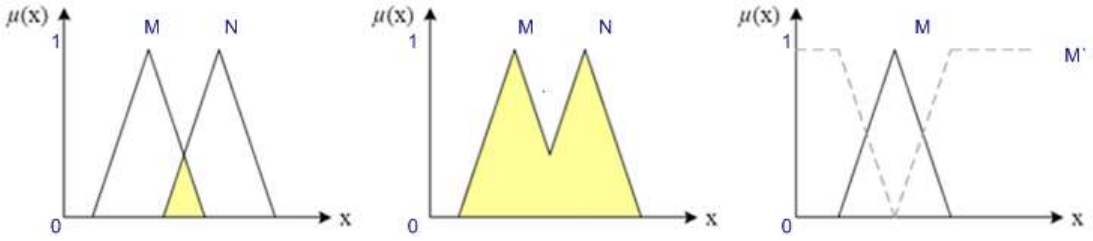


Figure 2.14: Fuzzy Operators, (a) and, (b) or, (c) not

$$\forall x \in X: \mu_{A \cap B}(x) = min(\mu_A(x) \cap \mu_B(x)) \quad (2.4)$$

$$\forall x \in X: \mu_{AUB}(x) = max(\mu_A(x) U \mu_B(x)) \quad (2.5)$$

$$\forall x \in X: \mu_{A'}(x) = 1 - \mu_A(x) \quad (2.6)$$

While variables in mathematics usually take numerical values, in fuzzy logic applications, non-numeric *linguistic variables* are often used to facilitate the expression of rules and facts. A linguistic variable is a variable whose values are words or phrases in a natural or artificial language rather than being numerical [Zadeh, 1965; Zadeh, 1968]. According to Godjevac [1997], a linguistic variable is defined by:

- Its name: "x".
- Its term set: "TS(x)", which is the set of linguistic values or labels of "x".
- The base variable "u", which supports the linguistic values of "x". In others words, the membership functions for the linguistic values of "x" are defined in the domain of "u".
- The universe of discourse "U" associated with the base variable "u".

Here, "x" should not be confused with "u": "x" is the name of a linguistic variable (such as distance, angle, etc) whereas "u" is the name of the base variable giving physical sense to "x" (i.e. meters, degrees, etc). In physical applications, "x" may adopt linguistic values (i.e. small, very large, etc) whereas "u" may adopt numerical values (i.e. 100m, -30°, etc) [Godjevac, 1997]. In a fuzzy controller, inputs and outputs are defined as linguistic variables. For instance, suppose that a controller is designed to guide a vehicle

towards a reference point in a plane. The inputs of the controller are the distance between the reference point and the vehicle, and the speed of the vehicle. The output of the controller is the power supplied to the vehicle's motor. The term set of the linguistic variables distance and speed for instance, can be defined, as follows:

Table 2.1: Linguistic variables and their corresponding linguistic terms

Linguistic Variable	*Linguistic Terms*	*Abbreviations of Terms*
Distance	*Small, Medium , Big*	*S,M,B*
Speed	*Low, Medium, High*	*L,M,H*
Power(Braking)	*Hard, Medium, Light*	*H,M,L*

Since *distance* is a length, the base variable "u" associated with the linguistic variable *distance* may adopt values expressed in terms of length units. Assuming the length measurement ranges from 0 to 1000 cm, the universe of discourse is U=[0,1000]. A linguistic value belonging to distance makes physical sense through the definition of its membership function that confines its domain in terms of the variable "u". This restriction is the "meaning" of such a linguistic value. The number of sets and the choice of their membership functions depend on the type of the problem that is required to be solved.

There is no standard design method that can be followed to obtain either the most effective membership function types for their numbers. By increasing the number of membership functions, the behaviour of a fuzzy system may be enhanced. However this increases the number of rules and consequently increases computational time required.

2.10.1.2 Fuzzy rule-base

A fuzzy proposition is a statement expressed in a natural or artificial language. In contrast to classical logic propositions, a fuzzy proposition may adopt a truth-value from the interval [0,1]. For the former example from vehicle guidance, shown in Table 2.1, the following sentences are fuzzy propositions:

Distance is very Big , Speed is Low.

where the meanings of these propositions are determined by the corresponding membership functions. Fuzzy controllers normally deal with several input variables defined in different universe of discourses. Therefore, the compound fuzzy propositions that are formed using linguistic connectives such as *and, or, not,* etc, are more frequently encountered such as:

Distance is very Small and Speed is Low; *Distance is Big or Speed is High.*

The generation of the fuzzy rule is the second step in a fuzzy system and depends on the knowledge of experienced human operators, the fuzzy model of the plant concerned and the analysis of the system. The rule base is composed of two parts namely, the if-part and according to Godjevac [1997], a linguistics If-Then rule. This can be demonstrated as follows:

antecedent part (premise), expressed by: if <fuzzy proposition>,
consequent part, expressed by: then <fuzzy proposition>,

where the fuzzy propositions at the antecedent and consequent parts may be simple or compound. In a fuzzy controller, the antecedent part is related to the inputs of the controller whereas the consequent part is related to the outputs. Let us reconsider the former example from vehicle guidance [Godjevac, 1997]. Suppose then that in the corresponding controller there is a fuzzy rule defined as: *If Distance is very Small and Speed is High then Braking Power is Hard*.

2.10.1.3 Fuzzy inference

Fuzzy inference provides the conclusion from the rule-base and forms the intermediate stage between the fuzzification and defuzzification of the fuzzy system. There are two methods used to find the rules conclusion; namely Max-Min inference and Max-Product inference. Max-Min inference uses the Minimum operator to combine the antecedent of

the If-Then rules, which produces modified fuzzy sets for the outputs. These modified sets are then combined using the Maximum operator. For a set of *r* rules, the aggregated output using the Max-Min inference will be given as follows [Ross and Hoboken, 2004]:

$$\mu_k(y) = \max_k[\min(\mu_{A^k}(i), \mu_{B^k}(j))] \quad k=1,2,3,...,r \qquad (2.7)$$

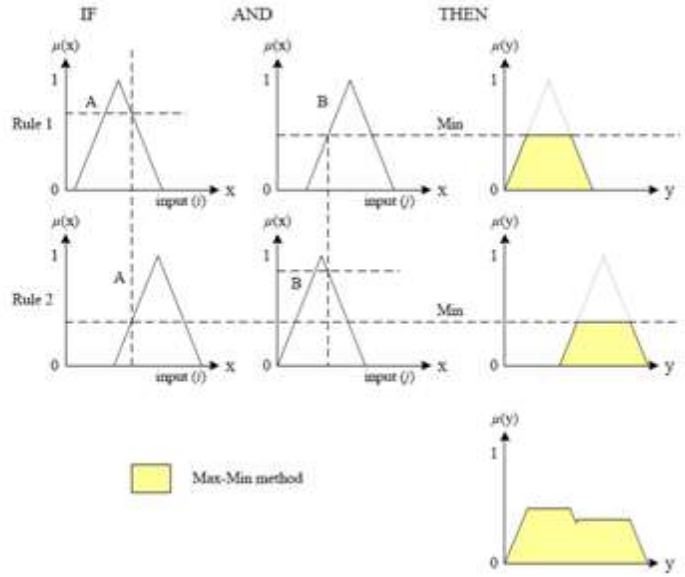


Figure 2.15: Example for the max-min inference method [Ross and Hoboken, 2004]

Max-Product inference employs the standard Product operator to combine the antecedent of the If-Then rules. Then the Maximum operator is used to combine these modified sets. For a set of *r* rules, the aggregated output using the Max-Product inference will be given as follows [Ross and Hoboken, 2004]:

$$\mu_k(y) = \max_k[(\mu_{A^k}(i).\mu_{B^k}(j))] \quad k=1,2,3,...,r \qquad (2.8)$$

Figure 2.15 demonstrates the Max-Min inference processes for two input variables *i* and *j*. Each of them is represented by two triangular fuzzy sets using two rules.

2.10.1.4 Defuzzification

Defuzzification is the process of mapping from a space of inferred fuzzy control action to a space of non-fuzzy control actions where the calculated crisp value is that which best represents the inferred control action. A number of defuzzification strategies exist, and it is a simple matter to invent more. The most popular defuzzification methods are the Centre-of-Area, Centre-of-Largest-Area, Centre-of-Sums, and Mean-of-Maximum. These methods are based on two basic mechanisms: centroid and maximum. The centroid methods are based on finding a balance point, while the Maximum methods search for the highest peak of weight (area) of each fuzzy set. Ross and Hoboken [2004] detailed these methods as follows:

Centre-of-Area (COA): COA essentially calculates the centroid of the total area representing the fuzzy output set as given by:

$$\mu^* = \frac{\int \mu(y).ydy}{\int \mu(y)dy} \qquad (2.9)$$

where $\int \mu(y).dy$ is the area of the output fuzzy set .

Centre-of-Largest-Area (CLA): CLA evaluates each implication result and then computes the centroid of the largest area to represent the output fuzzy, as given by:

$$\mu^* = \frac{\int \mu_m(y).ydy}{\int \mu_m(y)dy} \qquad (2.10)$$

where $\int \mu_m(y).dy$ presents the largest surface in the output fuzzy set.

Centre-of-Sum (COS): This process calculates the algebraic sum of individual output fuzzy sets instead of their union, as illustrated in the following equation:

$$\mu^* = \frac{\int y \sum_{k=1}^{n}(\mu_k(y)d_y)}{\int \sum_{k=1}^{n}(\mu_k(y)d_y)} \quad (2.11)$$

Mean-of-Maximum (MOM): MOM is often referred to as the Middle-of-Maximum. It is used when the maximum membership function is not unique, and is expressed as follows:

$$\mu^* = \sum_{m=1}^{M}\left(\frac{\mu_m}{M}\right) \quad (2.12)$$

where μ_m is the mean of all the maximums of the fuzzy output with the highest degree of truth, and M is the integer number of such peaks [Ross and Hoboken, 2004]. The selection of the defuzzification method to be used here is based on criteria which can be summarised as follows [Driankov et al., 1993; Ross and Hoboken, 2004]:

- Continuity, such that a small change in the input does not lead to a big change in output
- Computational complexity, where the computational time needed is an important criterion for the practical choice of fuzzy inference method.
- Plausibility, where the method is considered to be plausible if the support for the output fuzzy set has the highest degree of membership.
- Weight counting (disambiguity), in that weight information is not lost due to an inability to decide. (For instance, if there are two large areas, then CLA will not be able to decide).

2.10.2 Neural Network

A neural network, inspired by aspects of the structure of biological neural networks, is a powerful data modelling tool that is able to capture and represent complex input/output relationships. The field of research into neural network was established before the advent of computers. Neural networks have a remarkable ability to derive meaning from complicated or imprecise data, and can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

They have been applied in many fields, including aerospace, financial, defence, electronics, and robotics and computer vision. For instance, a vision-guided mobile robot navigation system, called NEURO-NAV, based on a neural network was described by [Meng and Kak, 1993]. In NEURO-NAV where primitive navigational tasks such as hallway following, and landmark detection are implemented using neural networks. In a more recent study, Janglova [2004] introduced the intelligent control of an autonomous robot which was claimed to move safely in a partially structured environment. The method essentially constructs a collision-free path for the robot based on two separate neural network architectures, where the first determines free-space using an ultrasound ranger finder, while the second determines possible navigation steps towards the goal [Janglova, 2004b]. Chi and Lee [2011] defined a neural network based obstacle control system that able to guide the mobile robots traverse through a maze with arbitrary obstacles [Chi and Lee, 2011]. Additional research in neural network based obstacle avoidance techniques can be found in [Lynch et al., 1999; Trieu et al., 2008; Fazl-Ersi and Tsotsos, 2009].

Camera calibration is a key step in 3D computer vision, which involves extracting metric information from 2D images and usually entails the solution of complex non-linear equations. However, artificial neural networks demonstrate outstanding non-linear mapping performance which can avoid these processes and makes it unnecessary to know parameters of the cameras, such as focus and distortions as well as the geometry of the system. An example of the implementation of intelligent camera calibration for a monocular vision camera is defined by [Li Guo and Li Guang, 2011] .

Their algorithm employed the Harris corner extraction algorithm to obtain input and output data for a multi layer neural network architecture. Other relevant examples of neural network based camera calibration can be found in [Lynch et al., 1999; Cai et al., 2010; Xiong et al., 2010]. The background knowledge of artificial neural networks associated with this dissertation will be detailed in the following section. Further discussion of neural networks, including their operation and the use of control in the area of calibration, obstacle avoidance and distance estimation, is provided in Chapter 5.

2.10.2.1 Background to Artificial Neural Network

The human nervous system consists of small cellular units, called neurons. When connected in tandem, these form a nerve fibre. A biological neural net is a distributed collection of these nerve fibres. A neuron receives electrical signals from its neighbouring neurons, processes those signals and generates signals for other neighbouring neurons attached to it. The operation of a biological neuron, which decides the nature of the output signal as a function of its input signals, is not yet clearly understood. However, most biologists are of the opinion that, after receiving signals, a neuron estimates the weighted average of the input signals and limits the resulting amplitude of the processed signal using a non-linear inhibiting function [Fu, 1994]. Further details about biological neurons can be found in Anderson [1972].

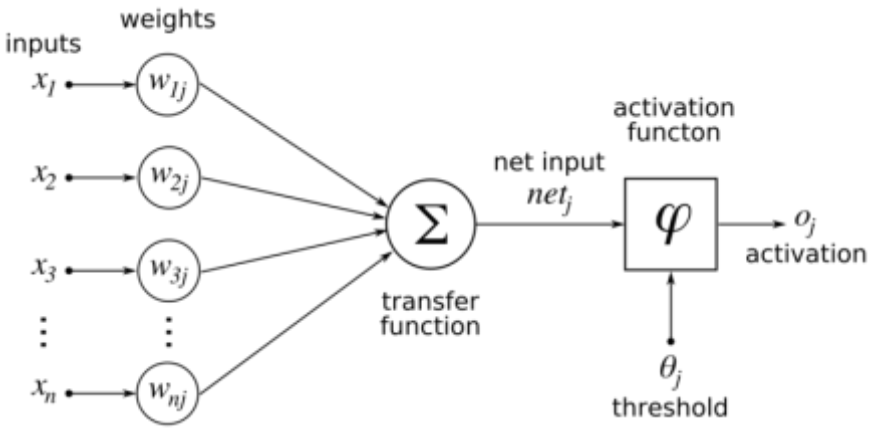


Figure 2.16: Artificial neuron model [Chrislb, 2005]

Artificial neurons are similar to their biological counterparts. They have input connections which are summed together to determine the strength of their output, which is the result of the sum being fed into an activation function, as illustrated in Figure 2.16. Though many activation functions exist, the most common is sigmoid activation function, which outputs a number between 0 (for low input values) and 1 (for high input values). The result of this function is then passed as the input to other neurons through more connections, each of which are weighted. These weights determine the behaviour of the network. The most common activation functions are given as follows [Hagan et al., 1996]:

Linear: This function is also called 'purelin' activation function providing linear relationships between the input and output, which is defined as follows:

$$f(v) = v \qquad (2.13)$$

Sigmoid: This function can range between 0 and 1, and an example is the log-sigmoid function which is defined as follows.

$$f(v) = \frac{1}{1+e^{-v}} \qquad (2.14)$$

Symmetric Sigmoid Function**:** This function can range from -1 to 1, and the hyperbolic tangent sigmoid is defined as follows**:**

$$f(v) = \frac{e^{v}-e^{-v}}{e^{v}+e^{-v}} \qquad (2.15)$$

Step function: This is a special type of function whose graph is a series of line segments. This kind of step activation function is useful for binary classification

schemes. In other words, if the aim is to classify an input pattern into one of two groups, a step activation function as given below can be used as a binary classifier.

$$f(v) = \begin{cases} 1, & v > 0 \\ \text{None}, & 0 < v \\ 0, & v = 0 \end{cases} \qquad (2.16)$$

2.10.2.2 Topologies of artificial neural networks

Depending on the nature of the problems involved, artificial neural networks can be organized in different structural arrangements (topologies). Common topologies can be classified into two groups namely; feed-forward and recurrent neural networks. Feed-forward networks mainly comprise single layer perceptron and multiple layer perceptron topologies. The most popular recurrent topologies are simple recurrent and Hopfield networks**.** These topologies are defined below [Konar, 2000].

Single Layer Perceptron: The earliest kind of neural network is a single-layer perceptron network, which consists of a single layer of output nodes where the inputs are fed directly to the outputs via a series of weights. In this way it can be considered the simplest kind of feed-forward network. The sum of the products of the weights and the inputs is calculated in each node and, for instance, if the value is above some threshold (typically 0) the neuron fires and takes the value 1; otherwise it takes the value -1.

Perceptrons can be trained by a simple learning algorithm that is usually called the delta-rule**.** It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent. Single-unit perceptrons are only capable of learning linearly separable patterns [Fu, 1994].

Multi Layer Perceptron: This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. This means that

each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function. The universal approximation theorem for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds only for restricted classes of activation functions, such as sigmoidal functions. Multi-layer networks use a variety of learning techniques, the most popular being back-propagation. Here the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error-function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case one can say that the network has learned a certain target function [Hagan et al., 1996]. To adjust weights properly one applies a general method for nonlinear optimization task that is called gradient descent. For this the derivation of the error-function with respect to the network weights is calculated and the weights are then changed such that the error decreases (thus going downhill on the surface of the error function). For this reason back-propagation can only be applied on networks with differentiable activation functions.

Simple Recurrent Network: A simple recurrent network (SRN) is a variation of the multi-layer topology. Contrary to the feed-forward networks, the dynamical properties of the network are important. In some cases, the activation values of the units undergo a relaxation process such that the neural network will evolve to a stable state in which these activations do not change. In other applications, the change of the activation values of the output neurons is significant, such that the dynamical behaviour constitutes the output of the neural network [Hagan et al., 1996].

Hopfield neural networks: The Hopfield net is a recurrent neural network in which all connections are symmetric. This network has the property that its dynamics are

guaranteed to converge. If the connections are trained using Hebbian learning then the Hopfield network can perform robust content-addressable memory, and it is robust to connection alteration.

2.10.2.3 Learning using neural networks

Artificial neural nets have been successfully used for recognizing objects from their feature patterns. For the classification of patterns, neural networks should be trained prior to the phase of recognition process. The process of training a neural net can be broadly classified into three typical categories, namely supervised, unsupervised and reinforcement.

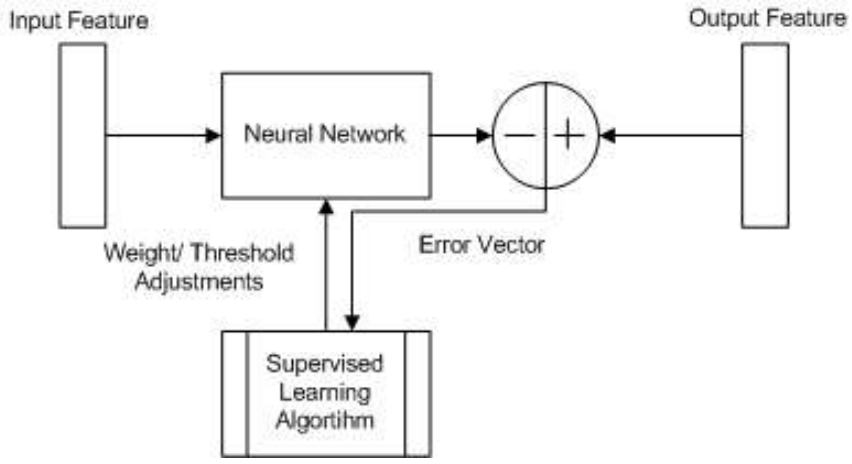


Figure 2.17: The supervised learning process

Supervised Learning: The supervised learning process shown in Figure 2.17 requires a supervisor that submits both the input and the target patterns for the objects to be recognized. For instance, to classify objects into "ball", "skull", and "orange", the features of these objects must be submitted, such as average curvature, the ratio of the largest solid diameter to its transverse diameter, and so on, as the input feature patterns. Conversely, to identify one of the three objects, one may use a 3-bit binary pattern where each bit corresponds to one object. Given such input and output patterns for a number of objects, the task of supervised learning calls for the adjustment of network parameters such as weights and non-linearities, which can consistently satisfy the input-output requirements for the entire object class, which is spherical objects in this

example). The most common supervised learning algorithms is the back-propagation training algorithm [Konar, 2000].

Unsupervised Learning: If the target pattern is unknown, many recognition problems require the process of unsupervised learning which attempts to generate a unique set of weights for one particular class of patterns. For instance, consider a neural net of recurrent topology having *n* nodes. Assume that the feature vector for spherical objects is represented by a set of *n* descriptors, each assigned to one node of the structure. The objective of unsupervised learning process is to adjust the weights autonomously, until an equilibrium condition is reached when the weights do not change further [Konar, 2000].

The process of unsupervised learning, therefore, maps a class of objects to a class of weights. Generally, the weight adaptation process is described by a recursive functional relationship. Depending on the topology of neural nets and their applications, these recursive relations are constructed intuitively. The Hopfield network is a typical example of unsupervised learning [Hagan et al., 1996].

Reinforcement Learning: This learning process may be considered as an intermediate form of the above two types of learning. In this process, the learning machine needs to undertake some action on the environment and receives a feedback response from the environment. The learning system grades its action good (rewarding) or bad (punishable) based on the environmental response and adjusts its parameters accordingly. Generally, parameter adjustment is continued until an equilibrium state occurs, following which there will be no more changes in its parameters.

2.10.2.4 Back-propagation algorithm

Back-propagation is one of the most popular techniques used for training neural networks, and an example of a three-layer back-propagation network is illustrated in Figure 2.18. This technique is primarily useful for feed-forward networks which have

no feedback, or put more simply that have no connections which loop. Since it is a supervised training algorithm, both the input and the target patterns must be given. For a given input pattern, the output vector is estimated through a forward pass on the network. After the forward pass is completed, the error vector at the output layer is estimated by taking the difference between the target pattern and the generated output vector. A function of errors of the output layered nodes is then propagated back through the network to each layer for the adjustment of weights in that layer. The weight adaptation policy used in back-propagation algorithms is derived from the principle of steepest descent approach [Fu, 1994; Hagan et al., 1996].

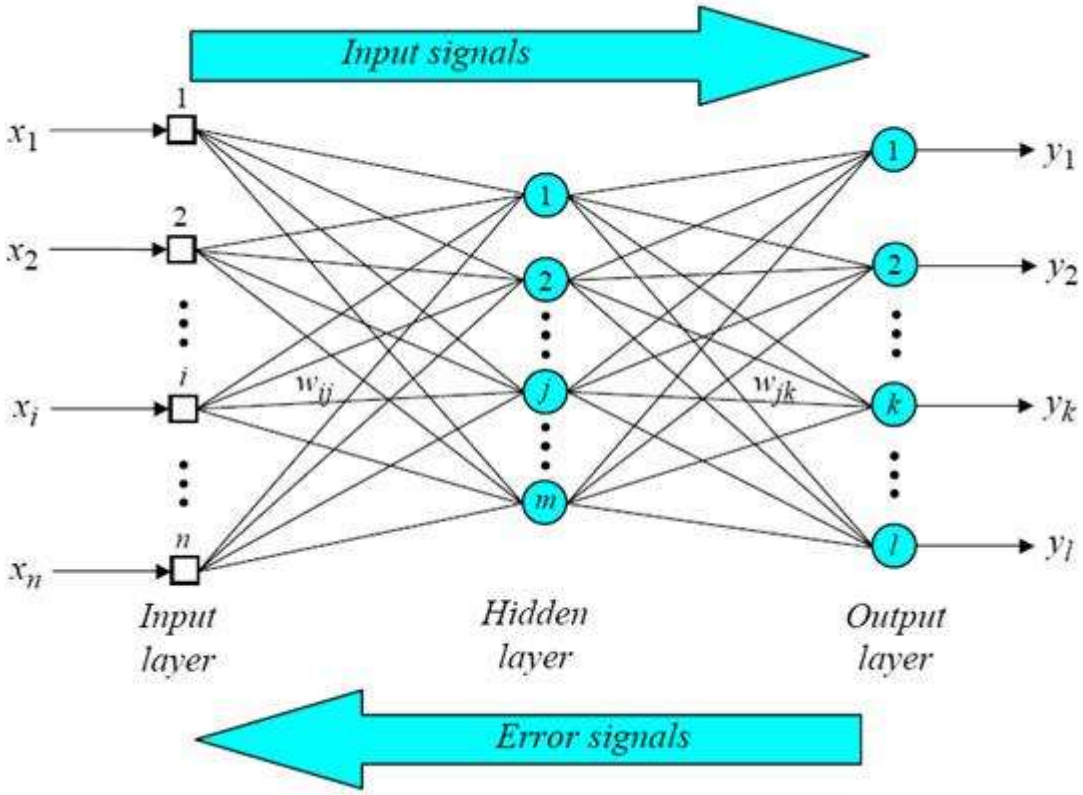


Figure 2.18: An example of three-layer back-propagation network

The steepest descent method is used for finding the minimum. It consists of computing the gradient of the function, then taking a small step in the direction of a negative gradient, which hopefully corresponds to a decreased function. The gradient of a multivariate function is the vector of partial derivatives, one for each variable. The

gradient is a vector in the space of all variables that the function depends on. It points in the direction of steepest increase [Hagan et al., 1996].

Consequently, this is essentially a two pass algorithm where the forward pass computes the outputs of all nodes, working from the inputs to the output node where the error is recorded, and the weights are held fixed during the forward pass. However, in the backwards pass, the weight correction starts from the final layer (output node) back towards the inputs.

2.11 Summary

Mobile robots have had a long history since the *Shakey*, the first mobile robot, was established. In this chapter relevant information regarding research, particularly in vision based mobile robotics field has been presented. Mobile Robot architectures, the foundations of their control and navigation systems enabling the robot to safely navigate to its goal position were introduced. The review mainly has focused on three major trends in visual based mobile robot navigation: map-based navigation, map-building-based navigation and mapless navigation. Focusing on the topic of mapless navigation, the most important techniques were summarized including visual homing and visual servoing concepts. These algorithms fall naturally into four categories: optical flow, appearance-based, object recognition and navigation techniques based on feature tracking. All of these have been defined and associated with relevant studies. Additionally, obstacle avoidance techniques using qualitative information have been defined. Feature-based algorithms require reliable solutions to the problems of consistent feature extraction and correspondence to ensure successful operation. Therefore, the current state of one of the strongest local feature detectors has been introduced. Sensors assist the mobile robot to acquire information about the external surroundings, while fuzzy logic and neural network facilitate robust and smooth motion for mobile robot. Finally, it was explained how the knowledge described has been integrated into software architectures to achieve specific tasks.

CHAPTER 3

VISION BASED OBSTACLE AVOIDANCE

This chapter focuses on a new vision based obstacle avoidance technique combining optical flow and appearance-based methods. The first section provides a brief description of existing vision based obstacle avoidance techniques. The next section details the proposed optical flow and appearance-based algorithms, followed by the design of these algorithms for obstacle avoidance. The final section of the chapter presents results for the proposed architecture of experiments conducted using Microsoft Robotics Studio. The results confirm that the proposed method can provide an alternative and robust solution for mobile robots using a single monocular camera as the only sensor used avoiding obstacles.

3.1 Vision Based Obstacle Avoidance Techniques

One of the key research problems in mobile robot navigation concerns methods for obstacle avoidance. In order to cope with this problem, most autonomous navigation systems rely on range data for obstacle detection. Ultrasonic sensors, laser rangefinders and stereo vision techniques are widely used for estimating range. However, all of these have drawbacks. Ultrasonic sensors suffer from poor angular resolution, and laser range finders and stereo vision systems are relatively expensive. Moreover the computational complexity of stereo vision systems is another key challenge [Ulrich and Nourbakhsh, 2000]. In addition to their other shortcomings, range sensors are not capable of differentiating between different types of ground surfaces such as pavements and adjacent flat grassy areas. Overall the computational complexity of the avoidance algorithms and the cost of sensors are the most critical factors for real time applications. The use of monocular vision based systems can avoid these problems and are able to provide appropriate solutions to the obstacle avoidance problem. There are two general types of vision based obstacle avoidance techniques; those that compute apparent motion, and those that rely on the appearance of individual pixels for monocular vision

based obstacle avoidance systems. The first group is called optical flow based techniques, in which the main idea is to control the robot using optical flow data, from which the heading direction of the observer and time-to-contact values are obtained [Guzel and Bicker, 2010]. One way of using these values is by acting to achieve a certain type of flow. For instance, to maintain ambient orientation, the type of optic flow required is to detect no flow at all. If some flow is detected, then the robot should change the forces produced by its effectors so as to minimize this flow, based on the Law of Control [Contreras, 2007] .

A second group of techniques is called the appearance-based methods, which in essence rely on qualitative information. They utilize basic image processing techniques which consist of detecting pixels different in appearance from those of the ground and then classifying them as obstacles. The algorithms used perform in real-time, provide a high-resolution obstacle image, and can operate in a variety of environments [DeSouza and Kak, 2002]. The main advantages of these two types of conventional methods are their ease of implementation and ready availability for real time applications.

3.2 Optical Flow for Obstacle Avoidance

Optical flow, as illustrated in Figure 3.1, is an approximation of the motion field, summarizing the temporal changes in an image sequence**.** Optical flow estimation is one of the central problems in computer vision. There are several methods which can be employed to determine optical flow, namely: block-based, differential, phase correlation and variational methods [Barron et al., 1994; Atcheson et al., 2009]. There has been wide interest in the use of optical flow for vision-based mobile robot navigation. The visual control of motion in flying insects has been shown to provide important clues for navigational tasks such as centred flight in corridors and the estimation of distance travelled, encouraging new biologically-inspired approaches to mobile robot navigation using optical flow. Behaviour such as corridor centring, docking and visual odometry have all been demonstrated in practice using visual motion for the closed loop control of a mobile robot [Szenher, 2008]. In recent years, there has been growing amount of literature on optical flow based mobile robot navigation. Bernardino and Santos-Victor

[1998] used biologically inspired behaviours based on stereo vision for obstacle detection. A trinocular vision system for mobile robot navigation has been also proposed [Argyros and Bergholm, 1999]. These methods, in some ways, emulate corridor following behaviour; nevertheless their main disadvantage is the need to employ more than one camera. Alternatively, a number of studies relying on monocular vision have proposed the employment of optical flow techniques for mobile robot navigation [Szabo et al., 1996; DeSouza and Kak, 2002; Souhila and Karim, 2007; Guzel and Bicker, 2010].

a) Destination **b)** Source **c)** Animation

Figure 3.1: Illustration of flow vectors and motion animation, (a) destination image, (b) source image, (c) movement animation

Differential methods are widely used for these navigation tasks, and are mainly based on partial derivatives of the image signal and/or the flow field sought and higher-order

partial derivatives. McCarthy and Barnes [2004] conducted a comprehensive study analyzing the performance of differential methods in mobile robot navigation. The results demonstrated similar levels of performance in the most popular differential methods [McCarthy and Barnes, 2004]. Accordingly, in the present study a multi resolution version of the conventional Horn-Schunk algorithm [Horn and Schunck, 1981], which is one of the most popular optical flow estimation methods, is used to estimate flow vectors in steering the robots.

3.2.1 Horn-Schunk method for obstacle avoidance

The Horn–Schunck algorithm yields a high density of flow vectors. On the negative side, this process is more sensitive to noise than local methods.

The main idea behind the technique assumes that, for a given scene point, the corresponding image point intensity *I* remain constant over time, which is referred to as the conservation of image intensity [Atcheson et al., 2009]. Therefore, if two consecutive images have been obtained subsequent time intervals, the basic idea is to detect motion using image differencing. If any scene point projects onto an image point (x, y) at time *t* and onto an image point $(x + \delta x, y + \delta y)$ at time $(t + \delta t)$, the following equation is inferred based on the assumption of the conservation of image intensity.

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \qquad (3.1)$$

Expanding the right-hand side of Eq. 3.1 using a Taylor series about (x, y, t), and ignoring the higher order terms then by rearrangement gives the following expression:

$$I(x, y, t) = I(x, y, t) + \delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} + \varepsilon \qquad (3.2)$$

where ε illustrates the second and higher order terms in δx, δy and δt. Further rearrangement gives the following equation:

$$\delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} = 0 \qquad (3.3)$$

A simpler expression is obtained by dividing by δt throughout where movement along the horizontal ($\frac{\delta x}{\delta t}$), and vertical ($\frac{\delta y}{\delta t}$) directions are u and v respectively. Conducting these rearrangements and denoting partial derivatives of I as I_x, I_y and I_t gives the differential flow equation shown in following expressions:

$$I_x u + I_y v + I_t = 0 \qquad (3.4)$$

where, I_x, I_y and I_t are the partial derivatives of image brightness with respect to x, y and t, respectively. Having one equation with two unknowns δx, δy for each pixel presents an aperture problem of the optical flow algorithms. To find the optical flow, another set of equations is needed using some additional constraint. All optical flow methods introduce additional conditions for estimating the actual flow [Horn and Schunck, 1981]. Depending on the approach, a regularizing term associated with smoothness is added to the general flow equation. Horn and Schunk [1981] stated that neighbouring pixels have the same velocity as moving objects, so the brightness pattern of an image changes regularly. This constraint is demonstrated by minimizing the squares of gradient magnitudes. The smoothness of an optical flow area can also be estimated by calculating the Laplacian of optical flow vectors speed in both horizontal and vertical directions denoted by u and w respectively, illustrated in following expressions:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \qquad (3.5)$$

where $E_i = I_x u + I_y v + I_t$ and $E_s = \nabla^2 u + \nabla^2 v$. The aim here is to minimize the total error given by the following expressions (3.6), which includes σ as the regularization parameter controlling the association between the detail and smoothness. High values of σ lead to the smoothness constraint being dominant and result in a smoother flow.

$$(u, v) = argmin \iint \left({E_i}^2 + \sigma^2 {E_s}^2 \right) dxdy \qquad (3.6)$$

3.2.1.1 Estimating the partial derivatives

One of the most significant challenges now is to calculate with precision I_x, I_y, the first-order differentials. While there are many formulae for approximate differentiation, those concerning the neighbouring points in quadrants in successive images give approximate estimates of the gradient based on the two frames as follows [Nixon and Aguado, 2008]:

$$I_{fx} \approx \left(I(0)_{x+1,y} + I(1)_{x+1,y} + I(0)_{x+1,y+1} + I(1)_{x+1,y+1} \right)$$

$$I_{sx} \approx \left(I(0)_{x,y} + I(1)_{x,y} + I(0)_{x,y+1} + I(1)_{x,y+1} \right)$$

$$I_x \approx \frac{I_{fx} - I_{sx}}{4}$$

$$I_{fy} \approx \left(I(0)_{x,y+1} + I(1)_{x,y+1} + I(0)_{x+1,y+1} + I(1)_{x+1,y+1} \right)$$

$$I_{sy} \approx \left(I(0)_{x,y} + I(1)_{x,y} + I(0)_{x+1,y} + I(1)_{x+1,y} \right)$$

$$I_y \approx \frac{I_{fy} - I_{sy}}{4} \qquad (3.7)$$

Additionally, the time differential, I_t is given by the difference between two pixels along the two faces of the cube, as

$$I_{ft} \approx \left(I(1)_{x,y} + I(1)_{x+1,y} + I(1)_{x,y+1} + I(1)_{x+1,y+1} \right)$$

$$I_{st} \approx \left(I(0)_{x,y} + I(0)_{x+1,y} + I(0)_{x,y+1} + I(0)_{x+1,y+1} \right)$$

$$I_t \approx \frac{I_{ft} - I_{st}}{4} \qquad (3.8)$$

where *I(0)* and *I(1)* refer to two successive images for both Equations 3.7 and 3.8.

In addition, the Laplacian is estimated by subtracting the value at a point from a weighted average of the values at neighbouring points [Gonzalez and Woods, 2002].

Thus an approximation to Laplacians of *u* and *v* can be considered as follows [Nixon and Aguado, 2008]:

$$\nabla^2 u = (\ \bar{u}_{x,y} - u_{x,y}\ ) \quad \text{and} \quad \nabla^2 v = (\ \bar{v}_{x,y} - v_{x,y}\ ) \qquad (3.9)$$

Nixon and Aguado [2008] define $\bar{u}$ and $\bar{v}$ as follows:

$$u_{l=} \frac{(u_{x-1,y} + u_{x,y-1} + u_{x+1,y} + u_{x,y+1})}{2}$$
$$u_{r=} \frac{(u_{x-1,y-1} + u_{x-1,y+1} + u_{x+1,y-1} + u_{x+1,y+1})}{4}$$

$$v_{l=} \frac{(v_{x-1,y} + v_{x,y-1} + v_{x+1,y} + v_{x,y+1})}{2}$$
$$v_{r=} \frac{(v_{x-1,y-1} + v_{x-1,y+1} + v_{x+1,y-1} + u_{x+1,y+1})}{4}$$

$$\bar{u}_{x,y} = u_l + u_r$$
$$\bar{v}_{x,y} = v_l + v_r \qquad (3.10)$$

3.2.1.2 Minimization

As previously mentioned, the main aim is to minimize the sum of the errors in the equation for the rate of change of image brightness and the estimate of the departure from smoothness in the velocity flow. In order to obtain appropriate values for optical flow velocity (u,v), total error (E^2) is differentiated so as to be minimized, which is given in the following expressions [Horn and Schunck, 1981]:

$$E^2 = {E_i}^2 + \sigma^2 {E_s}^2 \qquad (3.11)$$

Horn and Schunk [1981] accomplished the minimization of the function by differentiating the total error function as shown in Equation 3.12. Further details can be found in [Horn and Schunck, 1981; Weinstock, 2008].

$$\frac{\partial E^2}{\partial u} = 2(I_x u + I_y v + I_t)I_x - 2\sigma^2(\bar{u} - u)$$
$$\frac{\partial E^2}{\partial v} = 2(I_x u + I_y v + I_t)I_y - 2\sigma^2(\bar{v} - v)$$

Setting these two derivatives equal to zero leads to two equations using *u* and *v*

$$(\sigma^2 + I_x^2)u + I_x I_y v = (\sigma^2 \bar{u} - I_x I_t)$$
$$(\sigma^2 + I_y^2)v + I_x I_y u = (\sigma^2 \bar{v} - I_y I_t)$$

(3.12)

This is linear in *u* and *v* and may be solved for each pixel in the image. A direct solution of these equations, such as using Gaus-Jordan elimination [Bogacki, 2005], would be very costly. Instead, an iterative Gauss Seidel approach is used to reduce the cost and obtain the flow vectors, [Horn and Schunck, 1981]. Since the solution depends on the neighbouring values in the flow field, it must be repeated once the neighbouring pixels have been updated. The following iterative scheme is derived:

$$u^{n+1} = \bar{u}^n - \left(\frac{I_x \times (I_x \bar{u}^n + I_y \bar{v}^n + I_t)}{\sigma^2 + I_x^2 + I_y^2}\right) ; \; v^{n+1} = \bar{v}^n - \left(\frac{I_y \times (I_x \bar{u}^n + I_y \bar{v}^n + I_t)}{\sigma^2 + I_x^2 + I_y^2}\right)$$

(3.13)

where the superscript *(n+1)* denotes the next iteration which is to be calculated and *(n)* refers the preceding estimated result [Horn and Schunck, 1981; Nixon and Aguado, 2008].

3.2.1.3 Multi-scale optical flow estimation

The multi-scale coarse-to-fine approach is incorporated in most modern algorithms for optical flow estimation, in order to support large motion and for improved accuracy. This approach relies on estimating the flow in an image pyramid which is constructed by repeatedly down-sampling the image by a factor of two. The optical flow can then be found for the smallest image in the pyramid, and this is then used to unwrap the next smallest image. Interpolation is used for the fractional pixel locations. This process is then iterated until the original image resolution is reached [Lucas and Kanade, 1981]. The advantage of the pyramid structure with regard to optical flow is that it can efficiently capture large motions in a large contiguous area of an image. An example illustrating the algorithm is presented in Figure 3.2. Szelinski [2010] presented the detailed working schema of the coarse to fine algorithm, illustrated in Figure 3.3.

The following pseudo code can be considered to define the algorithm:

1. Create a Gaussian pyramid for both frames.
2. Repeat until reaching the finest level.
3. Apply corresponding flow algorithm at the current level of the pyramid.
4. Propagate flow by using bilinear interpolation to the next level, where it is used as an initial estimate.
5. Go back to step 2.

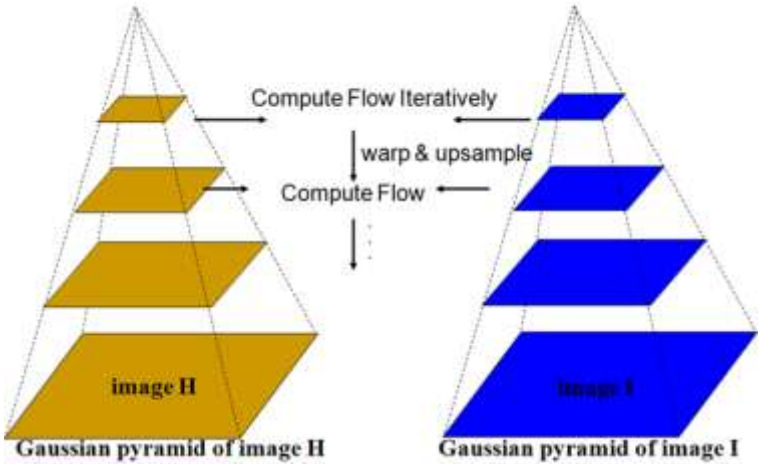


Figure 3.2: Multi scale coarse to fine approach

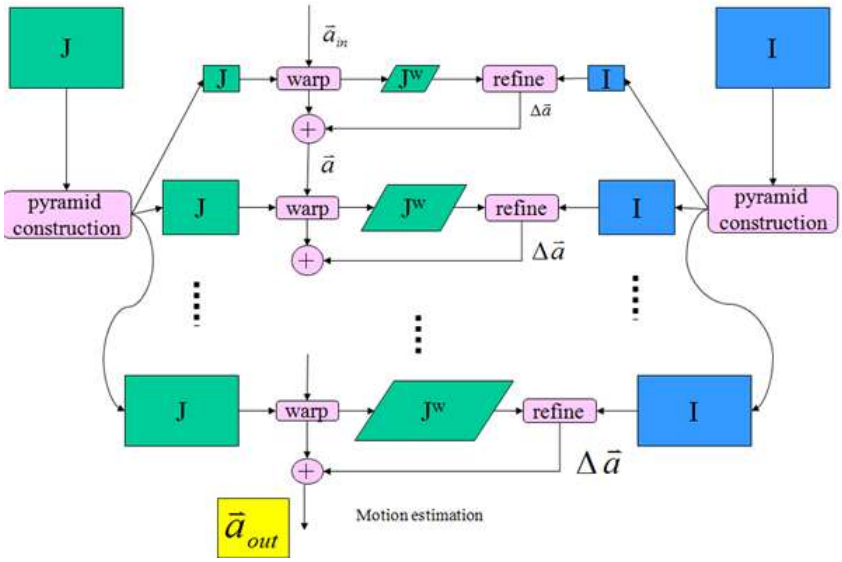


Figure 3.3: Detailed working schema of the multi scale coarse to fine algorithm [Szelisnski, 2010]

The pseudo code of the Horn-Schunk [1981] optical flow algorithm has given by Nixon and Aguado [2008], is as follows:

Horn-Schunk Algorithm:

Require *img1 and img2 are gray scale images, and each has R rows and N columns*
Ensure: u_x, v_y *stores flow vectors*
Procedure *Compute Flow*
Initial assignment for the parameters
Load images (img1,img2)
Until the end of the iterations
For x from 1 to rows
For y from 1 to columns
Calculate derivatives I_x, I_y, I_t
Calculate averages A_u, A_v
Update Estimates using derivatives and averages, to obtain temporal flow,(t_x,t_y*)*
end_for
end_for
For x from 1 to rows
For y from 1 to columns
Update(u_x,u_y) based on t_x,t_y
end_for
end_for
end_until
EndProcedure

The *CImg* library facilitates image analysis in any format and provides useful functions for image processing, and it is integrated into the developed software. A multi scale

version of this optical flow algorithm is provided by the *CImg* library which was integrated into the corresponding navigation problem with major modifications.

3.2.2 Applying optical flow for obstacle avoidance

The aim of this section is to adapt the previously discussed multi-scale optical flow technique in behavioural based robot architecture for obstacle avoidance and mobile robot navigation in a partially cluttered environment. The proposed system relies on a single monocular vision camera and tries to understand its environment by analyzing the data taken from image sequences. A block diagram of the proposed navigation algorithm is shown in Figure 3.4.

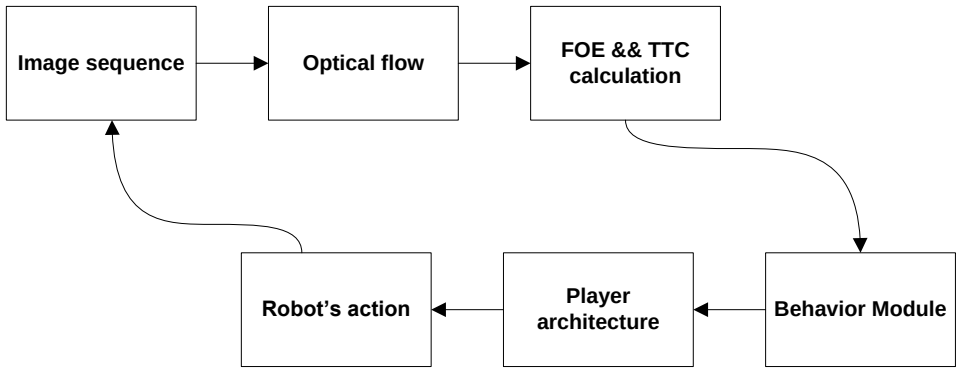


Figure 3.4: Flowchart of the optical flow based navigation algorithm

Flow vectors are utilized to navigate autonomous systems based on the Balance Strategy, shown in the following equation (Eq. 3.14), and the depth information which is extracted from the image sequence using Focus of Expansion (FOE) and Time To Contact values (TTC). The balance strategy is essentially a control law that can be used by mobile robots to avoid obstacles, to trail moving targets, or to escape from approaching enemies. The fundamental idea behind the balance strategy is that of the motion parallax so that, when the agent is translating, closer objects give rise to faster motion across the retina than further away objects. It also takes advantage of perspective in that closer objects also take up more of the field of view, biasing the average towards their associated flow [Temizer, 2001]. The agent turns away from the side of greater

flow, which indicates a possible approach to a stationary object. In order to achieve this purpose, the agent can adjust the direction and the magnitude of its rotation by looking at the difference between the magnitude of right and left sides optical flows, this control law can be expressed as follows [Duchon et al., 1998; Temizer, 2001]:

$$\Delta(F_l - F_r) = \left(\frac{\sum|w_L| - \sum|w_R|}{\sum|w_L| + \sum|w_R|}\right) \quad (3.14)$$

where w_L and w_r present the magnitude of right and left flows respectively.

3.2.2.1 FOE and TTC calculation

When one moves through a world of static objects the visual scene is projected on the retina and appears to flow past. In fact, for the translational motion of the camera, image motion everywhere is directed away from a singular point corresponding to the projection of the translation vector. This is called the *focus of expansion* (FOE), which is the point from which all optical flow vectors emerge, and both components of the optical flow vector are null at such a point (u= 0 and v= 0) [Kröse et al., 2000]. Essentially, in order to find the FOE, the calculated optical flow field is searched for a specific point in which the directions of the vectors in the field cross each other. The estimation of *time-to-contact* (TTC) is another useful tool for autonomous mobile robot navigation, where accurate estimates of TTC for approaching objects are crucial. TTC uses visual information to judge distance and speed of action with respect to time [Duchon et al., 1998]. The source of this visual information comes from the movement of the agent towards an object or of the object/surface towards the agent. These movements provide the visual systems with important information about the constantly changing environment, allowing appropriate actions to be produced. For instance, when the agent moves at a constant speed, the TTC value with a point of interest, in terms of the numbers of remaining frames that will be grabbed by the vision systems before contact occurs, can be basically estimated from the ratio of the distance of that point in the image plane from the focus of expansion to the rate of change in this distance (divergence from the FOE) [Temizer, 2001; Souhila and Karim, 2007]. The TTC calculation is also widely used in the field of robotics. Knowledge of the robot velocity

or its initial distance from the object is not required; however, the approach only works properly in the case of a static environment, and it also implies that the robot is moving with a constant velocity. For the present research, in order to derive more reliable and sensitive results from proposed algorithm, the image is segmented into vertical regions. The optical flow-based image segmentation is also called the segmentation of movement, which consists in grouping the image pixels that perform the same movement [De Oliveira Caldeira et al., 2007]. The number of regions is manually estimated depending on the problem and image resolution, three were used in this research. Thus, an image with n×m resolution is divided into three equal columns such that each column region has $\frac{n}{3}$×m optical flow vectors. Having the coordinates of the FOE and the optical flow field, the TTC corresponding to the i_{th} region of the image (τ_i) is calculated by Eq. 3.15.

$$\boldsymbol{\tau_i} = \sqrt{\frac{(\boldsymbol{x_i}-\boldsymbol{FOEx})^2+(\boldsymbol{y_i}-\boldsymbol{FOEy})^2}{\sqrt{\boldsymbol{u_i^2}+\boldsymbol{v_i^2}}}} \qquad (3.15)$$

where x and y are the centre points of the considered region; $FOEx$ and $FOEy$ presents the x and y coordinates of the FOE point in the image, and u and v are the optical flow components of the i_{th} region. Additionally, total flow magnitudes are assigned to the corresponding regions as follows:

$$Lflow = \left(\sum_{i=1}^{n/3} \boldsymbol{\rho_i}\right)$$
$$Cflow = \left(\sum_{i=n/3}^{2n/3} \boldsymbol{\rho_i}\right)$$
$$Rflow = \left(\sum_{i=\frac{2n}{3}}^{n} \boldsymbol{\rho_i}\right) \qquad (3.16)$$

where ρ_i refers to the total magnitude of flow vectors in the i_{th} region. Consequently, each region is considered with corresponding TTC values and flow magnitudes extracted by the current image sequence.

The values of these regions are used to activate the behaviour module in which three independent task-achieving behaviours are performed.

3.2.2.2 Behavioural module and implementation algorithm

In mobile robotics, earlier work focussed on the "sense-model-plan-act" strategy, requiring intensive computation to infer the location and identity of objects, updating a central world model, and planning a course of action to achieve some defined goal state [Duchon et al., 1998]. In contrast, recent studies found that it is more beneficial to decompose the navigation task into the multiple independent task-achieving modules, called behaviours [Brooks, 1986]. Behavioural architectures were previously discussed in Chapter 2.

In order to provide reliable navigation, the behavioural architecture, employed in this study is composed of three layers (behaviours), namely: steering, change direction and stop. The architecture is inspired by the conventional subsumption architecture where each layer implements a particular goal of the agent, and higher layers are increasingly abstract [Brooks, 1986]. The stop behaviour has the highest priority whereas the steering behaviour has the lowest priority.

State machines are a good way of implementing and presenting robotic architectures consisting of several behaviours, as illustrated in Figure 3.5. According to the proposed architecture, the robot is designed to wander around any cluttered indoor environment whilst not colliding with any obstacle. It reacts to the presence of an obstacle in the environment by adjusting its turning rate. The robot starts its initial movement with forward motion behaviour at a constant speed (v_c) and initial (0°) heading angle. It navigates in the forward direction using steering behaviour (see Eq. 3.17) until it encounters an obstacle. When an obstacle is detected by the vision system, either steering or change direction behaviour is enabled depending on the size of the obstacle. For instance, change direction behaviour is initiated when any single object such as a wall or table spans the entire field of view. In this case the system perceives that the

average of flow clusters is similar and, the standard deviation and average of TTC values are below a certain threshold value.

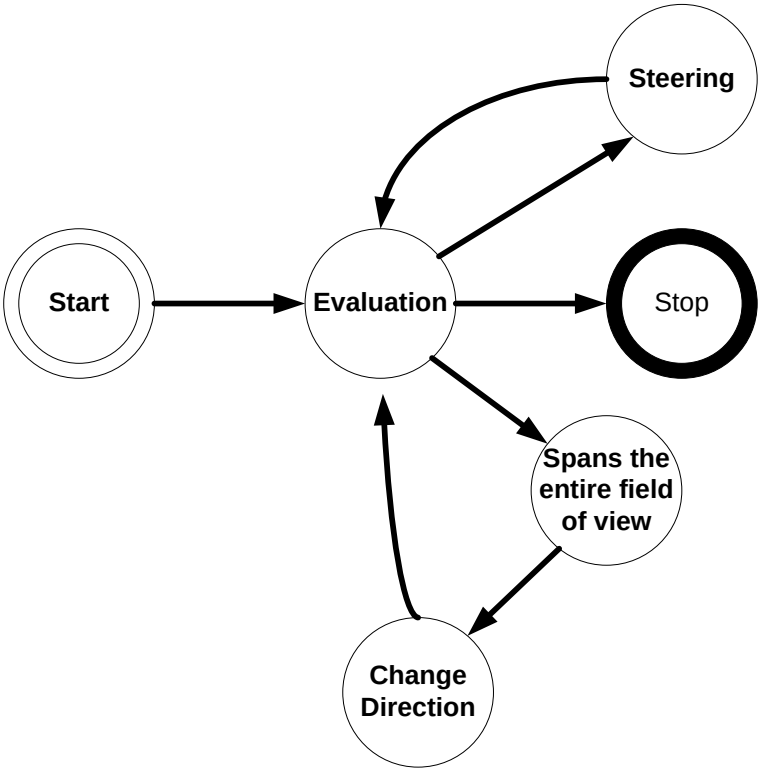


Figure 3.5: Flowchart of the optical flow based navigation algorithm

To avoid the obstacle, the robot first ceases forward motion and makes a constant turn (c_{turn}) which depends on the environment, followed by activation of the steering behaviour. Conversely, if the standard deviation of the TTC values are high, and the rate of side flows are not small, the robot performs a turn with respect to the control law, with the range varying from $\pm n°$, which gives a new heading angle as determined by the following expressions [Guzel and Bicker, 2010]:

$$\theta_{new} = -\left(\frac{\sum|w_L| - \sum|w_R|}{\sum|w_L| + \sum|w_R|} \times n\right) \qquad (3.17)$$

where $\sum|w_L|$ and $\sum|w_R|$ are the sums of the magnitudes of optical flow in the visual hemi-fields on both sides of the robot's body. Besides this, if there is no obstacle in its environment the robot is steered by this control law.

Stop behaviour is activated which essentially ceases the robot's motion, in the case of the mission being accomplished, which is evaluated as the robot navigates the environment over a certain time span whilst avoiding obstacles, or if the robot collides with any obstacle in the environment during the navigation task. Furthermore the robot can be stopped manually by the operator. Accordingly, this behaviour is enabled in case of success or failure in finishing the process.

3.2.3 Evaluation of flow vectors for mobile robot navigation

Two different scenarios are discussed in this section regarding the functionality of optical flow vectors in order to conduct vision based navigation. The first example, illustrated in Figures 3.6 and 3.7, shows the behaviour of the algorithm whenever an obstacle is detected along the path of the autonomous vehicle.

The second example presents a situation in which any single object spans the entire field of view, shown in Figure 3.8.

Figure 3.6: Scenario 1 (side obstacle), (a) source image, (b) destination image

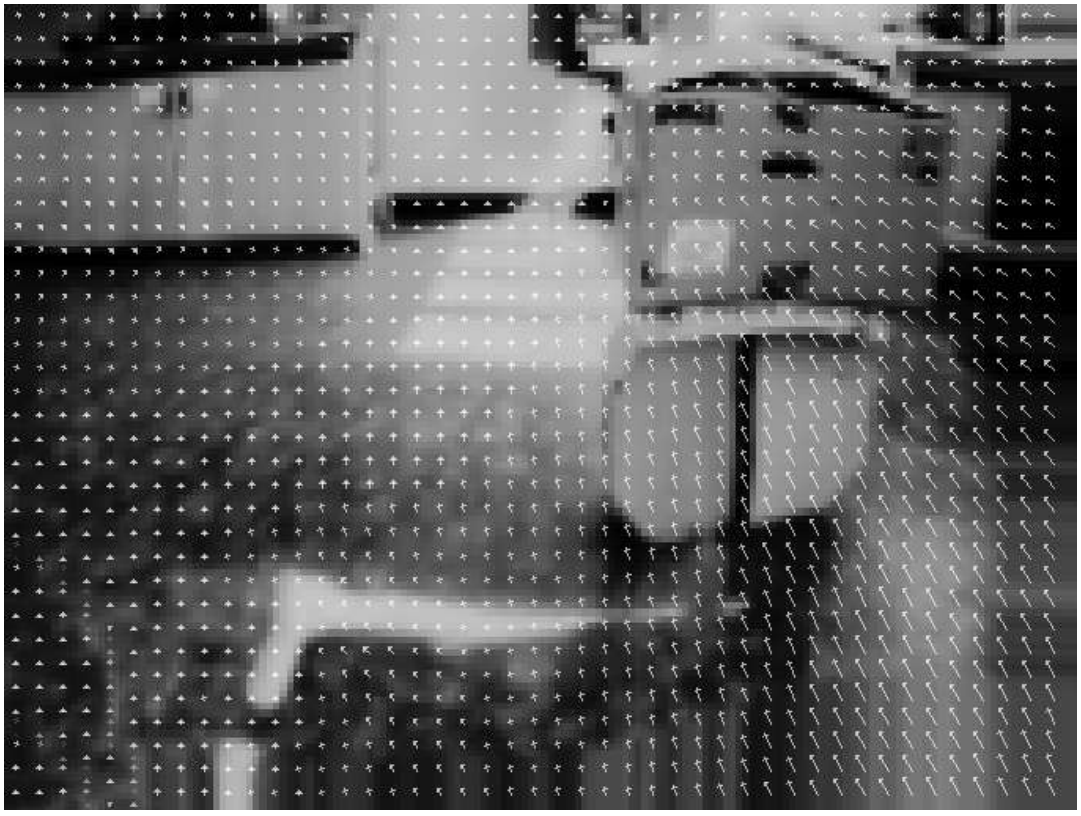
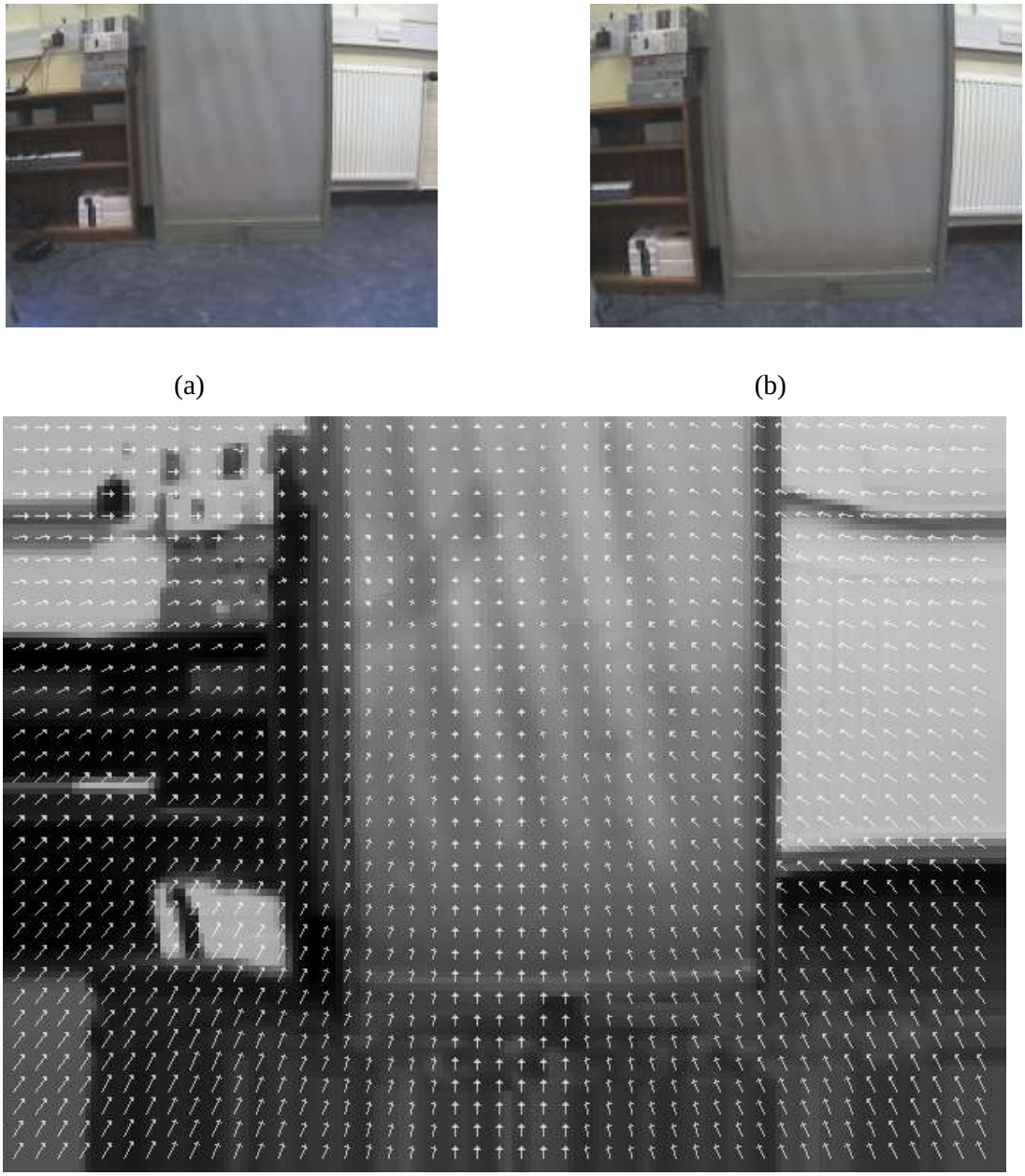


Figure 3.7: Scenario 1 (side obstacle); (standard deviation of TTC = 20.9), flow rate (left / right = 0.43), (Average of TTC parameters (left=63,center=11, right=17))

The first scenario displays the characteristics of the flow vectors in the event of any obstacle appearing in the robot's path as it manoeuvres, as illustrated in Figure 3.7. The magnitudes of the flow vectors on the right side are somewhat higher than the left side, and represents the possibility of colliding with an obstacle appearing on the right side of the robot. In addition, TTC values of right and centre clusters are low, which reveals the possibility of the presence of an obstacle at close range. However despite the low mean value of TTC parameters due to left cluster, standard deviation of TTC parameters is still high for change direction behaviour.

The results of the second scenario, shown in Figure 3.8 (c), present a situation when a single object spans the entire field of view. As the corresponding results shows, while the rate of side flow is close to 1, the TTC parameters are correspondingly rather low, presenting the high probability of colliding with a very large obstacle which cannot be overcome by standard turning manoeuvres. Therefore the *change direction* behaviour is enabled so as to avoid the obstacle.



(a) (b)

(c)

Figure 3.8: Objects spans over the entire field of view, (a) source image, (b) destination image, (c) flow vectors; (standard deviation of TTC = 5.6), flow rate (left / right = 1.12), (Average of TTC parameters (left=26,center=32, right=37))

3.3 Appearance Based Methods for Obstacle Avoidances

Appearance-based methods which identify locations on the basis of sensory similarities are a promising potential solution to mobile robot navigation. One of the main idea behind the strategy is to head the robot towards an obstacle-free position using similarities between the template and the active images [F. Vassallo et al., 2000]. This is called template matching, and is discussed in the next section. The similarity between the image patterns can be obtained by using feature detectors, involving corner based detectors, region based detectors and distribution based descriptors [Alper et al., 2006]. However, most of these techniques consume a lot of processing time which is not appropriate for real time systems. In order to handle the performance problem, algorithms are designed based on the appearance of individual pixels. The classification of obstacles is carried out by using differences between pixels in the template and active image patterns and any pixel that differs in appearance from the ground is classified as an obstacle. The method requires three assumptions that are reasonable for a variety of indoor and outdoor environments, which are:

- Obstacles must be different in appearance from the ground.
- The ground must be flat.
- There must be no overhanging obstacles.

The first assumption distinguishes obstacles from the ground, while the second and third assumptions are required to estimate the distances between detected obstacles and the robot. There are several models for representing colour. The main model is the RGB (Red, Green, Blue) schema which is used in most image file formats; however colour information in this model is very noisy at low intensity. The RGB format is frequently converted to HSV (hue, saturation, and value) or HIS (hue, intensity, saturation). Hue is what humans perceive as colour, Saturation is determined by a combination of light intensity and the extent to which it is distributed across the spectrum of different wavelengths and value is related to brightness. In HIS, I is an intensity value with a range from 0 to 1 where 0 represents black and white 1. These colour spaces are

assumed to be less sensitive to noise and lighting conditions. The flow chart for the appearance-based obstacle detection systems is illustrated in Figure 3.9.

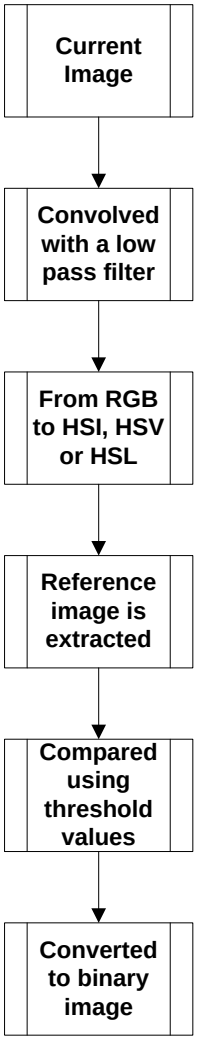


Figure 3.9: Flow chart of the appearance-based obstacle detection algorithm

The input image is first convolved with a smoothing filter in order to reduce noise effects, and then the smoothed image is converted to HIS, HSV or any relevant colour space with respect to the developed algorithm [Ulrich and Nourbakhsh, 2000; Fazl-Ersi and Tsotsos, 2009]. A reference area is obtained from this image which might be any defined geometric shape such as trapezoids, triangles or squares [Saitoh et al., 2009]. Finally, a comparison between the reference image and the current image is made by using some predefined threshold values. One comprehensive technique which has been proposed [Ulrich and Nourbakhsh, 2000] employs image histograms to compare the

reference area and the current image. For example, assume that the bin value, Hist(H(x, y)), of the generated histogram and the threshold value, T_H are compared, where H(x, y) is the H value at pixel (x, y). If Hist(H(x, y)) > T_H, then the pixel P(x, y) is classified into the safe region; otherwise it is classified as in the obstacle region. In order to simply the problem, the results are represented in a binary image in which the safe path is represented as white but obstacles are represented with black, as illustrated in Figure 3.10.

Figure 3.10: Appearance-based obstacle detection method

Figure 3.11: Effects of lighting conditions and unexpected stains on the floor

However, identifying regions purely on the basis of sensory similarity is too simplistic; different places may look very similar, even with a rich sensing methodology, due to lighting conditions, shadows from illumination, and so on. Furthermore, for dynamic

environments there might be unexpected stains on the ground which may be determined to be an obstacle and may lead the robot to an unsafe path. An example of how a stain on the floor can affect the output of the segmentation is illustrated in Figure 3.11.

3.3.1 Template matching

Template matching is a simple and popular technique in computer vision and image processing, where small parts of an image which match a template image are identified. It can be used in mobile robot navigation or as a way to detect edges or objects in images. A basic method of template matching uses a convolution mask which can be easily performed with grey images. The convolution output will the highest at places where the image structure matches the mask structure, such that large image values get multiplied by large mask values. This method is normally implemented by first picking out a part of the search image to use as a template. For instance, the input and output images are called I(x, y) and O(x, y) respectively, where (x, y) represent the coordinates of each pixel in the images and the template is called $T(x_t, y_t)$, where (x_t, y_t) represent the coordinates of each pixel in the template. The technique simply moves the centre of the template $T(x_t, y_t)$ over each (x, y) point in the search image and calculates the sum of products between the coefficients in I(x, y) and $T(x_t, y_t)$ over the whole area spanned by the template. As all possible positions of the template with respect to the input image are considered, the position with the highest score is the best position, and this is represented in the output image. Several techniques can be used to handle the translation problem; including the SSD (sum of squared differences), NCC (normalized cross correlation) and SAD (sum of absolute differences) [Wen-Chia and Chin-Hsing, 2009]. NCC basically measures the similarity of two variables and is defined as follows:

$$NCC = \frac{\sum_{i=0}^{N-1}(x_i - \dot{x}) \times (y_i - \dot{y})}{\sqrt{\sum_{i=0}^{N-1}(x_i - \dot{x})^2 \times \sum_{i=0}^{N-1}(y_i - \dot{y})^2}} \qquad (3.18)$$

where N is the template image size, and $\dot{x}$ and $\dot{y}$ represent average gray levels in the template and source image respectively.

The goal is to find the corresponding (correlated) pixel within a certain range disparity that minimizes the associated error and maximizes the similarity. This matching process involves computation of the similarity measure for each disparity value, followed by an aggregation and optimization step [Zitovai and Flusser, 2003]. An example related to the correlation based technique is illustrated in Figure 3.12.

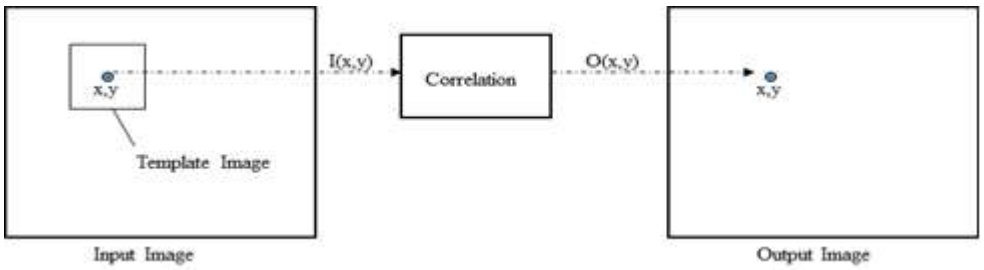


Fig. 3.12: Correlation-based template matching

Root of SSD is an alternative and robust template matching method widely used in image registration algorithms, and can be defines as follows:

$$\text{RSSD}=\sqrt{\sum_{i=0}^{N-1}(x_i - y_i)^2} \qquad (3.19)$$

where *N* is the template image size; *x* and *y* represent the corresponding pixel values in the template and the active image respectively.

3.3.2 Implementation of obstacle avoidance technique using appearance based approach

One of the aims of this study is to design and implement a purely reactive obstacle avoidance system based on qualitative information, which must be compatible with optical flow based navigation systems so as to construct the proposed hybrid obstacle avoidance architecture. Ulrich's method [Ulrich and Nourbakhsh, 2000] is a simple but reasonably efficient appearance-based obstacle region detection method, which has been recently modified to navigate a wheelchair based mobile robot [Saitoh et al., 2009].

However, this method is not purely reactive and its histogram based comparison is more vulnerable to lighting conditions than correlation based techniques. Therefore, in the research, Ulrich's method is modified somewhat in order to be able to run it with an optical flow based control architecture.

According to the algorithm given in Figure 3.9, the acquired image is first smoothed by a Gaussian filter to eliminate noise in the image [Ulrich and Nourbakhsh, 2000; Saitoh et al., 2009]. Then a copy of the original image is transformed from RGB space to HSV space. In order to provide the consistency between this method and the optical flow based method, the implementation has been carried out based on the principle of balance strategy used in optical flow based architecture, where the image is divided into *n* clusters. However, only the lower half of the image is considered in order to reduce processing time, and the upper part of the image is discarded. For instance, 4, 8 or 16 clusters can be obtained from a 640x480 resolution image which each cluster consists of 160x240, 80x240 or 40x240 pixels respectively. The same classification method can be applied to lower or higher resolution images. In this study for both 320x240 and 176x144 resolutions, *n* is set to 8 based on trial and error method.

The next step is to provide a reference area which is always required to be free of obstacle. Therefore, to minimize the risk of violating this constraint, the reference area cannot be deep. Accordingly, a reference area which is the same size as one of the vertical regions is provided. This reference image illustrates a free path and during each processing cycle of the main algorithm, it is compared with clusters extracted from the active image using the template matching technique based on the H, S value ranges respectively. Hue and saturation bands are less sensitive to changes in illumination than the value band [Ulrich and Nourbakhsh, 2000]. From results of the comparison between the reference image and the active images, corresponding clusters are allocated as either free or occupied. By way of illustration, the algorithm is applied to an appropriate scenario, shown in Fig 3.13. The results of clustering and template matching are illustrated in Figure 3.14, where (f) represents a free path and (o) a blocked (occupied) path. The correlation results with respect to the each cluster are illustrated in Figure 3.15. By way of comparison, SSD is applied for the Hue component and NCC is applied

to the Saturation component. The latter involves higher computational complexity compared to SSD since it requires numerous multiplication, division and square root operations.

Figure 3.13: An example image for the proposed appearance-based method

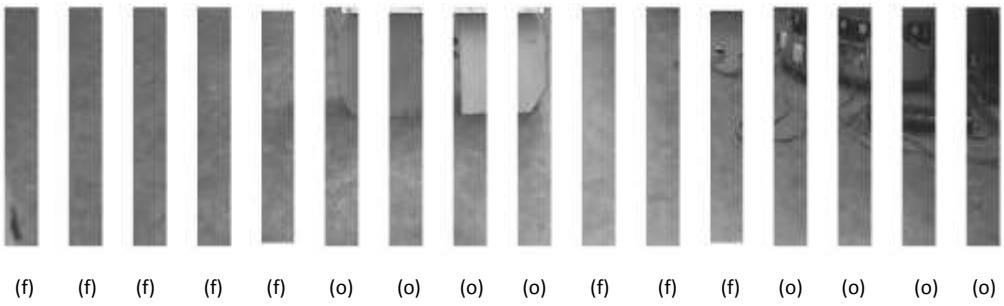


Figure 3:14: Allocation of image segments given example scenario

Finally, in order to construct the segmentation of the current image, correlation results are compared with previously defined threshold values. However, due to the structure of

appearance-based algorithms, lighting conditions and noise are very important. Thus, for each test environment, appropriate threshold values must be recalculated.

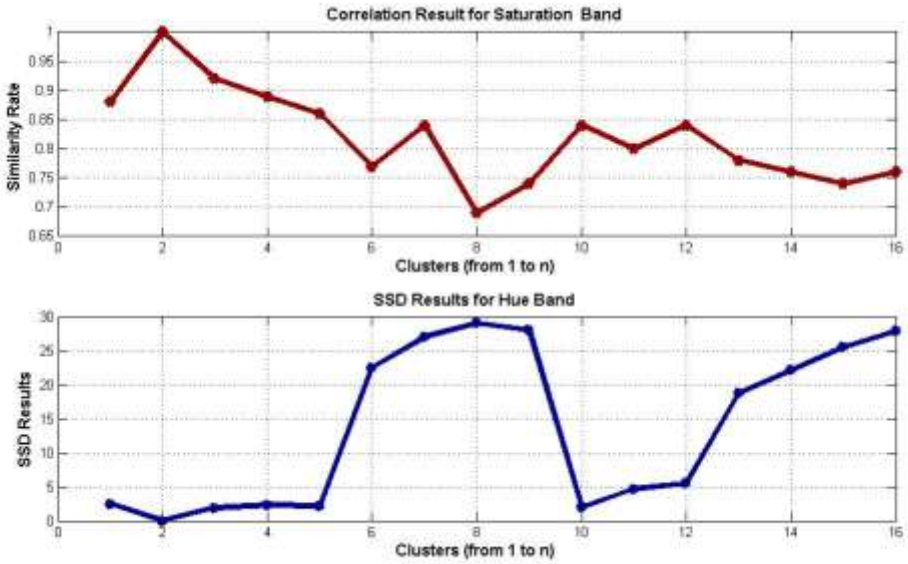


Figure 3.15: Correlation results for each cluster

Using the results obtained from this example, appropriate threshold values for this environment can be assigned as 0.8 for S and 10 for H components. To provide more reliable results, both components must be considered in the map construction operation.

The results indicate that both correlation based similarity measuring techniques produce comparable outcomes. Overall system performance enhancement is one of the primary motivations in real time applications. Therefore, for the real experiments, RSSD is applied for both components which is somewhat faster than using NCC.

3.4 Integration of Appearance Based Method with Optical Flow Architecture

Optical flow based methods suffer from two major problems. The first and most important of these is illumination, which is markedly affected by variations in lighting and shadows [Contreras, 2007]. Another major issue is sensitivity to noise and distortion. Various integrated methods for solving these problems have been proposed;

nevertheless it is still a key challenge in employing optical flow methodologies for mobile robot navigation. Appearance-based methods have significant processing and performance advantages which make them a good alternative for vision based obstacle avoidance. Nevertheless, these techniques still suffer from illumination problems and are highly sensitive to floor imperfections, as well as to the physical structure of the terrain**.**

To overcome these drawbacks, an alternative method has been proposed which essentially relies on a fusion of both techniques, as illustrated in Figure 3.16. The main strategy behind this proposal is to integrate the results obtained from an appearance-based method into the proposed optical flow based architecture. In order to achieve this integration, flow equations are updated with respect to an estimated binary image. However, the binary image illustrated with Boolean logic (F/O) needs to be converted into logical expressions in order to be reasoned over. The method used in this study obtains the extreme values (the highest and lowest average magnitude values) from flow clusters. These are subsequently replaced with Boolean values for the binary image in which the highest value is replaced with 'O' members and the lowest value is replaced with 'F' members. The following algorithm illustrates how the estimated Boolean values from the appearance-based method are converted into flow values.

Conversion Algorithm :

Calculate maximum flow F_{max} *and minimum flow* F_{min} *from the current image*
Until the conversion is completed
  If the current segment is free
    Replace F *with* F_{min}
  else
    Replace O *with* F_{max}
end_until

The conversion procedure for each segment can be formalized as follows:

$$c_i = \begin{cases} Fmax, & s_i = O \\ Fmin, & s_i = F \end{cases} \qquad (3.20)$$

where s_i represents the i_{th} segment extracted from the corresponding binary image and c_i is its updated equivalent.

Eq. 3.21 is used to calculate the new heading angle, including the corresponding member of the map. The new heading angle and the updated version of the control equation (θ_{unew}) can be expressed as follows:

$$\theta_{unew} = -\left(\frac{\sum|w_{uL}| - \sum|w_{uR}|}{\sum|w_{uL}| + \sum|w_{uR}|} \times n\right) \quad (3.21)$$

where $\sum|w_{uL}|$ and $\sum|w_{uR}|$ are the sums of the magnitudes of optical flow and converted map regions with respect to the extreme flow values in the visual hemi-fields on both sides of the robot's body. These can be detailed as follows:

$$w_{uL} = \sum_{i=1}^{n}[w_L(i) + w_{AL}(i)]$$
$$w_{uR} = \sum_{i=n+1}^{2n}[w_R(i) + w_{AR}(i)] \quad (3.22)$$

where w_L and w_R represent the average magnitudes of flow vectors in the left and right clusters respectively, whereas w_{AL} and w_{AR} represent the converted segments from the binary image (*n* is the number of clusters and is set to 4).

The flowchart of the overall control architecture is illustrated in Figure 3.16, the Image Sequence is used by the Optical Flow Module to calculate flow vectors and corresponding parameters such as Focus of Expansion (FOE) and Time to Contact (TTC). Simultaneously, the last obtained image is correlated with a template in order to estimate the free (F) and occupied (O) parts of the current image based on the appearance-based obstacle detection method. The Conversion Module converts the output of the appearance-based obstacle detection output into flow based values. The Control Law is generated based on the inputs provided by both the optical flow and conversion modules (see Eq. 3.22). Finally, the Behaviour Module selects the appropriate behavior based on its arbitration mechanism to steer the robot towards a free space.

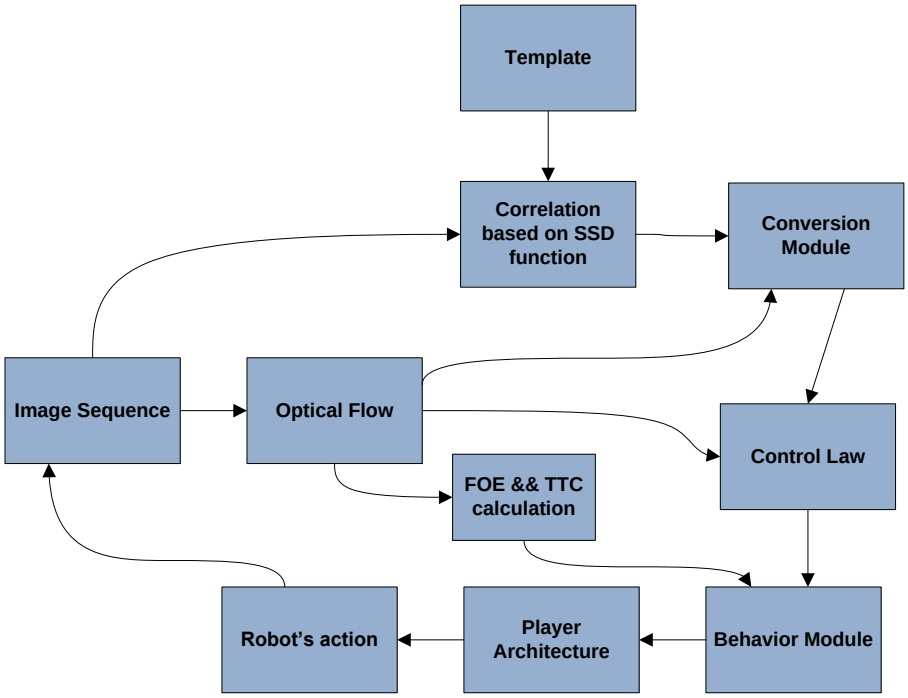
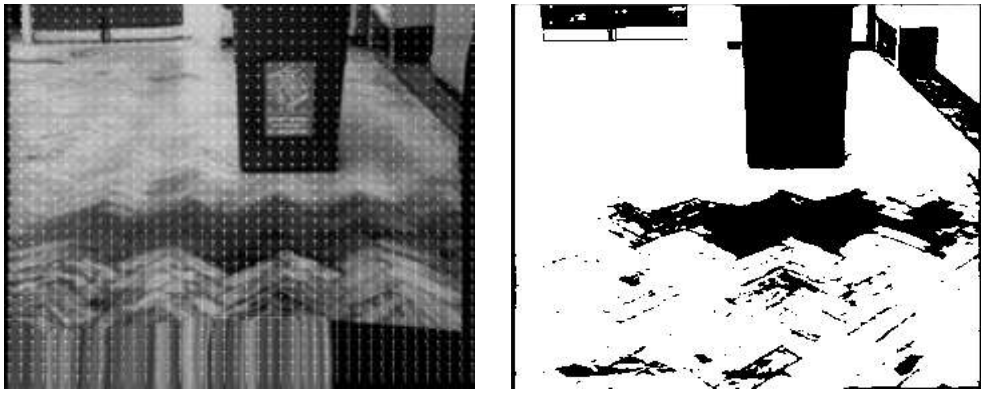


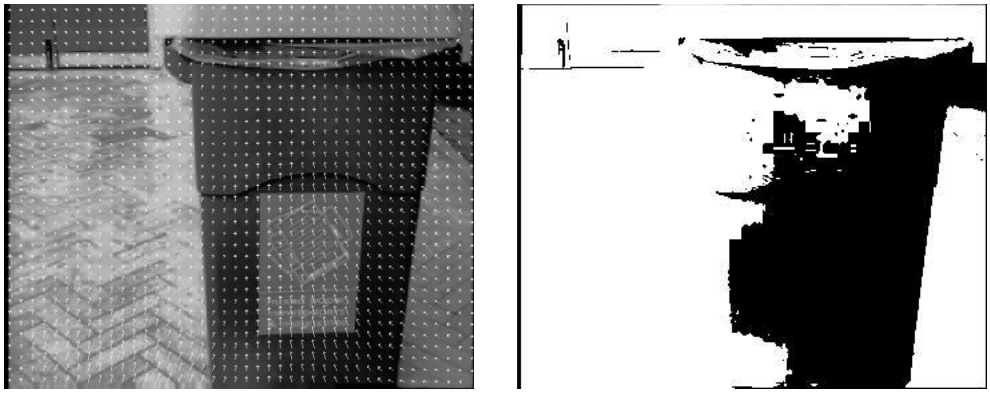
Figure 3.16: Flowchart of the proposed hybrid architecture

Figures 3.17-3.20 presents the output of both detection algorithms for different frames captured from a navigation scenario. The control parameters of each frame are included in Table 3.1, and the higher resolutions of flow vectors are included in Appendix A.



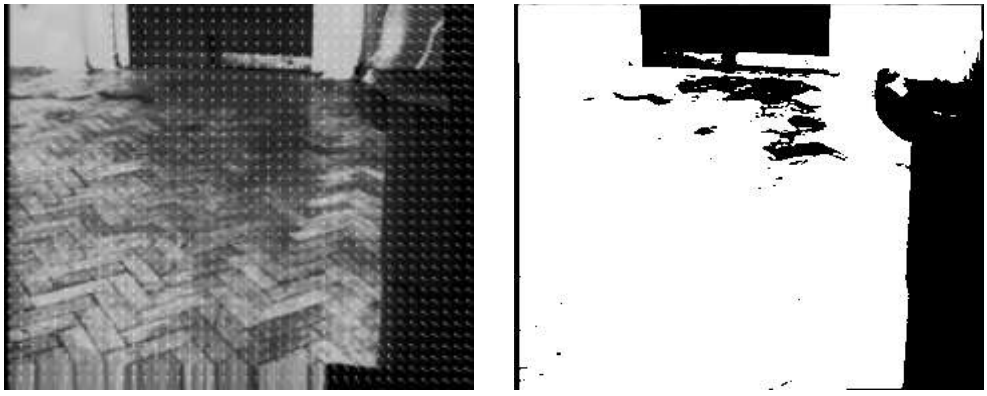
(a) **(b)**

Figure 3.17: Frame 1, (a) flow vectors, (b) binary output from appearance-based method



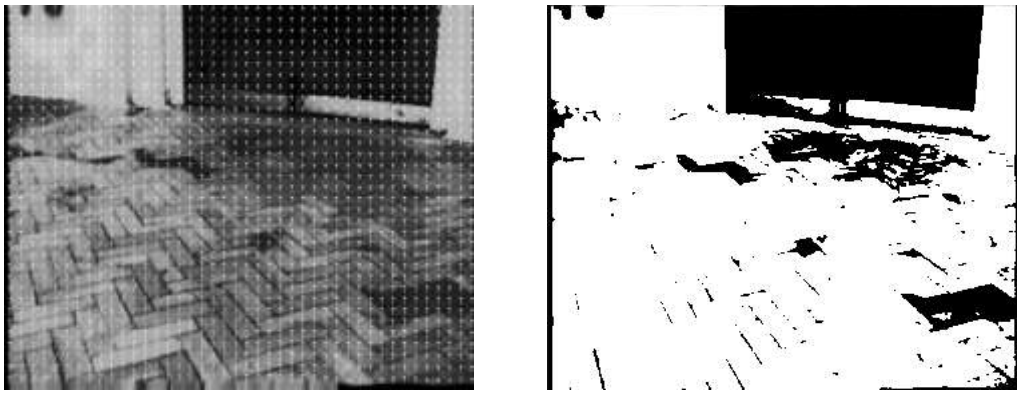
(a) **(b)**

Figure 3.18: Frame 32, (a) flow vectors, (b) binary output from appearance-based method



(a) **(b)**

Figure 3.19: Frame 53, (a) flow vectors, (b) binary output from appearance-based method



(a) **(b)**

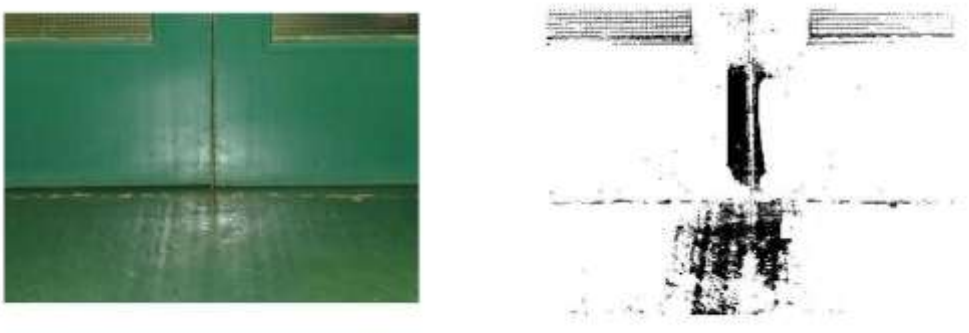
Figure 3.20: Frame 97, (a) flow vectors, (b) binary output from appearance-based method

Table 3.1: Estimated steering angles for experiments

Left (+) / Right (-)	Optical Flow	Hybrid Architecture
Frames	*w(deg/sec)*	*w(deg/sec)*
1	0.63	0.82
32	2.9	5.4
53	6	7.3
97	-0.93	-0.77

The hybrid technique has the ability to negotiate and avoid walls and doors by benefiting from the results of the optical flow based navigation technique using the frontal optic flow to estimate the so-called time-to-contact before a frontal collision is likely to occur. Furthermore it possesses the ability to avoid lateral obstacles in both a safer and smoother manner than with the conventional optical flow technique.

Figure 3.18 presents such a scenario where the robot, using the optical flow based method, is not able to avoid the obstacle, because the system does not generate an appropriate steering angle. The major difficulty with the optical flow method in mobile robot navigation is that, despite the assumption of constant illumination; lighting conditions may significantly change due to environmental factors which optical flow techniques are known to have difficulty in handling which may thus cause miscalculations. However the hybrid system integrates the results of the appearance-based method into the control law which enforces the overall control strategy. For this scenario, the hybrid system generates a sharper avoiding manoeuvre which allows the robot to pass the obstacle without colliding with it. Figure 3.19 presents another scenario in which the hybrid method generates a safer avoidance manoeuvre when compared with the conventional optical flow method. This is because the hybrid architecture involves merging the optical flow method with the appearance-based method and this results in a better response to the lateral obstacle. Figures 3.17 and 3.20 present the scenarios where the environments are partly open and safe. The results reveal that the control parameters generated by both methods for corresponding scenarios are similar (see Table 3.1).



(a) (b)

Figure 3.21: Pattern similarity, (a) orjinal image, (b) binary Image

Despite their success with lateral obstacles, conventional appearance-based obstacle detection methods tend to fail to detect objects such as walls and doors that span the entire field of view. This is because the appearance-based methods perform a classification of obstacles using differences between pixels in the template and the active image patterns, where any pixel that differs in appearance from the ground is classified as an obstacle. Additionally, region segmentation has some other drawbacks, one of which is that the thresholding technique requires a significant contrast between the background and foreground in order to be successful. This technique essentially works well for the environments which consist of one dominant colour. Accordingly, if the colour of the doors or walls is similar to the floor pattern, the algorithm may easily fail to complete the navigation task. Figure 3.21 illustrates an example where the appearance-based method is not able to distinguish between the door and the floor in a precise manner due to the similarity in colours of their patterns.

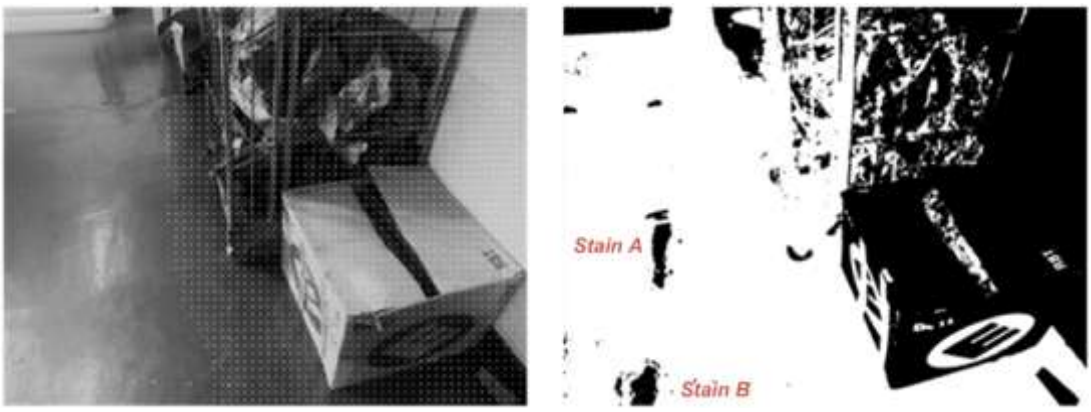


Figure 3.22: First large obstacle, (a) original image, (b) binary Image

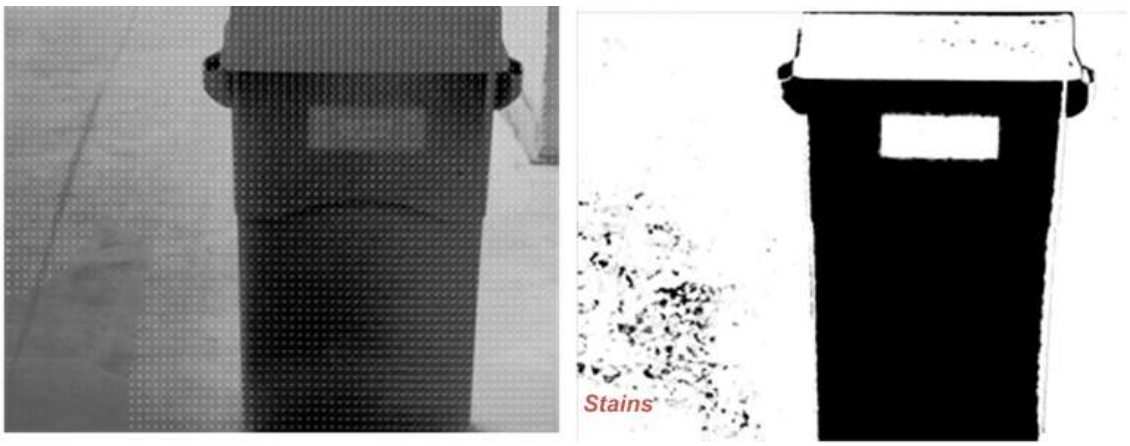


Figure 3.23: Second large obstacle, (a) original image, (b) binary Image

Two additional examples are illustrated in Figures 3.22 and 3.23, where the path of the robot is obscured by large obstacles. The results shown in Figure 3.22 indicate that both techniques can detect the obstacles. However, due to the lighting conditions, the second technique fails in the segmentation of some parts of the image where reflections are present on the white floor, as shown in Figure 3.22 (b). On the other hand, the first technique estimates the obstacles by succesfully using the magnitudes of flow vectors. Figure 3.23 demonstrates another scenario in which the obstacle is rather close to the goal. Here, the second method is more useful than the conventional optical flow based technique, despite the extracted stains as shown in Figure 3.23 (b). This is because the appearance-based methods are based on pixel differences which can provide image segmentation independent of distance to the goal.

As has been discussed above, the first method focuses on the practical use of optical flow and visual motion information in performing obstacle avoidance task in real indoor environments. However, when the obstacle becomes very close to the robot, the gradients usually cannot be calculated accurately which may result in the incomplete calculation or allocation of flow vectors. To evaluate the performance of the proposed navigation method, a series of simulation experiments are discussed in the following section.

3.5 Modelling and Simulation using Microsoft Robotics Studio

In order to estimate the capability of the proposed work for expressing useful tasks, the system has been evaluated using a simulator based on the Microsoft Robotics Studio (MRS) environment. The MRS provides a 3D programming and modelling simulator called Visual Simulation Environment (VSE), illustrated in Figure 3.24. This simulation environment is used to assess the performance of the proposed methodologies, as discussed in subsequent chapters.

The VSE includes a comprehensive graphical simulator which is also supported by a powerful physics simulator. Physics simulation is performed using a physics engine developed outside of Microsoft [Jackson, 2007]. Entities in the simulated world specify a physical description complete with friction coefficients, mass, and centre of gravity. Simulated contact takes into account force, torque, momentum, and resistance when updating the positions of each frame. VSE services are authored using any .NET compatible language. Both the graphics and physics simulation can be performed through software on Windows XP and higher. Additionally, graphics cards supporting DirectX 9 will accelerate rendering. The physics engine also allows hardware acceleration through the optional integration of the physics processing unit. Further details can be found in Microsoft [Jackson, 2007].

A series of simulations has been conducted to verify that the robot is able to navigate in its working environment and achieve its goal without collisions. A simulated *Corobot* mobile robot, which is able to moves on 4 wheels was used in these experiments, as shown in Figure 3.25. The technical details of the *Corobot* mobile robot are provided in Appendix B. For the experiments, the robot is equipped with an internet camera having 320x240 resolution and a 2D laser range finder which a 180-degree field of view, and 360 units of angular resolutions (2 units/degree). However, the internal camera employed by the robot in these simulations is a single monocular camera without a panning feature, and the robot has to rotate to the given position instead of panning the camera when the search behaviour is activated (see chapters 4 and 5). Fortunately, the simulation based experiments provide a more flexible yet still ideal platform to test the

algorithms. An interval of ±90° is employed in conducting simulation experiments reported in the following chapters.

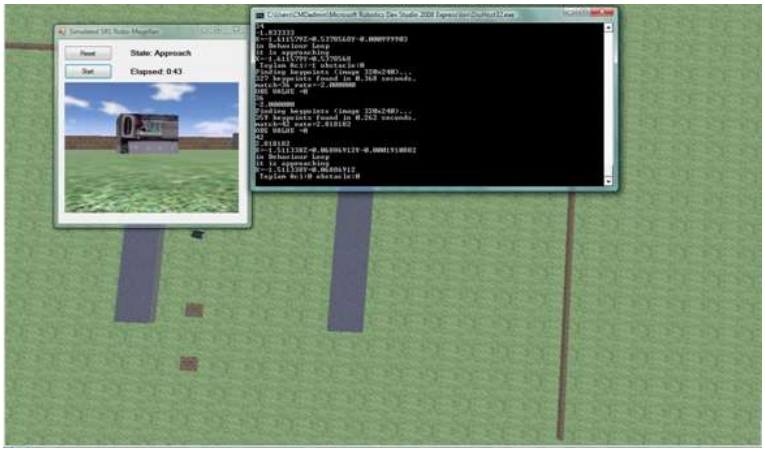


Figure 3.24: Simulation screenshot from the VSE

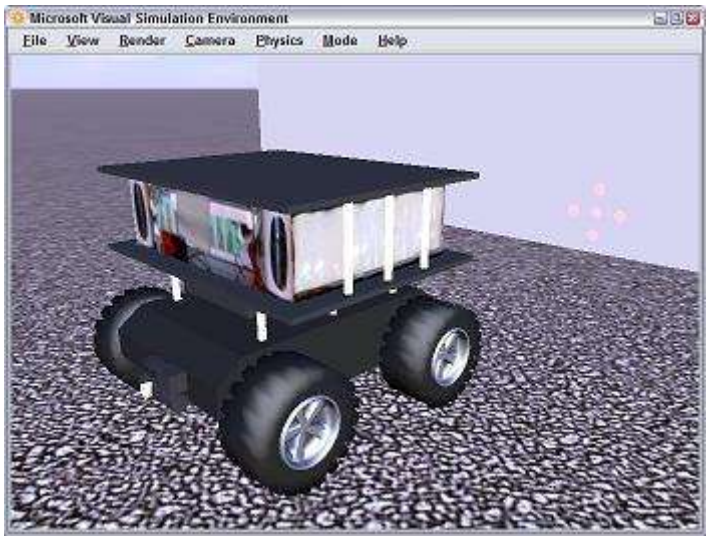


Figure 3.25: Four wheel simulated Corobot mobile robot

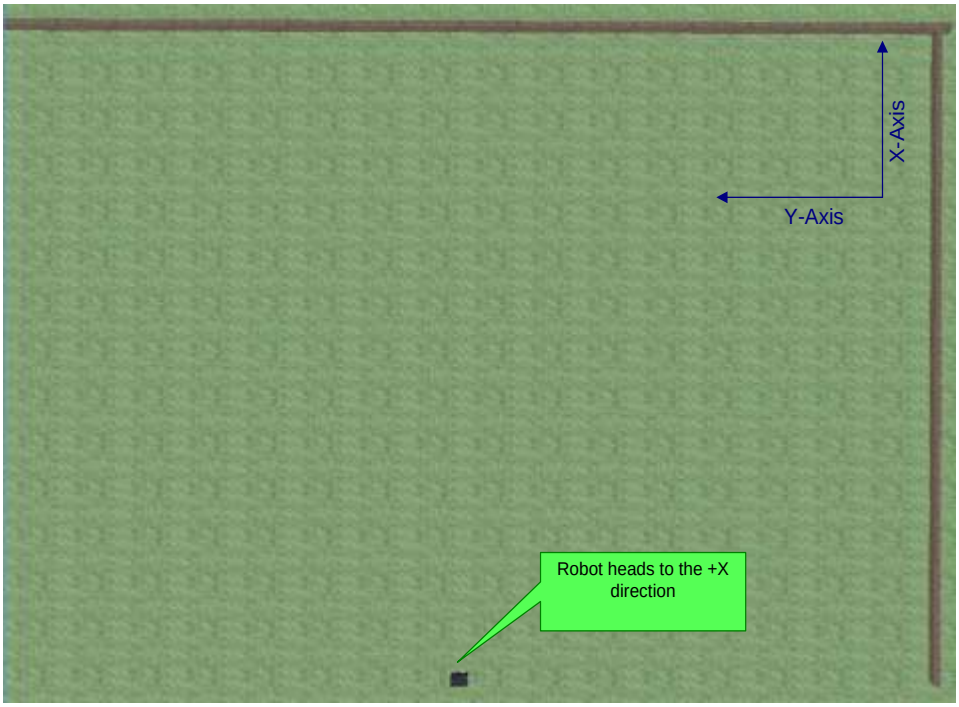


Figure 3.26: Simulation environment for experiments

In this section, simulations are conducted to assess the performance of the proposed vision based obstacle avoidance technique. Three different scenarios have been developed to achieve this purpose, each of which is conceived as having an increased level of complexity. The robot must navigate in the simulated environment whilst avoiding obstacles until the mission is accomplished or terminated. The scenarios are defined as follows:

Scenario 1: This is the preliminary scenario in which the mobile robot is required to navigate from its start point (-5, 1) in an organized and uncluttered environment. The only obstacle is a wall surrounding the working environment, as shown in Figure 3.27.

Scenario 2: The mobile robot is required to navigate from start point (-5,1) while two unexpected obstacles are placed along its path, as shown in Figure 3.28.

Scenario 3: This is a complex scenario in which the mobile robot navigates from start point (-5,1) in a partly cluttered environment whilst several unexpected obstacles are placed along its path, as shown in Figure 3.29.

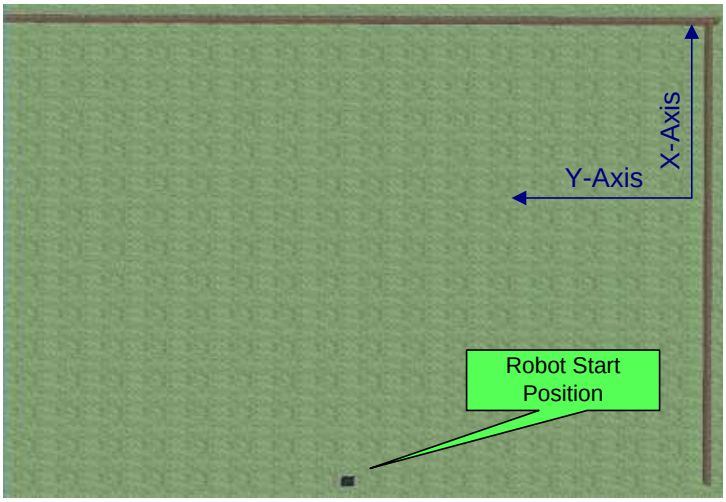


Figure 3.27: Scenario 1, the robot moves from the start position

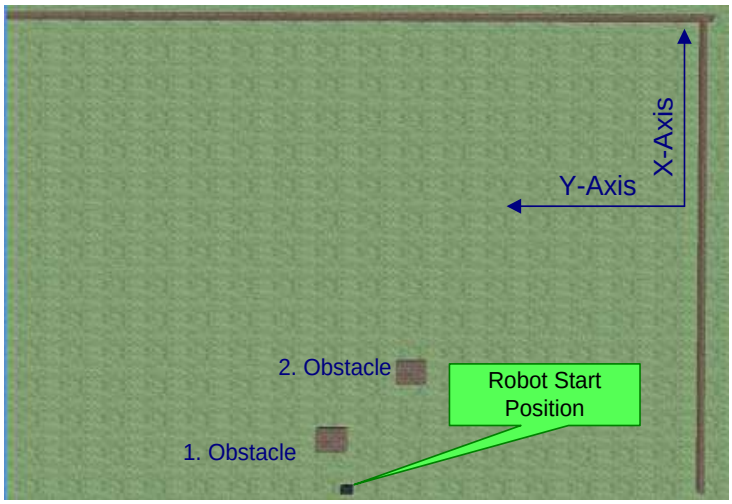


Figure 3.28: Scenario 2, the robot moves from the start position

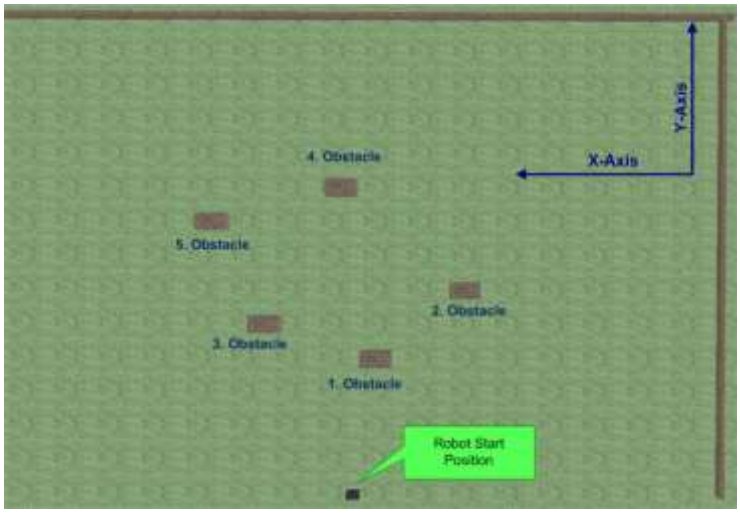
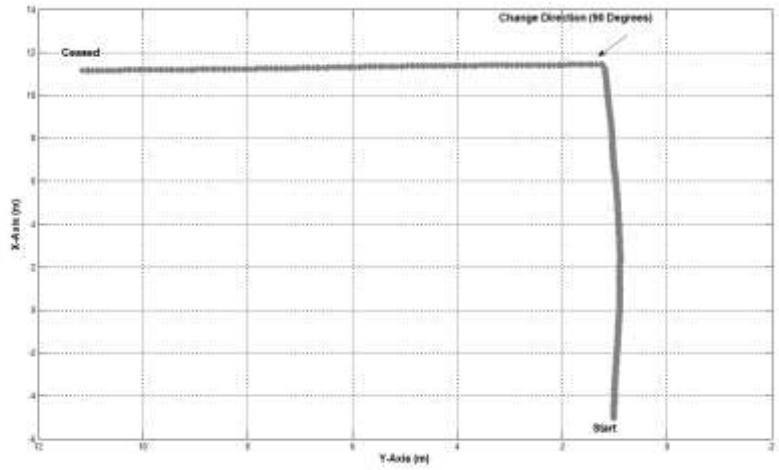


Figure 3.29: Scenario 3, the robot moves from the start position

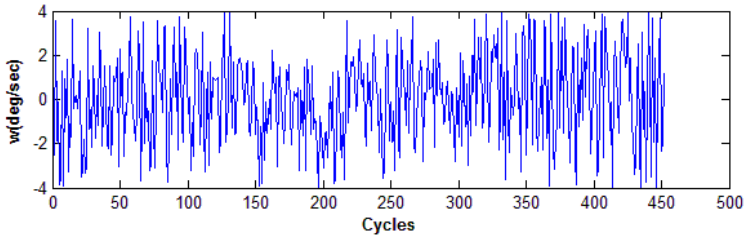
Table 3.2: Parameters for experiments

Parameters	***Descriptions***
Start Position (x,y)	Docking Area (-5,1)
Initial heading angle	$\theta = 0^o$
Constant speed	$v_c = 0.3$ m/s
Constant turn value	$c_{turn} = 90^o$
Maximum range for turning	$n = \pm 20°$

To carry out the simulation, several parameters are set, namely the robot's start position and its initial heading angle, constant turning value *(c_{turn})* used in the change direction behaviour, constant forward speed of the robot *(v_c)* and maximum turning angle range *(n)*, used to generate the new heading for each processing cycle. The simulated robot has a differential drive configuration controlled by a combination of linear and angular velocity. As previously mentioned, the linear velocity is constant *(v_c)* and angular velocity is obtained from the expressions for the heading angle *(w)* as given by Eq. 3.21. Table 3.2 presents the initial values of the parameters used to conduct the simulation based experiments. The simulated robot was stopped manually in each scenario, and the estimated trajectory is displayed on a 2D graph showing the characteristics of the algorithm. In addition, the corresponding value of *w* (deg/sec) for each processing cycle is displayed on a separate graph. As Table 3.1 shows, maximum turning angle is small in order to give a smoother trajectory, and this limits positional errors which may be caused by extreme manoeuvres that may lead the robot to depart from its original trajectory. The simulated obstacles used in Scenarios 2 and 3 are boxes having dimensions of 1000 mm x 750 mm.



(a)



(b)

Figure 3.30: Scenario 1, (a) robot's trajectory, (b) control parameters (between 267th and 268th cycles for change direction)

Figure 3.30 (a) presents the navigation results from the first scenario. This is the preliminary scenario for the robot that evaluates the characteristics of the algorithm and the performance of the behavioural based architecture. The robot begins its task by directly moving forward until it encounters the wall. During this forward movement, a series of small manoeuvres which do not affect the robot's direction is generated by the proposed control strategy until the robot encounters the wall. Once the wall is perceived, spanning the entire field of view, the robot turns 90^{o} to the left (c_{turn}, see Table 3.1) in order to avoid it. Subsequently, the robot keeps moving forward without losing directional stability until it stops. Figure 3.30 (b) gives the corresponding control parameters for this scenario.

Additionally, Scenario 1 was carried out using the conventional optical flow based control architecture to provide a comparison. The estimated trajectory and the

corresponding control parameters for this method are illustrated in Figures 3.31 (a) and 3.31 (b) respectively.

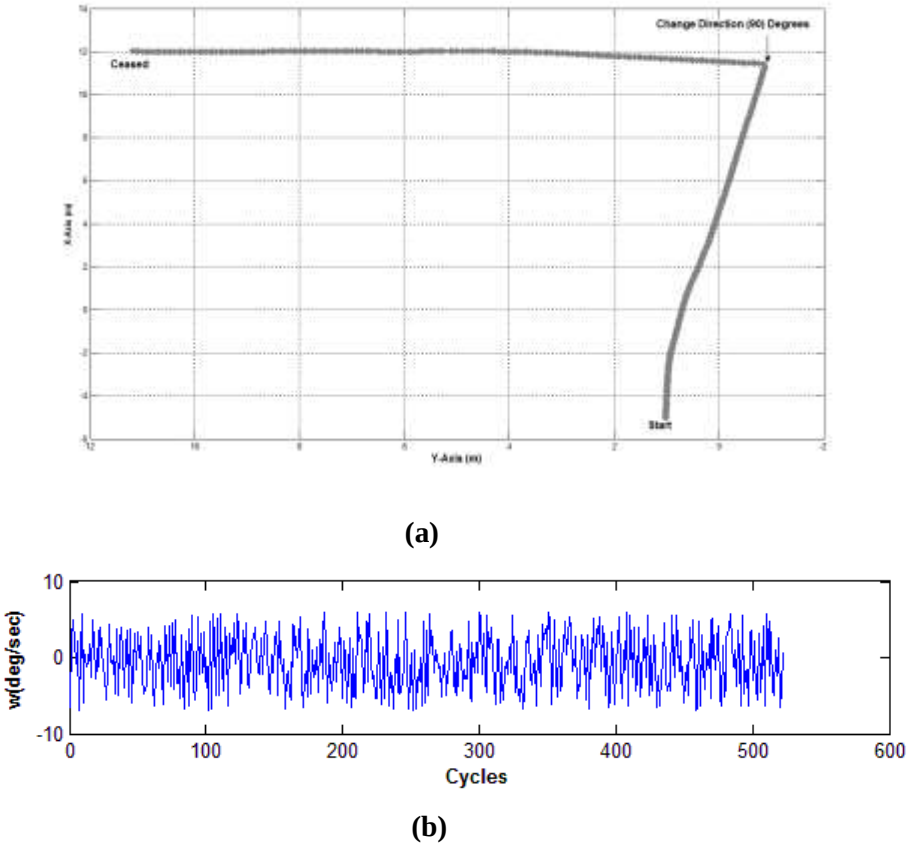


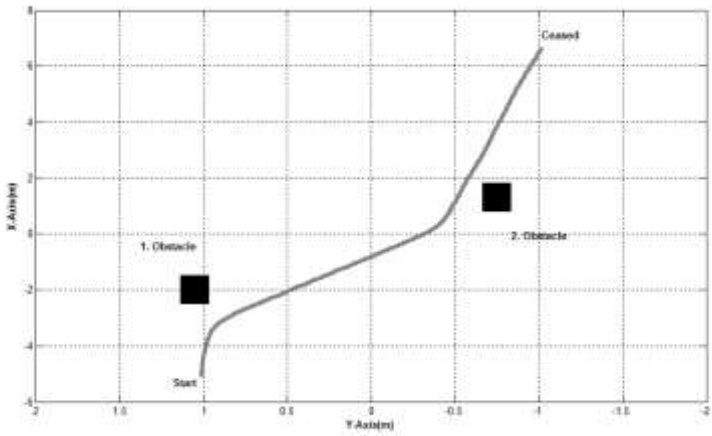
Figure 3.31: Scenario 1 with conventional method, (a) robot's trajectory , (b) control parameters (between 300th and 301th cycles for change direction)

The corridor following based comparison is a popular performance evaluation technique and is widely used to analyze of vision based control architectures in which the robot is required to move along a path without changing its heading direction and position error is calculated based on the deviation between the start and final positions [Mikolajczyk and Schmid, 2005].

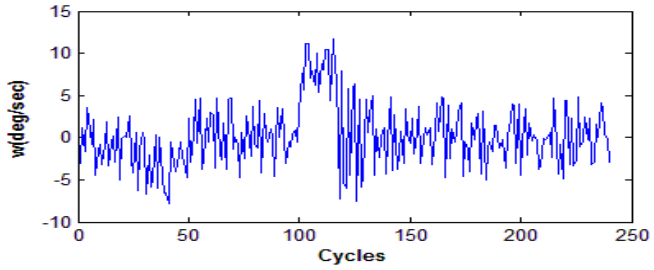
Position errors are given in Table 3.3, which in essence reveal that the hybrid architecture is able to conduct a corridor centring task somewhat better than the optical flow based architecture and in doing so yields a smaller error. Figure 3.32 presents the navigation results from the second scenario in which two obstacles are positioned close to the robot in the test environment. This test is intended to reveal how the robot interacts with obstacles and what obstacle avoidance strategy is used.

Table 3.3: Centring error results for scenario 1

Methods	Total Position Error (m)
Optical Flow	2.54 (m)
Hybrid	0.63 (m)



(a)



(b)

Figure 3.32: Scenario 2, (a) robot's trajectory, (b) control parameters

Conventional optical flow based navigation algorithms tend to fail when the distance between the robot and the obstacle is too close at the start. Accordingly, there could be two possible trajectories using optical flow based architectures. Either the robot may collide with an obstacle or diverge from the desired trajectory. An example of such a case is illustrated in Figure 3.33, in which the robot collides with the second obstacle and is therefore unable to complete the task. In contrast, the proposed hybrid architecture overcomes this problem. This is because the integration of appearance-based results to the conventional control architecture evidently enhances the overall performance of the system. As can be seen from the corresponding Figure 3.32 (a), once

the robot starts moving, the first obstacle is immediately detected and it is successfully avoided by smooth consecutive manoeuvres. After this the robot continues moving until the second obstacle is perceived and it then performs another turning manoeuvre to avoid the second obstacle. This ensures a safe path and the robot keeps moving until it stops. The control parameters of this scenario are illustrated in Figure 3.32 (b).

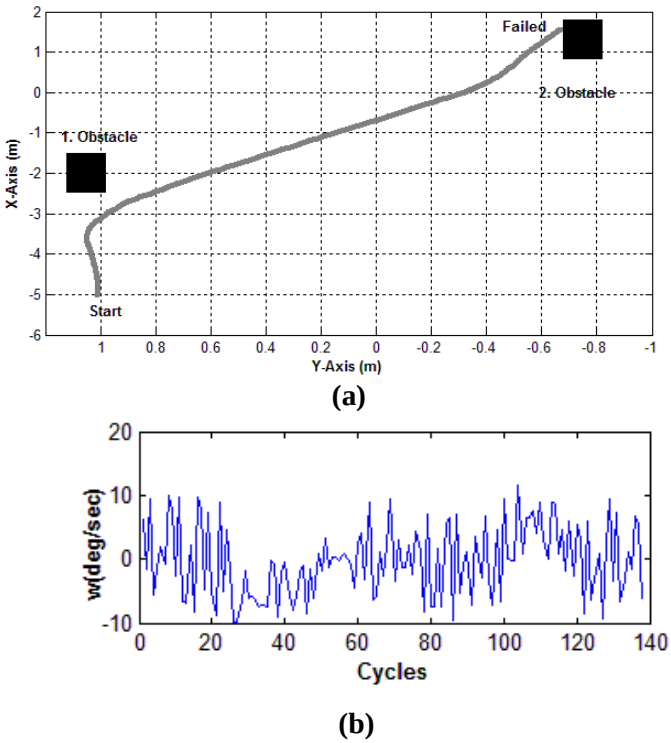
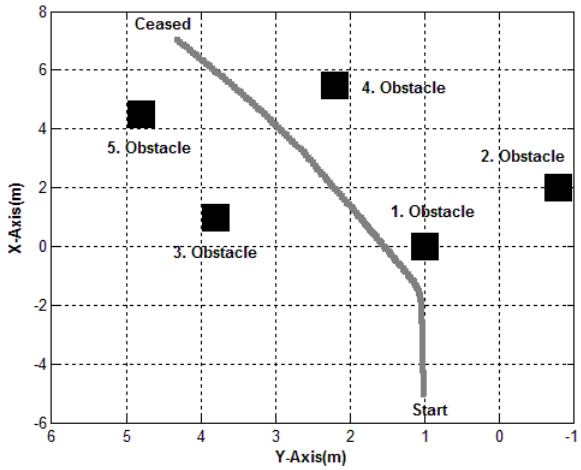


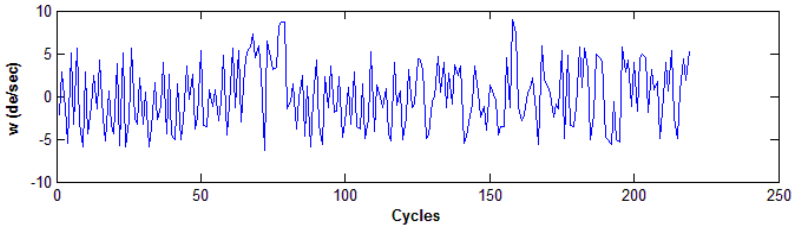
Figure 3.33: Scenario 2 with conventional method (results in failure), (a) robot's trajectory, (b) control parameters

Figure 3.34 displays the test results for the third and most complex scenario, the purpose of which is to estimate the performance of the proposed technique in a partly cluttered environment. For this experiment, five obstacles are located along the robot's path. As the robot start moving, it discovers a collision free path along to the left and keeps moving towards the right side in a smooth manner. On the other hand the robot, under conventional optical flow based control architecture is unable to complete the task, as illustrated in Figure 3.35. The test results of the simulations show that the proposed algorithm is able to safely navigate the robot within its working environment. The first experiment carried out in an unobstructed simulation environment established the basic characteristics of the proposed architecture, and demonstrated that the robot is

able to navigate along a collision free path with minimal positional error. The second scenario was designed to evaluate the performance of the system in the presence of external obstacles in close proximity to the robot. This scenario provides a good illustration of improvement in performance from the conventional optical flow algorithm and the proposed algorithm. The third scenario is designed to establish how the robot navigates and can successfully overcome obstacles in a partly cluttered environment.

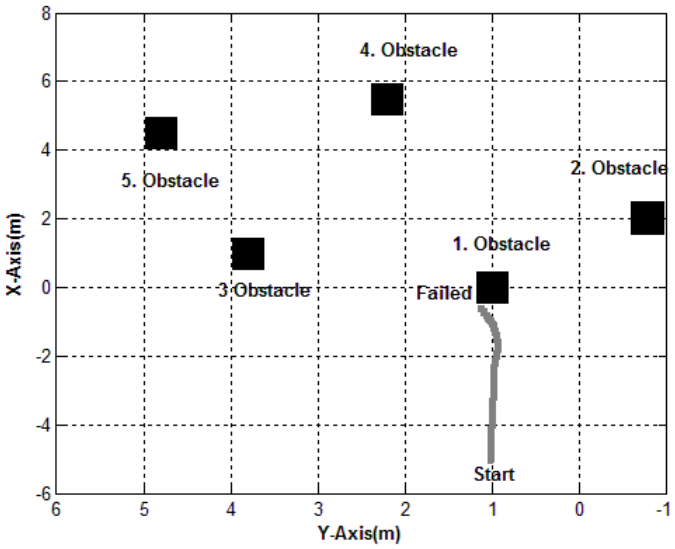


(a)

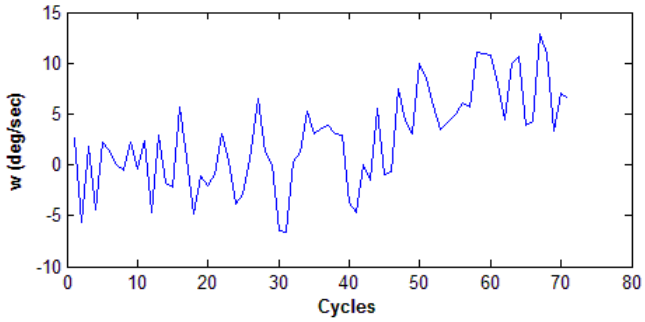


(b)

Figure 3.34: Scenario 3 ‘complex scenario’, (a) robot’s trajectory, (b) control parameters



(a)



(b)

Figure 3.35 Scenario 3 'complex scenario' with conventional method (results in failure), (a) robot's trajectory, (b) control parameter

3.6 Summary

This chapter has presented the development of a novel vision based obstacle avoidance architecture which integrates a high performance appearance-based obstacle avoidance method with conventional optical flow based navigation architecture. Several preliminary simulation based experiments have also been described which compare the different methodologies. Although the experiments demonstrate the potential performance improvement of the proposed system, the simulation is very much ideal,

whereas in a real physical scenario there would be lighting and illumination issues, as well as the physical dynamics of the robot. Consequently the simulation experiments may be somewhat misleading or incomplete without testing the robot in realistic conditions.

CHAPTER 4

VISION BASED MOBILE ROBOT NAVIGATION USING SIFT

This chapter focuses on a vision based mobile robot architecture using principles of image based visual servoing. The first section discusses the basis of the Scale Invariant Feature Transform, including a brief introduction to local features and a detailed explanation of the Scale-Invariant Feature transform (SIFT) algorithm. The next section details the feature based navigation technique using the SIFT algorithm to overcome the navigation problem of mobile robots, followed by the integration of the proposed technique using a subsumption architecture. The proposed architecture comprises several modules, facilitating the mobile robot's navigation and ensuring that it maintains a safe distance from any obstacles while finding the goal from its current position to its destination. The final section of the chapter describes the experimental results for the proposed architecture. All experiments were conducted using the Microsoft Robotics Studio. The simulation results reveal that this system can safely and effectively navigates the mobile robot in partly cluttered environments.

4.1 Local Features

A local feature is basically an image pattern which differs from that of its immediate neighbourhood. It is usually associated with a change of a single image property or several properties concurrently, although it is not necessarily defined precisely by this change. The image properties commonly considered are intensity, colour, and texture. Figure 4.1 shows some examples of local features in a contour images (left) as well as in a gray-value image (right) [Tinne and Krystian, 2008]. Local features can be illustrated mainly by edges, corners (interest points) and contours. Edges characterize boundaries in an image and an edge can be of almost arbitrary shape, and may include

junctions. To apply an edge detecting algorithm to an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties of an image. A comparison of two well known edge detectors, Canny [Canny, 1986] and Sobel [Green, 2002] is illustrated in Figure 4.2. The terms *corners* and *interest points* are used somewhat interchangeably and refer to point-like features in an image, which have a local two dimensional structure; an example with respect to interest point detection is illustrated in Figure 4.1(a). The name *corner* has been used since early algorithms were first performed but this term is used by tradition, for instance a small bright spot on a dark background may be detected as a corner (interest point). Contours provide a corresponding description of image structures in terms of regions, as illustrated in Figure 4.1(b). Besides this, contours descriptors able to obtain a preferred point (a local maximum) in the intensity landscape which means that many contours detectors may also be regarded as interest point operators, and can detect areas in an image which are too smooth to be detected by a corner detector [Maire, 2009].

(a) **(b)**

Fig. 4.1: Illustration of local features in gray-value and coloured images (a) Interest points with corner detection [Tuytelaars and Mikolajczyk 2008], (b) Contour detection [Maire, 2009]

Tuytelaars and Mikolajczyk [2008] divided local feature detectors into three main groups. This is not the only way of categorizing the detectors but it does emphasize different properties required by the usage scenarios. The first group concerns a specific type of local feature, such as image corners from a captured image, whereby the

matching of these local feature points are obtained principally by cross-correlating the image patches around them. For instance, detected edges in an aerial image often correspond to roads, whereas contour detection can be employed to identify impurities in inspection tasks. The second group comprises local feature detectors that can provide a limited set of well localized and individually identifiable points. Local features may not be relevant until their location can be determined accurately and in a stable manner over time.

(a) **(b)** **(c)**

Figure 4.2: A comparison of two edge detectors, (a) original image, (b) canny applied image , (c) Sobel operator applied image

Examples include tracking applications, pose estimation and image alignment. The KLT tracker is a typical example for this group [Tomasi and Kanade, 1991]. The final group comprises detectors which are able to employ a set of local features as a robust image representation that allows objects or scenes to recognized without the need for segmentation. In this case, the goal is not to match them on an individual basis, but rather to analyse their statistical characteristics.

For instance, Schiele and Crowlet [1996] employed multi-dimensional receptive field histograms for object recognition. This is a probabilistic object recognition technique, and does not require correspondence matching of images [Schiele and Crowley, 1996]. Texture analysis, image retrieval, and video mining are other application domains in this category. In addition, feature detectors may be classified based on the types of image features used for detection.Table 4.1 illustrates the classification of feature detection algorithms, mentioned in this chapter.

Table 4.1: Classification of some feature detectors

Feature Detectors	*Edges*	*Corners*	*Blob*
Canny	**X**		
Sobel	**X**		
Tomasi and Kanade		**X**	
Laplacian of Gaussians		**X**	**X**
Difference of Gaussians		**X**	**X**

Feature detection is one of the most challenging aspects of machine vision and refers to methods used to compute abstractions of image information in order to make local decisions at every image point, whether there is an image feature of a given type at that point or not. Once features have been detected, a local image patch around the feature can be extracted, which may involve quite considerable image processing effort. The result is known as a feature descriptor or feature vector. An important development in feature detection and description has been to introduce to the literature the use of the Scale-Invariant Feature Transform (SIFT) [Lowe, 1999; Lowe, 2004].

4.2 Scale-Invariant Feature Transform

The SIFT is an intensity based feature description algorithm that depends on intensity patterns to find points or regions which satisfy some criteria of uniqueness and stability [Hongli et al., 2007]. Applications of the algortihm include object detection, robot navigation, 3D modelling, video/image tracking and gesture recognition, and it first proposed by Lowe [1999, 2004]. Any object in an image is able to provide several features which are points of interests on it that can be extracted to provide a feature description of the object. This description extracted from a training image can then be employed to identify the object when attempting to locate it in a test image containing many other objects. It is important in performing reliable recognition that the set of features extracted from the training image is robust with respect to changes in image scale, noise, illumination and local geometric distortion. Lowe's [1999] patented method can robustly identify objects even among clutter and under partial occlusion because the most notable improvements provided by SIFT are its invariance to image scaling and

rotation and partial invariance to changes in illumination and 3D camera viewpoint. Features are well localized in both the spatial and frequency domains which reduce probability of disruption by occlusion, clutter, or noise. In addition, the features are extremely distinctive. This allows a single feature to be accurately matched with high probability against a large database of features, which is the basis of many applications in computer vision and image processing [Lowe, 2004].

The evaluations carried out so far suggest that SIFT-based descriptors which are region-based are the most strong and distinctive [Mikolajczyk and Schmid, 2005], and are therefore particularly suitable for feature matching and object detection. However, the main drawback of the algorithm is its computational complexity which usually discourages its real-time utilization. There are four major stages of computation used to generate the set of image features: Scale-space extreme detection, Keypoint localization**,** Orientation assignment and Key point descriptor. The overview of the algorithm involving these major steps is illustrated in Figure 4.3.

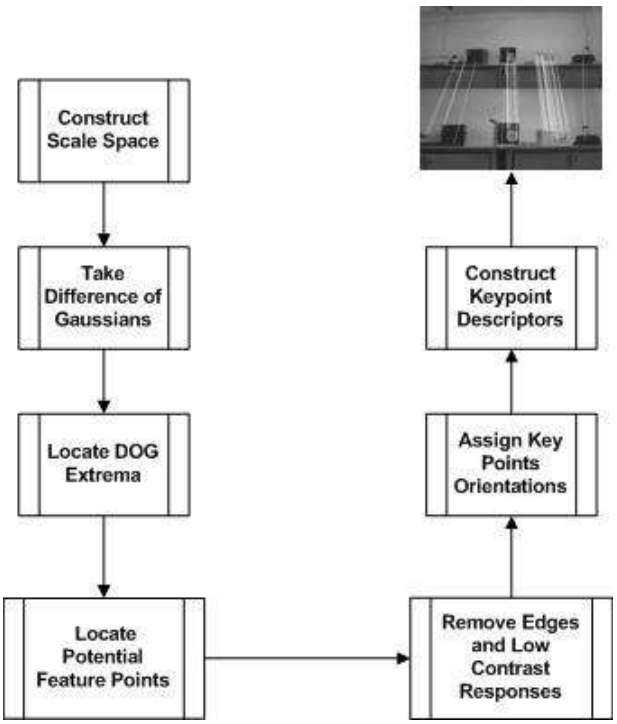


Figure 4.3: Overview of the SIFT algorithm

4.2.1 Scale-space extreme detection

This stage involves the potential interest points (keypoints), which are invariant to scale and orientation in the SIFT framework. According to Lowe [1999], the first stage of interest point (keypoint) detection is to identify locations and scales that can be assigned under differing views of the same object.

To obtain locations that are invariant to scale change of the image can be achieved by searching for stable features across all scales, employing a continuous function of scale known as scale space [Lowe, 1999;2004]. Witkin [1983] proposed a definition as a special type of multi-scale representation, which includes a continuous scale parameter and maintains the same spatial sampling at all scales [Witkin, 1983]. A variety of reasonable assumptions indicates that the Gaussian function is the only possible scale-space kernel [Lindeberg, 1994]. The scale space of any image is defined as a function, $L(x,y,\sigma)$, which is obtained from the convolution of a variable-scale Gaussian, $G(x,y,\sigma)$, with an input image $I(x,y)$. This is illustrated in the following expressions, where * is the convolution operation in x and y.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{4.1}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2 + y^2)/2\sigma^2} \tag{4.2}$$

Mathematically, "blurring" refers to the convolution of the Gaussian operator and the image. Scale-space is described by octaves in that each octave comprises progressively blurred images, and for each octave image it is resized to half of the original image, which can be regarded as a sub-sampling operation. An example of the construction of a scale-space is illustrated in Figure 4.4.

The next step is to employ an appropriate method to generate another set of images in order to detect interest points based on those blurred images. The normalized Laplacian

of Gaussian (LoG), $\sigma^2\nabla^2 G$ which produces the most stable image features compared to a range of other possible image functions, such as the gradient, Hessian, or Harris corner function. The technique is fundamentally based on calculating the second order derivatives, which locates edges and corners on the image. However the second order derivative is computationally intensive. An efficient method to overcome this problem was proposed by Lowe [1999, 2004] who revealed that the *Difference-of-Gaussian* (DoG) function provides a close approximation to scale-normalized LoG . To compute *DoG(x,y,σ)*, the difference of two successive Gaussian-blurred images, separated by a multiplicative constant factor *k*, is convolved with the input image, as given by the following equations:

$$DoG(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \quad (4.3)$$

This can also be simplified in the following expression:

$$DoG(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)) \quad (4.4)$$

where $L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y)$ and $L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y)$.

The final step is to obtain local extreme points from DoG images. In order to achieve this, for each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images, as illustrated in the left part of Figure 4.5.

Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images, illustrated on the right hand side of the Figure 4.5. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process is repeated. Lowe [1999, 2004] suggests that 4 octaves and 5 blur levels are ideal for the algorithm. Extreme points are identified as local maxima or minima of the DoG image across scales. Each pixel in the DoG images is compared to its 8 neighbours at the same scale, plus the 9 corresponding neighbours at neighbouring scales. If the pixel is a local maximum or minimum, it is selected as a candidate keypoint, as illustrated in Figure 4.6.

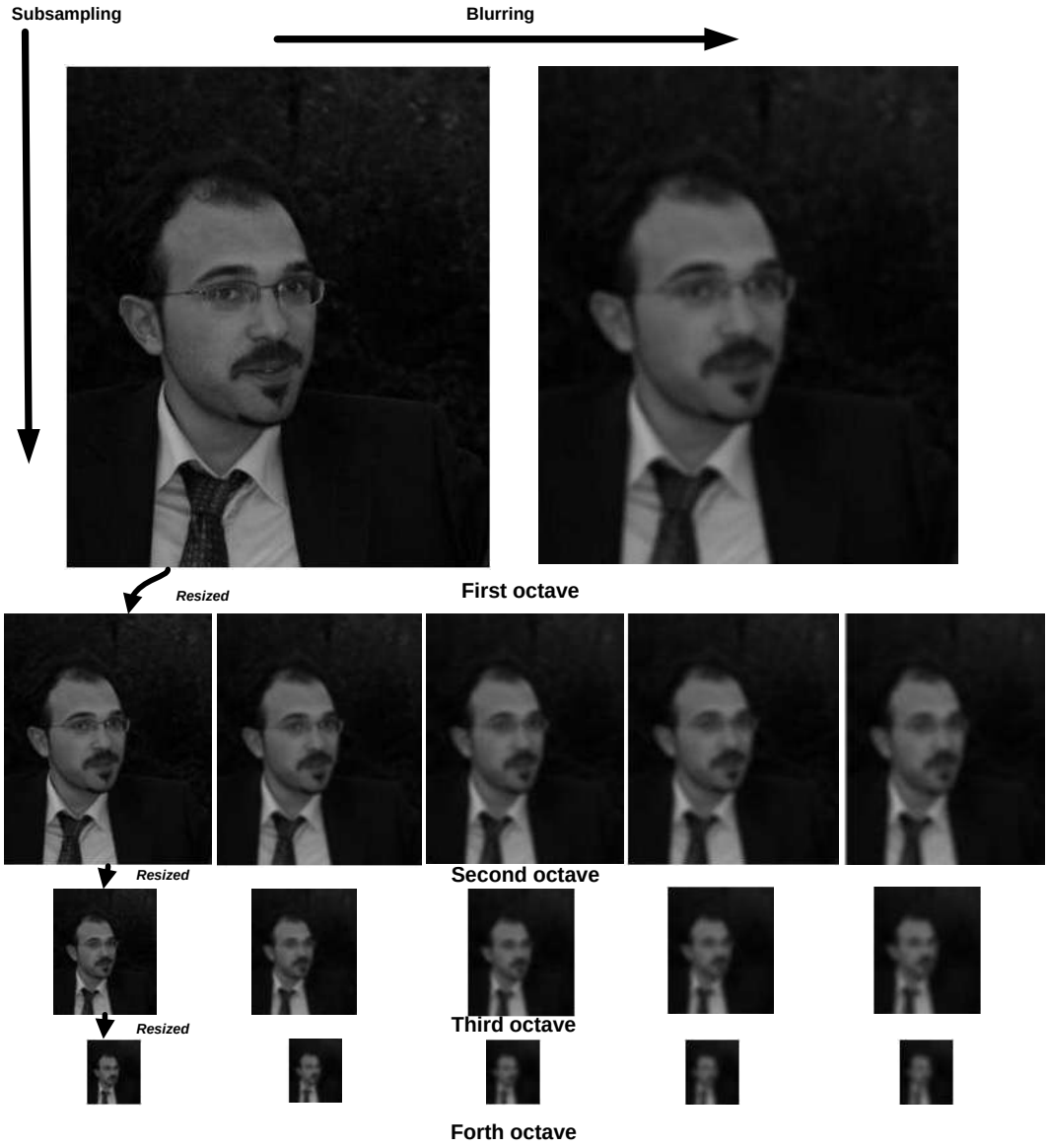


Figure 4.4: Scale spaces in SIFT (first octave is simplified, just including first and last blur levels)

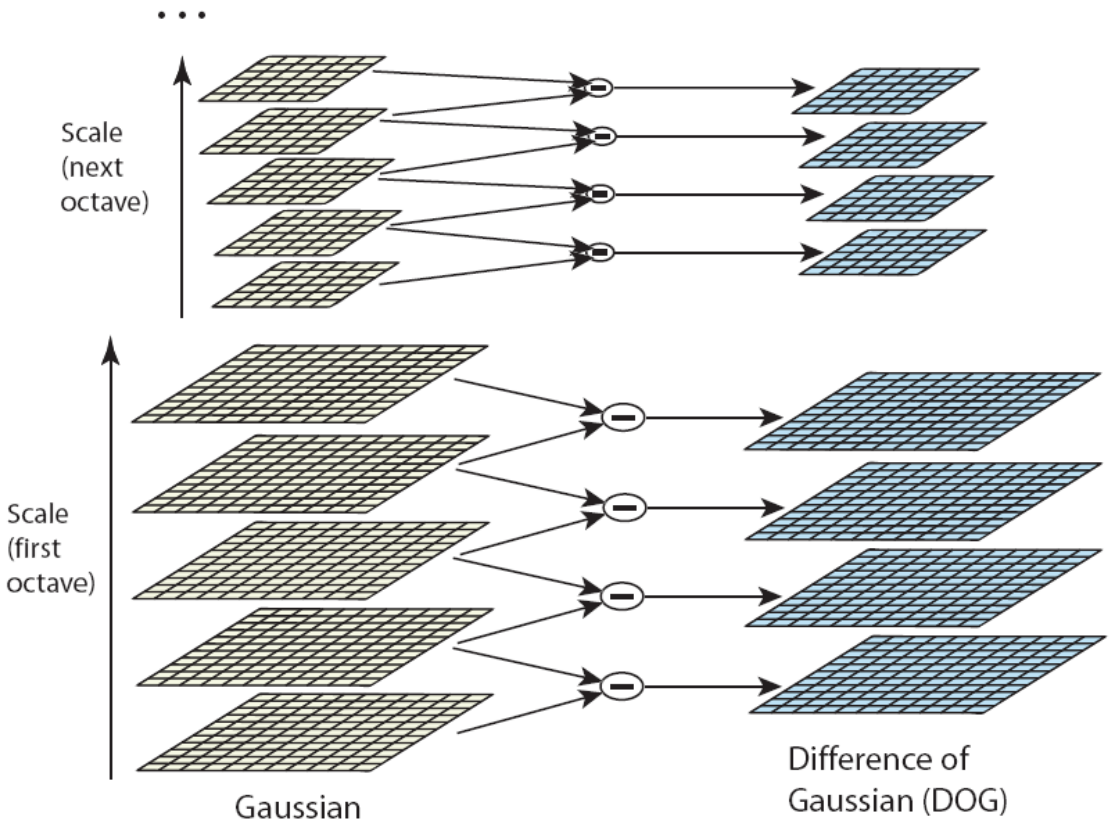


Figure 4.5: Calculation of DOG images [Lowe, 2004]

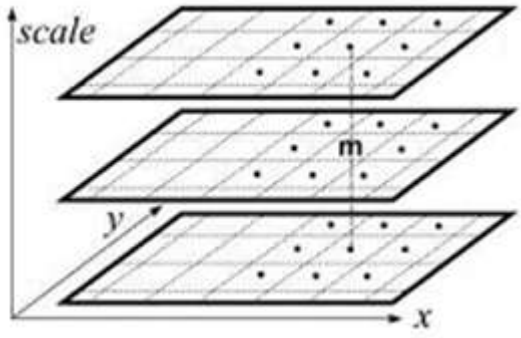


Figure 4.6. Keypoint identification

4.2.2 Keypoint localization and edge elimination

Once extreme points are detected by comparing a pixel with its neighbours, the next step is to perform a detailed fit to the nearby data for location, scale, and ratio of

principal curvatures. This allows points to be rejected that have low intensity or are poorly localized along an edge. Lowe [1999] located keypoints at the location scale of the central sample point. However the local maxima and minima almost never lies exactly on a pixel. It lies somewhere between the pixel. But it cannot simply accessed data "between" pixels; so the subpixel locations must be located mathematically.

In, Lowe [2004] this method was enhanced to strip away any unstable keypoint by employing a Taylor expansion of the scale-space function to reject those points that are not distinctive enough or are unsatisfactorily located near the edge. In order to achieve this, Lowe [2004] adapted a technique proposed by Brown and Lowe [2002] for fitting a 3D quadratic function to local sample points to determine the interpolated location of the maximum. The interpolation is carried out using the Taylor expansion of the $DoG(x, y, \sigma)$ scale-space function, with the candidate keypoint as the origin [Lowe, 2004]. The next step is to eliminate keypoints which either do not have enough contrast or which lie on an edge. In order to reject low contrast images, a simple thresholding technique is employed; if the magnitude of the intensity at the current pixel in the DoG which is being checked for minima/maxima is less than a threshold value, it is rejected. Finally, the keypoints that are poorly located on edges are excluded which increases the efficiency and also the robustness of the algorithm. The main principle behind the edge elimination process is to calculate a principal of curvatures, measuring the maximum and minimum bending of a regular surface at each point. These gradients are perpendicular to each other. A keypoint may be classified using these gradients and can be defined as follows:

- For flat regions both gradients will be small.
- Edge responses have one big gradient, are perpendicular to an edge, and one small gradient.
- Corners are the most consistent points and both of the gradients are big.

Lowe [2004] utilized a Hessian Matrix to obtain corner points in order to check whether or not a point is a corner with high accuracy. Mathematically, the curvature is retrieved

from the eigenvalues of the second-order Hessian Matrix *H* shown in the following expression:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (4.5)$$

where the derivatives are estimated by taking differences of neighbouring sample points, The ratio of principal curvature is directly related to the ratio of the trace and determinant of a matrix. The trace of an *n*-by-*n* square matrix *H* is defined to be the sum of the elements on the main diagonal from the upper left to the lower right of *H*, and the determinant. If the result is above a certain threshold the keypoint is rejected [Harris and Stephens, 1998]. Let *a* be the eigenvalue with the largest magnitude and *b* is the smaller one, then:

Tr(H) = $D_{xx+}D_{yy} = a+b$

Det(H) = $D_{xx}D_{yy} - (D_{xy})^2 = a \times b$

If r is the ratio between the largest and smallest one, then $a = r \times b$ gives the following expressions:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(r \times b+b)^2}{rb^2} = \frac{(r+1)^2}{r} \quad (4.6)$$

The quantity $\frac{(r+1)^2}{r}$ is at a minimum when the two eigenvalues are equal, therefore:

$$if \frac{Tr(H)^2}{Det(H)} > \frac{(r+1)^2}{r}, rejected \quad (4.7)$$

4.2.3 Orientation assignment

Legitimate key points have been obtained so far, which have been tested for stability. The next step is to assign a consistent orientation to each keypoint, providing rotation invariance. The purpose is to collect gradient directions and magnitudes around each keypoint to discover the most prominent orientation(s) in that region which will then be assigned to the keypoint. There is a right proportion between the size of the 'orientation collection region' around the keypoint and the scale of this keypoint. Gradient magnitude and orientation are calculated respectively for each scale invariant image sample, *L(x,y)*, as shown in the following expressions:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \qquad (4.8)$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \qquad (4.9)$$

According to Lowe [2004], an orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint. The orientation histogram has 36 bins covering the 360 degree range of orientations. For instance, assuming that the gradient direction at a certain point in the orientation collection region is 15.675 degrees, and then it will go into the 10-19 degree bin. The amount that is added to the bin is proportional to the magnitude of gradient at that point. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint. Once this operation is performed for all pixels around the keypoint, the histogram will have a peak at some point. Peaks in the orientation histogram correspond to dominant directions of local gradients, and any local peaks which are above 80% of the highest peak are converted into a new keypoint which has the same location and scale as the original, but different orientation. As a result, for locations with multiple peaks of similar magnitude, there will be multiple keypoints created at the same location and scale but different orientations.

4.2.4 Keypoint descriptor

The previous steps, calculating key point locations at particular scales and assigning orientations to them, are also called upon to assign 2D image location, scale, and orientation parameters to keypoints. This ensures invariance to image location, scale and rotation. The next step is to generate a descriptor vector for each keypoint that is distinctive and partially invariant to the remaining variables including illumination and 3D viewpoint. First a set of orientation histograms are created on 4x4 pixel neighbourhoods with 8 bins each. These histograms are computed from magnitude and orientation values of samples in a 16 x 16 region around the keypoint such that each histogram contains samples from a 4 x 4 sub-region of the original neighbourhood region. The magnitudes are further weighted by a Gaussian function with σ equal to one half the width of the descriptor window. The descriptor then becomes a vector of all of the values of these histograms, which leads to a SIFT feature vector with 8x4x4 = 128 elements. Figure 4.7 displays an example a 2x2 descriptor array computed from an 8x8 set of samples whereas, as previously mentioned, the real experiments conducted with the algorithm employing 4x4 descriptors are computed from a 16x16 sample array.

To cope with the problem of illumination the vector is normalized to unit length. A change in image contrast in which each pixel value is multiplied by a constant will multiply gradients by the same constant, so this contrast change will be cancelled out by vector normalization. A brightness change in which a constant is added to each image pixel will not affect the gradient values, as they are computed from pixel differences. Therefore, the descriptor is invariant to affine changes in illumination [Lowe, 2004]. Figure 4.8 displays two different examples of the key-point selection procedure and the 4x4 SIFT descriptor frames of these points. An example is illustrated in Figure 4.9 (a) using Lowe's [2004] conventional key point selection and visualizing software application to extract keypoints from a low-resolution natural image, where each extracted keypoint is associated with a vector, and each vector demonstrates the location, scale, and orientation of the relevant keypoint.

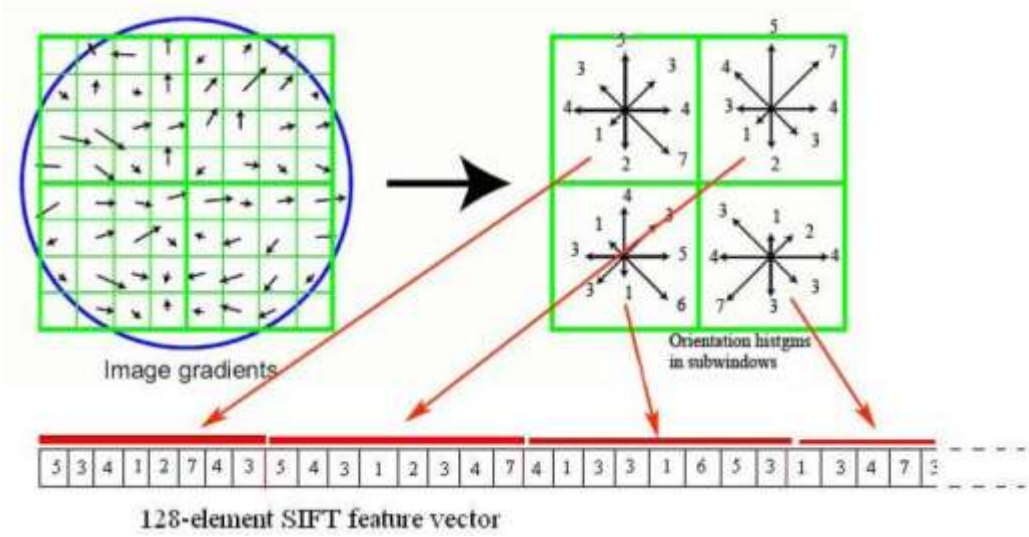
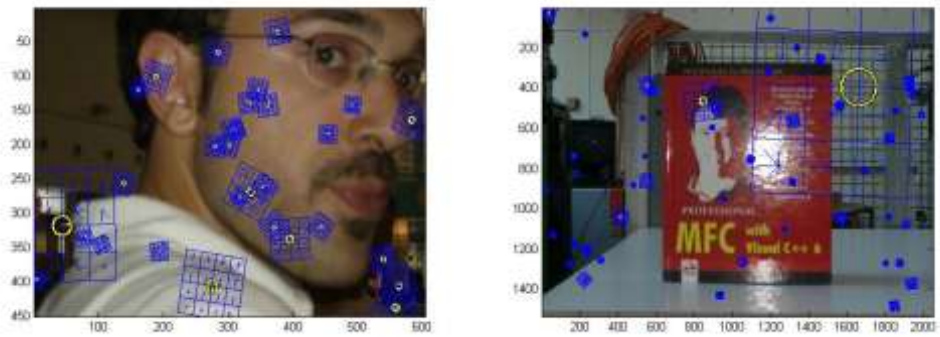


Figure 4.7: An example of a 2x2 descriptor array for the SIFT algorithm



(a) **(b)**

Figure 4.8: Some of the detected SIFT descriptors for, (a) the first image, (b) the second image

4.2.5 SIFT matching

Lowe [2004] proposed the nearest neighbour algorithm, where a candidate is located by computing and ranking in ascending order the angle between the descriptors using a vector dot product. False matches can be initially rejected using the likelihood ratio test if the ratio between the potentially best matched descriptor to its next best is above a previously defined threshold, which can be represented by Equation (4.10).

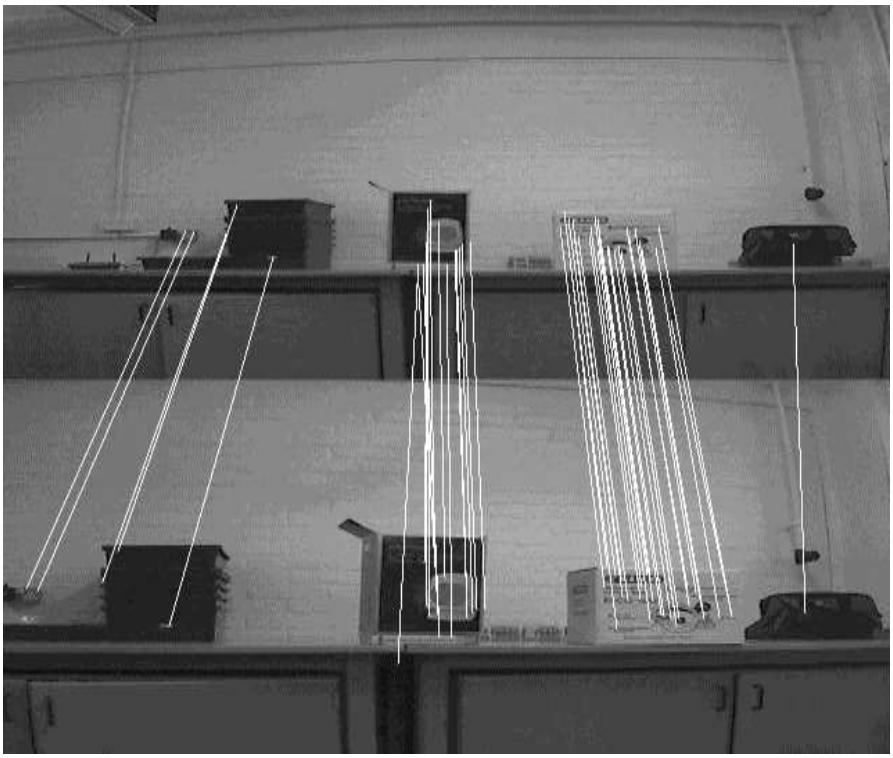
$$\text{Match}=\begin{cases} true\ , & if\ \frac{val_1}{val_2} < tr \\ false\ , & otherwise \end{cases} \quad (4.10)$$

where $va1_1$ is the best matched descriptor, $va1_2$ is the next best matched and *tr* is the threshold value.

An example utilizing the SIFT matching algorithm is illustrated in Figure 4.9 (b). In the conventional method, Lowe [2004] rejected all matches in which the distance ratio is greater than 0.8, which eliminates 90% of false matches while discarding less than 5% of the correct matches [Lowe, 2004] .

In addition, to verify matches between two different range images captured of different scales or angles, a Hough transform [Duda and Hart, 1972] can be employed to identify clusters of features that have a reliable interpretation of an object by a voting procedure. This is applied to compute the similarities between the two sets of descriptors based on the features exhibiting the same relevant parameters involving translation, orientation and scale. If any of the clusters has three or more entry points, it is possible to apply a robust fitting procedure in which a linear least squares solution is performed for the parameters of the affine transformation relating the model to the image. Outliers are discarded, due to the agreement between each image feature and the model, giving the parameter solution. Each match is required to agree within half the error range that was used for the parameters in the Hough clusters. After discarding the outliners, the linear least squares solution is performed with the remaining points and the process iterated. If fewer than 3 points remain after discarding outliers, then the match is rejected. This helps to maximize the performance of object recognition for small or highly occluded objects. Further details can be found in Lowe [2004].

(a)



(b)

Figure 4.9: SIFT, (a) Keypoints are extracted from a low resolution Image, (b) an example for SIFT Matching algorithm

4.3 Evaluation of the SIFT Algorithm

Lowe [2004] derived a 2D method for image feature generation based on the Scale Invariant Feature Transform which transforms an image into a large collection of local feature vectors, each of which is invariant to image translation, scaling and rotation. This method is reasonably efficient and reliable for feature extraction and object detection tasks. The evaluations performed suggests that SIFT-based descriptors, which are region-based, are the most robust and distinctive and are therefore best suited for feature matching [Tao et al., 2010]

Mikolajczyk and Schmid [2005] evaluated interest point descriptors by comparing descriptors computed on regions extracted with recently proposed scale and affine-invariant detection methods. The tests were designed for the matching and recognition of the same object or scene. According to the test result, Gradient Location and Histogram (GLOH) algorithm, an extension of the SIFT descriptor designed to increase its robustness and distinctiveness, closely followed by conventional SIFT algorithm, demonstrated the robustness and the distinctive character of the region-based SIFT descriptor. While the ranking of the descriptors is similar for different matching strategies, the conventional SIFT algorithm gives relatively better results if the nearest neighbour distance ratio is used for thresholding. Another comparison provided by Juan and Gwun [2009], included the Speeded Up Robust Features (SURF) algorithm. The results show that despite the computational advantages of the SURF algorithm it is not stable to rotation and illumination changes. Nevertheless, the conventional SIFT algorithm presents its stability in most situations although it is slower than SURF.

The conventional SIFT algorithm is computationally intensive, due to the serial timing of Gauss blur when constructing scale space, and multiple loops when generating descriptors. Furthermore, convolution and complex arithmetic operations dealing with floating-point data, such as exp, floor, sin, cos, are known to be time-consuming. For that reason, it is hard for conventional SIFT to attain appropriate real-time performance on any mobile robot platform. The performance of the algorithm can be enhanced with multiprocessing programming techniques or libraries. One of those is OpenMP library

which is a free application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran on many architectures and platforms. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behaviour. The enhancement methods based on this library and the performance evaluation test results of the Fast SIFT algorithm are discussed in Chapter 6.

4.4 Navigation via SIFT based on Monocular Vision

Vision is potentially the most powerful sensing capability in providing reliable and safe navigation. For indoor navigation, researchers rely on artificial landmarks such as coloured or geometrical objects to achieve safe navigation. Many approaches which employ artificial landmarks are easy to both design and implement; however, the main disadvantage of these approaches is their dependence on specific tasks. Feature based mobile robot navigation provides a good alternative to these methods which makes no explicit attempt to localise itself and thus requires no landmark map. The main problem in this approach is to solve the feature extraction and correspondence problem consistently. Accordingly, distinctive features are extracted from both the reference image and the snapshot (current image). Each identified feature in the snapshot is then usually paired with one feature in the reference image (the correspondence problem), and the robot is finally steered towards the goal, depending on the control algorithm. However, such techniques rely on omnidirectional vision which captures images at low resolution. Alternatively, stereo vision-based techniques acquire robust in-depth information, and are very common. However these techniques suffer from several disadvantages, involving the computational cost of the stereo vision systems, and synchronization problems between the cameras and their calibration (see Chapter 2).

In this section a new alternative method inspired by the image visual servoing control architecture relying on SIFT based feature tracking algorithms is introduced. The main idea is to generate control variables involving linear velocity (m/sec) and angular velocity (deg/sec), based on matching results between the current image and the goal image. The proposed control algorithm is entirely qualitative which does not employ the

traditional concepts of jacobians, homographies, or fundamental matrices. Preliminary test results verify that the proposed algorithm estimates the turning rate and linear velocity of the mobile vehicle with reasonable accuracy and affordable computational time. A flow chart of the proposed algorithm is illustrated in Figure 4.10.

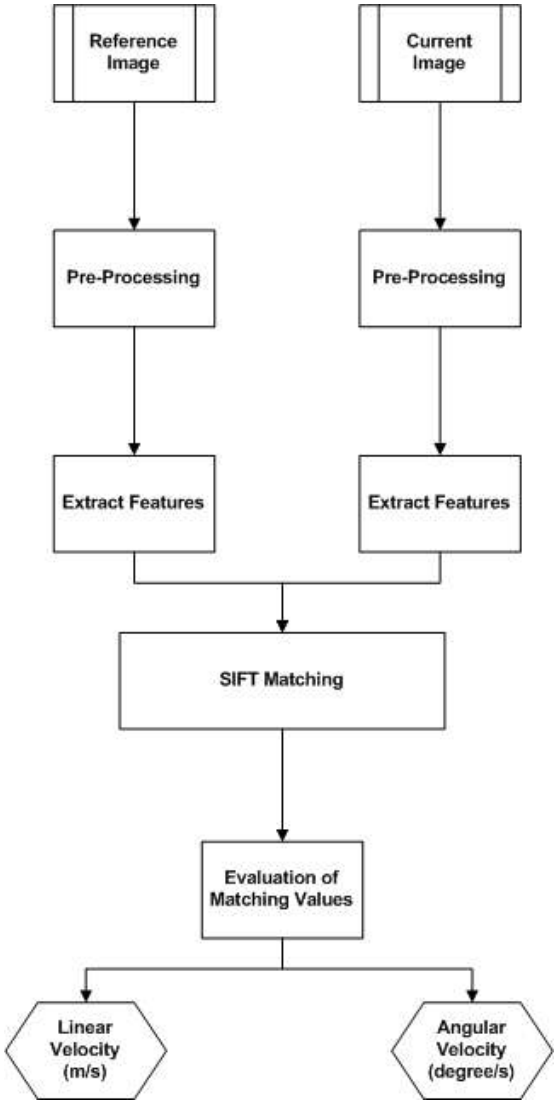


Figure 4.10: Flowchart of the SIFT-based control system

The first part of the algorithm enhances the input image against any possible illumination or noise. This is called pre-processing. The input image is convolved with a filter based on the first derivative of a Gaussian to obtain a blurred version of the image, which removes unexpected noises and smoothes images. Subsequently, histogram

equalization is applied to the filtered image to adjust its contrast. The second function of the algorithm extracts key features from enhanced images using a SIFT algorithm. As mentioned previously, SIFT is one of the most powerful and popular feature detection algorithms, but due to its computational cost it is not suitable for real-time applications. In order to cope with this problem, a cross-platform library that computes fast and accurate SIFT image features, which is optimized with OpenMP is used instead of the conventional SIFT implementation.

The performance of this enhanced algorithm is quite impressive. To validate performance with different image resolutions, several experiments were performed. After extracting features from both current and active images, an improved version of Lowe's [2004] matching algorithm is utilized to match the features. The performance of the conventional matching algorithm has been improved by OpenMP which is discussed in Chapter 6.

The proposed control strategy resembles the conventional balance strategy [Duchon et al., 1998; Temizer, 2001] (see Chapter 3). The next step is to evaluate matched points and assign each matched point on the current image to a corresponding position. There are four clusters having the same size namely: Left(c_l), More-Left(c_{ml}), Right(c_r) and More-Right(c_{mr}) respectively. Clusters are obtained by dividing the image vertically and each of these clusters is considered with the total number of matched features, which is used to generate control variables. The distribution *(d)* of the matched values is a key factor in order to estimate the next possible turning rate, which can be illustrated by the following expressions:

$$d = \begin{cases} -\dfrac{M_r}{M_l}, & M_l < M_r \ , \quad if \ \ M_l = 0 \ then \ \ M_l = 1 \\ \dfrac{M_l}{M_r}, & M_r < M_l \ , \quad if \ \ M_r = 0 \ \ then \ M_r = 1 \end{cases}$$

(4.11)

where M_l and M_r are the total count of the matches on the left hand and right hand parts of the image. To obtain a more robust and sensitive control equation, distribution *(d)* might be redefined, including all matching clusters which is then able to approximate turning rate with higher accuracy, as shown in the following expressions:

$$d = \begin{cases} -(s_{w1} x \frac{c_r}{c_l} + s_{w2} x \frac{c_{mr}}{c_{ml}}), M_l < M_r \ , if \ \ c_{l\,(and/or)}\ c_{ml} = 0 \quad then \ \ c_{l\,(and/or)} \quad c_{ml} = 1 \\ s_{w1} x \frac{c_l}{c_r} + s_{w2} x \frac{c_{ml}}{c_{mr}}, \ M_r < M_l \ , \ \ if \ \ c_{r\,(and/or)}\ c_{mr} = 0 \ then \ \ c_{r\,(and/or)} \quad c_{mr} = 1 \end{cases}$$

(4.12)

where $0 < s_{w1} \leq 1$, $0 < s_{w2} \leq 1$ and $s_{w2} > s_{w1}$

In order to estimate the next possible turning rate *(w)*, *d* is multiplied by a model parameter value which varies between 1 and 2 and can be defined as follows:

$$w = d \times s_w, \ where \ 0 \leq s_w \leq 2 \qquad (4.13)$$

When the robot approaches its goal with the capability of keeping it in the field of view, matching strength usually tends to increase. Therefore, the matching strength can be adapted to arrange the linear velocity. In order to achieve this, a simple but efficient velocity model is proposed, illustrated in the following expressions:

$$v = \begin{cases} m_t * k_v, \ v_{th} < m_t \\ v_i, \ v_{th} \geq m_t \end{cases}$$

In order to constrain the linear velocity, the velocity $'v'$ is compared with $'v_{max}'$, as shown in following expression:

$$v = min(v, v_{max}) \qquad (4.14)$$

where v_i = minimum linear velocity.

k_v = constant used to convert matching value to linear velocity .

m_t = number of total matched points.

v_{th} = linear velocity threshold parameter.

v_{max} = maximum accepted linear velocity .

The proposed algorithm is designed for mobile vehicles in which the only interaction with the motors is carried out by using the robot's forward speed (m/sec) and its angular velocity (turning rate) (deg/sec). At the end of each processing cycle, these two control variables completely define the output behaviours. The algorithm principally proposes a reliable solution to the Image based visual servoing (IBVS) problem which uses visual information to control the vehicle's pose with respect to a specific goal. Visual homing is also a type of visual servoing [Szenher, 2008]. Thus the terms are used interchangeably. The main idea behind Visual Homing strategies is to infer the direction and/or distance to the goal location from the disparity between the current and goal images. The control variables of the proposed algorithm can be easily adapted to a visual homing strategy with, minor modifications. As discussed previously in Chapter 2, homing vectors are estimated for each processing cycle until the discrepancy between the current and reference images falls below a certain threshold value. Each homing vector comprises a rotation angle to decrease the orientation difference between two images.

As no metric landmark information is used, the homing vector $\vec{H}$ is often inaccurate The agent therefore moves by some distance (either fixed or calculated based on current sensor information) in the direction of homing vector, $\vec{H}$. In order to make an approximation to homing vectors, Equation 4.13 can be used to obtain rotation angle instead of angular velocity by changing interval of the model parameters, shown in the following expressions:

$$a = d \times s_{wa}, \; where \; 0.5 < s_{wa} \leq 1 \qquad (4.15)$$

In addition, instead of estimating *linear velocity*, Equation 4.14 can also be adapted with a minor modification to be used to estimate forward translation (forward displacement) or the magnitude of each homing vector, as illustrated in the following expressions:

$$ft = \begin{cases} m_t * k_{ft}, & ft_{th} < m_t \\ ft_i, & ft_{th} \geq m_t \end{cases}$$

In order to constrain the forward translation, ft is compared with ft_{max}, *as* shown in following expression:

$$ft = min(ft, ft_{max}) \qquad (4.16)$$

where ft_i = minimum forward displacement .

k_{ft} = constant used to convert matching value to forward translation

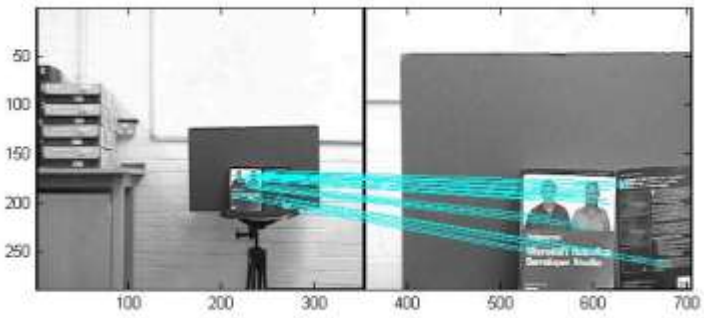
m_t = number of total matched points.

ft_{th} = linear velocity threshold parameter.

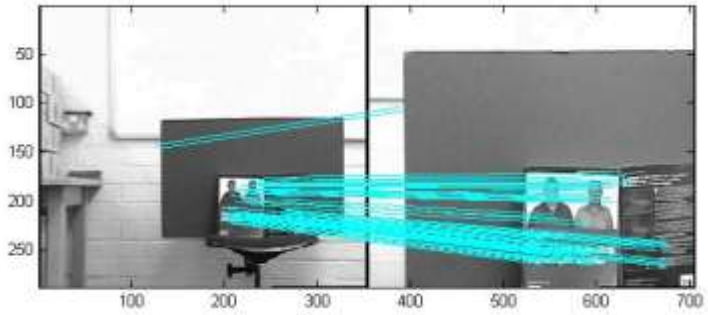
ft_{max} = maximum accepted forward displacement.

Unlike conventional visual homing techniques, the proposed method is designed for monocular vision based navigation systems. Despite the fact that a wide angle of view permits a mobile robot to interact with a curved path even around sharp corners, hairpin turns or other complicated curves, the main disadvantages of the omnidirectional vision systems are geometric distortion and poor resolution. Monocular vision is able to cope with these problems, although its drawback is that the target objects may be outside the field of view of the camera. In order to overcome this problem and to provide reliable navigation, the SIFT based algorithm is performed with a monocular camera equipped with *pan* and *zoom* functions, which are discussed in the following section. Figure 4.11 illustrates three different frames; each of these captured from different locations to the goal, and was matched with a reference image. The steering angle (w) and linear velocity based on the Equations 4.12 and 4.14 were calculated for each frame using parameters shown in Table 4.2. According to the results, it can be shown that when the

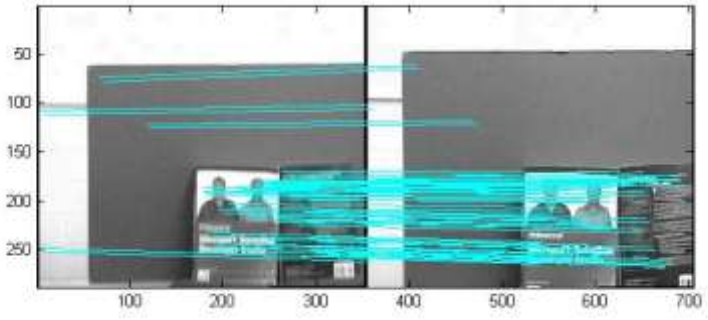
similarity between the goal and the current image increases, control parameters are updated successfully.



(a) w= -6.24(deg/sec); v=0.2080(m/sec)



(b) w = -10.8 (deg/sec); v=0.312 (m/s)



(c) w = -8.8 (deg/sec) ; v= 0.36 (m/s) (vmax)

Figure 4.11: Different snapshots are matched via a reference image from a to c (left side is for the current and the right is for the reference images)

4.5 Design of a Reactive Architecture using Subsumption Architecture

There are several approaches to designing a behavioural-based architecture, which have been discussed in Chapter 2. In this study, the architecture has been designed based on the subsumption architecture in which each layer or behaviour implements a particular goal of the robot and higher layers are increasingly abstract [Brooks, 1986].

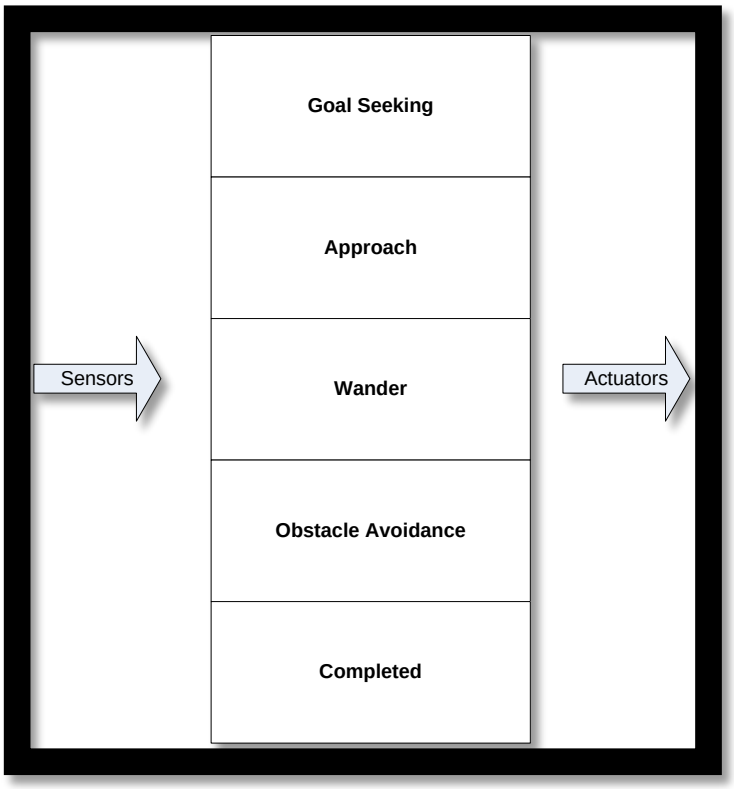


Figure 4.12: Behavioural design of the proposed architecture

Each layer's goal subsumes that of the underlying layer, and their interaction with each other is illustrated by using finite state machines (FSM). Finite state machines define several states (behaviours) that represent a current situation for the robot. Certain events in the outside world can change the state. For instance, the robot could have an approach state whereby it is moving about the environment trying to get closer to the goal. When the laser range finder detects nearby obstacle, the state may change from approach to obstacle avoidance, and the avoidance algorithm will move the robot away

from the obstacle. When the obstacle has been avoided, the robot state will change back to approach. The architecture, designed for this study comprises five behaviours, namely: goal seeking approach, wander, obstacle avoidance and completed, as illustrated in Figure 4.12. The state diagram of the behaviours is shown in Figure 4.13.

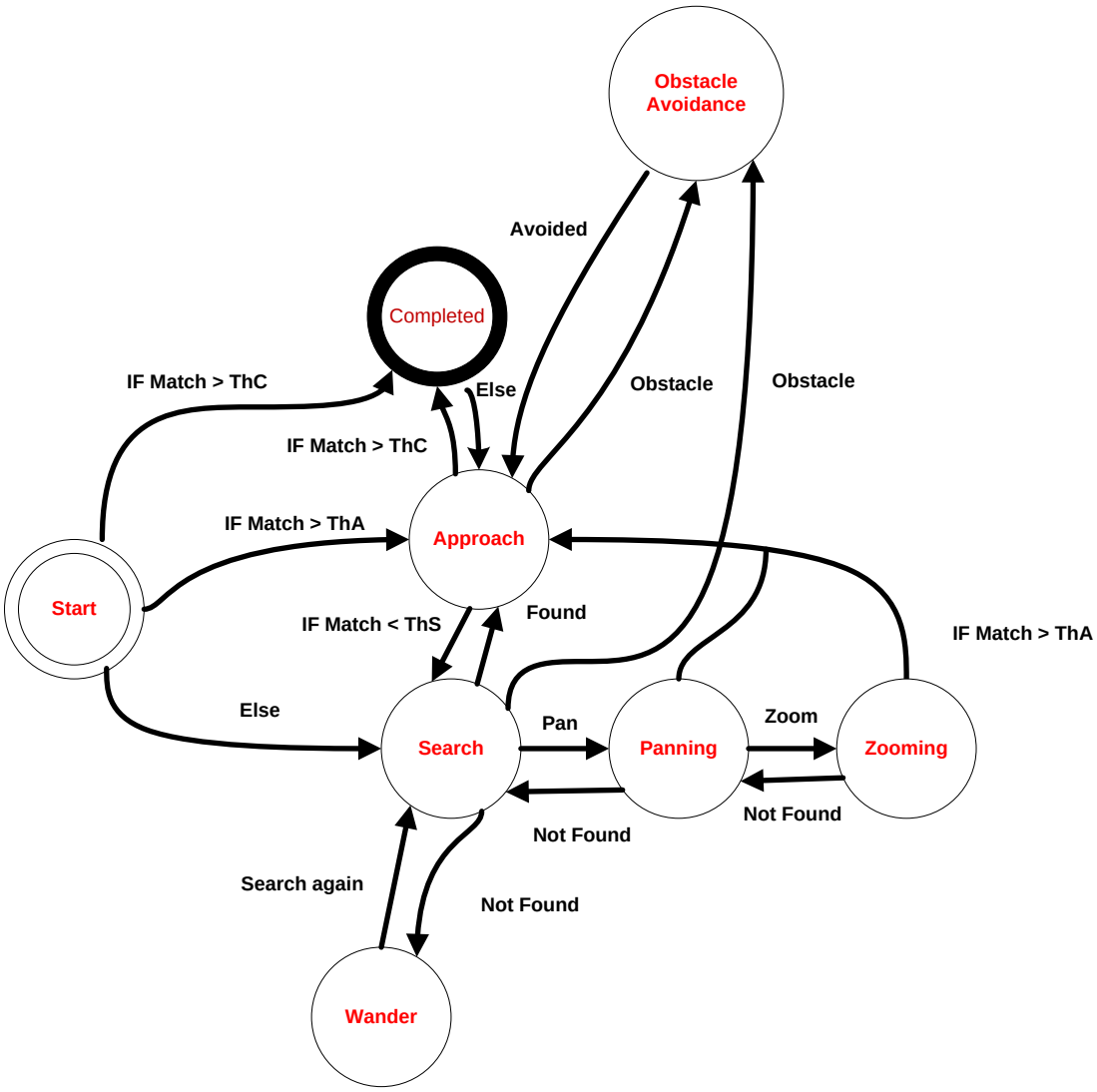


Figure 4.13: The state diagram for behaviours

4.5.1 Goal seeking

Reactive-based architectures are widely used in autonomous navigation. Goal seeking is one of the key behaviours of these architectures and is used to find and acquire the target object in complex environments. The goal seeking behaviour is designed for monocular vision systems equipped with pan and zoom functions. The main drawback of monocular vision systems, as mentioned previously, is their limited field of view which is not appropriate for goal-based navigation in dynamic and partially cluttered environments. To enhance the field of view of the monocular vision system, the pan function of the camera is adapted to the navigation algorithm. The main objective of this behaviour is to seek the goal in order to compute control parameters of the robot on a real-time basis. The state diagram for this behaviour is shown in Figure 4.13, and it checks for the existence of any clue about the goal, based on the SIFT algorithm. The strength of the matching results obtained from output of the algorithm, is utilized to determine the next possible state or behaviour. If the computed matching value is higher than a predefined threshold value, *ThA* or *ThC*, the robot enters the approach or completed state. Otherwise, the panning stage of the current behaviour is activated, which entails panning the camera left and right at a predefined (small angle in both cases, to enhance the field of view of the robot's vision. The most reliable way to obtain high accuracy at this stage is to pan until reaching the limitations of the physical sensor. However, the larger angle the camera turns through the more processing time it consumes. For real-time applications, processing time and the physical limits of the camera are important limitations. Therefore, a high quality camera is essential to improve the performance of the proposed system, as discussed in Chapter 6. In the real experiments for this study, an interval of ±90° is employed in order to obtain a good search performance with affordable processing time based on the capacity of the vision sensor.

To increase the accuracy of the navigation, a zooming stage which involves changing the focal length of the lens to bring the subject closer or further away in the frame, is activated, depending on the strength of matching value. Consequently, if the consistency between the panning and zooming stages is obtained relating to the goal,

and the matching value is more than a predefined threshold value, the approach behaviour is activated. Otherwise, if the target cannot be acquired, the wander behaviour is activated to displace the robot's position randomly.

4.5.2 Approach

This is the main behaviour of the proposed control architecture which directs the robot to its goal. This behaviour is only activated when the existence of the target is detected to a high accuracy. In order to navigate in a smooth way the turning rate *(w)* and linear velocity *(v)* are adjusted respectively. The state diagram for this behaviour is shown in Figure 4.13, and if the strength of the matching results is more than an appropriate threshold value, *ThC*, the behaviour completed is activated.

On the other hand, if the difference between the previous and current matching results is less than a predefined threshold value, *ThS*, the first panning stage, is activated to seek the goal. If this fails, then progressive zooming stages are activated, and if these stages succeed in tracking the goal, the robot keeps navigating in this manner until the completed behaviour is activated; otherwise the wander behaviour is activated to relocate the robot randomly. This is discussed in more detail in the following section.

4.5.3 Wander

Wandering is a form of random steering, and is the main state of the robot while it navigates within the environment. However it is not a preferable behaviour in this research which in essence navigates the robot aimlessly until it encounters the goal. For instance, at time t_1, the robot may be in the process of turning to the right, and at time t_2 it will still be turning in almost the same direction. The steering vector takes a random walk from one direction to another. To produce the steering vector for the next instance, a random displacement is added to the previous value [Reynolds, 1999]. An example of wander behaviour is included in Appendix C.

4.5.4 Obstacle avoidance

This behaviour utilizes information from the laser range finder. The perceptual schema for the output of this behaviour generates an avoiding manoeuvre for the robot. After each scan, the laser range finder returns a corresponding point for each unit of angular resolution which represents the distance between the robot and any obstacle that the laser detects. This behaviour is activated whenever the laser range finder returns a value within a distance of influence. The angle between the robot's heading angle and *i*-th unit of angular resolution depends on the laser's configuration. For instance, if the range finder has a 180° field of view and 360 units of angular resolution, as shown in Figure 4.14, the angle of the 270th unit of angular resolution is equal to 45° with respect to the robot's heading. In this navigation architecture, a simple and efficient algorithm is employed to deal with the obstacle avoidance problem using a laser range finder. According to the algorithm, the output of the laser is first simplified by classifying angular resolution units into *n* clusters (O_c) where, for instance the range finder has a 180° field of view and 360 units of angular resolution; thus each cluster is responsible for $\frac{360}{n}$ units of angular resolution and a $\frac{360}{2n}$ degree field of view. Subsequently, the average range value of each cluster is calculated, and then the average values of each cluster with a distance of influence is summed up based on the corresponding field of view value in order to estimate the next possible turning rate for the avoiding manoeuvre. It can be expressed mathematically as follows:

$$w = l_g \times \sum_{i=1}^{n} O_c\,(i)\,,\ if\ O_c(i) < d_o \qquad (4.17)$$

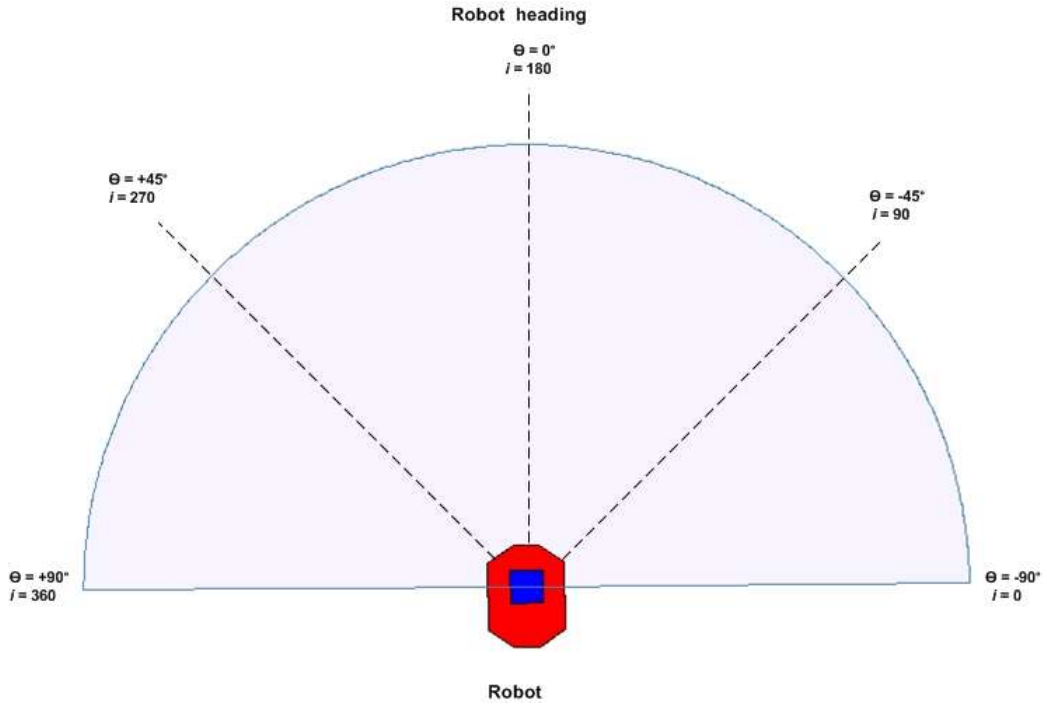
where n = number of clusters.

l_g = laser- smoothness parameter that adjusts turning rate , set l_g =1.

$O_c(i)$ = i_{th} obstacle cluster.

d_o = distance of influence.

The obstacle avoidance behaviour is invoked and is activated whenever the robot encounters any obstacles, as illustrated in Figure 4.15. The only exception that may occur is when the completed behaviour is enabled (see Section 4.5.5).



Note: For the laser ranger finder having a 180° degree field of view and 360 unit resolution , the angle between the robot's heading and the i-*th* unit of angular resolution, $\Theta = (\frac{1}{2} \times i) - 90°$

Figure 4.14: Configuration of the laser range finder

One of the key points related to avoidance behaviour is to localize the position of the robot based on the previous bearing angle after an avoiding manoeuvre, decreasing the consumed time to reach the goal. The localization technique basically attempts to locate the robot with respect to its odometry readings, and to incorporate odometry data into the system. The function is defined as follows:

$$Error = \Delta_{odometry}(\Delta_y, \Delta_\theta) \qquad (4.18)$$

where Δ_y = Position difference along y-axis .

Δ_θ = Bearing angle difference.

After an avoiding movement, the robot steers in the proper direction to decrease the error provided by the $\Delta_{odometry}$ function. However, the odometry readings may become increasingly unreliable over time as errors accumulate and compound. To improve these readings, a simple bearing only measurement technique relying on monocular vision is employed with odometry readings in which the reference image and the current image are compared to enhance the localization accuracy of the robot [Deans, 2005].

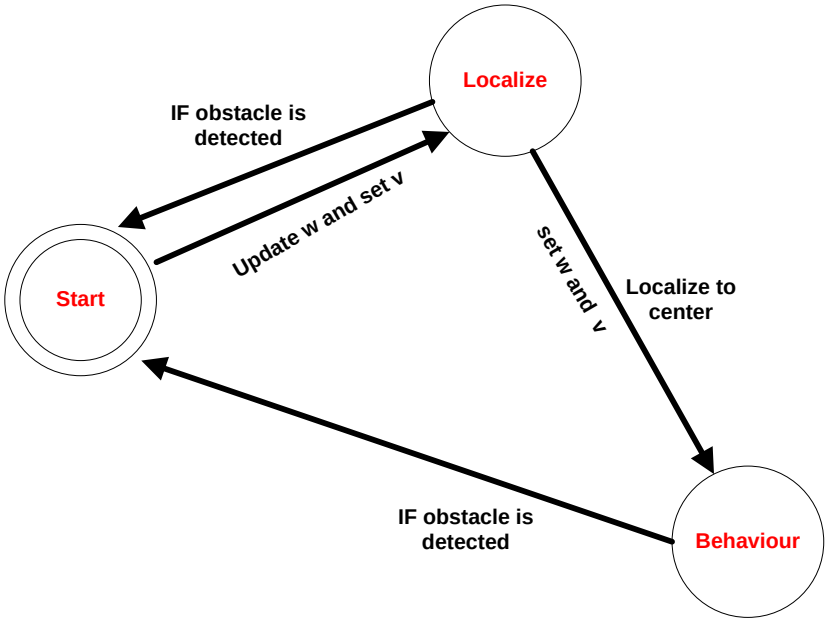


Figure 4.15: State diagram for obstacle avoidance behaviour

The vision based localization algorithm used in this study relies on the SIFT algorithm, where robust localization is performed by fusing odometry data and SIFT matching results. Hence the matching strength between the reference image and the current captured image is compared to the threshold value of the previously performed behaviour during the localization manoeuvre. For instance, if the matching strength obtained by the images is equal to or over this threshold value during the localization manoeuvre, the current state is immediately interrupted and the new heading angle is generated by using Equation 4.13.

On the other hand, if the goal is not detected during the localization manoeuvre, which may be caused by the presence of an obstacle in the robot's environment, it keeps heading in the same direction for a short time instead of activating search behaviour,

which increases the sustainability of the system's performance. An example of obstacle avoidance and localization are demonstrated in Figure 4.16.

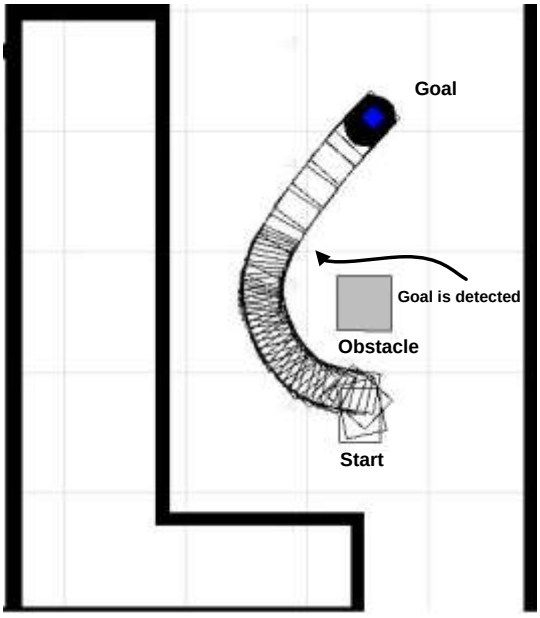


Figure 4.16: An example for obstacle avoidance behaviour

A constant value is assigned to the linear velocity (o_v) during the avoidance and localization procedures to provide the stability. Beside this, angular velocity (o_w) is also maintained constant during the localization procedure until such time as the robot encounters the goal (see Figure 4.16).

4.5.5 Completed

This behaviour is illustrated using a FSM, as shown in Figure 4.17. The objective is to complete the proposed task, which is activated by either goal seeking or approach behaviours based on the strength of the matching value. When the behaviour is activated, the robot continues navigating until its goal is found**,** in a smooth and timely manner by adapting its algorithm. One of the key issues with this behaviour arises when the robot approaches its destination. As the distance between the target and the robot's actual position decreases, obstacle avoidance behaviour may be invoked which steers the robot towards another location. As previously mentioned, the system is designed based on subsumption architecture in which each layer's goal subsumes that of the

underlying layers. Hence, in the case of completed behaviour activation; obstacle avoidance behaviour must be suspended, in order for the robot to reach its goal. The system completes its task when the matching strength exceeds a predefined threshold value. Therefore, the stopping criteria of the system robot can be defined as follows:

$$Fs = \begin{cases} 1, & m > M_c \\ 0, & otherwise \end{cases} \qquad (4.19)$$

where Fs = *final stage, a boolean value*

m = *matching strength*

M_c = *threshold value for matching strength*

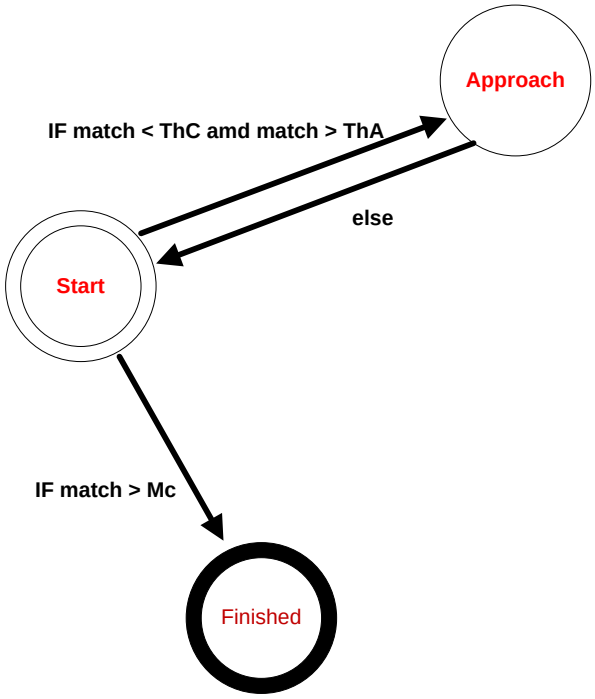


Figure 4.17: State Diagram for completed behaviour

4.6 Prediction of the heading angle

Visual servoing requires the target object to be in the field of view of the camera at all times. At the same time, it is also required that controllability of the robot's pose can be

achieved. However, in real experiments visual based sensing systems can face various problems, such as those resulting from lighting conditions in the environment or vibration caused by the robot's motion. These problems influence the accuracy of the image acquired and the performance of the corresponding feature extraction algorithm. Thus a target prediction system algorithm is employed to overcome these given problems. The system basically maintains the best heading angle direction towards the goal taking into consideration matching strength. According to the algorithm, the best matching results generate the most reliable heading direction. Thus, as the target is lost or the features of the corresponding image are not analysed precisely, the robot makes a manoeuvre in the direction of the previously recorded best position in order to keep the target in the field of view. The position prediction algorithm is as follows:

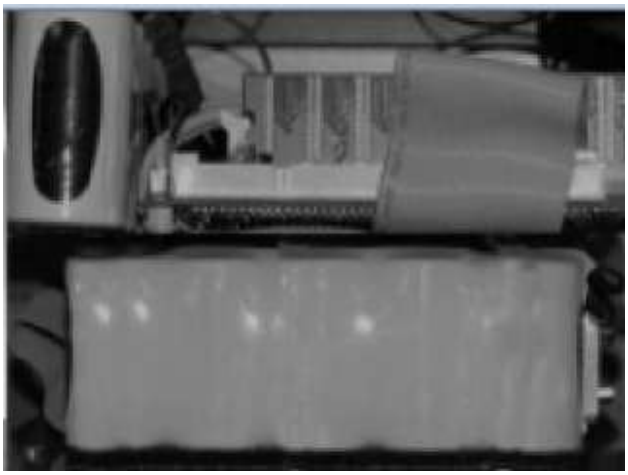
Prediction Algorithm:

Make initial record for the heading direction θ_{best} *using best match strength* (m_{sb})
Until the goal is found
  If the goal is in the field of view
    If the current match strength (m_{sc}) > (m_{sb})
      Replace θ_{best} *with* $\theta_{current}$
  else
      Replace $\theta_{current}$ *with* ($\theta_{current} - \theta_{best}$)
end_until

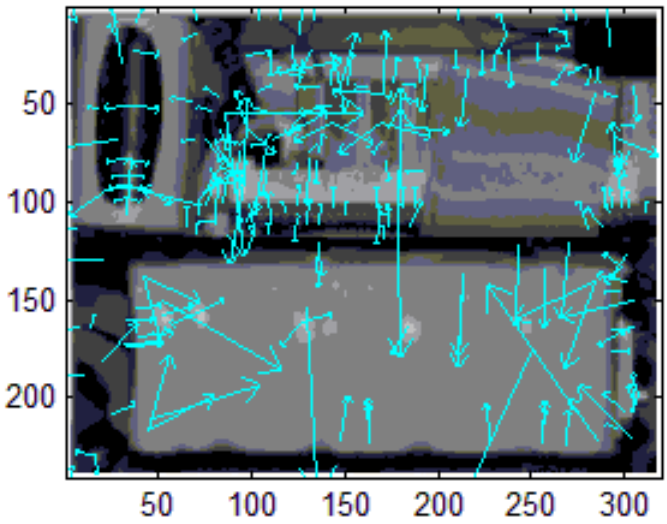
4.7 Modelling and Simulation using Microsoft Robotics Studio

To estimate the capability of the proposed work for expressing useful tasks, the system has been evaluated again in the Microsoft Robotics Studio simulation environment. A number of simulations have been conducted to verify that the Corobot mobile robot is able to navigate in its working environment and achieve its goal without collisions. The simulated mobile robot is equipped with a set of behaviours, namely: goal seeking, approach, wander, obstacle avoidance, completed. The main control algorithm employs SIFT features to navigate the robot towards a specific target. This resembles an image-based visual servoing technique and aims to provide a simple but efficient solution for

vision based mapless navigation problem. The main goal object used in the experiments is shown in Figure 4.18.



(a)



(b)

Figure 4.18: Goal image with 320x240 pixels resolution,
(a) image,(b) SIFT features extracted

Table 4.2: Initialization of the robot control algorithm

Parameters	***Descriptions***
Start position (x,y)	Starting position of the robot in the simulated area
Goal position (x,y)	Position of the goal in the simulated area
Initial heading angle (θ)	Starting heading angle of robot, θ=0^o
Distance of influence of object	d_o = 0.65 m
Maximum velocity	v_{max}= 0.36 m/s
Minimum velocity	v_{min}= 0.08 m/s
Maximum matching value (stop criteria)	45 for reaching goal
Minimum matching value (start criteria)	5 for starting navigation
Velocity constant	k_v =0.008 adjust velocity
Steering constants	s_w = 0.7, s_{w1} = 0.4, s_{w2} = 0.6 adjust steering
Avoidance behaviour parameters	o_w = 12 deg/s ; o_v = 0.1 m/s

To conduct the experiments, the following parameters are defined: the robot's starting position, initial heading and goal position, a set of control equations parameters (s_w ,s_{w1}. s_{w2} ,k_v), maximum and minimum matching values and distance of influence of the object, (d_o), and finally avoidance behaviour control parameters (o_w ,o_v). Table 4.2 presents either a definition or the initial values of aforementioned parameters used in performing the experiments. The parameters for the control algorithm are obtained by a trial and error method, and can be easily modified based on the dynamics or limitations of any specific robot.

Three different test scenarios have been devised to evaluate the performance of the system in the experiments, each of which is conceived with an increased level of complexity, namely:

Scenario 1 (S1): The mobile robot is required to navigate from the 'Start Position' to the 'Goal Position', where the robot is not able to detect the goal at the starting position and a large wall and an external obstacle are located in its path, as shown in Figure 4.19.

Scenario 2 (S2): The mobile robot navigates from the 'Start Position' to the 'Goal Position' (19,-1.0) in a partially cluttered enviorement and has to avoid three unexpected obstacles located in its path, as shown in Figure 4.21.

Scenario 3 (S3) The mobile robot is required to navigate to three different goals successively in a partially cluttered enviorement and to avoid three unexpected obstacles located in its path, as shown in Figure 4.23.

Table 4.3 displays the starting and goal positions in each scenario. The evaluation of each scenario is presented graphically including estimated trajectory, control variables '*w*' and '*v*' and matching strength during navigation.

Table 4.3: Definition of scenarios

Scenario	*Start (x,y)*	*Goal (x,y)*
S1	*(6.5,7.5)*	*(17.0,11.5)*
S2	*(8.0,-1.5)*	*(19.0,-1.0)*
S3	*(7.0,-5.0)*	*(12.0,-5;19.0,8,0;16.0,-16.0)*

Scenario 1 (S1)

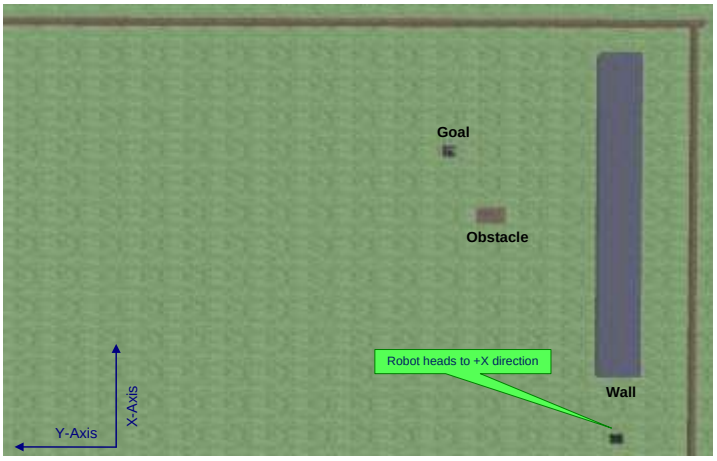
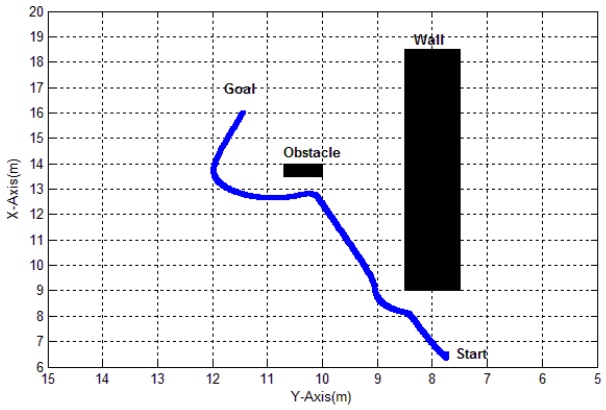
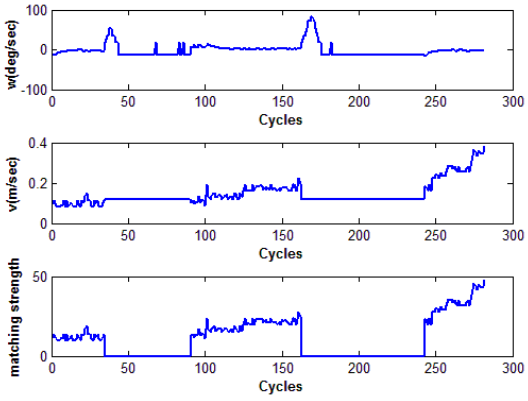


Figure 4.19: Scenario 1, the robot moves towards the goal from its start position



(a)



(b)

Figure 4.20: Scenario 1, (a) estimated trajectory, (b) control parameters

The scenario shown in Figure 4.19, demonstrates the robot's ability to negotiate towards the goal which is located out of the field of view of the robot. In addition, the robot must avoid a large wall and an external obstacle which is positioned so as to obstruct its path. The obstacle is a rectangle with dimensions of 500 mm x 700 mm.

Figure 4.20 (a) presents the estimated trajectory of the robot using the proposed architecture for SC1. The robot initially starts searching for the goal along its initial heading direction. Once the goal is detected, the robot turns 45^{o} towards the goal position and heads towards it. The robot keeps moving until it senses the wall. After this, it manages to avoid the wall safely and carries on approaching the goal until it encounters the obstacle. Once this obstacle has been avoided, the robot again detects the goal and achieves its mission. The corresponding control parameters are illustrated in Figure 4.20 (b), showing the characteristics of the navigation procedure in which the relationships between velocity and matching strength as well as the change in angular velocity due to the obstacles are evident.

Scenario 2 (S2)

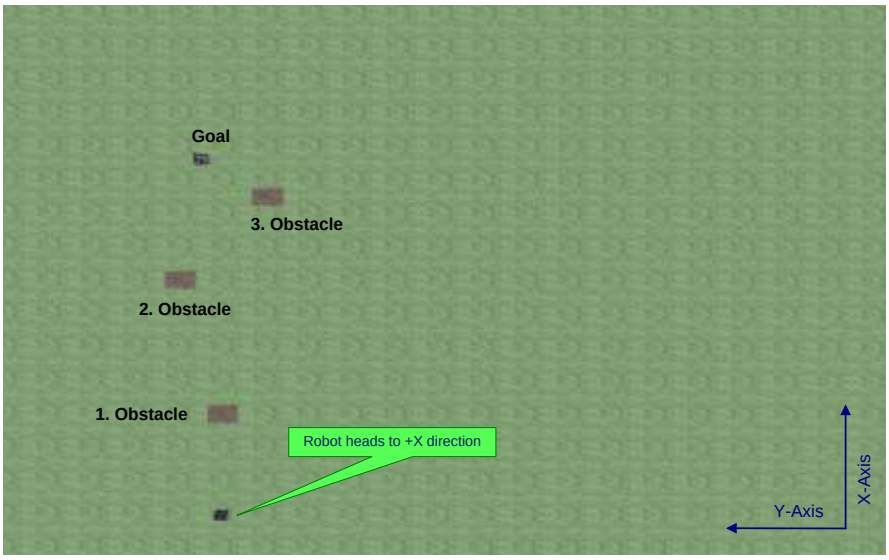
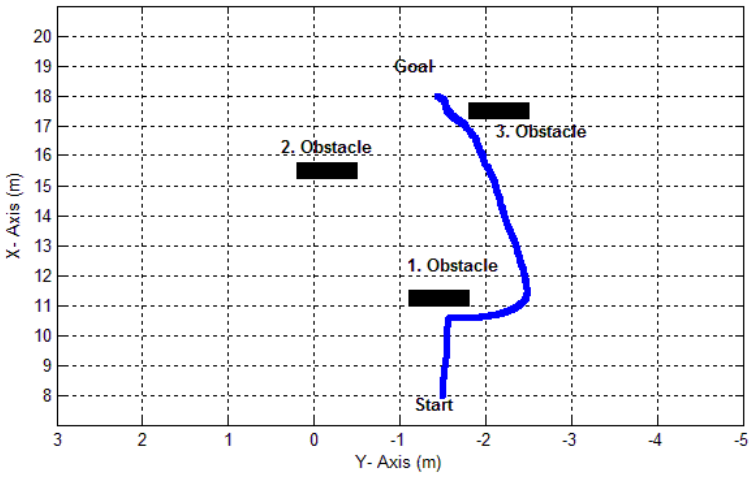
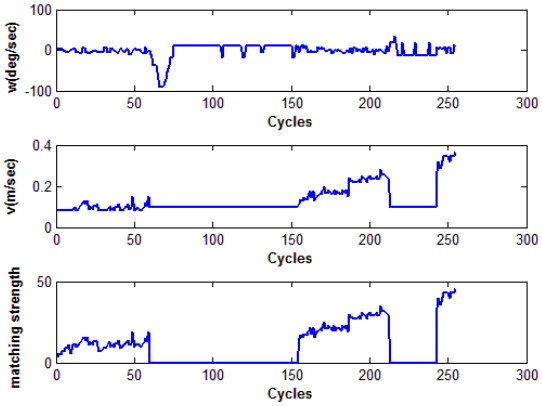


Figure 4.21: Scenario 2, the robot moves towards the goal from its start position

The scenario demonstrates the robot's ability to avoid three rectangular obstacles positioned so as to obstruct its path, as shown in Figure 4.21. The estimated trajectory and the corresponding control parameters for this scenario are illustrated in Figures 4.22 (a) and 4.22 (b) respectively. The robot begins navigation by moving forward until it senses the first obstacle. The robot avoids the obstacle successfully and then localizes itself towards the goal again and continues moving until perceiving the third obstacle. In negotiating this obstacle, the robot avoids it and proceeds to complete the task successfully. The robot does not encounter the second obstacle.



(a)



(b)

Figure 4.22: Scenario 2, (a) estimated trajectory, (b) control parameters

Scenario 3 (S3)

This scenario is designed to evaluate the performance of the proposed algorithms in the case of a global navigation problem in which there are three successive goals. Each of these are obscured by an external obstacle so as to increase the challenge inherent in the scenario, as shown in Figure 4.23. The estimated trajectory and corresponding control parameters for each goal are illustrated in Figure 4.24. The robot moves towards the first goal until it perceives the first obstacle, whereupon the robot avoids it and continues moving towards its goal. It then reaches the goal and completes its first task. After this the robot starts searching for the second goal which is detected after rotating the camera clockwise. The robot then rotates to its right (30^o) to engage with the goal, and starts moving toward 'Goal B' while successfully avoiding the second obstacle. Having reached 'Goal B', the system again enables the search behaviour for the third goal, and then rotates clockwise to its right (70^o) to engage the goal. Once so engaged it heads towards the goal until it perceives the third obstacle in its path which the robot avoids from the obstacle and reaches its final goal.

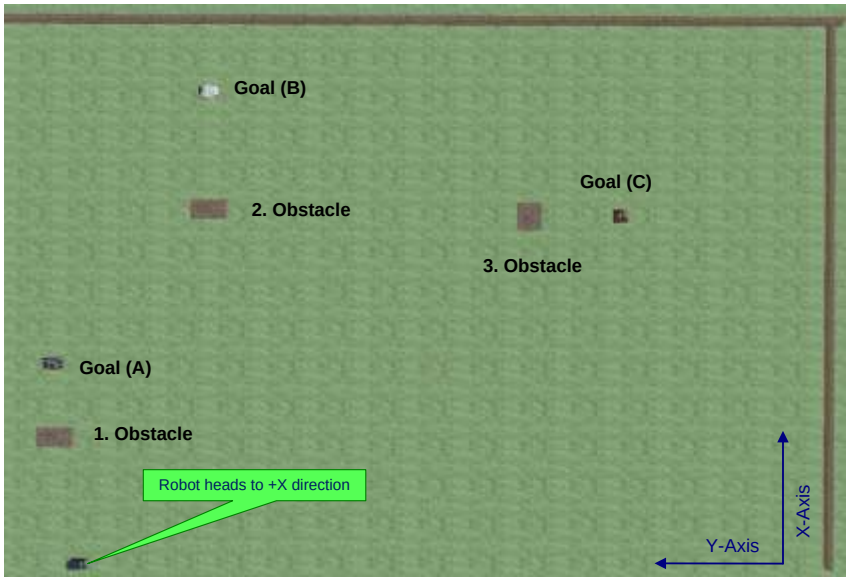
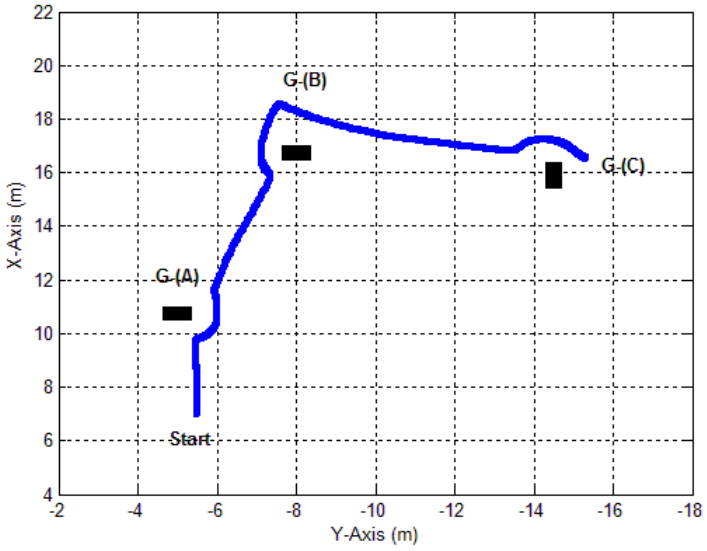


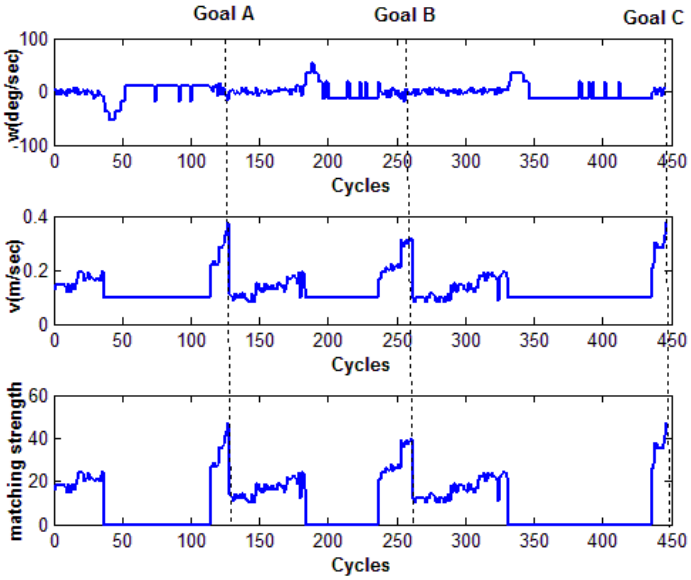
Figure 4.23: Scenario 3, the robot moves towards the multiple goals respectively namely, Goal (A), Goal (B), and Goal (C)

The results of these simulations demonstrate that the proposed system is able to safely navigate the mobile robot to different locations whilst avoiding obstacles within its

working environment. The robot was able to find the locations of the goal, and as well as this it did not collide with any walls or obstacles.



(a)



(b)

Figure 4.24: Scenario 3, (a) estimated trajectory, (b) control parameters

An additional test scenario has also been designed to simulate a trap-situation that the robot may experience when navigating in partially cluttered environments. According to the following scenario, the robot is required to reach a goal which is obscured by a large

obstacle. The robot in essence must pass along a narrow path towards the goal, as illustrated in Figure 4.25. The estimated trajectory and control parameters are illustrated in Figure 4.26. The robot perceives the wall on its left side whilst heading towards the goal, and then avoids the wall successfully. After this it moves towards the wall until it detects the big obstacle obstructing its path. In negotiating this obstacle, the robot becomes confused and is unable to avoid the obstacle. This scenario demonstrates the limitations of the proposed control architecture under the subsumption architecture in conditions that the robot may realistically experience. The problem here is that each layer works independently without consideration the strategic plan.

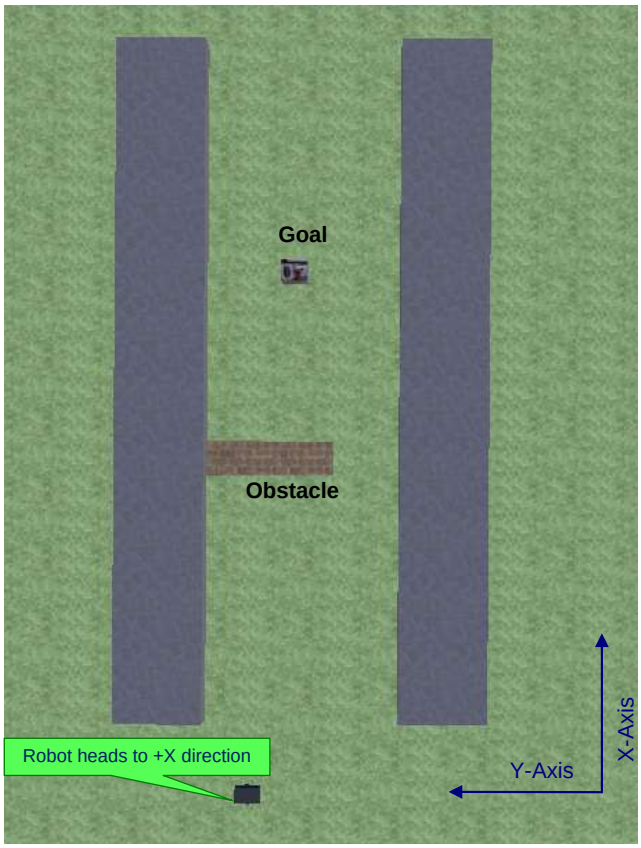
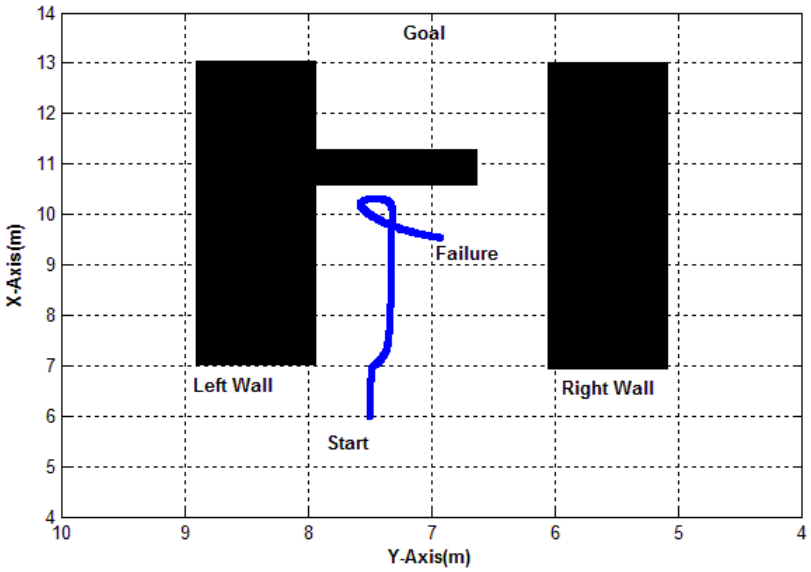


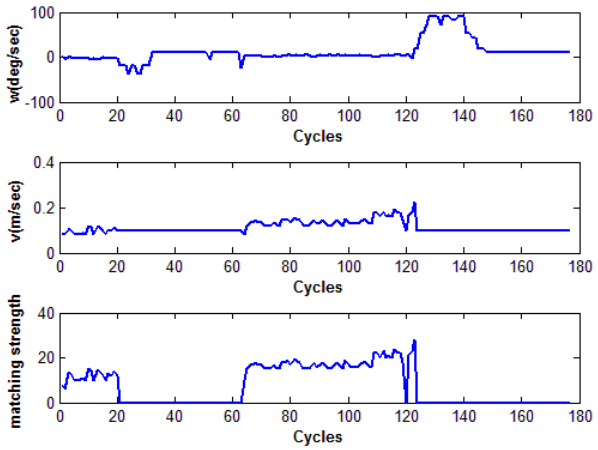
Figure 4.25: Scenario 4, the robot moves towards the goal from its start position

The robot is sometimes trapped in a 'dead-lock' situation, repeating the same reaction many times, or else does not perform rationally based on the shortfalls in the obstacle avoidance and navigation strategies. To overcome this problem in a complex scenario

under a full reactive architecture, the qualitative methods can be altered or amended using artificial intelligence approaches, as will be discussed in the following chapter.



(a)



(b)

Figure 4.26: Scenario 4, (a) estimated trajectory, (b) control parameters

4.8 Summary

An integral part of the objective of this study is to integrate reliable and powerful object detection and feature extraction algorithms into vision based mobile robot navigation systems, whilst relying on a mapless strategy. To achieve this objective, an enhanced

version of the SIFT algorithm, which is able to perform with high accuracy and reasonable processing time, is adapted to a vision based behavioural mobile robot system in order to control the robot. The control variables of the robot are generated based on the position and strength of the matched features. Although, as demonstrated by the test scenarios 1-3, the simulated robot was able to successfully navigate using the Microsoft Robotics Studio simulator, however these experiments do not guarantee that a real robot will perform in the precisely same manner when test are conducted in real conditions, because the simulated robot and the sensors are modelled as ideal. In particular, in real-world lightning and illumination conditions may dramatically decrease the performance of the system. However, the simulation results postulate that the proposed architecture can achieve an acceptable level of performance under real conditions. For the final scenario, the robot was unable to complete its task. This demonstrates the limitations of the proposed system.

The following chapter introduces a novel intelligent navigation strategy which also employs SIFT matching results as the primary input for navigation issue, and is designed based on behaviour based architecture, in an attempt to improve its robustness.

CHAPTER 5

INTELLIGENT NAVIGATION USING SIFT

This chapter describes a novel architecture that employs fuzzy logic and artificial neural networks for vision based mapless mobile robot navigation. Unlike the previously defined architecture, this architecture performs an intelligent solution using the notions of 'feature tracking' and 'visual servoing' to take advantage of scale-invariant features.

The K-Means classification algorithm is applied to matched features to eliminate mismatches as regards scale parameters. This enhancement is a key step which directly increases the overall reliability of the system. In order to predict a robust steering direction toward the goal with respect to the extracted scale-invariant features, an artificial neural network (ANN) technique based on multi-layer perception (MLP) is employed which uses a back propagation learning algorithm. In addition, a technique to adjust the distance of influence parameter using MPL architecture is described, providing safer avoidance manoeuvres. Another important contribution discussed in this chapter is to estimate distance using the scale parameters of scale-invariant features. In addition a fuzzy controller is utilized to estimate the global velocity of the robot.

The first section provides a brief description of the intelligent framework proposed in this study. The next section details the layers of the proposed intelligent framework which enhances the performance of the control architecture, and results are presented concerning how the simulated mobile robot navigates in its environment.

5.1 Design of an Intelligent Framework for Vision Based Mobile Robot Navigation

The details of the proposed intelligent navigation framework are presented in this chapter. This section provides a brief description of the framework and highlights the modules used within it. A flowchart of the proposed architecture is illustrated in Figure 5.1, a central concept of which is to incorporate appropriate soft computing techniques into the vision based navigation problem to allow the robot to move in a more robust and smooth manner. The framework consists of a number of modules which pass data from one to another. The system is equipped with a single pan-tilt-zoom monocular camera which acquires data in image format and a laser range finder which calculates distance in a specific scanning area.

The first module matches the current and template images by employing the SIFT algorithm, and the best candidate match for each keypoint is found by identifying its nearest neighbour in the database of keypoints from training images. The nearest neighbour is defined as the keypoint with a minimum Euclidean distance for the invariant descriptor vector, as described in Section 4.2.5. However, many features of an image will not have correct match, because some of them arise from background clutter and others may not have been detected in the training images [Lowe, 2004]. Therefore, different methods should be applied to discard features that do not have good matches to the database. The popular K-means clustering algorithm is based on the partition of data. In this study, the matched features are enhanced by implementing a pre-processing step, utilizing the K-means clustering algorithm to eliminate mismatches obtained from the SIFT matching module. This module first classifies the matched features into clusters of scale parameters, and then eliminates any mismatches by employing a conventional thresholding technique.

An intelligent method of mobile robot steering control whilst navigating along an unknown path, keeping the goal in the field of view, is developed by combining two different methods. The first of these employs a neural network to estimate the

corresponding turning rate with respect to the location of the matched features, and the second employs a simple linear regression technique to address the problem of calibrating a camera mounted on a robot (detailed in section 5.2). Furthermore, a novel module responsible for estimating the distance from a single monocular camera employs a feed-forward neural network to obtain the distance data. The network has two input nodes for the average of scale parameters and an active zooming factor, and has one output corresponding to the value of physical distance to the goal. This module is mainly used to complete the navigation task when the robot approaches the goal within the tolerance distance, and is also used as an input by the Fuzzy Inference system to estimate the linear velocity. The proposed method is based on the computation of a fuzzy perception of the environment, dealing with the precision of the sensorial system. The Inference system uses three inputs of distance to the goal, distance to the obstacle and matching strength, and offers linear velocity as output. The final module is used to calculate the distance of influence parameter, d_o, employed by the obstacle avoidance behaviour which adapts the algorithm according to a range of conditions. The inputs estimated from the sensors are then all passed to a Behavioural Module which processes information in parallel. Brook's subsumption allows the software designer to determine which lower-level, self-managed behaviours should be subsumed by other higher-level layers of the architecture [Brooks, 1986]. The Behavioural Module comprises five behaviours, and its design was inspired and adapted from the previously defined architecture (see Section 4.5). The only exception was the 'Completed' behaviour that employs the proposed distance estimation method for a stopping criteria instead of matching strength. The output manoeuvre of the robot is determined by this module, after which the Player Architecture block is enabled to provide communication between the high level commands and low level control.

Consequently, the intelligent framework provides several new solutions to the vision based mapless navigation problem. First of all, it enhances the output steering parameter by adopting a feed-forward neural network instead of requiring calculation in a tedious calibration process, with no specialized knowledge of 3D geometry and computer vision being needed. A novel method is proposed to estimate the linear distance using a monocular vision camera, based on scale parameters of extracted interest points. This

method is a new solution to the problem of distance estimation for a monocular vision camera. Furthermore a new fuzzy control system is proposed to estimate and adjust linear velocity depending on the parameters of matching strength, distance to the goal and distance to the obstacle. The adaptive obstacle avoidance behaviour is designed to allow the robot to negotiate narrow paths and navigate in a safer manner.

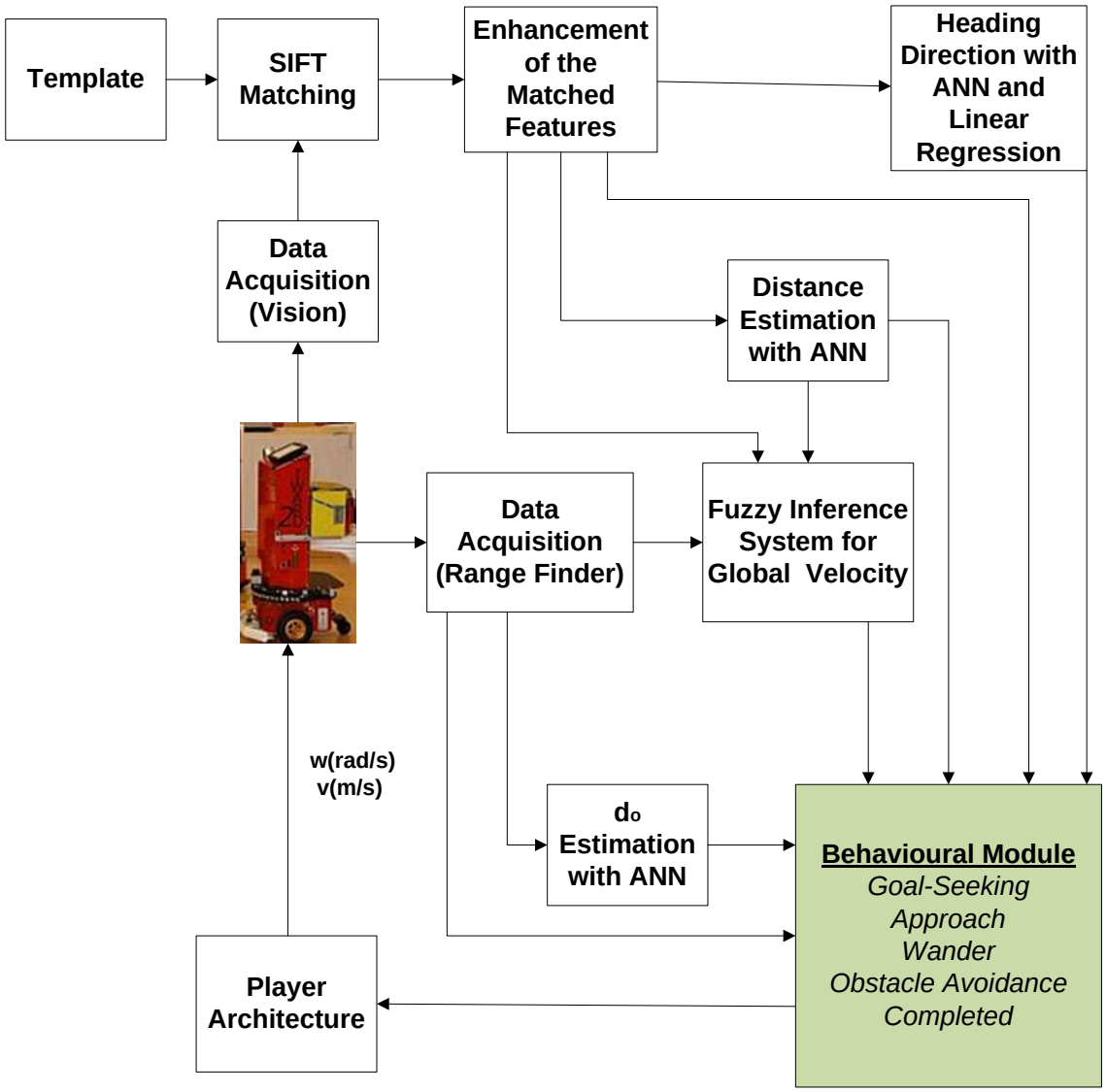


Figure 5.1: Overall system architecture

5.2 Robust Estimation of Heading Direction of a Mobile Robot using ANN and Linear Regression

This section introduces an intelligent estimation of control variables, using the SIFT algorithm for navigation, to overcome the problem of mapless navigation in partially cluttered environments. The vision based navigation algorithm for the proposed architecture is based on two techniques of feature tracking and visual servoing.

In the navigation of a mobile robot, it is essential to accurately determine its location and orientation in order to follow the desired path. Obtaining a precise estimation of the association between the extracted features and the next possible turning manoeuvre is a key challenge especially for monocular vision based system. The conventional route is to calibrate the camera which however, requires solution of several complex mathematical equations [Eric et al., 2007] and very precise camera parameters. Furthermore it is not easy to estimate the value of distance with monocular vision, which is a key requirement for the camera projection matrix. Alternatively, appropriate control laws can be generated using a statistical assessment of the correspondences between matched points. An original control law technique involving this approach was discussed in the previous chapter. Alternatively, the methods may use a teach-replay approach in which the robot is manually led along a desired path in the teaching phase, then the robot autonomously follows that path in the replay phase [Zhichao and Birchfield, 2006]. However, due to lack of an explicit model of the geometric world, its geometric accuracy is limited. In the proposed method, an original algorithm is used which consists of a multilayered neural network to train parameters of matched points. This generates a turning rate control variable without using complex and tedious calibration techniques.

5.2.1 Conventional method for camera calibration

Accurate camera calibration is a key requirement to achieve accurate visual measurements. The relationships between the actual position points and the matched

image points are complex. Accordingly, the camera parameters have to be calculated by a precise imaging model. However, the more precise the imaging model required, the more complicated the calibration becomes. How the camera is calibrated essentially determines the relationship between what appears on the image plane and where it is located in the 3D world. In order to explain the conventional calibration techniques, a pin-hole model is assumed for the camera mounted onto the robot, so that its optical axis is aligned to the robot's forward direction and also parallel to the ground plane [Zhang, 2000]. Conventional camera calibration involve a procedure for determining the internal camera geometric and optical characteristics (intrinsic parameters) and the 3D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters). Further details can be found in Appendix E.

5.2.2 The proposed algorithm to estimate heading direction

Many studies using vision for robot navigation try to build accurate models of the scene, using accurately calibrated systems [Lynch et al., 1999; Cai et al., 2010; Xiong et al., 2010]. However the camera calibration process is complex, sensitive to the calibration errors and may need an explicit model of the environment.

An alternative method has been employed to estimate the heading direction based on neural network. A multi-layered feed-forward neural network has the ability to form complete mapping from a set of input patterns to a set of output patterns. In this study, an original artificial neural network (ANN) design is developed which consists of a multi-layered feed-forward network to overcome the heading direction problem. This eliminates the tedious calibration process and does not require specialized knowledge of 3D geometry and computer vision. A pin-hole model is assumed for the camera which is mounted onto the robot so that its optical axis is aligned to the robot's forward direction and also parallel to the ground plane. A multi-layered feed-forward network is then designed based on scale-invariant features [Lowe, 1999; Lowe, 2004] to provide the association between the 2D image coordinates and the 3D reference (world) coordinates.

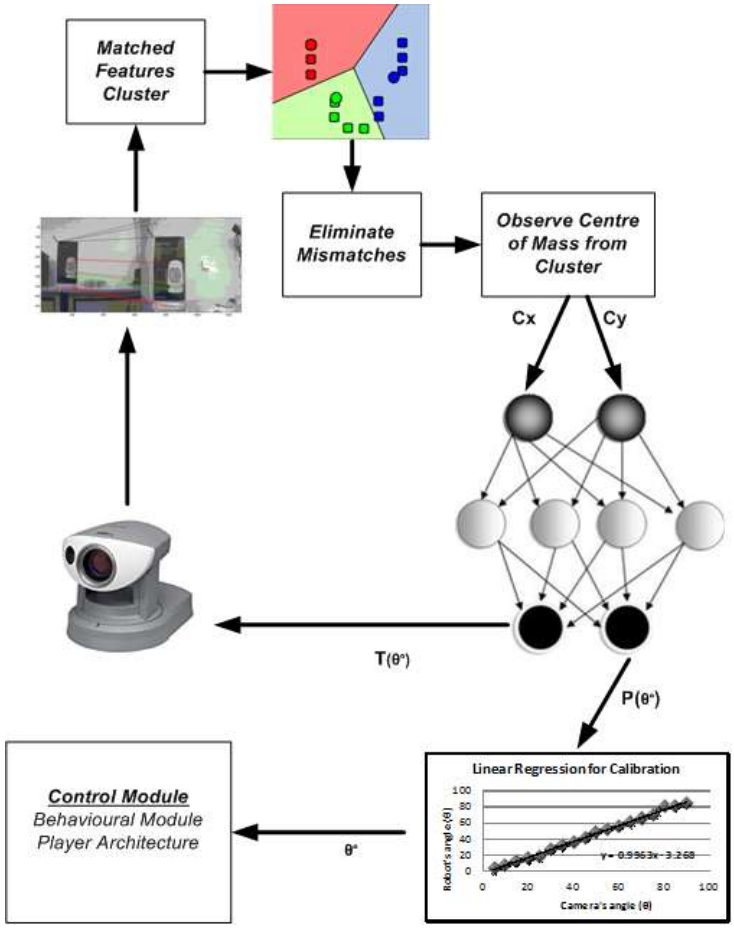


Figure 5.2: Estimation of heading direction using ANN and Linear Regression

The final transformation process between the image workspace and the robot workspace is calculated in two steps. The first step is to conduct the transformation between the camera's coordinate system and robot's coordinate system, which is essentially provided by a simple linear regression technique due to the location of the camera. This essentially performs the transformation between the heading direction of the camera (pan angle) and the robot.

The second step aims to establish the relationship between the image coordinates and the world coordinates which essentially generates appropriate pan and tilt angles regarding the centre of mass of the landmark. Deriving accurate results from the calibration processes and estimation of the precise control variable requires the consistent assessment and enhancement of extracted features, which basically involves removing mismatches, to increase the accuracy of the matching process. The assessment

step with the extracted features on the other hand requires observing an appropriate location from the feature cluster. The overall system architecture has been displayed in Figure 5.1. A flow chart of the heading direction estimation, including ANN design, linear regression analysis and SIFT features, is illustrated in Figure 5.2. The outputs of this system are utilized as inputs by the control module to generate corresponding control parameters.

5.2.2.1 Assessment of scale-invariant features

The first step is to eliminate mismatched features from the matched feature cluster, as shown in Figure 5.3. In order to fulfil this aim, matched points are classified with respect to their scale parameters. The classification of matched features with regard to scale and orientation parameters was first proposed by Lowe [2004], who employed the Hough transform to classify objects in a scene. Nevertheless, the problem is different in the present approach since there is only one object to detect, which illustrates the goal. Accordingly, a simple but efficient classification technique is utilized with respect to scale parameters to overcome the elimination problem. The basic idea is to classify clusters using the K-Means classification algorithm, which is a simple algorithm that has been adapted to many problem domains regarding scale parameters, and then to remove the inconsistent features from these clusters.

For instance, It is assumed that there are n sample feature vectors $x_1, x_2, ..., x_n$ all from the matched class and it is known that they fall into k compact clusters, $k < n$. Let m_i be the mean of the vectors in cluster i. A minimum-distance classifier can be used to separate them. That is, it can be indicated that x is in the i_{th} cluster, if $|| x - m_i ||$ is the minimum of all the k distances. This statement suggests the following algorithm for finding the k means:

K-Means Algorithm:

Make initial guesses for the means $\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_k$
Until there are no changes in any mean
    Use the estimated means to classify the samples into clusters
    For i from 1 to k

Replace $\boldsymbol{m}_i$ *with the mean of all of the samples for cluster i*
end_for
end_until

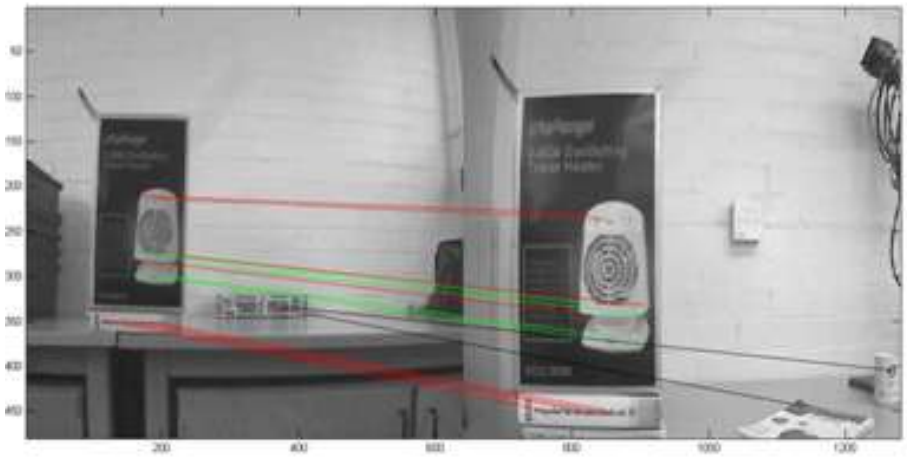


Figure 5.3: Matching example, including different clusters (1_{th} cluster (black), 2_{th} cluster (green), 3_{th} cluster (red))

An example, illustrating how to apply the K-Means algorithm to classify the scale parameters for the SIFT algorithm is demonstrated in Figure 5.3. Figure 5.4 shows the allocation of clusters. The corresponding figures represent the assignment of each feature to the matching sub-clusters based on scale parameters, and each cluster is illustrated using a different colour. The next step is to determine and remove clusters, including any mismatches. In order to estimate these clusters, a simple thresholding technique is used such that if the mean value of any cluster is higher than a predefined threshold value, all members of the corresponding clusters are removed from the matching database.

In addition, clusters having one member are also eliminated. The example, given in Figure 5.3 utilizes '7' as the threshold value. Thus the cluster illustrated in black is removed from the database, eliminating two mismatched features. However, in some situations the selection of an inappropriate threshold value can have adverse effects on the performance of the matching process and eliminating true matches. Despite the possibility of eliminating true matches, this simple conventional classification technique

removes most of the mismatched features, and consequently enhances the overall accuracy of the system.

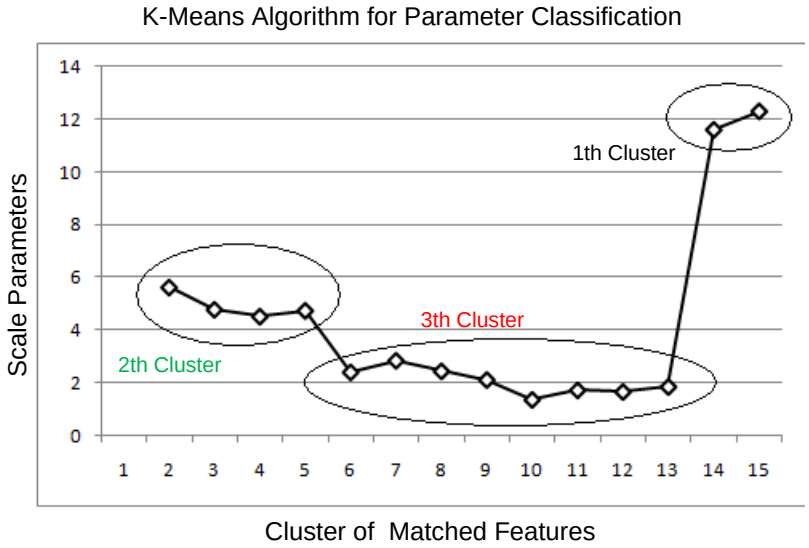


Figure 5.4: Three clusters (black, green, red) are obtained having '10.46', '4.97' and '1.51' mean values respectively

5.2.2.2 The estimation of heading direction using ANN

The purpose of this section is to simplify the otherwise tedious and complex calibration steps using an artificial neural network. Scale-invariant features obtained by the SIFT algorithm are used to automatically detect calibration points. Then a back projection neural algorithm is used to map the relationships between the image coordinates and the world coordinates in terms of pan and tilt angles. The conventional methods to estimate the intrinsic and extrinsic parameters almost all involve uncertainty. Almost all lenses used for image acquisition have some degree of distortion as well as varying focal lengths. The uncertainty about extrinsic parameters is mainly due to the roughness of the floor, geometrical error in fixing the camera on the robot, and the erroneous setup of the landmark. Image deformation in the real environment degrades the accuracy of the analytic estimation algorithm because it depends heavily on the exact location of image data. Therefore, the pinhole model and idealizations may render the solution inadequate; acceptable estimation accuracy cannot be guaranteed in real situation [Koh et al., 1994].

To rectify this problem, several different algorithms have been developed to establish a mathematical and geometrical relationship between the physical 3-D co-ordinates and its corresponding digitized 2-D co-ordinates using ANN [Junghee and Choongwon, 1999; Zou et al., 2005]. The basic idea with these techniques is to detect distinctive features using different image extraction techniques to train a multilayer feed-forward neural network, which is able to approximate any arbitrary continuous function with any desired degree of accuracy. The most extensively used techniques basically extract several feature points, for instance on a checkerboard grid from different angles and distances. The shooting angle, which in simple terms means where the agent stands in order to take the photograph, is determined by the angle between the camera's optical axis and the template plane normal. After this an appropriate network design is constructed, utilizing image points (u,v), and shooting angle (θ) as inputs, and world coordinates (X,Y,Z) as outputs, as illustrated in Figure 5.5.

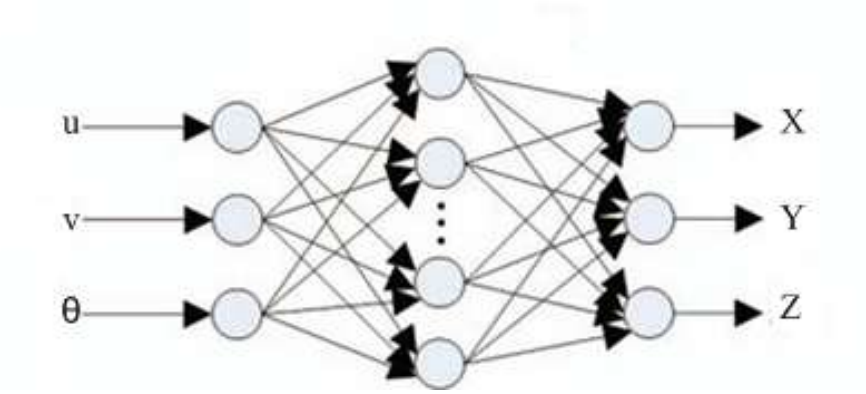


Figure 5.5: The general structure of the neural network for camera calibration

The conventional mode of using an ANN to calibrate any simple monocular vision camera usually relies on a map, including several additional assumptions to facilitate calibration [Koh et al., 1994; DeSouza and Kak, 2002; Zou et al., 2005]. For instance, an accepted assumption is to tilt the camera downwards which in essence keeps the distance of the corresponding landmark constant, as illustrated in Figure 5.6.

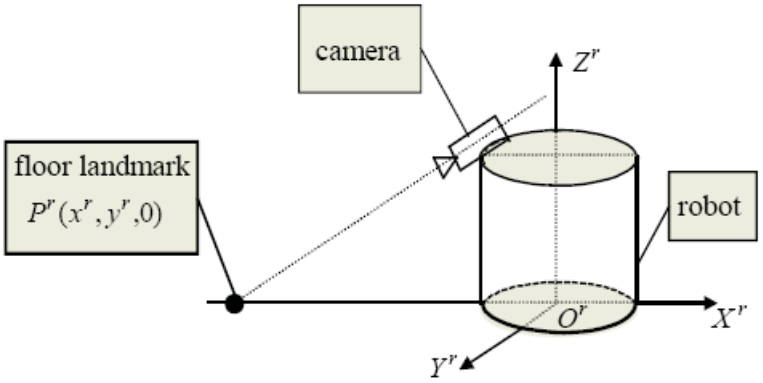


Figure 5.6: The robot coordinate system and a camera axis [Zou et al., 2005]

Nevertheless, the requirements for the proposed system are different from the conventional situations in that the methodology is inspired by a mapless navigation strategy where the exact positions of the landmarks are unknown [DeSouza and Kak, 2002]. In addition, the ANN design is developed for a pan-tilt camera instead of single monocular vision camera. Koh et al. [1994] also propose a method using ANN to calibrate a camera which rotates with two degrees of freedom (pan and tilt), and extracts feature points from the image for landmark tracking. The method, however, entails an artificial landmark and various geometrical assumptions [Koh et al., 1994]. A different method is accordingly proposed to overcome the mapping problem between the physical (3-D) and corresponding digitized two-dimensional (2-D) co-ordinates.

A multi-layered feed-forward neural network is utilized having two input nodes from the image coordinates (u,v). It has two output nodes corresponding to the pan and tilt angles (P_θ,T_θ). The mapping function of the neural network, $N(.)$ can be represented by:

$$M = N(S) \qquad (5.1)$$

where $M = [P_\theta, T_\theta]^T$, $S = [u,v]^T$

Consequently, the technique requires collecting sample output pan and tilt movements in degrees to obtain any given coordinates in the image plane. An example of the proposed network design is illustrated in Figure 5.7. The exact 3-D position of the

camera can be estimated from the output of the network, with minor modifications. This however, is not functional in this application, but alternatively can be used with any map based navigation algorithm.

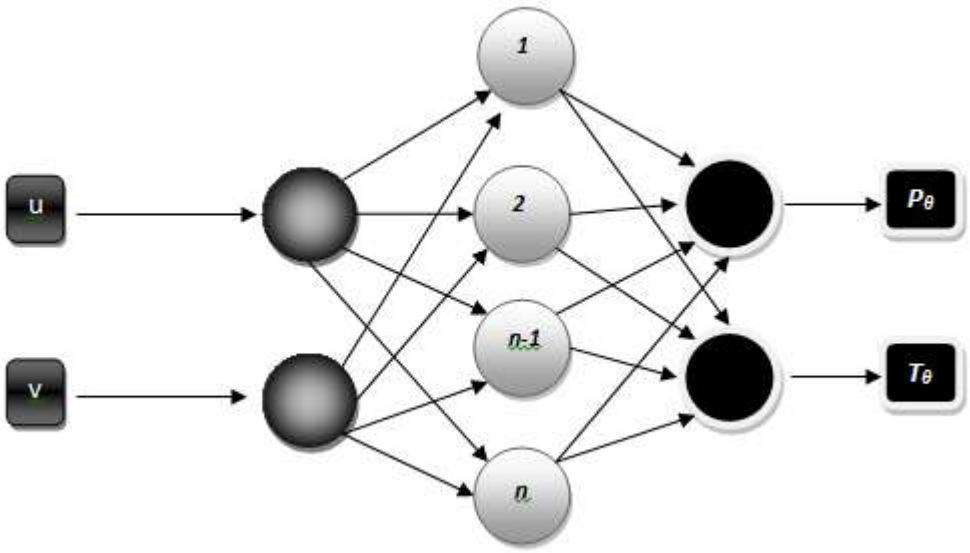


Figure 5.7: The general structure of the proposed neural network design for camera calibration, including one hidden layer with n neurons

The training is performed off-line by using a Levenberg-Marquardt back-propagation algorithm on an error function that is the sum of error squared over the entire training sample [Matlab, 2001]. Details of the algorithm can be found in Appendix D. The sigmoid and linear functions are employed as activation functions for hidden and output neurons respectively. The performance analysis of the proposed ANN design for the Axis-213 camera and the simulated camera are presented in Chapter 6, and two different networks are designed that are as simple as possible to reduce computational time. Table 5.1 demonstrates the basic specifications for the resolutions employed. Training data are presented to the network during training, and the network is adjusted according to its error. Validation values are used to measure network generalization, and to halt training when generalization no longer improves. Testing data do not have any effect at the training stage which provides an independent measure of network performance during and after training. The validation performance of the proposed ANNs for each resolution is given in Chapter 6.

Table 5.1: Specifications of the proposed network topologies

Camera Type	*Resolution*	*Data*	*Topology*	*Train*	*Validation*	*Test*
Axis-213	***176x144***	***155***	***2-6-2***	***125***	***15***	***15***
Simulated	***320x240***	***85***	***2-4-1***	***69***	***8***	***8***

5.2.2.3 Linear regression technique for calibration

Regression is a simple statistical tool used to model the dependence of a variable on one or more explanatory variables. This functional relationship may then be formally stated as an equation, with associated statistical values that describe how well it fits the data, and is used for the transformation between the robot and camera. A simple linear regression equation can be expressed as follows:

$$y = b_0 + b_1 x \tag{5.2}$$

To conduct a regression analysis, coefficients must be solved, as shown below.

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x}).(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$b_0 = \bar{y} - b_1 \bar{x} \tag{5.3}$$

where,

$\bar{x} = mean\ of\ x\ values,$

$\bar{y} = mean\ of\ y\ values.$

It is assumed that the camera is mounted on the robot with its optical axis is aligned to the robot's forward direction and also parallel to the ground plane, as illustrated in Figure 5.6.

The camera's pan axis should ideally sit at the rotation axis of the robot to provide accurate navigation; therefore the linear regression technique is used to provide calibration between the two axes. According to the proposed linear regression model, the independent variable '*x*' represents the estimated pan movement obtained by the camera in order to reach a specific coordinate in the world. Therefore there must be a model that estimates the dependent variable '*y*', the robot's rotation along the Z axis with respect to its pan movement. To estimate the model, scale-invariant features of the environment are extracted and stored in a database throughout the panning movement. The range of pan movement, used in the real experiments is 180° (around the horizontal axis). One of the key issues is to collect an appropriate number of samples from the panning space. Subsequently, the same procedure is applied to the robot, and the robot rotated along its Z axis in order to estimate the best matching with corresponding panning position. Finally, linear regression is applied to all of its recorded positions to generate the association between the camera and the robot. The objective is to find a line going through all of the points. The results of the linear regression between the camera and the robot for different experimental sets, as used in the real experiments, are detailed in Chapter 6.

5.3 Scale Parameters for Distance Estimation Based on ANN

One of the most challenging problems for monocular vision based systems is to determine the distance to the goal. In stereo or trinocular vision spatial information can be derived from the comparison of different images, whereas in monocular vision the analysis can be performed only by studying the fundamental characteristics of an image. This analysis relies on statistical investigations, such as the Time to Contact (TTC) calculation, based on gray level distribution, which was discussed in Chapter 3, or the distribution of RGB channels [Jarvis, 1983; Cantoni et al., 2001].

Optical flow based techniques can alternatively be utilized to calculate depth information from the shapes of the objects positioned in the working space, however the performance of this approach can be poor due to inherent assumptions about knowledge of shapes [Gokturk et al., 2004]. An alternative method is to utilize a defocusing

technique which in essence measures blurred information to determine depth. Buzzi and Guichard [2004] proposed a method employing some of the basic approaches to calculating blur in order to quantify it in an image, and once the 'blurriness' of an object is estimated, depth information can then be easily calculated. However, these types of applications are not appropriate for real time applications [Aslantas and Pham, 2007]. Determining whether or not an object is inside the focal distance can be problematic as they have the same measure of blur, although this can be solved by adopting a system with two cameras of different focal lengths [Buzzi and Guichard, 2004].

Another method uses supervised learning to estimate 1D monocular depth information in unstructured outdoor environments, however, this requires complex processing and training steps [Jeff et al., 2005]. In addition, the SIFT algorithm produces a scale parameter for each key point extracted. For each matched pair of key points in the training and recognition images, the quotient of the keys' scale parameter gives an estimate of their relative apparent size and hence their distance [Sjöö et al., 2009]. This proposal represents a distance estimation technique based on scale parameters, performing a good approximation of distance, but the technique requires the width of the object in the training image to be obtained in pixels. In addition, mismatched key point pairs can produce incorrect scale parameters.

An original and efficient distance estimation technique inspired by these methods has been proposed, which overcomes their drawbacks and also works for any monocular camera equipped with zoom functionality. The technique is based on a multi-layered feed-forward neural network design which has two input nodes: the average of scale parameters (S_m,Z_f) and an active zooming factor. It has one output corresponding to the value of physical distance (d) to the goal in metres. The mapping function of the network can be expressed as follows:

$$D = N(S) \qquad (5.4)$$

where $D = [d]^T$, $S = [S_m, Z_F]^T$

The architecture of the proposed algorithm is illustrated in Figure 5.8. The technique involves a pre-processing step which utilizes a K-means based classification algorithm to remove mismatches. The pre-processing step initially eliminates mismatches, followed by estimating the centre of mass value (S_m) from the enhanced scale parameter space. Training is carried out off-line using the error back-propagation algorithm, and the sigmoid and linear functions are employed as activation functions for hidden and output layers respectively. The general structure of the proposed network design is illustrated in Figure 5.9, and Table 5.2 demonstrates the basic specifications for the networks designed.

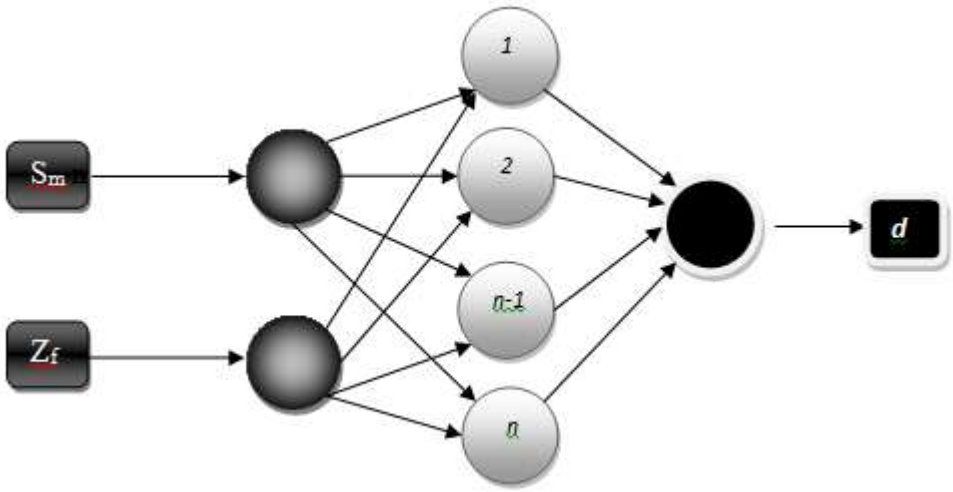


Figure 5.9: General structure of the proposed neural network design for distance estimation, including one hidden layer with n neurons

Table 5.2: Specifications of the proposed network topologies for distance estimation

Camera Type	*Resolution*	*Data*	*Topology*	*Train*	*Validation*	*Test*
Axis-213	***176x144***	***128***	***2-5-1***	***104***	***12***	***12***
Simulated	***320x240***	***128***	***2-4-1***	***104***	***12***	***12***

The maximum estimated distance is 8 metres and the system operates at two different zoom levels. Data samples are collected every 10 cm up to 600 cm and 800 cm for these zoom levels respectively. The minimum distance to the goal is set to 60 cm, which is the distance tolerance parameter (d_t), as discussed in Section 5.7. The AXIS-213 camera has a 26x optical zoom and the rate of magnification of images can easily be

changed with VAPIX, using (5.5) to convert the VAPIX command to the optical zoom value.

$$Optical\ Zoom = \frac{(\mathrm{MOZ}-1)*(\mathrm{VZV}-1)}{(9999-1)} + 1 \qquad (5.5)$$

where MOZ = *maximum optical zoom* value and VZV = VAPIX zoom value.

.

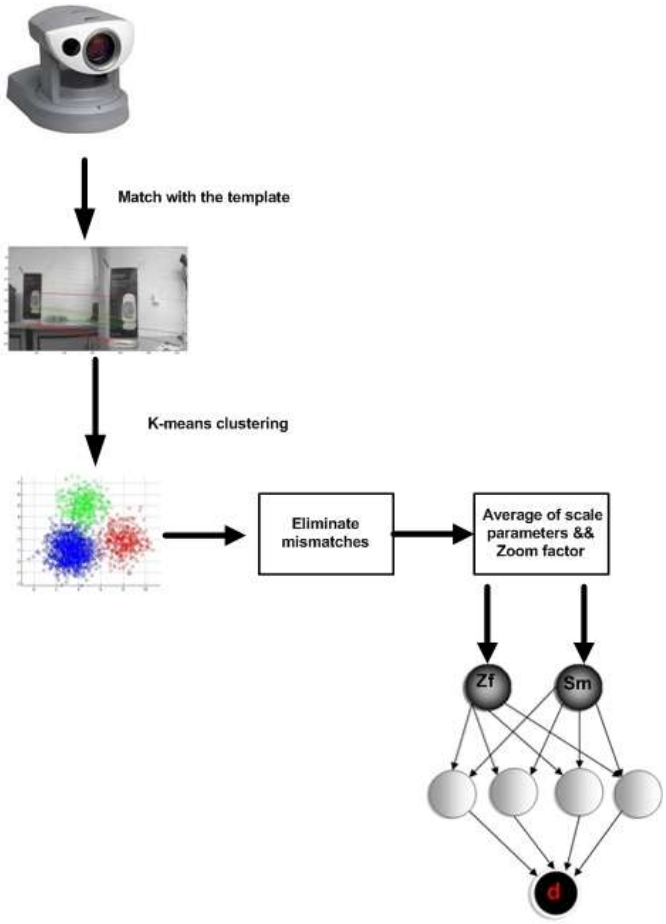


Figure 5.8: The architecture of the proposed distance estimation algorithm

The network is trained using the Levenberg-Marquardt back-propagation algorithm, which is an iterative technique that locates the minimum of a function expressed as the sum of squares of nonlinear functions. It has become a standard technique for nonlinear least-squares problems and can be thought of as a combination of steepest descent and the Gauss-Newton method (see Appendix D). One of the main advantages of neural

networks are their ability to generalize, this means that a trained net can classify data from the same class as the learning data that it has never seen before. In real world problems, only a small proportion of possible patterns is available to generate a neural network. Therefore, to achieve the best generalization, the data set should be split into three parts, namely training, validation and test sets. The learning should be terminated in the minimum of the validation set error which shows the best generalization. If learning is not halted, overtraining can occur and the performance of the overall system decreases.

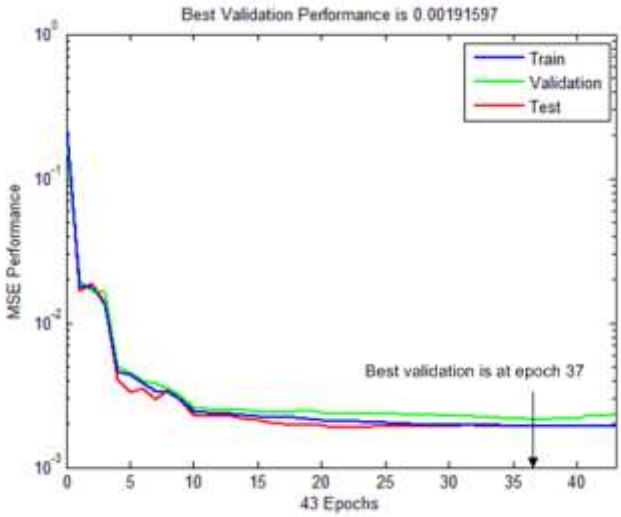


Figure 5.10: Training results for distance estimation

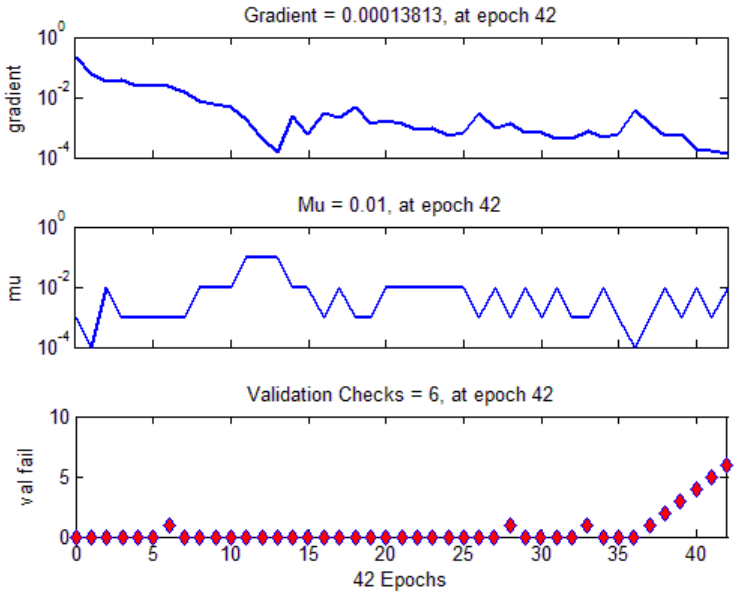


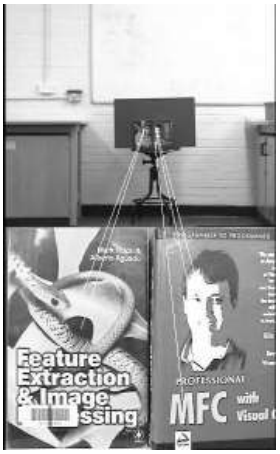
Figure 5.11: Training states of the algorithm with an error of 0.00191597

The corresponding zoom factors employed for the algorithms are *11x (4000)* and *16x (6000)* where the VAPIX equivalents of these values are shown in round brackets. The training analysis of the proposed network is illustrated in Figure 5.10, from which the results indicate that the network approaches the best validation point at the 37_{th} epoch (iteration) with an error of 0.00191597. The training stages of the algorithm, including gradient, adaptive value '*mu'* and the validation checks during epochs, as illustrated in Figure 5.11. The network is trained with this configuration several times, resulting in a standard deviation (σ) of 0.00017. The trained network with the given configuration is able to reliably estimate the distance from a single monocular camera using SIFT features.

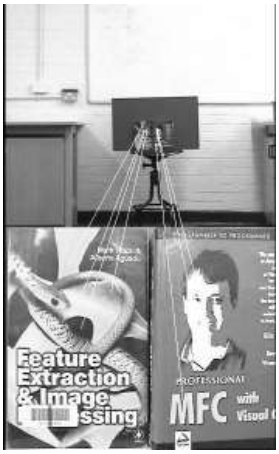
An example from the training data set for distance estimation is illustrated in Figure 5.12, and the corresponding input and output parameters, including zoom level, scale average and the physical distance can be found in Table 5.3. The training results of the simulated camera for calibration and distance estimation are included in Appendix M.

Table 5.3: An example data set for distance estimation

Scale parameters with 11x zoom level		
Frame	***Scale average***	***Physical distance (m)***
(a)	***7.98***	***6.0***
(b)	***7.61***	***5.60***
(c)	***4.47***	***4.0***
(d)	***3.58***	***3.2***
(e)	***1.33***	***0.6***



(a)



(b)

(c)

(d)

(e)

Figure 5.12: Training set for distance estimation via scale parameters, frames (a) to (e)

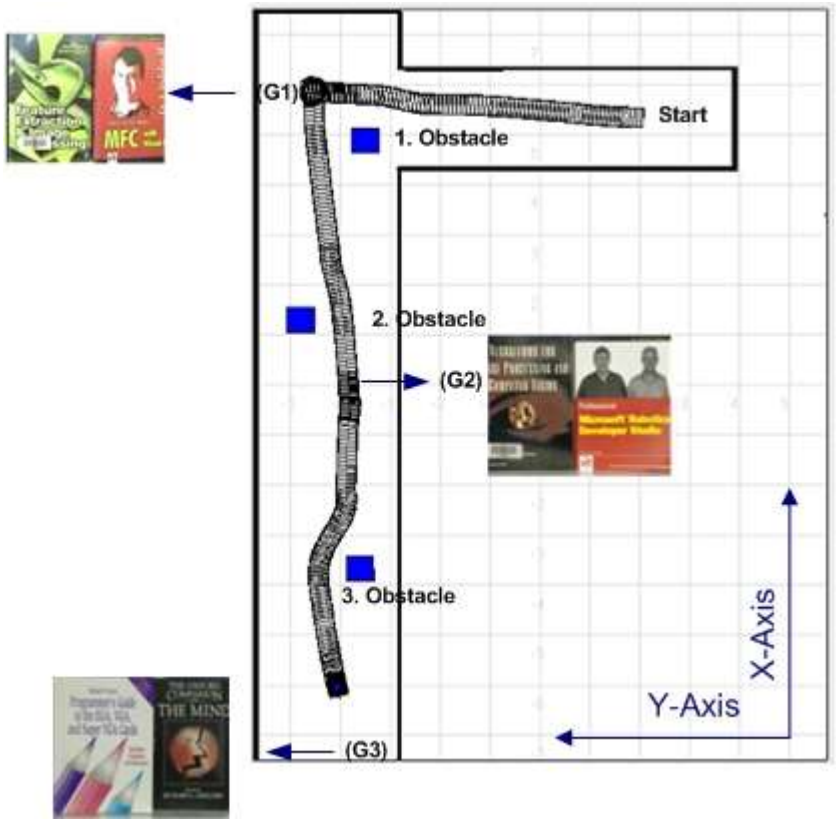


Figure 5.13: Distance estimation in a real navigation scenario

Table 5.4: Distance estimation results from a real experiment

Distance estimation with 16x zoom level	
Goal	***RMSE (m)***
G1	***0.3409***
G2	***0.3822***
G3	***0.7156***

A final example is illustrated in Figure 5.13 to demonstrate the performance of the proposed distance estimation technique in a real navigation problem. Three different goals are located in the robot's path (see Section 6.3.2). The details of this scenario are described in Section 7.3.1.2. Table 5.4 presents the root mean square error (RMSE) between the estimated distance and the actual values obtained for each goal. A lower value of RMSE indicates a better fit of the model. The results demonstrate that goals G1 and G2 produce lower values, as expected, which were used in the training stage of the network. On the other hand, goal G3 results in a reasonable RMSE value, despite not

being used in the training stage. Accordingly, the trained network can work successfully with different goals, providing flexibility and reliability to the proposed method.

To conclude, the proposed distance estimation technique is a robust way of estimating distance from a single monocular camera. This technique is adapted to the Fuzzy Inference system, designed to estimate linear velocity (see Section 5.5), and it is also used in the Behavioural Module of the architecture as detailed in Section 5.6.

5.4 ANN based Approach for Obstacle Avoidance

According to the philosophy of the subsumption architecture, as long as no obstacles are detected, the robot will successfully head towards the goal. If an obstacle is detected, however, *the obstacle avoidance* behaviour is activated which steers the robot away from the obstacle. To overcome the obstacle avoidance problem, simple but efficient obstacle avoidance methods have been employed in the previous chapter (see section 4.5.4). In many situations, this technique works perfectly well, Nevertheless, some drawbacks of this technique have been observed during the evaluation of the proposed architecture.

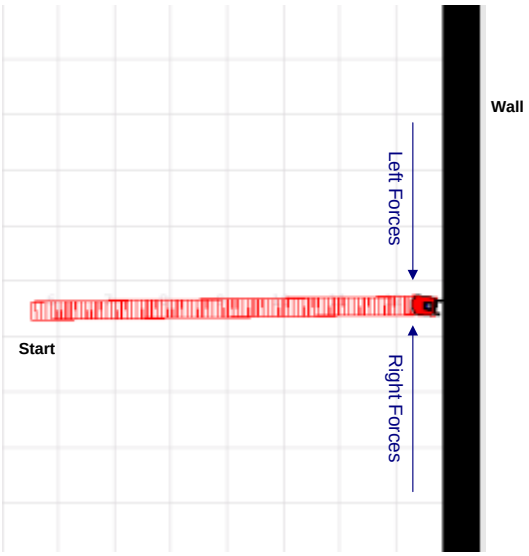


Figure 5.14: The right and left forces cancel each other out and the avoidance algorithm fails

For instance, when the left and right forces which compel the robot to make manoeuvres to the right and left respectively cancel each other out, the robot will probably hit the obstacle, as illustrated in Figure 5.14. This problem resembles the local minima problem in the potential field method, developed as an online collision avoidance approach, which is applicable when the robot does not have a prior model of the obstacle but senses it during motion. It is clear that its reliance on local information can trap it in a local minimum as the repulsive and attractive forces cancel each other out [Nattharith and Bicker, 2009].

Another example is illustrated in Figure 5.15, where the robot is jammed between the obstacle and the wall, and cannot escape through the gap. This is because the proposed algorithm compares each cluster with a distance of influence, and each of these is summed based on the corresponding value of the field of view value, which consequently results in an avoidance manoeuvre. However the algorithm considers only one parameter based on a constant value of distance of influence, d_o , which leads to the method being inadequate for steering the robot along narrow gaps smoothly, as seen in Figure 5.15. To increase the performance of the avoidance manoeuvre, a solution based on artificial neural network is proposed which varying the d_o parameter with respect to different situations. It results in performing smoother avoidance manoeuvres and overcomes the drawbacks mentioned above. To handle these problems, an artificial neural network architecture is designed whose input layer includes the range data cluster obtained from the laser sensor O_c (see Section 4.5.4); The mapping function of the network can be defined as follows:

$$D = N(S) \tag{5.6}$$

where $D = [d_o]^T$, $S = [O_c(2), O_c(3), O_c(4),\ldots., O_c(n)]^T$

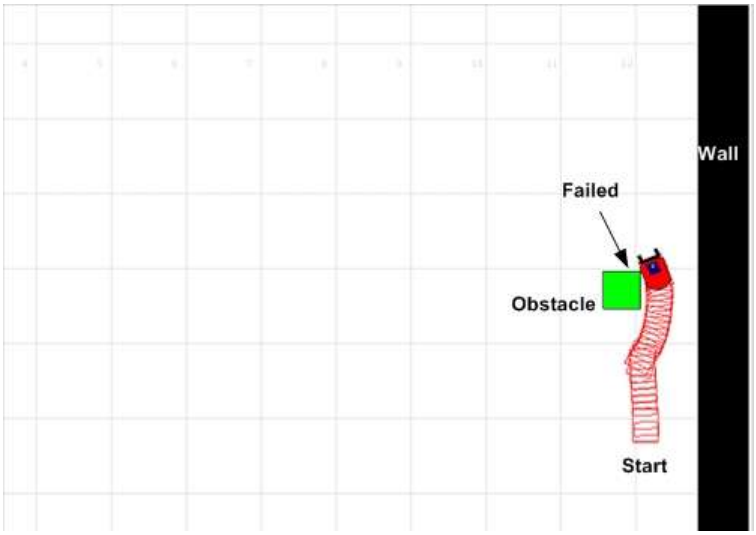


Figure 5.15: The robot is stuck and collides with the obstacle

The proposed neural network utilizes usable accessible space data as an input and providing values of d_o. To simplify the problem, only three different values are assigned to the output of the network The first of these values is the initial distance of influence parameter used for most of the cases. The second is used for trap situations and narrow gabs as illustrated in Figure 5.15, and the final one is for the situation where the left and right forces cancel out each other (see Figure 5.1).

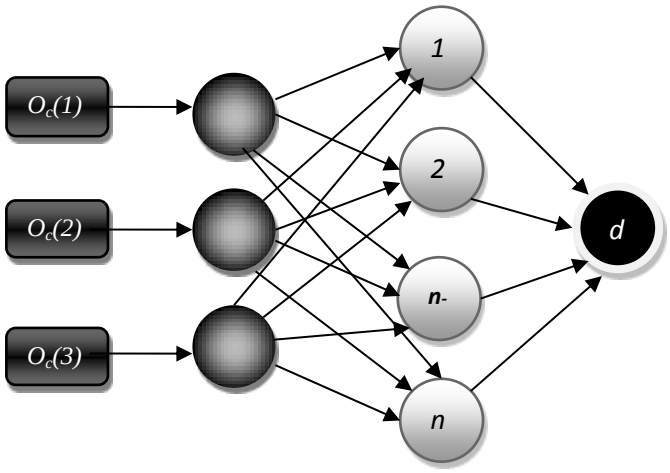


Figure 5.16: The general structure of the proposed neural network design for obstacle avoidance behaviour, $O_c(i)$ represent the range cluster (for simplicity i =3 for this case), including one hidden layer with n neurons

The first and the second values are employed to update the d_o . However the third parameter is utilized to indicate a possible collision with wall or door. Accordingly, when the network yields this output, the obstacle avoidance behaviour is invoked to make a random manoeuvre to either the left or right. Data collection is a key challenge where any human expert may fail or need to spend too much time to overcome data acquisition problems [Janglova, 2004a]. The proposed architecture is constructed on the basis of the previously used simple obstacle avoidance technique (see Section 4.5.4), which performs reasonably well in many situations, and the robot is required to follow a number of predetermined paths to gather data for training. These paths are selected by the designer to simulate the previously mentioned tasks. The movements of the robot are measured and formed as training patterns for each obstacle avoidance sub-task. The robot was made to follow a number of paths classified as general obstacle avoidance situations in order to collect data. The data set was divided into three sets of training, validation and test patterns based on the independent data collected from the different paths.

The performance evaluation results from these experiments are discussed in Chapter 6. The network is trained with the collected data, and the proposed neural network employed in this study is a feed-forward neural network with a back propagation training algorithm, like the previously mentioned architectures (see Sections 5.3.2 and 5.4). The general design of the proposed network is illustrated in Figure 5.16, in which the number of laser range finder cluster is set to 3 instead of 9 so as to simplify the image. Each cluster is responsible for a 20^o field of view. Table 5.5 demonstrates the basic specifications for the network, which was used in both real experiments and simulation tests.

Figure 5.17 presents the results of the proposed artificial obstacle avoidance behaviour when left and right forces cancel each other out. The characteristics of the range data obtained from the range finder with respect to the wall are illustrated in Figure 5.18 (a). The output of the networks triggers random steering direction, which helps to prevent collision. The distribution of data from the sensor after random steering is illustrated in Figure 5.18(b). The re-localization of the robot after the avoidance manoeuvre is

performed with respect to odometry readings and the position of the landmark as discussed previously in Chapter 4. In essence, the robot conducts its localization with regard to the centre following principle and landmark based localization (see 4.5.4). Figure 5.19 presents the output trajectory of the artificial avoidance technique, which generates a safe and smooth avoidance manoeuvre in such a complex situation. This scenario was initially conducted using the reactive method, but it did not navigate the robot safely along the path, as illustrated in Figure 5.15. Due to the narrow path, the robot becomes trapped between the wall and the obstacle, and fails to escape from the trap situation. On the other hand, in the intelligent solution, this path was accepted as a training path from which the corresponding range data was gathered to determine the characteristics of the problem. Accordingly, in the training phase, the human expert manually navigates the robot towards the trap and performs the avoidance manoeuvre. When the robot encounters such a trap, the distance of influence, d_o parameter is immediately reduced by the system, and the robot avoids the trap.

The more the network is trained, the better the results which are obtained. The intelligent method is able to successfully and smoothly accomplish obstacle avoidance problems, as shown in Figures 5.14 and 5.15.

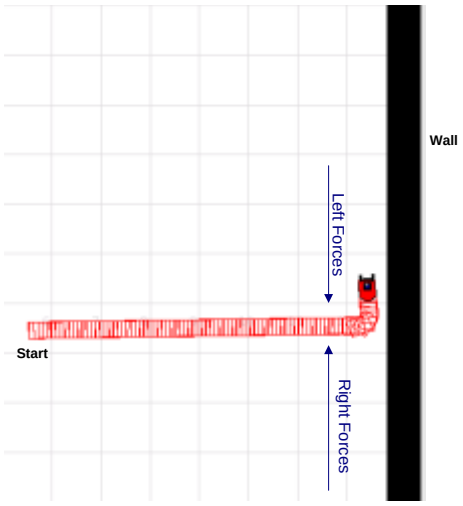
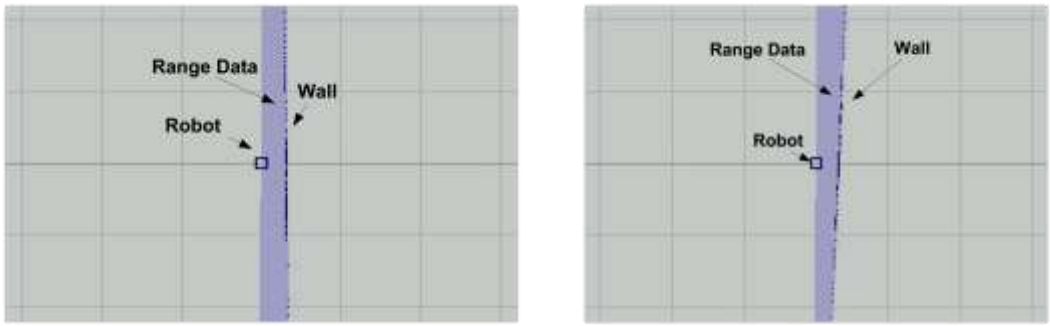


Figure 5.17: The intelligent avoidance manoeuvre succeeds to escape from the trap



(a) (b)

Figure 5.18: Output of range finder, (a) range data for network input , (b) range data just after random steernig to left for the network output

Table 5.5: Specifications of the proposed network topologies for avoidance

Range Finder	*Resolution*	*Data*	*Topology*	*Train*	*Validation*	*Test*
URG-04LX	***180***	***650***	***9-7-1***	***520***	***65***	***65***

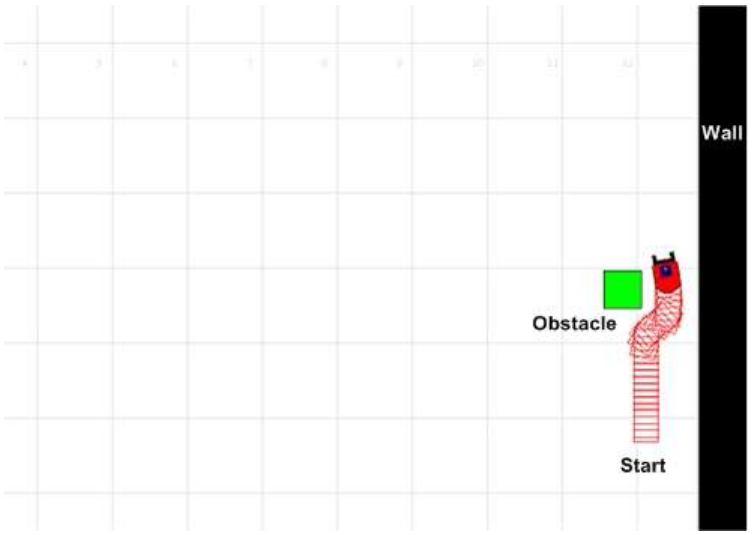


Figure 5.19: The intelligent avoidance algorithm performs a safe and smooth avoiding manoeuvre in a complex scenario

5.5 Estimation of Global Linear Velocity using Fuzzy Logic

The use of fuzzy logic in the design of navigation behaviours and the generation of control parameters for a mobile robot has recently gained a lot of attention from researchers, as these approaches attempt to mimic how humans make decisions [Li and Yang, 2003] .

The problem of estimating the heading angle for goal-based navigation and obstacle avoidance has been discussed in the previous sections. However, the other important control parameter is linear velocity (*v*). Keeping this constant is a well-known method [Brett et al., 2003]. When obstacles are nearby and/or closely aligned with the heading direction, additional speed control helps to avoid collisions or otherwise the robot would have to make sharper turns. In addition, reliable variation in linear velocity during navigation directly affects the overall performance of the system. Accordingly, in order to estimate the instant velocity regarding distance to the goal and obstacles in the environment, a fuzzy based control system is proposed. The fuzzy inference system for this behaviour takes three inputs, i.e. $S^t = (s_1^t, s_2^t, s_3^t)$, and offers linear velocity *(v_t)* as output. The first input, s_1^t, is the estimated distance between the goal and the robot, while s_2^t is the distance from the robot to the closest obstacle. The input s_3^t is the number of matched feature points between the goal and the current image. Based on the common experience of a skilled human operator, it can be shown that a lower value of s_1^t should result in a higher value of v_t, as the robot nears its goal. On the other hand, a higher value of s_2^t assumes a higher value of v_t , since the robot is a safe distance from the obstacle. A higher value of s_3^t implies the higher value higher value of v_t because the robot is close to its goal. The *i-th* fuzzy rule for linear velocity is defined as:

$$If[(s_1^t \; is \; P_a^t) \text{ and} (s_2^t \; is \; Q_b^t) \text{ and } (s_3^t \; is \; R_c^t)] \;\; then \;\; [v_t \text{ is } W_d^t] \qquad (5.7)$$

where,

and denotes the Min operators, and P_a^t , Q_b^t , R_c^t and W_d^t are fuzzy sets defined over the ranges of (s_1,s_2 s_3) and v_t respectively.

5.5.1 Design of membership functions for the linear velocity controller

This section describes the design of the membership functions for the fuzzification and defuzzification processes. Although different researchers have chosen different membership functions depending on the problems encountered in various applications, the trapezoidal and triangular shapes have been employed in this study to simplify the computation. It is important to realize that there are no established methods to adjust the shapes of membership functions. This process requires careful experimentation. In order to achieve this, the ranges of different sensory inputs to the fuzzy sets used were arranged so as to cover all relevant situations and to provide the system with greater flexibility in making the best decision. The trial and error method was used with different combinations of inputs and corresponding outputs are evaluated.

Three membership functions are defined for the velocity control problem, which are detailed in the following paragraphs. The first membership function for the given problem, shown in 5.20, is distance to goal (d_g) which illustrates the distance from the robot in its current position. Three geometric membership functions represent the robot's position to the goal, which are *near*, *midway* and *far*. The distance to the goal is estimated using the scale parameters, as previously mentioned in Section 5.3. Any value within a distance between 2 and 5m is considered to be *midway*. The *near* and *far* membership functions can overlap midway. Thus, the robot can be both *midway* and *near* or both *midway* and *far* from its goal. The overlap allows the use of multiple rules.

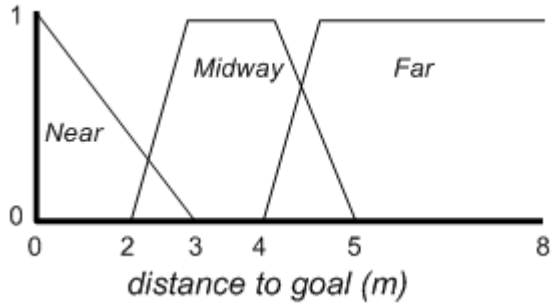


Figure 5.20: The distance to the goal membership functions (d_g)

The distance to the obstacle (d_{obs}) membership function illustrated in Figure 5.21 represents the closest distance from the robot to the obstacles where three membership functions stand for *near*, *midway* or *far* between the robot, and the nearest obstacle. All ranges within the distance of influence of the object (d_o), that is 1.2 m, are taken into account where a distance within 0.6 m is considered to be near.

The matching strength membership function illustrated in Figure 5.22 corresponds to the similarity between the current and the reference (goal) images using three membership functions stand for *small*, *moderate* or, *big*. A value of similarity under 10 is considered to be small. Any value of matching similarity between 5 and 25 is considered to be *moderate*. The *small* and *big* membership functions overlap midway.

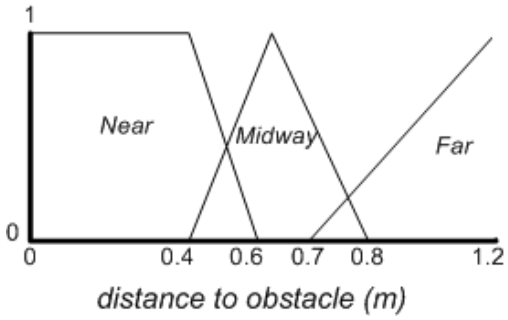


Figure 5.21: The distance to the obstacle membership functions (d_{obs})

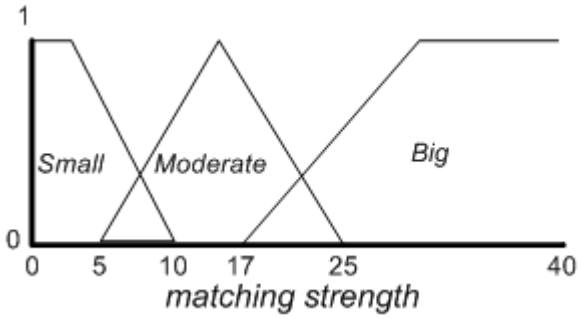
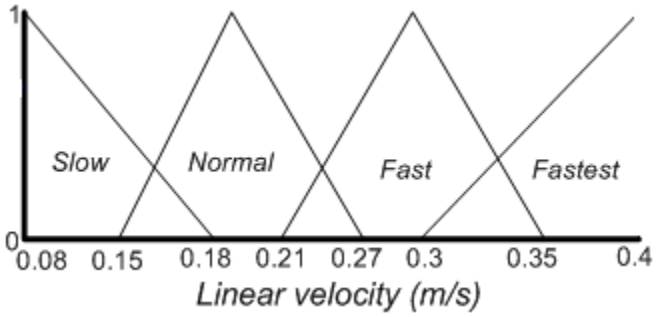


Figure 5.22: Matching strength membership functions (m_{str})

The output membership functions are used to calculate the linear velocity of the robot, which is defined from 0.08 m/s to 0.4 m/s depending on the dynamics of the robot, as illustrated in Figure 5.23. The membership functions are defined as *slow*, *normal*, *fast and fastest*. The next step is to define the appropriate fuzzy rules based on the details of the fuzzy inference system (FIS), a set of fuzzy rules are experimentally developed and adjusted until the outputs are judged to be satisfactory for different situations. Table 5.6 displays the fuzzy rules for the velocity problem.



F**igure 5.23:** Linear velocity membership functions (v_l)

Table 5.6: List of the linear velocity algorithm rules

Rule No	*IF (d_g)*	*AND (d_{obs})*	*AND (m_{str})*	*THEN (v_l) is*
1	*Near*	*Near*	*Small*	*Slow*
2	*Near*	*Near*	*Moderate*	*Slow*
3	*Near*	*Near*	*Big*	*Normal*
4	*Near*	*Midway*	*Small*	*Normal*
5	*Near*	*Midway*	*Moderate*	*Fast*
6	*Near*	*Midway*	*Big*	*Fastest*
7	*Near*	*Far*	*Small*	*Normal*
8	*Near*	*Far*	*Moderate*	*Fastest*
9	*Near*	*Far*	*Big*	*Fast*
10	*Midway*	*Near*	*Small*	*Slow*
11	*Midway*	*Near*	*Moderate*	*Slow*
12	*Midway*	*Near*	*Big*	*Slow*
13	*Midway*	*Midway*	*Small*	*Normal*
14	*Midway*	*Midway*	*Moderate*	*Normal*
15	*Midway*	*Midway*	*Big*	*Normal*
16	*Midway*	*Far*	*Small*	*Normal*
17	*Midway*	*Far*	*Moderate*	*Normal*
18	*Midway*	*Far*	*Big*	*Fast*
19	*Far*	*Near*	*Small*	*Slow*
20	*Far*	*Near*	*Moderate*	*Slow*
21	*Far*	*Near*	*Big*	*Slow*
22	*Far*	*Midway*	*Small*	*Slow*
23	*Far*	*Midway*	*Moderate*	*Slow*
24	*Far*	*Midway*	*Big*	*Normal*
25	*Far*	*Far*	*Small*	*Slow*
26	*Far*	*Far*	*Moderate*	*Slow*
27	*Far*	*Far*	*Big*	*Normal*

5.5.2 Defining the defuzzification method

In the given FIS system, the COS method was chosen to calculate the crisp value which takes into account the influence of every fuzzy rule. Figure 5.24 displays an example where three different rules are activated and result in three different fuzzy sets labelled A, B and C for the output variable, where B and C overlap. Two fuzzy rules have voted for a moderate output, but when applying the COA, MOM and CLA methods this information is lost and unjustified importance is given to a low output. However, COS

is able to overcome this problem. For the given example, the system would expect the moderate output to be the correct decision since two fuzzy rules out of three have voted for it. The COS defuzzification method offers the best results in this case, and for this reason was selected for use in this study. A more detailed comparison of defuzzification methods can be found in [Nattharith, 2010].

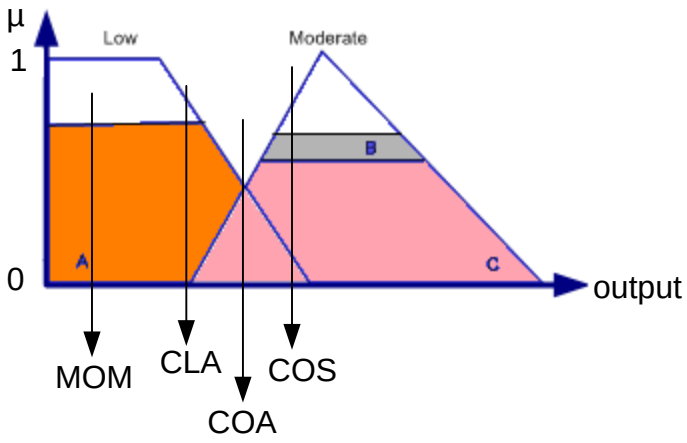


Figure 5.24: Result of different defuzzification methods

5.6 Design of Behaviours Based on Subsumption Architecture

The behavioural architecture consists of five behaviours comprising *goal seeking*, *approach*, *wander*, *obstacle avoidance* and *completed*. The FSM technique is used to visualize the behaviours and their association with other behaviours. The behavioural system is designed to be flexible, as defined in Chapter 4. However, the behaviour (*completed*) is modified and discussed in the following paragraphs. The *completed* behaviour, as previously defined in Section 4.5.5, is based on the matching strength parameter in which the system completes its navigation when this parameter exceeds a predefined threshold value. Nevertheless, this solution depends entirely on the number of correct matched points. This may sometimes be misleading or inadequate so that the robot stops in an inappropriate position. The use of a laser is an alternative method,

because it is a strong sensor in terms of estimating distance, but it must be placed close to the floor so as to detect obstacles. Therefore, if an object is not at that low height, the estimates may be wrong. Consequently, the approach works only for objects that are placed on the floor or are located close to walls (such as, a bookshelf). An example is illustrated in Figure 5.25 [Sjöö et al., 2009].

To address this issue, the matching strength parameter in this work is replaced with a distance parameter based on the scale parameters of SIFT features (see Section 5.4). The stopping criteria for this behaviour can be expressed mathematically as follows:

$$Fs = \begin{cases} 1, & d_{rg} < d_t \\ 0, & otherwise \end{cases} \qquad (5.8)$$

where Fs = *final stage, a boolean value*

d_{rg} = *distance between the robot and the goal*

d_t = *distance tolerance*

The overall system architecture used in the real experiments is illustrated in Figure 5.26 in which control parameters are employed by the behavioural module to steer the robot via Player Architecture.

The algorithm is designed for vehicles in which the only interaction with the motors is carried out using the robot's linear and angular velocities. Thus, the output of the system generates the robot's angular velocity, *w* (deg/sec), and linear velocity, *v* (m/sec). This proposed architecture produces a solution for the image based visual servoing (IBVS) problem in which the control law is based on the error between current and desired features on the image plane, and does not involve the estimation of the 3D pose of the target. On the other hand visual homing strategies, which are a type of visual servoing [Szenher, 2008], estimate direction and/or distance to the goal location in terms of the difference between the current and goal images. The control variables of the proposed algorithm can easily be adapted to a visual homing strategy with minor modifications.

Heading direction θ is also obtained from the output. Thus, the only modification needed is to obtain an appropriate distance for each homing vector which is basically performed by converting global linear velocity v (metres/sec), into distance, d (metres). For instance, if the estimated linear velocity is 0.25 m/sec, the robot moves 0.25 m in the direction of the homing vector $\overrightarrow{H}$.

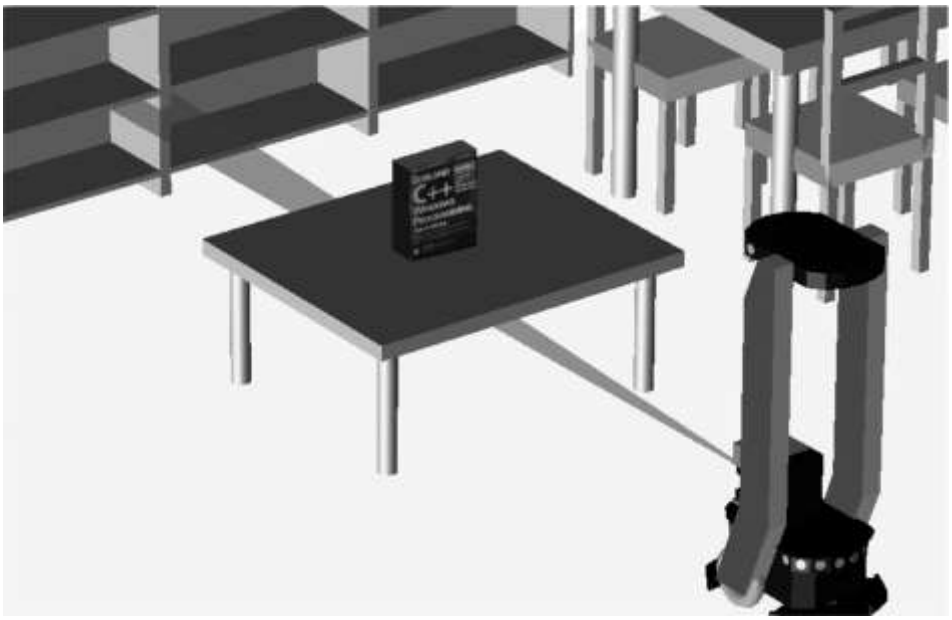


Figure 5.25: Instead of the distance to the object on the table, the distance to the shelf is measured [Sjöö et al., 2009]

5.7 Modelling and Simulation using Microsoft Robotics Studio

In order to estimate the capability of the intelligent navigation system, a series of experiments was conducted to navigate the simulated mobile robot in partially cluttered environments so that it can attain its goal without collisions. In the experiments, a simulated Pioneer mobile robot was used. The details of the robot's configuration were described in Section 4.6. A Matlab simulation was adapted to test the designed neural network architectures as well as to correct the output generated by the C++ coding for the architectures. Additionally, the Matlab Fuzzy Logic Toolbox was used to design the fuzzy logic controller. Once the membership functions have been identified in the

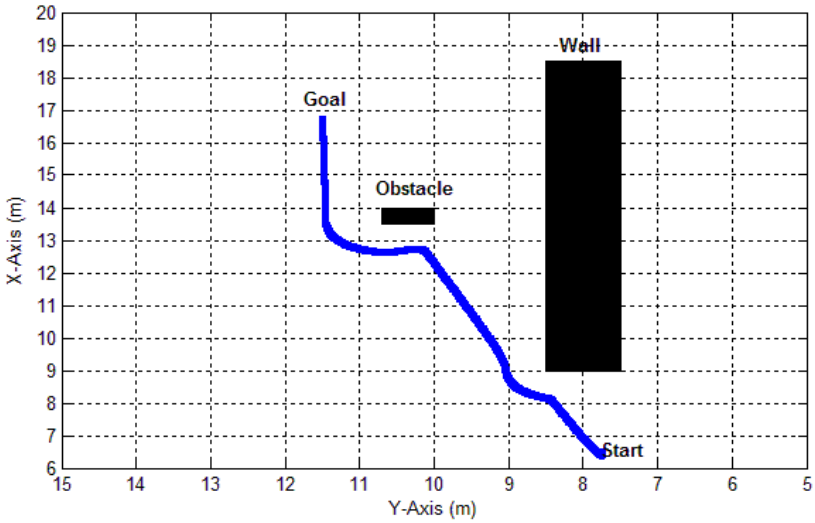
toolbox, the control is simply programmed as a set of linguistic rules relating inputs to outputs.

Table 5.7: Initialization of the robot control algorithm

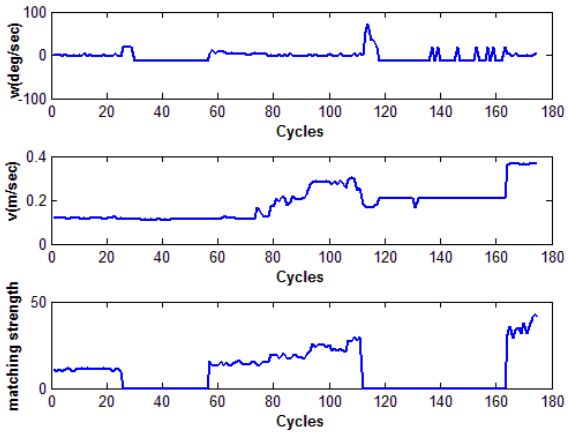
Parameters	***Descriptions***
Start Position (x,y)	Start position of the robot on the simulated area
Goal Position (x,y)	Position of the goal on the simulated area
Initial heading angle (θ)	Starting heading angle of robot, θ=0^o
Distance of influence of object	d_o = 0.5 m
Maximum Velocity	v_{max}= 0.37 m/s
Minimum Velocity	v_{min}= 0.11 m/s
Minimum matching value (Start Criteria)	5 for starting the navigation
Distance Tolerance	d_t= 0.6 m for reaching the goal

In order conduct the experiments and evaluate the proposed intelligent navigation system, the test scenarios used in Chapter 4 have been utilised, defining the initial parameters of the robot's starting and initial heading and goal positions, maximum and minimum matching values, distance of influence of the object, (d_o), and distance tolerance, (d_t).

Table 5.7 presents the definitions and initial values of the parameters used to perform the experiments. The experimental results for each scenario are presented in Figures 5.26-5.29 and these reveal that the proposed architecture enhanced by artificial intelligence safely navigates the robot mobile robot to different locations in its working environment. Figures 5.26 (b) – 5.29 (b) display the control parameters generated by the system for different test scenarios.



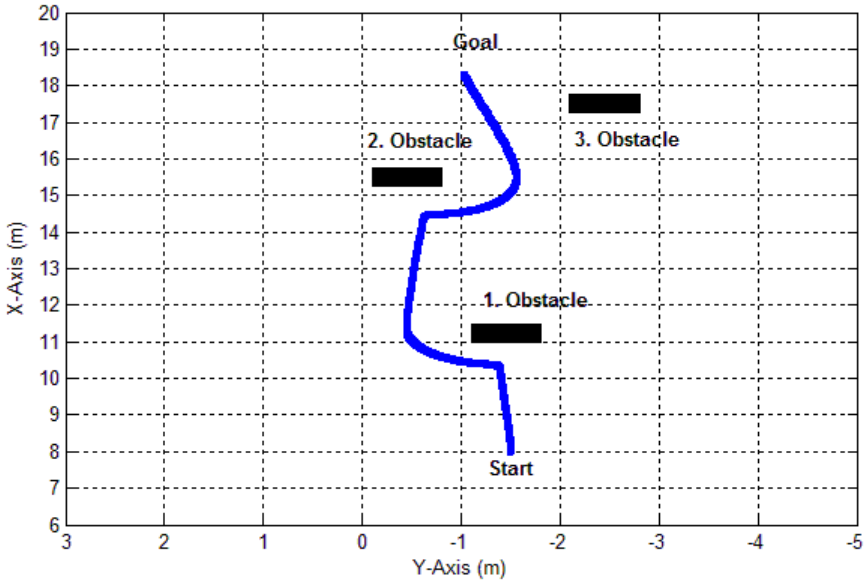
(a)



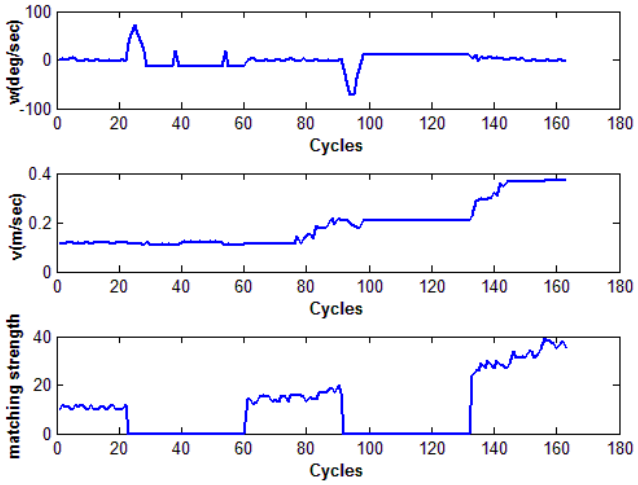
(b)

Figure 5.26: Scenario 1, (a) estimated trajectory, (b) control parameters

Figure 5.26 (a) illustrates the estimated trajectory of the robot employing the intelligent navigation architecture for SC1. The robot starts its navigation by searching for the goal; once it is detected the robot then rotates to its left to engage with it. After which it moves towards the goal until it perceives the wall, whereupon the robot avoids it and continues moving towards its goal. The system accelerates slowly whilst simultaneously decreasing the heading angle until the robot senses the obstacle. It then avoids the obstacle safely, and progressively increases the value of *v* in approaching its goal. This leads the robot to reach its goal successfully



(a)



(b)

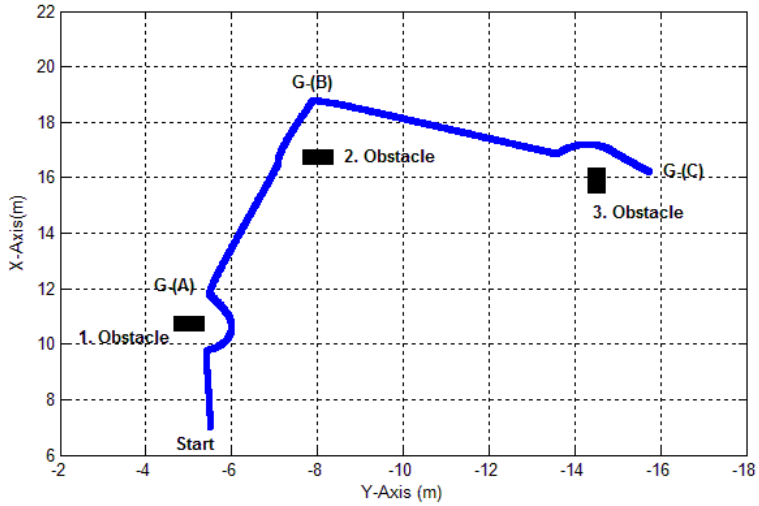
Figure 5.27: Scenario 2, (a) estimated trajectory, (b) control parameters

Figure 5.27 (a) presents the estimated trajectory of the robot for SC2. The robot is required to navigate from its starting point to its goal with obstacles positioned so as to obstruct its path. The robot travels towards its goal until it senses the first obstacle, and maintains *v* at a lower level as the obstacle in front of the robot is approached after which it is able to perform a safe manoeuvre to avoid it. In negotiating the obstacle, the value of *w* is increased progressively as the robot perceives it on its right. The robot

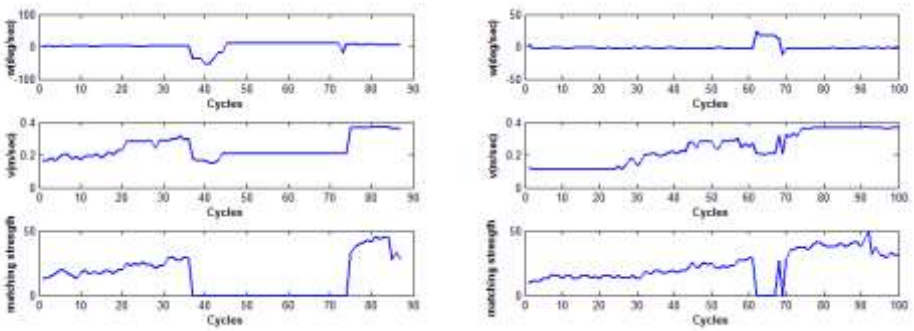
avoids the obstacle successfully and then localizes itself again. In moving toward the goal, the robot senses the second obstacle on its right, which appears within the range of d_o, which invokes the obstacle avoidance behaviour of the robot again. Once the second obstacle is avoided safely and the robot again approaches the goal, *v* is increased whilst *w* decreases. This finally leads the robot successfully to the goal. Unlike in the previous method (see section 4.6), the proposed architecture follows the left-hand path which takes it further away from the third obstacle.

Figure 5.28 (a) displays the navigation results for the final scenario which is designed to evaluate the performance of the architecture with a global navigation problem. The robot begins moving towards its primary goal until it senses the first obstacle. The system then decreases its velocity (*v*) so as to facilitate a safe manoeuvre. After avoiding the first obstacle smoothly, the robot approaches its goal with no obstacle in its field of view. This progressively increases the *v* parameter and directs the robot to safely reach its goal. Once the first goal is accomplished, the robot starts searching for the second goal. It then rotates clockwise to its right (90^o) to engage with the goal. After this the robot begins moving towards the 'Goal B' with an increasing value of *v* and small *w*. The second obstacle is passed smoothly and the robot attains its second goal. In order to negotiate with the final 'Goal C', the robot rotates clockwise to its right (75^o). Once so engaged it heads towards the goal until it senses the third obstacle in its path, which the robot avoids successfully and reaches its final goal.

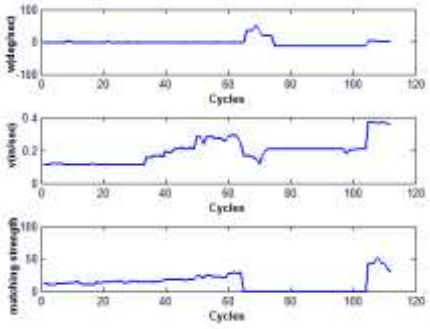
The robot navigates safely and the system increases *v* progressively as the robot approaches the goal and the heading angle *w* remains stable until it perceives the obstacle which, on the other hand, is avoided with a smooth manoeuvre. The robot clearly achieves better navigation results when compared to the test results without using intelligent methods, as shown in Section 4.6. The robot navigated at a closer distance to the goals while its trajectories were smoother without any sharp turns. Furthermore, unlike in the previous method, the robot using the intelligent approach is also able to escape from trap situation, as illustrated in Figure 5.29.



(a)



(b) **(c)**

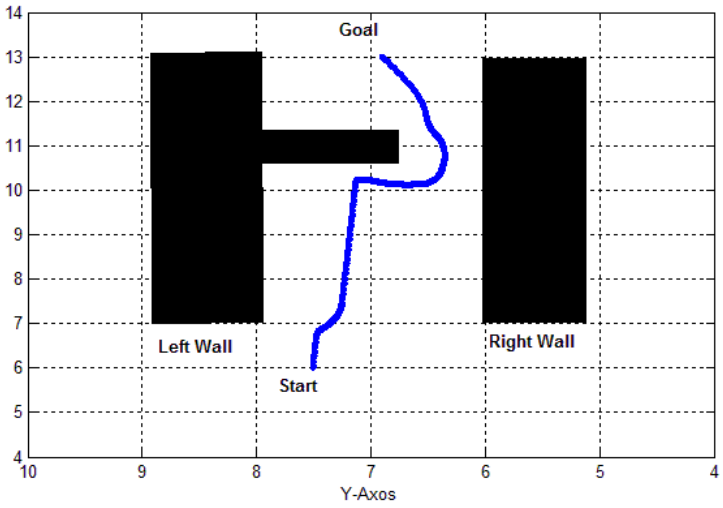


(d)

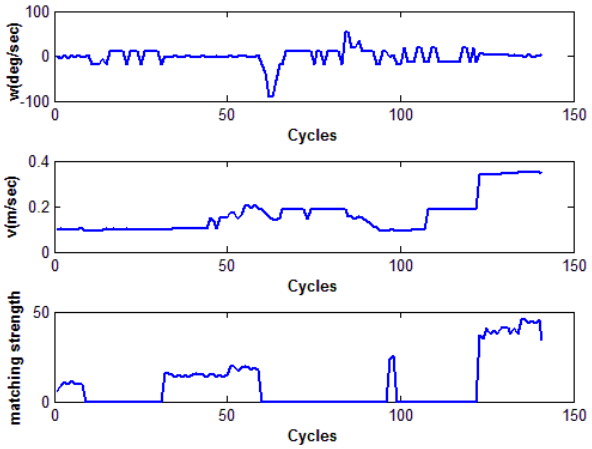
Figure 5.28: Scenario 3, (a) estimated trajectory, (b) control parameters for Goal A, (c) control parameters for Goal B, (d) control parameters for Goal C

The robot begins moving towards the goal until it perceives the wall (on its left) which it avoids successfully and heads towards the goal again. After this it moves towards the wall with increasing *v* and stable *w*. Once it detects the large obstacle obscuring its path,

and unlike in the previous method (see Section 4.6) the system decreases *v* and performs a sharp avoiding manoeuvre, providing rapid adjustment to *w*. It then re-localizes towards the goal whilst negotiating the narrow gap between the wall and the obstacle, which is avoided in a stable manner, as illustrated in Figure 5.29 (a). Finally, the robot approaches its goal and no longer perceives obstacles within a small distance, which leads it to reach its goal successfully.



(a)



(b)

Figure 5.29: Scenario 4, (a) estimated trajectory, (b) control parameters

5.8 Summary

In this chapter, artificial intelligence methods have been integrated with the SIFT based architecture and have demonstrated safe and successful navigation. To achieve this, artificial neural networks are employed to increase the accuracy of turning rates and to estimate distance. As well as this, they are employed to adjust the distance of influence parameter in order to increase overall performance. The proposed architecture also employs a fuzzy logic controller to adjust the linear velocity, which helps more robust path control of the robot. Additionally a simple linear regression technique for camera calibration is integrated to the system, and the *K-means* algorithm is used to eliminate any mismatches provided by the SIFT algorithm. The test results reveal that the proposed architecture is able to direct the robot to successfully complete its tasks. The following chapter describes the configuration of physical mobile robots used to conduct the experiments under real conditions.

CHAPTER 6

ROBOT CONFIGURATION AND SOFTWARE DESIGN

In previous chapters, the proposed navigation architectures have been tested in the Microsoft Robotics Developer Studio programming environment. Several scenarios have been considered in order to evaluate the characteristics and performance of the proposed systems. The aim of this thesis, to conduct novel research into robot navigation using monocular vision, has demanded an experimental system with the capability to support the modes of operation proposed. Therefore, in the remainder of the thesis, all of the navigation algorithms were applied in physical robot and tested under more realistic conditions. The overall hardware and software issues with the robot have to be addressed in real experimental test which are described in this chapter.

6.1 IWARD Project

The mobile robot used in this thesis has been developed as part of the Intelligent Robot Swarm for Attendance, Recognition, and Cleaning and Delivery (IWARD) project [Nattharith, 2010]. The aim of IWARD project is to create a robot team that is capable of fulfilling ward-related operational functionalities in hospitals. The robots operate autonomously in performing their activities but are also able to interact with health care staff through touch screens or by voice. IWARD focuses on the need of hospitals and health care centres to overcome the problem of staff shortages. There are ten partners involved in the IWARD project. The main contribution of Newcastle University to the IWARD project was to design and develop the mobile robot hardware platforms that form the IWARD robot team, which comprises two types: model 3-DX Pioneer robots and one bespoke robot called Nubot, designed and constructed at Newcastle. Further details of the project can be found in [Nattharith, 2010]. Figure 6.1 displays the IWARD mobile robot team. The real experiments were conducted by IWARD Pioneer 1. The configuration and specifications of the robot will be detailed in the following sections.

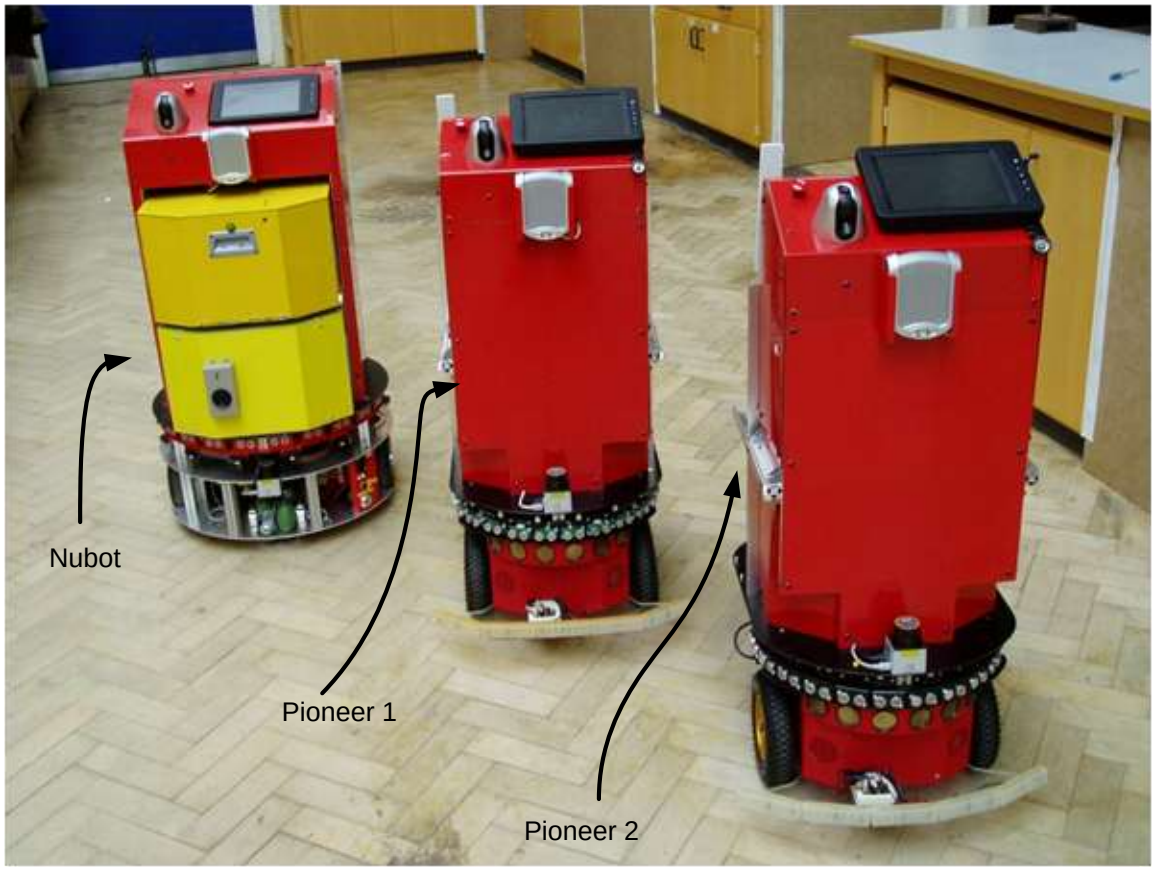


Figure 6.1: IWARD robotic team

6.2 IWARD Pioneer Robot (Pioneer 1)

The Pioneer 3-DX robot is a member of the Pioneer robot family manufactured by ActivMedia Robotics [ActivMedia, 2010]. It has a sturdy aluminum body, a balanced drive system with a two-wheel differential drive with casters, two reversible DC motors, a motor-power control board, an 8 element ultrasonic sensor array, high resolution motion encoders, and battery power, all managed by an onboard SH2 based microcontroller. The robot's maximum unloaded speed is 1.6 m/s and it has a maximum payload capacity of 23 kg including batteries. Additionally, an optional onboard PC (motherboard) is installed for wider and more robust autonomous navigation, and which provides for interactions the microcontroller locally to conduct high level operations. The onboard PC is required for processing data from the laser range finder, ultrasonic

range finder, camera and any additional interface. Figure 6.2 shows the standard version of the Pioneer 3-DX. Further technical details of the robot are given in Appendix F.

Figure 6.2: Standard Pioneer 3-DX

The Pioneer 3-DX robot has an onboard motherboard installed. The robot's control architecture has two levels using the onboard motherboard for communication software and the SH2-based microcontroller for low-level motor control [Nattharith, 2010]. The robot microcontroller is configured as the server in a client-server paradigm, and handles the low-level control such as speed and the acquisition of sensory data. The client software running on the onboard PC provides high level control; the onboard PC receives sensory data from the microcontroller and transmits the motor commands in return. The connection between the onboard PC and the microcontroller is provided by RS-232 standard serial communication. The Pioneer robot is fitted with Versalogic Cobra EBX-12 motherboards. This motherboard is a standard EBX form-factor board with four serial ports, 10/100Base-T Ethernet, monitor, keyboard and mouse ports, two USB ports and support for an IDE hard disk drive [Nattharith, 2010]. It has a Pentium M 1.8 GHz Processor and supports up to 2 GB of system RAM. Additional functionality includes sound video frame grabbing and wireless Ethernet. The main task

of the robot microcontroller is to maintain the independent speed and direction control of the robot motors and to keep track of its absolute positions, whilst maintaining communication with the onboard EBX-12 PC. The SH2 microcontroller is provided with I/O ports for attachment and close integration of the onboard PC and sensors, and supports other accessories. The robot microcontroller is also connected to the motor-power control board, which interfaces with PWM and motor directional command. This also supplies signal paths for standard and accessory onboard electronics.

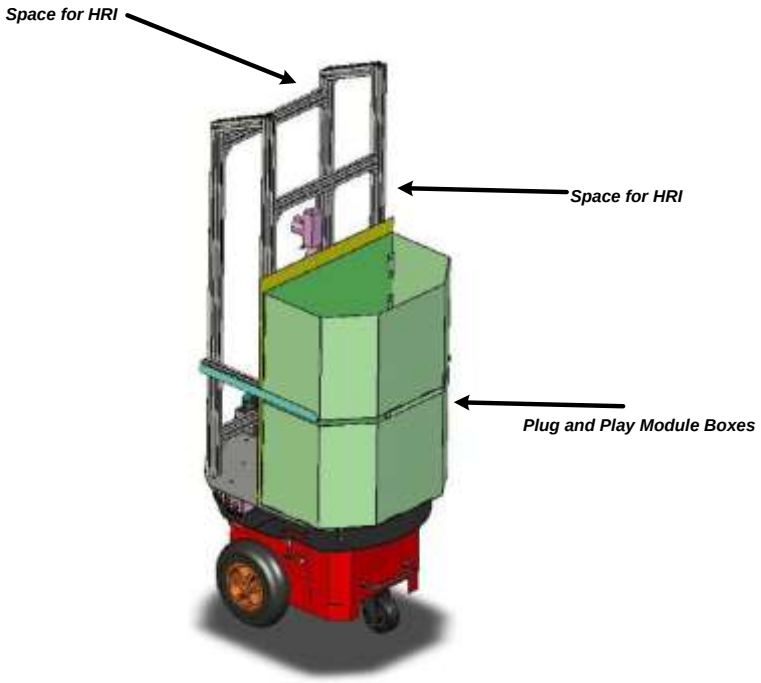


Figure 6.3: Design concept of the Pioneer 1 robot

The top of the Pioneer 1 robot is equipped with a superstructure to accommodate the plug and play module boxes. Flex Link XDBM 3x22 aluminium was used to construct the frame which can accommodate up to two module boxes mounted on the rear of the robot, as shown in Figure 6.3. This superstructure is also designed to support the Human Robot Interface (HRI) panel. The height of the HRI panel is 1000 mm above ground level to allow ease of access. Figure 6.1 displays the current Pioneer robot configurations in which the rear caster wheel was repositioned to provide improved

stability. For this project a platform is designed and attached to the front part of the robot to accommodate the AXIS-213 IP camera, and the design of the platform is provided in Appendix G. The robot and AXIS 213 camera are illustrated in Figure 6.5.

6.2.1 Robot sensors and peripheral device design

To achieve the navigation tasks, the robot is equipped with several sensors and adaptable components. An overview of the sensors used and peripheral devices included in the robot system are introduced in the following section.

Figure 6.4: Hokuyo URG-04LX

Laser range finder: The Hokuyo scanning laser range finder, illustrated in Figure 6.4, is installed on the Pioneer robot. The scanning area is 240 degrees field of view with an angular resolution of 0.36 degrees. It has a detection range from 60 mm to 4 m with a scanning refresh rate of up to 10 Hz. The specifications of the laser range finder are given in Appendix H. It is connected to the robot's onboard PC through a USB (Universal Serial Bus) port. In order to maximise its field of view, the laser is mounted at a height of 310 mm on the front of the robot.

Bumper: The bumper mounted on the front of the Pioneer robot was developed by Cardiff University [IST, 2006]. It essentially provides an emergency stop facility for the

robot if other sensors fail to prevent collision. The robot is halted immediately whenever any of the micro switches on the bumper are activated.

Ultrasonic Sensors: There is an onboard ultrasonic sensor array on the standard Pioneer 3D-X robots, however for the IWARD robots a new sensor array was developed using 16 Devantech SRF05 ultrasonic sensors and 8 Sharp GP2D120 IR sensors. The 180° forward facing field of view of the Pioneer robot was covered by 12 ultrasonic sensors and 8 infrared sensors whilst the rear detection was managed by 4 ultrasonic sensors. The SRF05 sensors, having a range of 30 mm– 4m and are employed for medium range detection, whereas the Sharp GP2D120 IR sensors, having a range of 100-800 mm, are used for the short range detection. Further details about the prototype design of this sensor can be found in [Nattharith, 2010].

Emergency-Stop: There is an Emergency-Stop (E-Stop button) on the robot which overrides deceleration and stops the robot immediately. This button is directly connected to the motor power control board.

Human Robot Interface: The HRI was designed to allow hospital personnel to communicate with the robot. The HRI components include a VGA touch-screen (model LinTX Plus 8.4 inch), and a pan-tilt camera, illustrated in Figure 6.6. Appendix I provides the specifications of the VGA touch-screen. A Digi-Lan pan/tilt network camera is mounted on the HRI panel, and is connected to the motherboard directly using an Ethernet connector. The pan-tilt camera is not used in this study. This is because its specifications are inadequate for navigation purposes.

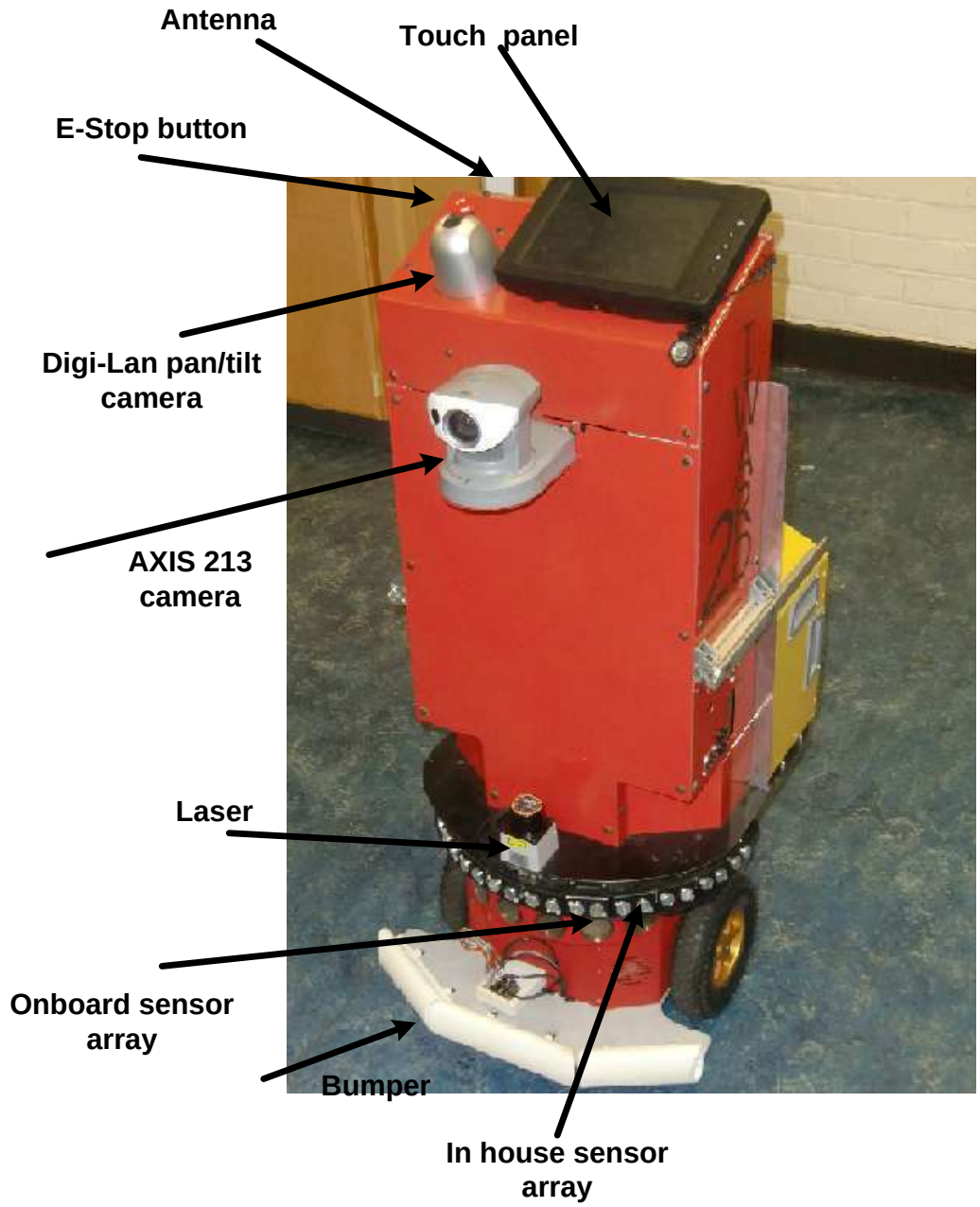


Figure 6.5: The pioneer robot with additional sensors and peripheral devices

Wireless Communication Interface: A remote host computer is required to monitor and control the client software running on the robot onboard PC motherboard The robot onboard PC and the remote computer are connected via a local wireless network, based on the IEEE 802.11g standard, operating in the 5 GHz frequency band with a maximum bit rate of 54 Mbit/s. The robot has a fixed IP address which is *192.168.2.102*. A built-in wireless network interface provides the wireless connection for the remote host

computer, whilst the onboard PC supplies the wireless connection via a PC104+PCMCIA adapter card. A wireless Ethernet card is inserted in the PCMCIA card slot and connected to an antenna which is located behind the HRI panel of the robot (see Figure 6.5).

6.2.1.1 AXIS-213 pan/tilt/zoom camera

The AXIS 213 PTZ Network Camera is a fully featured PTZ network camera for surveillance and remote monitoring, as shown in Figure 6.6. Images from the camera are made available on the network as real-time, full frame rate Motion JPEG streams and/or MPEG-4 video streams. The AXIS 213 also has an infrared (IR) lamp and a removable IR filter for day and night operation.

Video can be viewed in 5 resolutions up to 768x576. Up to 20 viewers can access the AXIS 213 PTZ simultaneously. As the AXIS 213 PTZ is designed for use in security systems, it is equipped with features such as IP address filtering and multilevel passwords. The AXIS 213 PTZ has a built-in Web server, providing full access to all features through the use of a standard Web browser. The camera enables advanced remote monitoring with pan, tilt and zoom through operator control from any PC connected to the local area network or the Internet. It provides wide coverage with its ability to pan 340 degrees, tilt 100 degrees and 26x optical zoom in on specific details. The technical specifications of the camera are shown in Appendix J.

Figure 6.6: AXIS-213 pan/tilt/zoom camera

6.3 Software Design

The robot onboard PC has a Player (Version 2.0.5) distributed architecture, which is based on a client-server structure running under the Linux Operating System and provides control over the physical sensors and actuators in the mobile robot. The player server connects to the robot microcontroller, sensors and actuators. It is necessary that the robot microcontroller must be interfaced with correspondingly. The Pioneer robot utilizes a protocol called the P20S based protocol which is provided by Player [Gerkey et al., 2001a], to establish such an interface. This allows the control of the Pioneer robot with Player. Player supports various robot sensors and components, including sonar, laser cameras and bumpers which developers are allowed to access and implement directly on the Player Server. For the Pioneer robot, both client and server run on the robot onboard PC. The client program provides communication with the Server via a standard Transmission Control Protocol (TCP) socket. All operations of the robot PC are monitored and controlled by a remote computer throughout a wireless network. Details of the Player architecture are given in Chapter 2.

The AXIS 213 is a Network camera which, as previously mentioned, has a built-in Web server that does not need a direct connection to a PC or any other hardware or software to capture and transfer images. It provides a programming interface called VAPIX, an open Application Programming Interface (API) which makes the Axis network video solutions cost efficient, flexible, scalable, future-proof and easy to integrate with other systems.

All Axis network cameras and video servers have an HTTP-based application programming interface. VAPIX provides functionality for requesting images, controlling network camera functions such as pan, tilt, zoom, and setting/retrieving internal parameter values. The purpose of the API is to make it easier for developers to build applications that support Axis video products. For example the ***http://myserver/axis-cgi/jpg/image.cgi*** command requests a default image in JPG format and the ***http://myserver/axis-cgi/com/ptz.cgi?rpan=10*** command is a request to pan the camera to the right by 10 degrees. All commands and parameters can be

accessed in the current VAPIX manual [AXIS, 2011]. The Player architecture does not provide any interface for AXIS-213 cameras. Therefore, C++ based client software called AXIS Camera Control Software (ACCS) has been implemented so as to utilize the camera for the necessary applications. ACCS essentially provides a connection to the server, and conveys requests to the web-server via VAPIX commands. It is a flexible program able to work with different members of the TCP protocol, such as HTTP and FTP. The software provides a parser procedure which basically reads and parses the URL to obtain the appropriate data. For instance, when a request to capture the current image is made, the parser data is saved in JPG format, and can be easily converted into the PGM format required for SIFT applications. The cross flowchart of the corresponding software is illustrated in Figure 6.7.

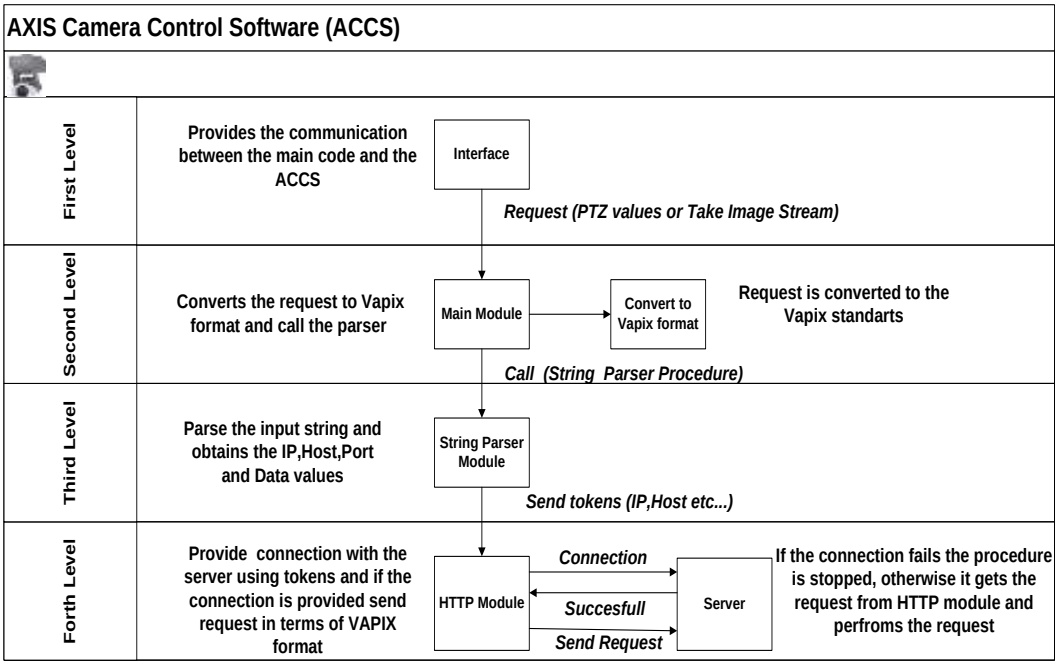


Figure 6.7: Cross flowchart of ACCS software

ACCS is classified into four levels, with each level responsible for different procedures, namely: Interface, Main Module, String Parser Module and HTTP Module, Server. The software is described according to each level as follows.

First Level: This comprises the input procedure called Interface, and essentially provides the communication between the ACCS and the corresponding software which employs it so as to control the camera. The Interface procedure collects the input in String format and passes the corresponding data to the second level.

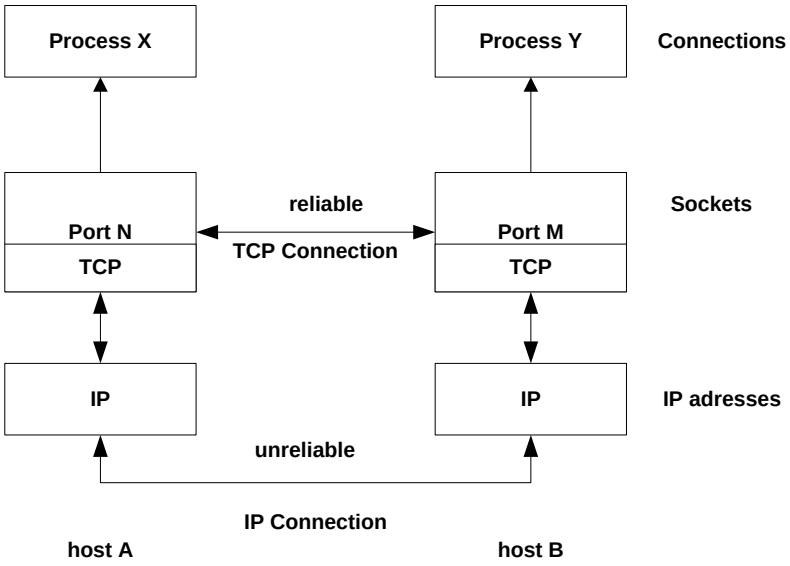


Figure 6.8: TCI-IP connection

Second Level: This is the main module which utilizes the input obtained from the preceding level and converts the input request into VAPIX format. Afterwards it runs the corresponding procedure of the succeeding level.

Third Level: This level performs the parsing procedure which receives the input string in VAPIX format and parses it to obtain the IP address, Host Port and Data values.

Fourth Level: This is main level that carries out the TCP-IP connection with the server located in the Camera. TCP provides a connection oriented, reliable, byte stream service. The term connection-oriented means that the two applications using TCP must establish a TCP connection with each other before they can exchange data. This is a full duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction. TCP includes a flow-control mechanism for each of these byte streams that allows the receiver to limit how much data the sender can transmit.

The basic structure of a TCP-IP connection is illustrated in Figure 6.8. The two processes communicate using TCP sockets where each side of a TCP connection has a socket which can be identified by the pair (IP address, Port number). Two processes communicating over TCP form a logical connection that is uniquely identifiable by the two sockets involved, using the combination (*local_IP_address, local_port, remote_IP_address, and remote port).* Once reliable communication is successfully established, the request is transferred to the camera (host B), and the camera performs the required task.

6.3.1 Software tools and libraries used in this project

This section provides a brief description of the two open source software libraries that have been used in this project in order to facilitate the software design in this project. The CIMG library is a reliable and rapid way of resolving image processing and computer vision issues. The second tool is Open Multi-Processing (OpenMP), which is an application programming interface (API) that supports multi-platform shared memory multiprocessing programs. These libraries are detailed in the following sections.

CIMG library: The CIMG Library is an image processing and computer vision library which is designed for those able to utilize C/C++ programming languages. An example application with its CIMG library is shown in Figure 6.9. The library provides useful classes and functions to load/save, display and process various types of images. It consists of a single header file "*CImg.h*" providing a set of C++ template classes that can be used for specific sources, to load/save, process and display images or list of images. The header file "*CImg.h"* contains all the classes and functions that compose the library itself. The library is very portable, and it is compatible with different operating systems such as Unix/Linux, Windows, and MacOS X. The specifications of this library are listed as follows:

- No pre-compilation of the library is needed, since the compilation of the functions is done at the same time as the compilation of the user' own C++ code.
- No complex dependencies have to be handled; it just includes the "*CImg.h"* file.
- The compilation is accomplished on the fly, which means that only functionalities used by the user program are compiled and appear in the executable program. This leads to very compact code, without any unused components.
- Class members and functions are inline, leading to better performance during the execution of the program [Tschumperlé et al., 1999].

An inline function requests the compiler has been requested to perform inline expansion. In other words, the program requests the compiler to insert the complete body of the function in every place that the function is called, rather than generating code to call the function only in the one place it is defined [Tschumperlé et al., 1999].

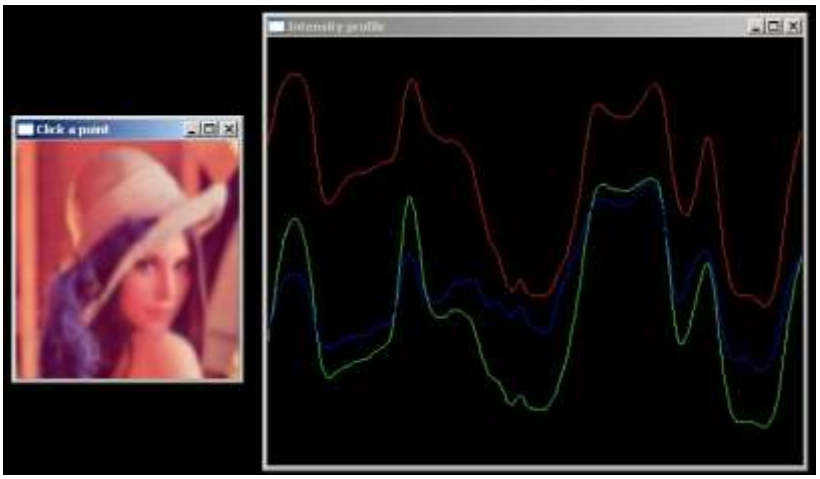


Figure 6.9: An example application with CIMG library

All library classes and functions are defined in the namespace "*cimg_library"*, which namespace encapsulates the library's functionalities and avoids any class name collisions that can happen with other header files. The CIMG library is utilized during this project to save and load images and to enhance the images with low-pass filters. It also provides appropriate functions to access and update coloured images which are used in the qualitative, appearance-based techniques see (Chapters 2 and Chapter 3).

Figure 6.10: Image registration algorithm, with multi-scale capability

The library also includes a function that provides multi-scale versions of the image registration algorithm. An example of a code provided by the library which also demonstrates a multi-scale optical flow algorithm modified and employed as a template for the Horn-Schunk multi scale algorithm (described in Chapter 3). An example of a screenshot of the corresponding multi scale algorithm with the Horn-Schunk method is illustrated in Figure 6.10, in which the flow vectors demonstrate the motion difference between two successive images.

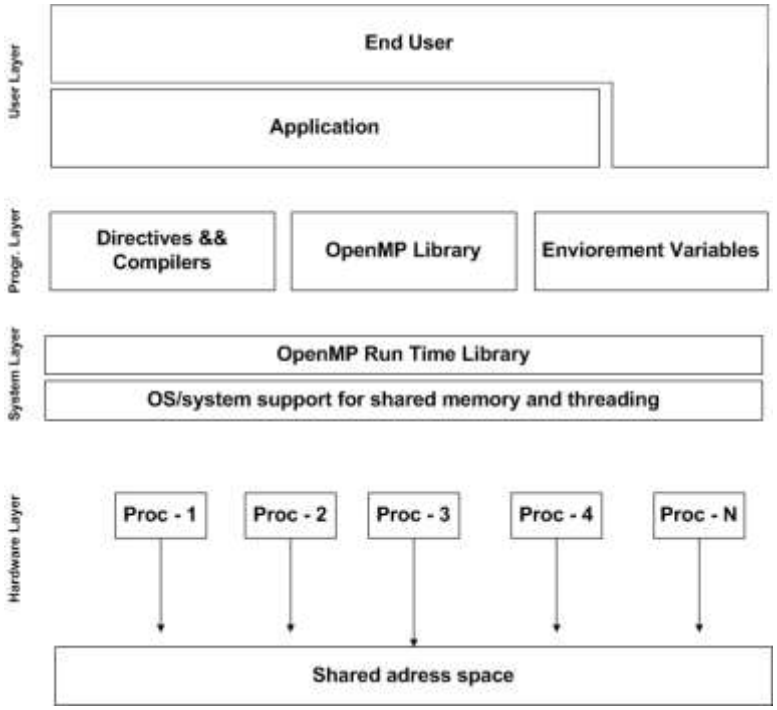


Figure 6.11: Architecture of the OpenMP API [OpenMP, 2010].

OpenMP (Open Multi-Processing): OpenMP is an Application Program Interface (API), jointly developed by a group of major computer hardware and software vendors to provide a portable, scalable model for developers of shared memory parallel applications [OpenMP, 2011]. The API supports C/C++ and FORTRAN on a wide variety of architectures, and is compatible with most major operating systems including Unix/Linux and Windows NT. The architecture of OpenMP is illustrated in Figure 6.11, and its functions are included in a header file labelled *"omp.h"*. OpenMP comprises three primary API components which are Compiler Directives, Runtime Library Routines and Environment Variables. The main goal of the API project is defined as follows:

- To provide a standard among a variety of shared memory architectures/platforms
- To establish a simple and limited set of directives for programming shared memory machines. Parallelism can be implemented by using 3 or 4 simple directives.
- To provide capability to incrementally parallelize a serial program, unlike message passing libraries which typically require an all or nothing approach

- To support for Fortran (77, 90, and 95), C, and C++ [OpenMP, 2011].

OpenMP is based upon the existence of multiple threads in the shared memory programming paradigm. A shared memory process consists of multiple threads and uses the fork-join model of parallel execution, as illustrated in Figure 6.12.

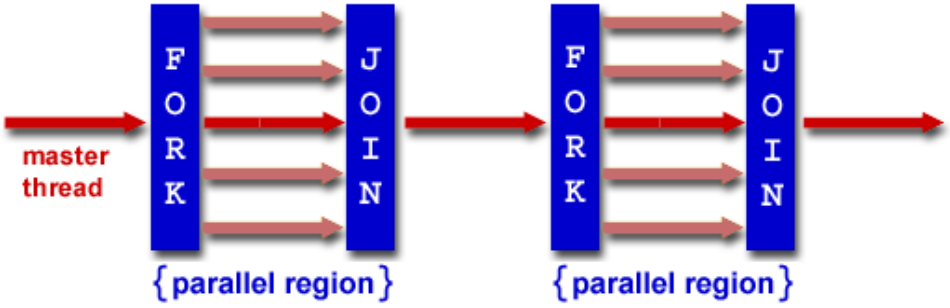


Figure 6.12: Fork join model of OpenMP [OpenMP, 2011]

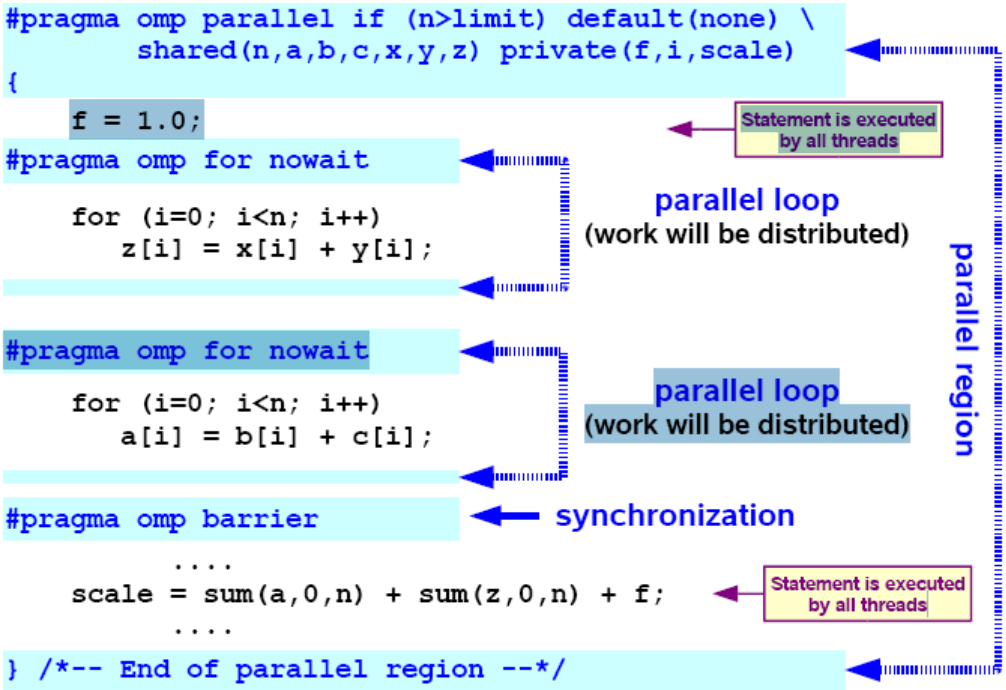


Figure 6.13 Loop synchronization of OpenMP [Barney, 2012].

All OpenMP programs begin as a single process, called the master thread. The master thread executes sequentially until the first parallel region construct is encountered. The

master thread subsequently creates a team of parallel threads, called *fork*. The statements in the program that are enclosed by the parallel region construct are then executed in parallel among the various team threads. As soon as the team threads complete the statements in the parallel region construct, they synchronize and terminate, leaving only the master thread (Join) OpenMP, called join [Tschumperlé et al., 1999]. A typical example of a parallel execution of a 'for loop' is illustrated in Figure 6.13. Loop optimization is a key problem for most programs in any programming language in terms of processing time. The example in figure 6.13 employs environment variables starting with *pragma omp* in order to provide the parallelization of the given code which basically updates three code statements sequentially.

For this project, OpenMP is utilized to enhance the performance of the conventional SIFT algorithm of Lowe [2004] which is not appropriate for real time applications. Therefore, OpenMP directives are employed with Lowe's algorithm. An open source implementation of the conventional SIFT algorithm, called the Fast Sift Library [Sourceforge, 2011], has been adopted to this project. The Library is mainly focused on extracting SIFT features from any PGM format images. The core elements of OpenMP are the constructs for thread creation, workload distribution (work sharing), data-environment management, thread synchronization, user-level runtime routines and environment variables. In C/C++, OpenMP uses "#*pragmas"* as previously mentioned.

One of the most useful directives is *omp parallel*, which explicitly instructs the compiler to parallelize the chosen block of code. In addition, the *omp for* directive instructs the compiler to distribute loop iterations within the team of threads that encounters this work-sharing construct. For instance, the *SubtractImage* procedure used in the library essentially performs the image differencing of two successive images and assigns the result to another image (imgdst = img0 - img1). The implementation requires three *for loop* operations (two of which are nested) so as to derive all corresponding pixels. The Fast Sift implementation employs the directive code starting with *#pragma*, as shown:

```
void SubtractImage(Image imgdst, Image img0, Image img1)
{
int rows = imgdst->rows, cols = imgdst->cols, stride = imgdst->stride;
float* _pixels0 = img0->pixels, *_pixels1 = img1->pixels, *_pdst = imgdst-
>pixels;
#pragma omp parallel for schedule(dynamic)
for(int j = 0; j < rows; ++j ) {
float* pixels0 = _pixels0+j*stride;
float* pixels1 = _pixels1+j*stride;
float* pdst = _pdst + j*stride;

for(int k = 0; k < (cols&~7); k += 8)
{
_MM_STORE_ALIGNED(pdst+k,_mm_sub_ps(_MM_LOAD_ALIGNED(pi
xels0+k), _MM_LOAD_ALIGNED(pixels1+k)));
_MM_STORE_ALIGNED(pdst+k+4,_mm_sub_ps(_MM_LOAD_ALIGNED
(pixels0+k+4), _MM_LOAD_ALIGNED(pixels1+k+4)));
}

for(int k = (cols&~7); k < cols; ++k)
pdst[k] = pixels0[k]-pixels1[k];
}
```

A parallel region has at least one barrier at its end, and may have additional barriers within it. At each barrier, the other members of the team must wait for the last thread to arrive. To minimize this wait time, shared work should be distributed so that all threads arrive at the barrier at about the same time. If some of that shared work is contained in *for constructs*, the *schedul**e*** clause can be used for this purpose. The dynamic schedule is appropriate for the case of a *for construct*, with the iterations requiring varying, or even unpredictable, amounts of work. The dynamic schedule is characterized by the property that no thread waits at the barrier for longer than it takes another thread to execute its final iteration. This requires that iterations be assigned one at a time to threads as they become available, with synchronization for each assignment. Further explanation and examples of implementations can be obtained from the Project's web page [Sourceforge, 2011].

The Fast Sift Library has been utilized to increase the computational performance of the SIFT extraction algorithm; however this algorithm does not support the matching procedure which compares the reference and current image. Accordingly, corresponding OpenMP directives have been implemented for the matching function as follows:

```
void FindMatches(Keypoint keys1, Image im2, Keypoint keys2)
{
   Keypoint k, match;
   Image result;
   int count = 0;

    /* Match the keys in list keys1 to their best matches in keys2.
   */
   #pragma omp parallel for
   for (k= keys1; k != NULL; k = k->next) {
     match = CheckForMatch(k, keys2);

     if(match != NULL)
        count ++;
     end

   fprintf(stderr,"Found %d matches.\n", count);
}
```

The FindMatches function is the main procedure employing extracted key points of reference and current images as inputs. The main for loop utilizes *the CheckForMatch* function to evaluate the similarity between each key point pair, and is parallelized using an OpenMP pragma.

CheckForMatch performs the matching between two images and is the key procedure employed by *Find Matches.* It finds the two closest matches and compares them with respect to specific threshold values. If the condition is satisfied, the corresponding key point is assigned as a valid match point. This function employs a for loop as expected, which is initially parallelized using an OpenMP pragma. The enhanced version of the function is illustrated as follows:

```
Keypoint CheckForMatch(Keypoint key, Keypoint klist)
{
int dsq, distsq1 = 100000000, distsq2 = 100000000;
Keypoint k, minkey = NULL;

/* Find the two closest matches, and put their squared
distances in
distsq1 and distsq2.
*/
#pragma omp parallel for
for (k = klist; k != NULL; k = k->next) {
dsq = DistSquared(key, k);
if (dsq < distsq1) {
```

```
distsq2 = distsq1;
distsq1 = dsq;
minkey = k;
} else if (dsq < distsq2) {
distsq2 = dsq;
}
}
/* Check whether closest distance is less than threshold*/
if (10 * 10 * distsq1 < 6* 6 * distsq2)
return minkey;
else return NULL;
}
```

The main improvement in performance with the use of the OpenMP library is achieved with systems having more than one processor. The library allows them to allot the work into each processor efficiently. The results shows that the default setting of the Fast SIFT library produces the same output as the conventional SIFT software. On a quad-core Core2Duo machine with OpenMP, the fast SIFT library runs approximately 6 times faster than Lowe's [2004] SIFT software for 640x480 pixel images. In this project, the onboard robot computer has a single processor where performance enhancement in terms of computational time is somewhat lower than a multi processors system as expected. Nevertheless, it is still able to reduce the overall computational time of the system by implementing the SIFT into a real time system.

6.3.2 Performance analysis of Fast SIFT library

To analyse the performance enhancement provided by the OpenMP API, both conventional SIFT and Fast SIFT implementations are tested with four different objects at different resolutions. The objects are illustrated in Figure 6.14, the first three of which will be used in the real experiments (Goal A, Goal B, Goal C). The main reason behind the selection of these objects is their distinctive patterns which makes them appropriate for the SIFT algorithm in partially cluttered environments.

The Goals are composed of images obtained from different book covers which are appropriate for SIFT based experiments. Each of these books is positioned in front of a blue cardboard paper surface so as to isolate the corresponding object from the

background. Goal A shown in Figure 6.14 (a), was used in all of the real experiments (see Chapter 7). The SIFT based analysis of the other goals regarding different resolutions are included in Appendix K.

(a) **(b)**

(c) **(d)**

Figure 6.14: Goals used in performance Evaluation for SIFT algorithms in JPG Format, (a) Goal A, (b) Goal B, (c) Goal C, (d) Goal D

6.4 Calibration analysis of the sensors for the INUS technique

Chapter 5 introduced intelligent based methods for the estimation of heading direction, and obstacle avoidance relying on a range finder. This section discusses and analyses the results from the algorithms with real sensors attached to the Pioneer 3-DX robot. The Matlab toolbox was utilized to train all networks. A two-layer feed-forward neural network with sigmoid hidden neurons and linear output neuron was designed and adapted for each problem. The networks were all trained using the Levenberg-Marquardt back-propagation algorithm. The neural networks are able to acquire knowledge from their surroundings by the adaptation of its internal parameters. The networks can learn from examples given to them, and generalize knowledge from them. Having the best generalization, the data set should be split into three parts which are training, validation and test sets. The learning procedure should be stopped in the minimum of the validation set error where the net generalizes best. If learning is not stopped, overtraining occurs which means that the network has learned not only the basic mapping associated with input and output data, but also the errors specific to the training set. If overtraining occurs, the network only memorizes the training set and loses its ability to generalize to new data.

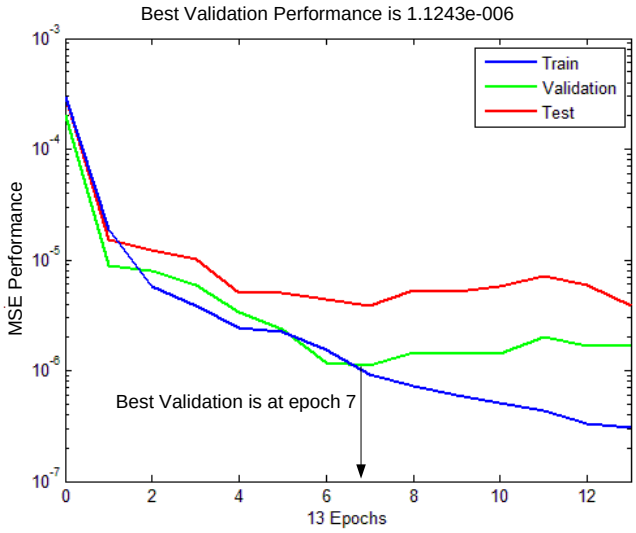
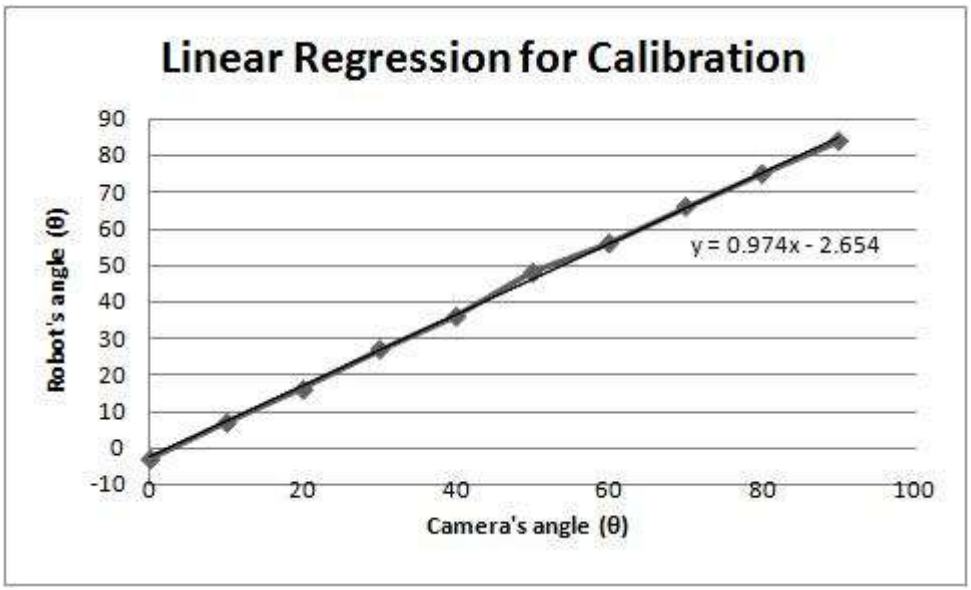
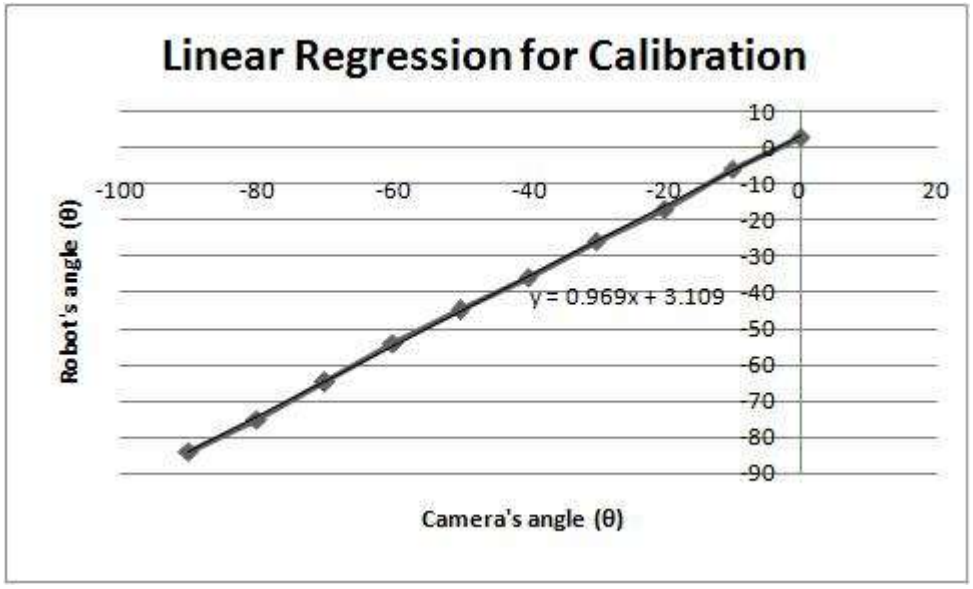


Figure 6.15: Training results for AXIS-213 camera calibration



(a)



(b)

Figure 6.16: Regression analysis to relate the AXIS-213 camera and the Pioneer 3D-x robot, (a) range from 0° to 90°, (b) range from 0° to -90°

The first step in the camera calibration is to employ scale-invariant features in order to automatically detect calibration points. A back propagation neural algorithm is utilized to map the image and the world coordinates in terms of pan and tilt angles. Different topologies have been implemented so as to provide an appropriate solution to the problem. The network shown in Table 6.1 is designed for 176x144 resolution.

Table 6.1: Basic specifications of the network for heading angle estimation

Camera Type	*Resolution*	*Data*	*Topology*	*Train*	*Validation*	*Test*
Axis-213	***176x144***	***155***	***2-6-2***	***125***	***15***	***15***

Figure 6.15 demonstrates the training result for the proposed neural network (see Table 6.1). The results reveal that the network approaches the best validation point at the 7_{th} epoch (iteration) with $1.1243x10^{-6}$ error. The outcome of these networks produces appropriate pan and tilt values with regard to specific coordinates in the image space.

The next step is to calibrate the camera with respect to the robot. The Axis-213 camera is mounted onto the robot with its optical axis aligned to the robot's forward direction and also parallel to the ground plane, as is illustrated in Figure 6.10. A simple linear regression model is used to relate the camera and the robot. The range of pan movement used in the real experiments is 180° around the horizontal axis. After conducting the analysis two regression models are obtained for ranges from 0 to 90° and 0 to -90° as shown in Figure 6.16.

Accordingly, the combination of these two linear modelling techniques leads the robot towards the specific coordinate on the image in which the specific pan angle obtained from the neural network is employed as the main input to steer the robot. The output of the network also generates the tilt angle for the camera to reach the exact position (see Chapter 5.2.2.2). Nevertheless the robot is unable to move towards the tilt direction due to its physical structure. Therefore, the camera needs to be tilted toward the goal in each processing cycle. However, the real experiments have indicated that the turning speed of tilt movement for the AXIS-213 cameras is not suitable for real time applications. Therefore the robot only utilizes the *yaw* angle along the Z axis to approach the goal. A goal tracking example related to this calibration is included in Appendix L

The final network architecture is designed for obstacle avoidance relying on range finders (see Chapter 5.5). The specifications of the network architecture are illustrated

in Table 6.2. The data is basically collected from different obstacle avoidance scenarios (see Chapter 5.4) in which the range finding data and the corresponding avoidance manoeuvres have been matched as inputs to the network. Figure 6.17 shows that the best validation point at the 11_{th} epoch (iteration) with $2.4014x10^{-5}$ error. The trained network is proposed to estimate the distance of influence (d_o) value for the obstacle avoidance behaviour.

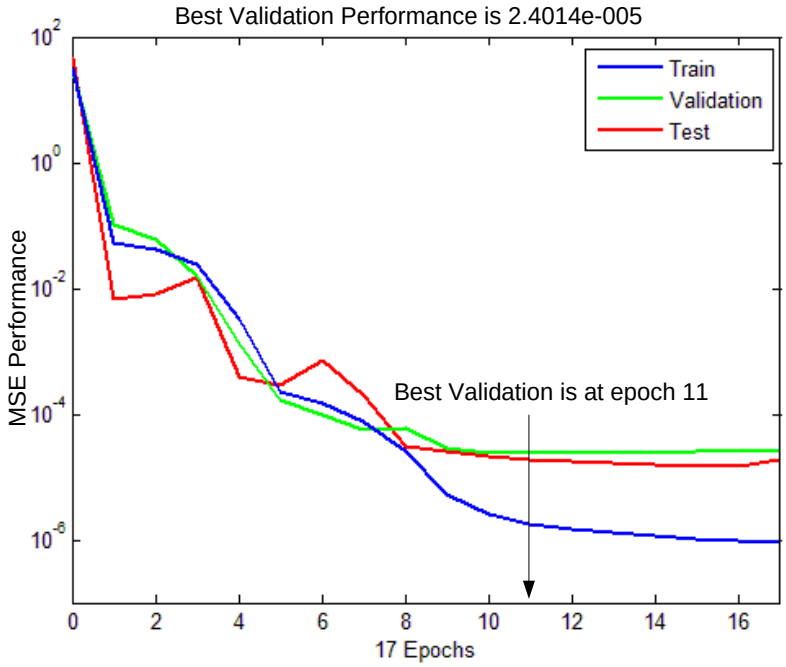


Figure 6.17: The training results for obstacle avoidance

Table 6.2: Basic specifications of the network for obstacle avoidance

Range Finder	*Resolution*	*Data*	*Topology*	*Train*	*Validation*	*Test*
URG-04LX	***180***	***650***	***9-7-1***	***520***	***65***	***65***

6.5 Summary

The robot configuration and actual design have been discussed in this chapter. Additionally, details of the software architecture as well as the open source libraries

employed have been provided. The main advantage of open source implementations is that they provide the freedom to access the source code allowing for the modification and improvement of the system which facilitated the software design in this project. Finally, the analysis and evaluation of AI algorithms implemented with the corresponding physical robot and sensors have been discussed. The following chapter introduces the experiments conducted under real conditions.

CHAPTER 7

IMPLEMENTATION AND EVALUTION OF PROPOSED NAVIGATION SYSTEMS

The navigation systems introduced in the earlier chapters have been tested extensively in real-world experiments. The experimental setups are defined in order to show the applicability of the methods developed for navigation using mapless strategies. The experiments are mainly classified into two groups. The first evaluates the performance of the hybrid vision based obstacle avoidance system, introduced in Chapter 3, the aim of which is to allow the mobile robot to navigate without collisions, in partially cluttered environments. The second group of experiments evaluates the performance of the SIFT based navigation systems discussed in Chapters 4 and 5. The Pioneer mobile robot introduced in Chapter 6 has been evaluated, using several scenarios that including random positions and different sizes and types of obstacles. The results of the first experimental group confirm that the proposed method provides an alternative and robust solution to avoid obstacles for mobile robots using a single low-cost camera as the only sensor used. The second group of experiments are aimed at demonstrating that the proposed navigation systems are both efficient and robust in permitting the robots to safely navigate from their starting positions to their goals.

7.1 Experimental Procedures

The navigation systems were uploaded onto the Pioneer mobile robot. All experiments were conducted in and around an area of the Robotics and Automation Laboratory (RAL) at Newcastle University, which has the physical dimensions of 15.60m x 17.55m, as illustrated in Figure 7.1. Hard board panels were used to simulate walls during the experiments. Figure 7.2 displays a schematic diagram of the test environment.

Figure 7.3 shows the corridor area located in front of the laboratory with the physical dimensions of 15.40m x 9.60m. A schematic diagram of the corridor is illustrated in Figure 7.4. In addition, Room G-45 located in the RAL has the physical dimensions of 10m x 5.20m, as illustrated in Figure 7.5. A schematic diagram of the room is illustrated in Figure 7.6

7.1.1 Performance evaluation

The following parameters derived from previous studies [Huq et al., 2008; Szenher, 2008] were used to evaluate the robustness and consistency of robot navigation performance.

Total navigation time (t_s): This parameter indicates the total duration of travel in seconds. A lower value of t_s is expected for fast navigation.

Total travel distance (d_t): This parameter presents the distance travelled by the robot from its starting position to its goal. A lower value of d_t is expected to optimize the travel distance.

Average rate of change of angular velocity ($\Delta\Omega$): This parameter establishes the average change of angular velocity (in deg/s^2) as the robot navigates from point to point. A lower value of travel $\Delta\Omega$ indicates a reliable angular velocity of the robot, which is given as follows:

$$\Delta\Omega = \frac{1}{n-1}\sum_{i=2}^{n}\frac{|\Omega(i)-\Omega(i-1)|}{t_c(i-1)} \qquad (7.1)$$

where $\Omega(i)$ = the angular velocity at the i-th decision cycle.

$t_c(i-1)$ = the length of the s-th decision cycle.

Position Error of the robot: This parameter establishes the distance between the home position where the reference image is captured and the location where the robot has stopped. A lower value of error indicates successful navigation, which is given as follows:

$$Error = \sqrt{(x_r - x_c)^2 + (y_r - y_c)^2} \quad (7.2)$$

where (x_r, y_r) = coordinates of the home position.

(x_c, y_c) = coordinates of the robot's final position.

Total number of collisions: (*C*) : This parameter should be zero in safe navigation.

Average velocity (v_a)**:** The average velocity of an agent moving through a displacement (Δd) during a time interval Δt can be expressed as follows:

$$v_a = \frac{\Delta d}{\Delta t} \quad (7.3)$$

Figure 7.1: Robotics and Automation Research Labrotary, Newcastle University (including hard-board panels)

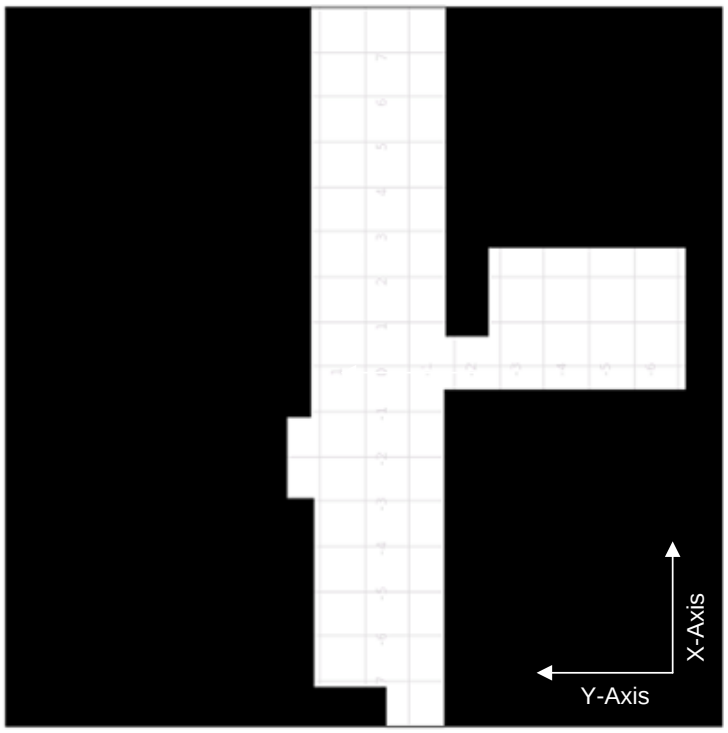


Figure 7.2: Schematic of navigation environment (including hard-board panels)

Figure 7.3: Corridor area outside the Labrotary

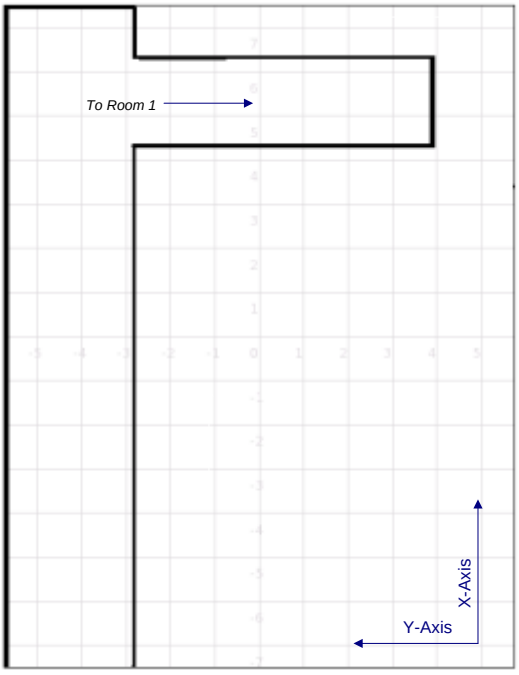


Figure 7.4: Schematic of the corridor environment

Figure 7.5: Room G-45

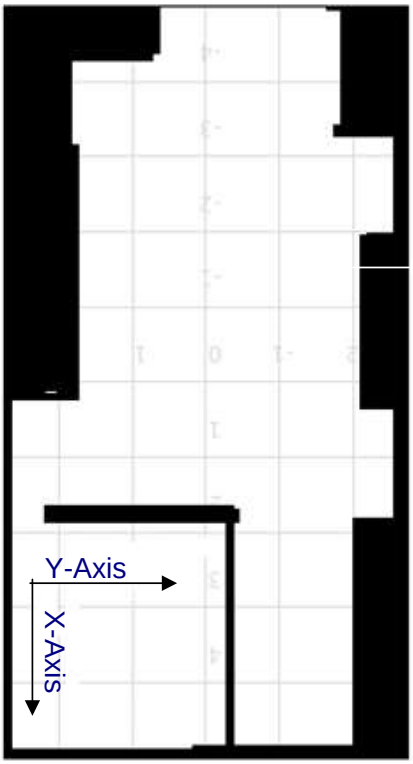


Figure 7.6: Schematic of Room G-45 in the RAL

7.2 Experimental Design and Results of the Hybrid Vision Based Obstacle Avoidance System

This section presents the design of the experiments used to evaluate the proposed hybrid vision based obstacle avoidance system. The experiments were conducted in the test environments shown in Figures 7.1 and 7.3. In order to verify the performance of the proposed system, the results for each scenario are compared with those of the conventional optical flow method. The main aim of these experiments is to navigate the robot in these environments with regard to the designed scenarios without hitting any obstacles until a certain amount of time has passed. The robot navigates in these experiments at a linear speed of 0.15 m/sec and the required time limit is 200 seconds to fulfil each scenario. Therefore, once the robot achieves to wander along the environment without colliding until the end of the time limit, it is accepted to complete the task successfully. All overhead lights in the laboratory and corridor environment are turned on during the capture of both snapshot and current images, in an attempt to maintain constant illumination over the entire experimental area. Images were captured at a resolution of 176x144 jpg format and then converted to *pgm* format.

Table 7.1: Initial parameters for experiments

Parameters	***Descriptions***
Initial heading angle	$\theta = 0^o$
Linear velocity(constant)	$v_c = 0.15$ m/s
Constant turn value	$c_{turn} = 90^o$
Maximum range for turning	$n = \pm 20°$
Minimum Time Limit	$T_l = 200$ sec (must move at least 200 sec)

Four different scenarios are discussed in this section, and an example presenting the limitations of the proposed architecture is demonstrated. Two vision based obstacle avoidance techniques were employed, namely the Hybrid (FS) and Optical Flow Based (OFB) as discussed in Chapter 3. In order to provide a precise comparison of the test results, each technique is integrated with the proposed control architecture and the

behavioural strategy discussed in Chapter 3. Table 7.1 displays the initial parameters used in the navigation algorithms used for conducting the experiments.

7.2.1 Definition of scenarios

Four different scenarios were set up to evaluate the performance of the proposed system. They are arranged in increased level of difficulty. Experiments were conducted in two different test environments as previously discussed.

Scenario 1 (S1) – requires the robot to navigate in the first open environment (with no external obstacles).

Scenario 2 (S2) – requires the robot to navigate in the first environment whilst having to negotiate two obstacles.

Scenario 3 (S3) – requires the robot to navigate in the second open environment (with no external obstacles).

Scenario 4 (S4) – requires the robot to navigate in the second environment whilst having to negotiate three obstacles.

7.2.2 Navigation test results

Each individual test was repeated five times and the average for each performance parameter was determined. The results for each series of tests were found to be very consistent, principally because the starting position, robot position and feature size, and position of the obstacles which are identical for different runs under the same scenario. The data used in each of the following trajectory plots, however, is taken from the last run in each of the corresponding scenarios.

A performance evaluation table presenting the average rate of chance of angular velocity ($\Delta\Omega$), total number of collisions (C) and total navigation time (t_s) is generated

for each scenario. For the corresponding ($\Delta\Omega$) and (t_s) parameters in each scenario, constant rate of turn values are excluded in order to simplify the results of evaluation parameters.

Scenario 1 (S1)

Figure 7.7 displays the scenario in which the robot is required to navigate in this open environment without colliding.

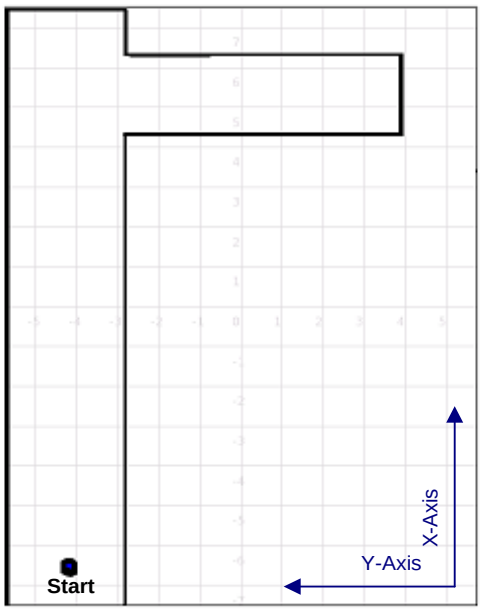
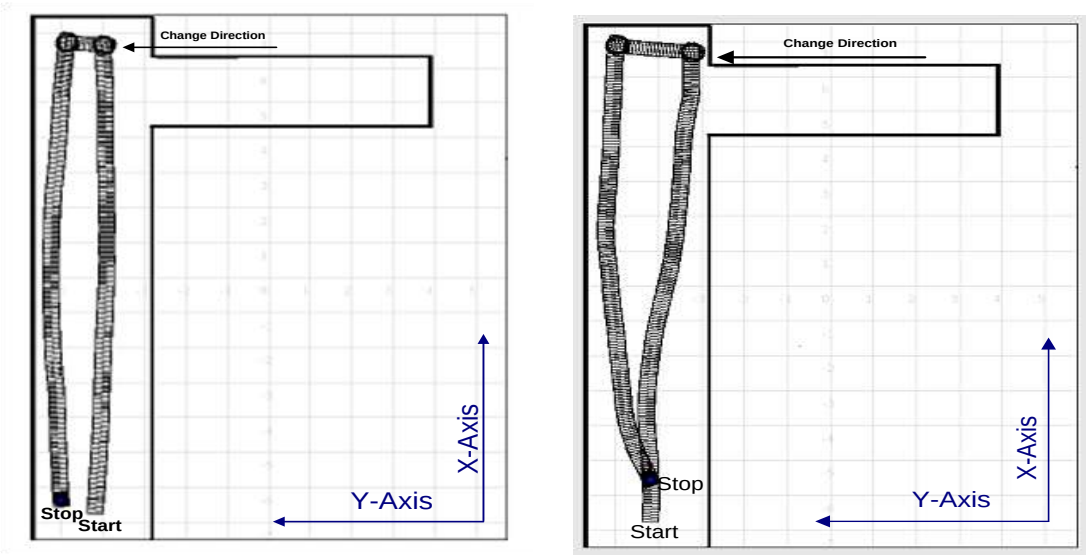


Figure 7.7: Scenario 1 for vision based obstacle avoidance problem

The trajectory of the first scenario with the FS technique is given in Fig 7.8 (a). The robot navigates with a smooth trajectory until it reaches the end of the corridor (door). When it encounters the door, *Change Direction* behaviour is activated, performing a 90° left turn in order to avoid the door. It then moves towards the left wall, which is also avoided with a 90° left-turn manoeuvre. After this the robot moves towards the start position whilst being pushed away from the left wall and this result in the curved trajectory of the robot. As usual due to the noise within the vision system, the robot could not follow a straight line, but it always remains inside an acceptable margin around the centre of the corridor. Furthermore, the OFB Technique, as illustrated in Fig.

7.8 (b), is applied to steer the robot whilst avoiding collision, which essentially completes the task with a similar trajectory to the FS method. However it follows a wider path and produces a trajectory with higher values of ($\Delta\Omega$). Table 7.2 summarises the performance results of each algorithm, showing the FS method performs navigation without any collision, and it also generates a lower value of $\Delta\Omega$ which represents the smoothest trajectory of the methods employed.



(a) **(b)**

Figure 7.8: Estimated trajectories for scenario 1 (a) FS method, (b) OFB method

Table 7.2: Performance measures for scenario 1

Methods	*$\Delta\Omega$ (deg/s^2)*	*Time (sec)*	*Collision*
Hybrid	*4.61*	*200*	*No*
Optical Flow	*6.73*	*200*	*No*

Scenario 2 (S2)

Figure 7.9 illustrates the second scenario in which the robot is required to navigate in the same environment while two unexpected obstacles are placed in its path. The results of the corresponding scenario employing the FS technique are shown in Figure 7.10 (a)

where the robot navigates along the corridor smoothly until it perceives the door. When it encounters the door, *Change Direction* behaviour is activated, performing a 90° left turn in order to avoid the door. Afterwards it moves forward until it perceives the wall, which is avoided successfully by another 90° left-turn manoeuvre. After this it continues moving toward the start position until it perceives the first obstacle. The robot avoids the obstacle using a sharp manoeuvre. Subsequently the robot continues moving with a smooth trajectory whilst avoiding the right wall. Figure 7.10 (b) presents the estimated trajectory generated using the OFB algorithm in which the robot successfully achieves to avoid the first obstacle. It then continues to move safely until it perceives the second obstacle. However, the robot then makes a sharp manoeuvre to the left which causes it to collide with the wall. Performance measures for this scenario are illustrated in Table 7.3, revealing that the FS method is again considered to be the most suitable of the two methods.

Table 7.3: Performance measures for scenario 2

Methods	*$\Delta\Omega$ (deg/s^2)*	*Time (sec)*	*Collision*
Hybrid	*4.67*	*200*	*No*
Optical Flow	*7.54*	*144*	*(3 times)*

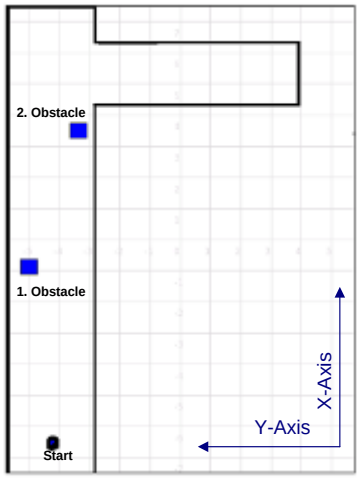
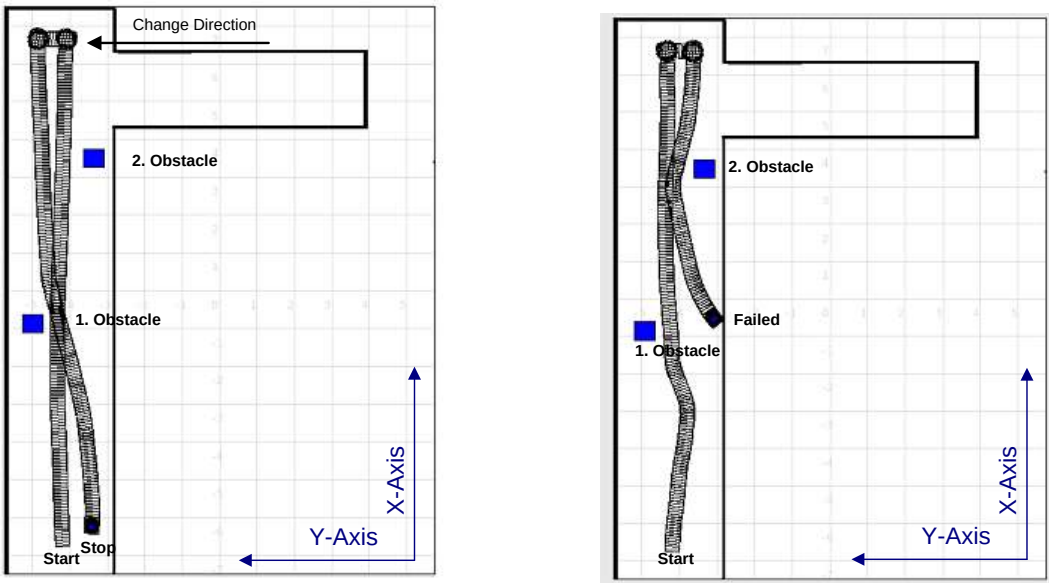


Figure 7.9 Scenario 2 for vision based obstacle avoidance problem



(a) **(b)**

Figure 7.10: Estimated trajectories for scenario 2 ,(a) FS , (b) OFB

Scenario 3 (S3)

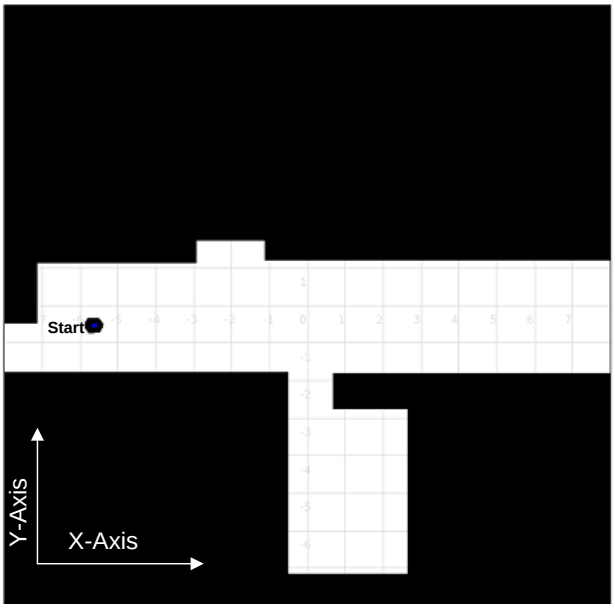
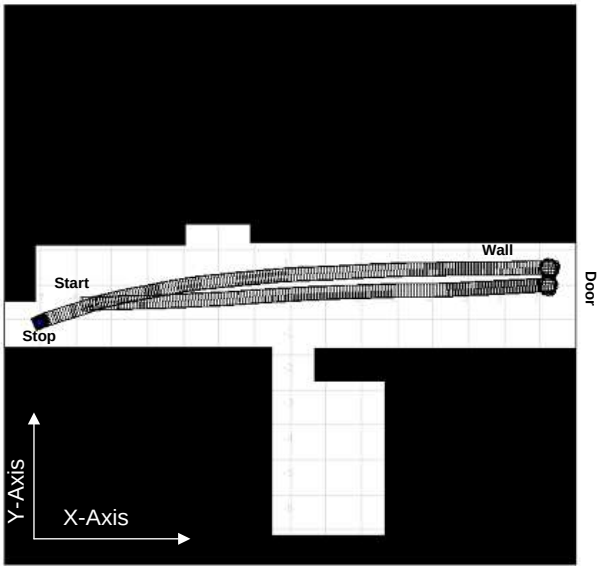


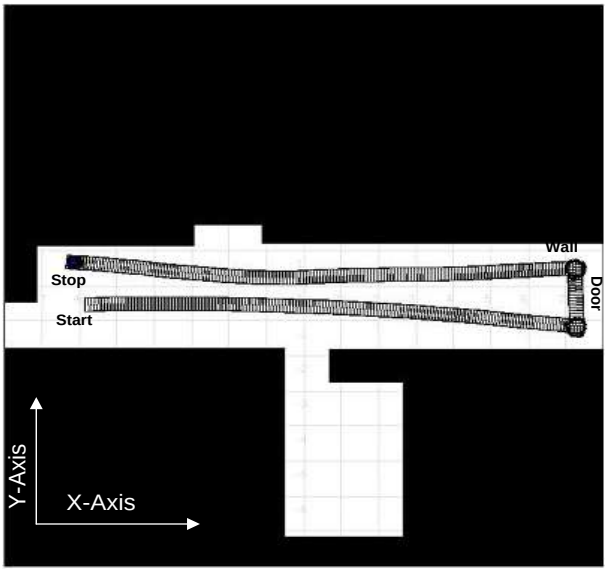
Figure 7.11: Scenario 3 for vision based obstacle avoidance problem

Figure 7.11 presents the third scenario which was conducted in the laboratory environment where the robot was required to navigate in this open environment. The

results of the corresponding scenario employing the FS technique are shown in Figure 7.12 (a). The robot navigates through the environment successfully without collision. It negotiates both the door and the wall avoiding them using a 90° left turn manoeuvre. It then proceeds to move forward along a left curved trajectory eventually getting back to its start point. The results demonstrate that the FS technique performs smooth and robust behaviour for this navigation task.



(a)



(b)

Figure 7.12: Estimated trajectories for scenario 3, (a) FS , (b) OFB

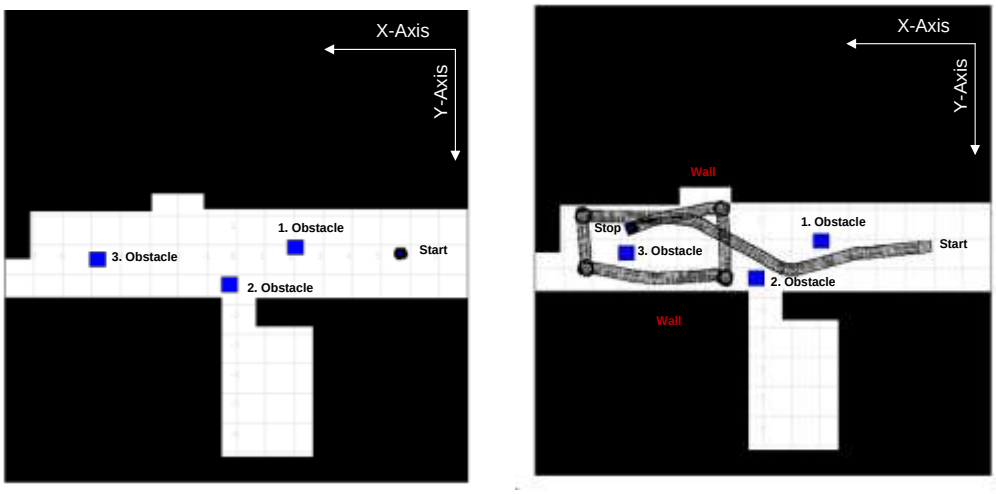
The OFB technique performs the navigation without colliding with any obstacle in a smooth manner, as shown in Figure 7.12 (b), so that it negotiates the door and walls respectively. The results demonstrate that performance in these experiments is surprisingly reliable for this scenario. Table 7.4 presents the performance measures for each method with this scenario, The FS technique performs the task for each repetition successfully. OFS fails once but its overall performance is better than expected. Nevertheless it generates a higher value of $\Delta\Omega$ compared to the FS method.

.

Table 7.4: Performance measures for scenario 3

Methods	*$\Delta\Omega$ (deg/s^2)*	*Time (sec)*	*Collision*
Hybrid	*4.19*	*200*	*No*
Optical Flow	*5.23*	*182*	*(1 times)*

Scenario 4 (S4)



(a) **(b)**

Figure 7.13: Estimated trajectories for scenario 4, (a) scenario 4, (b) FS

Figure 7.13 (a) illustrates the fourth scenario in which the robot is required to navigate in the laboratory environment with three unexpected obstacles are placed along its path. Figure 7.13 (b) presents the navigation results of the FS method for this scenario. The robot begins its navigation and then it detects the first obstacle. The robot avoids the

first and second obstacles successfully. Subsequently it avoids the second obstacle. After this the robot negotiates walls and the third obstacle, all of which are successfully avoided by following a rectangular path.

The navigation results for the OFB technique for this scenario are given in Fig 7.14, where the robot avoids the first obstacle but collides with the second. After this the robot passes the second room and is stopped. Performance measurements of the given scenario are illustrated in Table 7.5

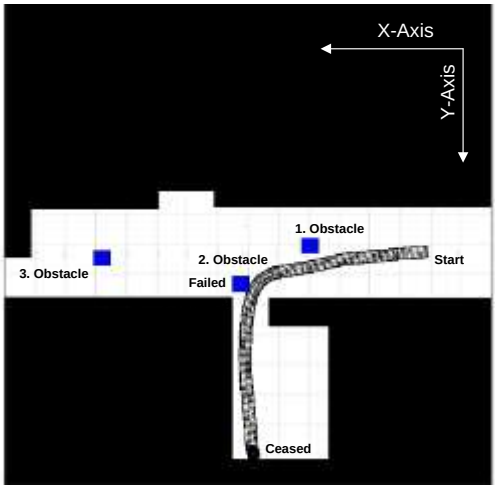


Figure 7.14: Estimated trajectories for scenario 4 using OFB

Table 7.5: Performance measures for scenario 4

Methods	*$\Delta\Omega$ (deg/s^2)*	*Time (sec)*	*Collision*
Hybrid	*6.88*	*191*	*(1 times)*
Optical Flow	*7.79*	*107*	*(3 times)*

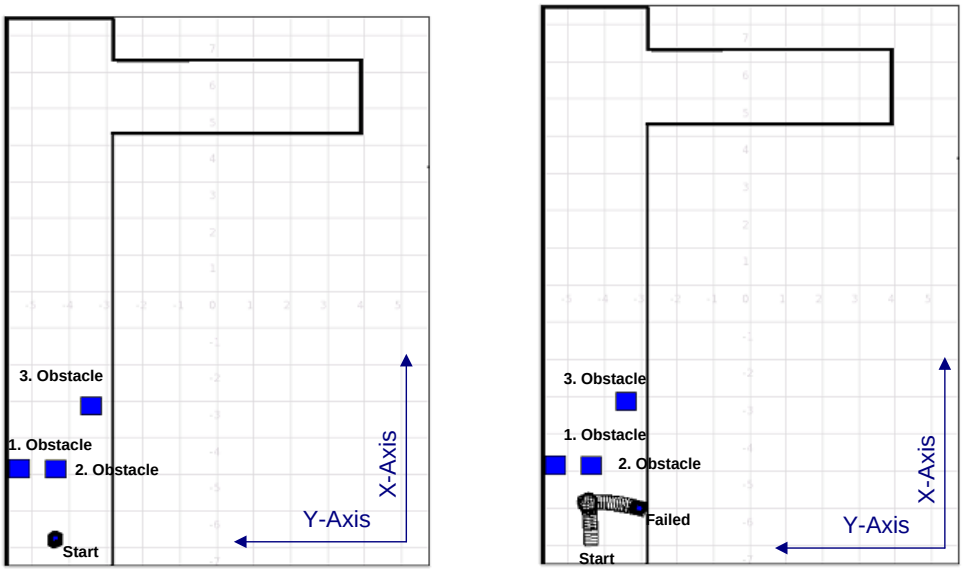
7.2.3 Comparison and evaluation of methods

In these test scenarios, the positions of all obstacles in the test environment are unknown to the robot. Four scenarios were selected for discussion in this section. The

results reveal that the OFB technique addresses the use of optical flow to supervise the navigation of mobile robots. It basically utilizes control laws in aiming to detect the presence of obstacles close to the robot based on information about changes in image brightness. The technique performs better than expected in steering the robot effectively, especially in the open environments as illustrated in Figures 7.8 (b) and 7.12 (b). The major difficulty with employing optical flow in mobile robot navigation is when key information is not obtained concerning whether or not motion vectors or changes of illumination change the intensity value of pixels. In addition, despite the assumption of having constant illumination, lighting conditions may significantly change due to environmental factors which optical flow techniques are known to have difficulty in handling. These may cause the miscalculation of flow vectors and can result in collision, as illustrated in Figures 7.10 (b) and 7.14. The OFB is capable of negotiating walls and doors successfully which provides flexibility in this method in partially cluttered indoor environments. However, the OFB technique is not able to avoid external obstacles deliberately located along the path of the robot, as much as the FS technique is able to perform.

It is proposed that the FS technique can improve on the performance of the OFB method, by fusing the results of two techniques in terms of the optical flow based control law. Figures 7.10 (a) and 7.13 (b) reveal the capacity of this technique in partially cluttered environments, including those with external obstacles. The aim of the technique is to integrate the results of the appearance-based detection technique and optical flow based navigation architecture. The technique has the ability to negotiate and avoid walls and doors, by benefiting from the results of the optical flow based navigation technique employing the frontal optic flow to estimate the so-called time-to-contact before a frontal collision is likely to occur (see Chapter 3). It is also able to avoid lateral obstacles more smoothly than with the conventional optical flow technique. The outcome of balance strategy tends to maintain equal distances to obstacles on both sides of the robot, exploiting the results of the appearance-based detection technique. The test results reveal that the overall performance of the system is better than that of the conventional technique, but it is still vulnerable to lighting conditions, illumination problems and floor imperfections.

The characteristics problems of these conventional methods may still affect the performance of the proposed method. Figure 7.13 (b) illustrates the characteristics of the proposed algorithm in cases of frontal obstacles spanning the entire field of view. As the robot reminds blind (does not make any decision) and the TTC value indicates the high possibility of collision, the behavioural module is triggered, according to which the *Change Direction* behaviour has a higher priority level than the *Steering* behaviour. Thus a 90^{o} turning manoeuvre is performed to avoid obstacles. The FS method does not extract the features of the images but only measures the differences between them, which makes the technique appropriate for real time applications, however this maybe a disadvantage in more complex situations.



(a) **(b)**

Figure 7.15: Experimental results for scenario 5(trap-situation), (a) scenario 5, (b) FS method

An example illustrating the limitations of the technique is shown in Fig 7.15 (a), in which the robot is required to negotiate three obstacles that are close to each other. Fig 7.15 (b) displays the navigation results of the robot where the robot is not able to avoid both the obstacles and collision with the right wall. This case represents a typical trap-situation for this method, where the robot is not able to avoid all obstacles in such complex situations. Table 7.6 highlights the percentage improvement in performance of the FS over the OFB for the Pioneer robot. Tables 7.2-7.5 consistently demonstrate

improved performance. Accordingly, the FS navigation method offers better overall performance in terms of safety and consistent motion when compared to the OFB method. The control outputs of each scenario can be found in Appendix N.

Table 7.6: Performance improvement of the FS over the OFB

Scenario No	*more consistent*	*safer navigation*	*longer navigation*
S1	*%31*	*% 0*	*%0*
S2	*%39*	*%150*	*%28*
S3	*%20*	*%25*	*%9*
S4	*%26*	*%100*	*%78*

7.3 Experimental Design and Results of SIFT based Navigation Systems

This section presents and discusses the test results of the SIFT based and Intelligent SIFT based navigation strategies discussed in Chapters 4 and 5 respectively.

Table 7.7: Initialization of the robot control algorithm for NUS

Parameters	***Descriptions***
Start Position (x,y)	Start position of the robot on the simulated area
Goal Position (x,y)	Position of the goal on the simulated area
Initial heading angle (θ)	Starting heading angle of robot, θ=0^o
Distance of influence of object	d_o = 0.65 m
Maximum Velocity	v_{max}= 0.24 m/s
Minimum Velocity	v_{min}= 0.064 m/s
Maximum matching value (Stop Criteria)	30, for reaching goal
Minimum matching value (Start Criteria)	3, for starting the navigation
Velocity constant	k_v =0.008 adjust velocity
Steering constants	s_w = 1.2, s_{w1} = 0.6, s_{w2} = 0.8 adjust steering
Avoidance behaviour parameters	o_w = 12 deg/s ; o_v = 0.1 m/s

The experiments were conducted in the test environments shown in Figures 7.3 and 7.5. The proposed SIFT based methods, navigation using the SIFT (NUS) and Intelligent Navigation using SIFT (INUS) were initially tested using the Pioneer robot. The results show that, when employing the proposed INUS method, the robot successfully avoided collisions and was able to reach all of the desired goals. On the other hand, the NUS performed fairly well despite its full reactive architecture and simple control strategy. Nevertheless it failed to successfully complete some of the complex scenarios, as discussed in the following sections. Since navigation results are very hard to quantify [Gat, 1995], the performance measures described in Section 7.1.1 were employed to evaluate navigation performance. A corresponding table of performance measures comparing the performance of the two methods is illustrated for each scenario; although these do not include the time spent searching for the goal. The experiments revealed that the robot under INUS gets closer to the goal than when using NUS. Accordingly, an updated version of the travelled distance parameter (d_t) including this difference is reported next to the (d_t) parameter in square brackets for each performance table and employed for performance evaluation.

Table 7.8: Initialization of the robot control algorithm for INUS

Parameters	***Descriptions***
Start Position (x,y)	Start position of the robot on the simulated area
Goal Position (x,y)	Position of the goal on the simulated area
Initial heading angle (θ)	Starting heading angle of robot, $\theta=0^o$
Distance of influence of object	$d_o = 0.5$ m
Maximum Velocity	$v_{max}= 3.7$ m/s
Minimum Velocity	$v_{min}= 0.11$ m/s
Minimum matching value (Start Criteria)	3, for starting the navigation
Distance Tolerance	d_t=0.6 m for reaching goal

The trajectories of the robot were plotted using data collected during the experiments. Values of linear velocity (m/sec), angular velocity (deg/sec) and matching strength generated by the output in each experiment are displayed for both methods. Tables 7.7

and 7.8 present the initial parameters used in the NUS and INUS algorithms. Parameters were estimated by a trial and error method. During the experiments, the sampling period of the length of a decision cycle was set at t_c = 220 ms for NUS and 300 ms for INUS. This is because the INUS method needs more time to generate control variables, mainly due to the complexity of the algorithm. For each decision cycle, the robot is controlled by an updated translational velocity command (v) and angular velocity command (w).

Images were captured at a resolution of 176x144 or 352x288 pixels, and stored in the *pgm* format with respect to the characteristics of the SIFT algorithm. The resolution of 352x288 pixels was utilized only for *search behaviour*, generating more interest points with the SIFT algorithm which allows the position of the goal to be obtained. Nevertheless the SIFT algorithm consumes a lot of processing time at this resolution, which is not appropriate for visual servoing and real time applications with the given robot configuration. Therefore, once the goal was detected, the image resolution of the captured goal was reduced from 352x288 to 176x144 pixels whilst navigating the robot. When the resolution of an image is down-sampled to a smaller resolution, the number of extracted features is significantly reduced. This, in essence diminishes the performance of both velocity controllers. On the other hand, Lowe [2004] claimed that three features are enough for robust matching across different scenes. Therefore, in these experiments, in order to compensate for the decline in numbers of matched features, a matching constant with a value of 5 was added to the total matching strength parameter. This updated parameter was only employed by the velocity controllers, and was not taken consideration when calculating the steering parameters of both methods.

The reference images (goal images) used in these experiments were described in Chapter 6. For each scenario, the robot starts its navigation by searching for the position of the goal where the panning features of the corresponding behaviour may be invoked with regard to its position. To increase the overall performance of the system in the real experiments, some minor modifications were implemented in terms of the parameter selection for search behaviour. Consequently, the searching interval for the panning function was set to 15°. In addition, in order to reduce the total processing time

of searching, two levels of zoom *4000(11x)* and *6000(16x)* were utilized for each panning level, to increase the overall performance of the system.

The main obstacles used in this experiment were rectangular boxes of dimensions of 550x500 mm. Different sizes of obstacles were used in one of the more complex scenarios designed to demonstrate and compare the capabilities of the algorithms, as discussed in section 7.3.2.

7.3.1 Experimental Implementation

These experiments are classified into two groups. The first group of experiments are preliminary navigation tests which ensure that the proposed navigation algorithms are able to perform fundamental tasks in a partially cluttered test environment. The second group of more complex experiments aim to evaluate the overall performance of these algorithms when different numbers of obstacles are placed along the robot's path, to see if it can detect and avoid these objects and make its way to the desired location. In addition, the robot is required to reach different goals sequentially, which allows the performance evaluation of the proposed local navigation methods in cases of global navigation problem.

7.3.1.1 Preliminary test results

The preliminary navigation tests were conducted in Room G-45, as shown in Figure 7.5. These experiments aimed to reveal the performance of fundemantal behaviours for each SIFT technique, as well as measuring which of them can yield better navigation performance in static environments. All overhead lights in the test environments were turned on during the capture of both snapshot and current images, resulting in constant illumination over the entire experimental area. Four different scenarios were created in order to evaluate the basic skills of the algorithms.

Preliminary Scenario 1 (PS1) – requires the robot to navigate towards its goal in an open environment with no external obstacles.

Preliminary Scenario 2 (PS2) – requires the robot to navigate towards its goal in an open environment with an initial heading angle (θ) of 90° with no external obstacles.

Preliminary Scenario 3 (PS3) – requires the robot to navigate towards its goal whilst an external obstacle is located on its path.

Preliminary Scenario 4 (PS4) – requires the robot to navigate towards its goal which is located at the end of the room.

PS1 is the shortest and easiest task while the PS2 involves searching for the goal and PS3 involves negotiating an obstacle. PS4 represents a more challenging situation, where the goal is located at the end of the room so that the robot needs to negotiate a narrow gap along its path. Table 7.10 displays the starting and goal position for each scenario.

Table 7.10: Definition of preliminary scenarios

Scenario	*Start (x,y)*	*Goal (x,y)*
PS1	*(0.0,0.0)*	*(5.0,0.0)*
PS2	*(0.0,0.0)*	*(5.0,0.0)*
PS3	*(0.0,0.0)*	*(5.5,0.0)*
PS4	*(0.0,0.0)*	*(8.0,1.0)*

Preliminary scenario 1

Figure 7.16 (a) displays the first scenario, in which the robot is required to follow a straight line from the start to the goal position. Figures 7.16 (b) and 7.16 (c) display the trajectories of the robot employing the NUS and INUS algorithms respectively. The control parameters for each method are also presented in Figures 7.16 (d) and 7.16 (e). Table 7.11 displays the performance measures for both navigation methods in successfully reaching the goal. These parameter values were calculated using data averaged from repeated experiments.

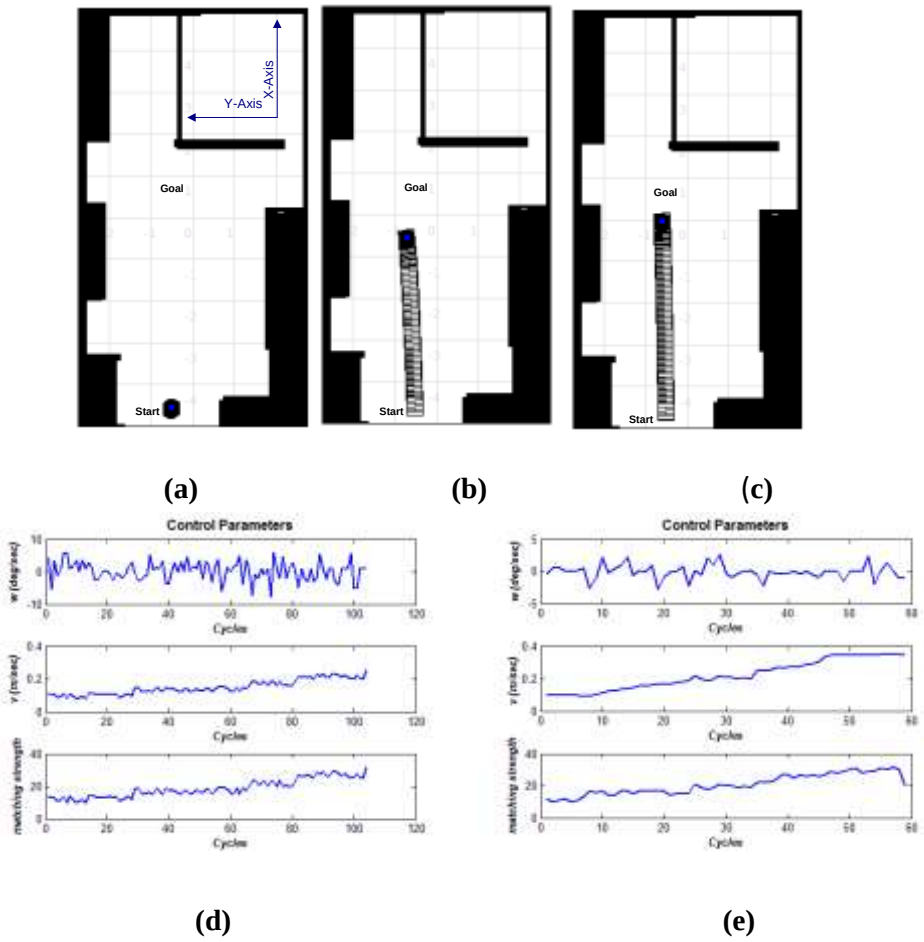


Figure 7.16: Experimental results for PS1, (a) the scenario, (b) estimated trajectory with NUS , (c) estimated trajectory with INUS, (d) control parameters for NUS, (e) control parameters for INUS

Table 7.10: Performance measures for PS1

Methods	*$\Delta\Omega$ (deg/s^2)*	*t_s(s)*	*d_t(m)*	*Error(m)*	*v_a(m/s)*	*Collision*	*Search(deg)*
NUS	*3.99*	*23.2*	*4.01 [4.61]*	*1.143*	*0.173*	*No*	*0*
INUS	*2.46*	*17.9*	*4.49*	*0.547*	*0.251*	*No*	*0*

Preliminary scenario 2

Figure 7.17 (a) displays results for the second scenario in which the robot's initial heading angle is set to 90^o to the right, and it is required to follow a straight line from

the start to the goal position as in the previous scenario. Figures 7.17 (b) and 7.17 (c) present the trajectories for the robot employing the NUS and INUS methods respectively. Figures 7.17 (c) and 7.17 (d) display the control parameters for each method. The robot initially starts searching for the goal with varying pan positions. Once it is detected, the robot turns 90^{o} towards the goal position and heads towards it. Table 7.11 displays the performance measures of the robot which was able to complete the missions safely.

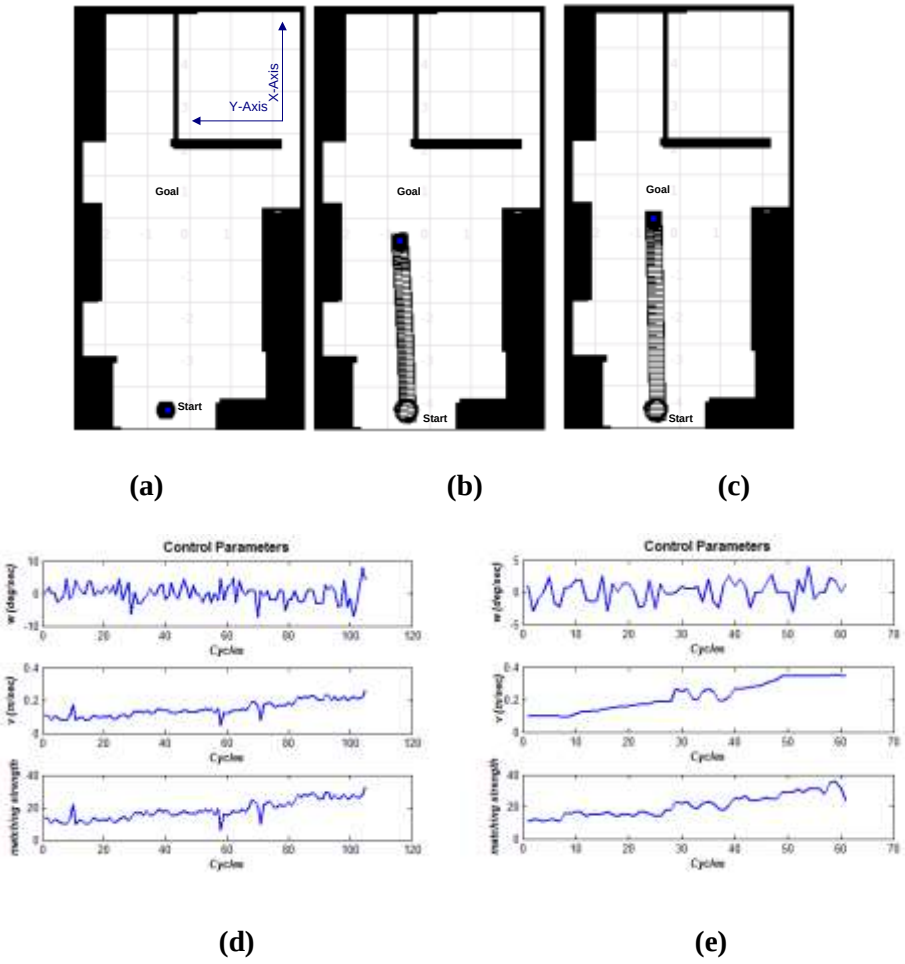


Figure 7.17: Experimental results for PS2, (a) the scenario, (b) estimated trajectory with NUS , (c) estimated trajectory with INUS, (d) control parameters for NUS, (e) control parameters for INUS

The results for the PS1 and PS2 scenarios basically show the performance of the proposed algorithms for navigation along a straight line. The velocity in both algorithms gradually increases in the open environment when approaching the goal according to the

characteristics of the corresponding control algorithms. The NUS method produces higher values of $\Delta\Omega$, *Error* and $t_{s,}$ and a lower value of v_a than when employing the INUS method.

Table 7.11: Performance measures for PS2

Methods	*$\Delta\Omega$ (deg/s^2)*	*t_s(s)*	*d_t(m)*	*Error(m)*	*v_a(m/s)*	*Collision*	*Search(deg)*
NUS	*4.27*	*23.0*	*4.03 [4.63]*	*1.107*	*0.175*	*No*	*90*
INUS	*3.31*	*18.3*	*4.57*	*0.51*	*0.249*	*No*	*90*

Preliminary scenario 3

This scenario requires the robot to navigate towards the goal whilst having to negotiate an external obstacle placed along its path, as shown in Figure 7.18 (a). The trajectories for both methods are displayed in Figures 7.18 (b) and 7.18 (c). This scenario demonstrates the robot's ability to avoid an external obstacle. Figure 7.18 (b) illustrates the trajectory of the robot under the NUS in avoiding the obstacle. When the obstacle is detected the obstacle avoidance behaviour is activated and the robot starts its avoiding behaviour whilst also negotiating the left wall which incidentally generates a repulsive force. Once the obstacles have been successfully avoided the robot detects the goal and attains its mission. Under INUS, the robot navigation is conducted in a smoother manner due to the global velocity control technique which adjusts the speed of the robot once the obstacles are detected. Here, the enhanced avoidance behaviour improves the robot's capability of passing through the gap smoothly and safely.

The control parameters of NUS are illustrated in Figure 7.18 (d), where the robot's velocity increases gradually with respect to matching strength until the obstacle is detected, and afterwards the robot avoids the obstacle with constant speed. Once the goal is perceived again the goal is approached with increasing speed. The main challenge facing this control technique is to prevent collisions which may be caused by failure to adjust speed while negotiating obstacles. The algorithm is supposed to increase speed proportionally according to the matching of similarity. Accordingly, the sensitivity of the control parameters is quite significant. To overcome these problems,

the parameters d_o and k_v must be designed carefully. The d_o is set to a constant value of distance in order to provide enough space for avoidance manoeuvres, and in addition k_v should be set to a reasonable value (see Table 7.7).

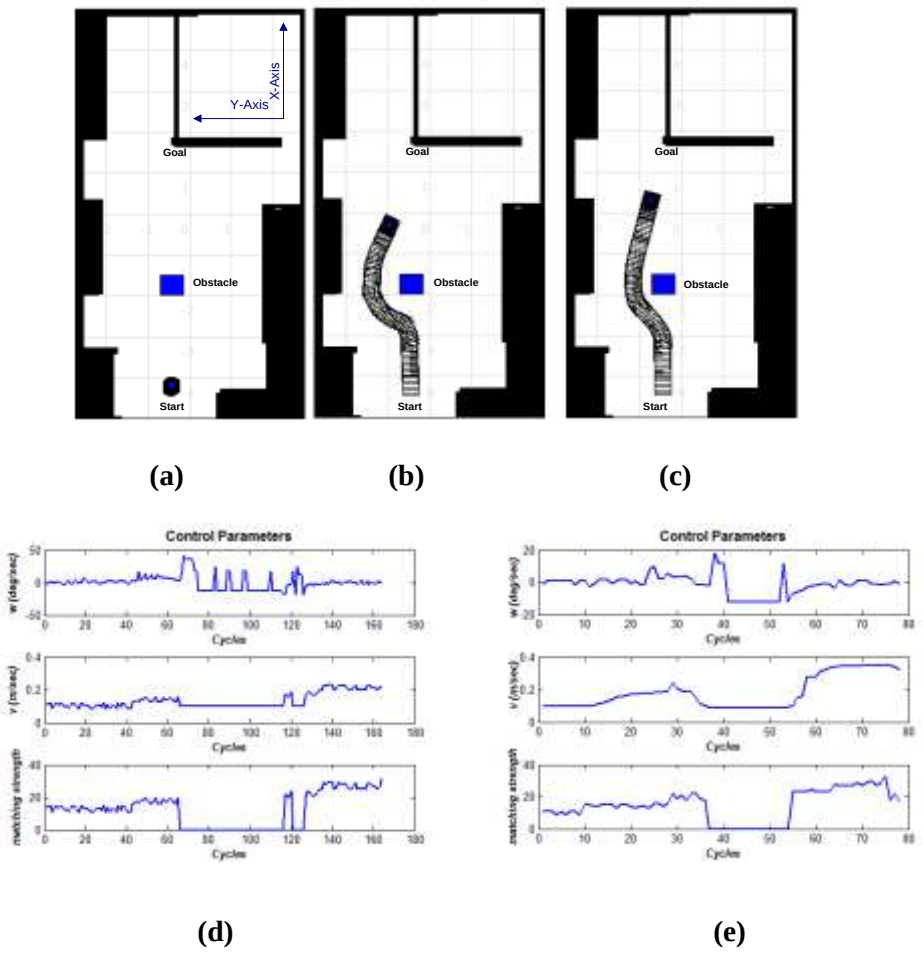


Figure 7.18: Experimental results for PS3, (a) the scenario, (b) estimated trajectory with NUS , (c) estimated trajectory with INUS, (d) control parameters for NUS, (e) control parameters for INUS

These changes, on the other hand, decrease overall performance in terms of velocity and smooth motion. The initial control parameters employed in the INUS technique are given in Figure 7.18 (e), where the speed is adjusted reasonably. This is primarily due to the proposed fuzzy inference system and avoidance algorithm. In addition, the turning manoeuvres are more consistent and smoother than when using the NUS based on the calibration technique discussed. Table 7.12 summarises the performance results for this

test which shows that the robot performance is better with INUS than when using the NUS, resulting in more consistent motion and shorter navigation.

Table 7.12: Performance measures for PS3

Methods	*$\Delta\Omega$ (deg/s^2)*	*t_s(s)*	*d_t(m)*	*Error(m)*	*v_a(m/s)*	*Collision*	*Search (deg)*
NUS	*7.95*	*36.3*	*5.2 [5.63]*	*1.12*	*0.142*	*No*	*0*
INUS	*4.91*	*25.01*	*5.04*	*0.69*	*0.209*	*No*	*0*

Preliminary scenario 4

Figure 7.19 (a) shows the final scenario described in this section, which requires various skills in completing the task. The robot initially searches for the goal located at the end of the room, and it is required to pass through a narrow gap to reach the goal. The trajectory generated by the NUS method and its corresponding control parameters are illustrated in Figures 7.19 (b) and (d) respectively. The results show that the robot manages to detect the position of the goal after its first search attempt. After this the robot heads towards the goal until it encounters the table. While the robot is able to avoid this obstacle safely, the narrow gap generates repulsive forces from both sides which temporarily decrease the robot's velocity. After which, the repulsive forces cancel each other out and the robot again engages with the goal to attain its mission. In the INUS case, as shown in Figure 7.19 (c), the robot begins its task by searching for the goal. After the goal is detected, the robot navigates until it negotiates the passage where it senses the table on its right. After smoothly avoiding this obstacle, the robot passes through the narrow gap and moves towards the goal. Figure 7.19 (e) displays the control parameters of the INUS method, which generates a more consistent angular velocity (w), resulting in a smoother path. The velocity graph reflects the basis of the proposed fuzzy inference system where the velocity (v) of the robot remains stable until it gets closer to the goal, after which it accelerates gradually until the obstacle is sensed. Once the obstacle is avoided the system progressively increases the velocity (v) as the goal is approached. In the final stage, the velocity remains stable as the robot is directed to achieve its goal. The matching strength, on the other hand, decreases dramatically in the final part of the navigation scenario. This is because of the reduced field of view

with respect to zooming, and excessive speed may prevent the robot from capturing complete images during the final cycles.

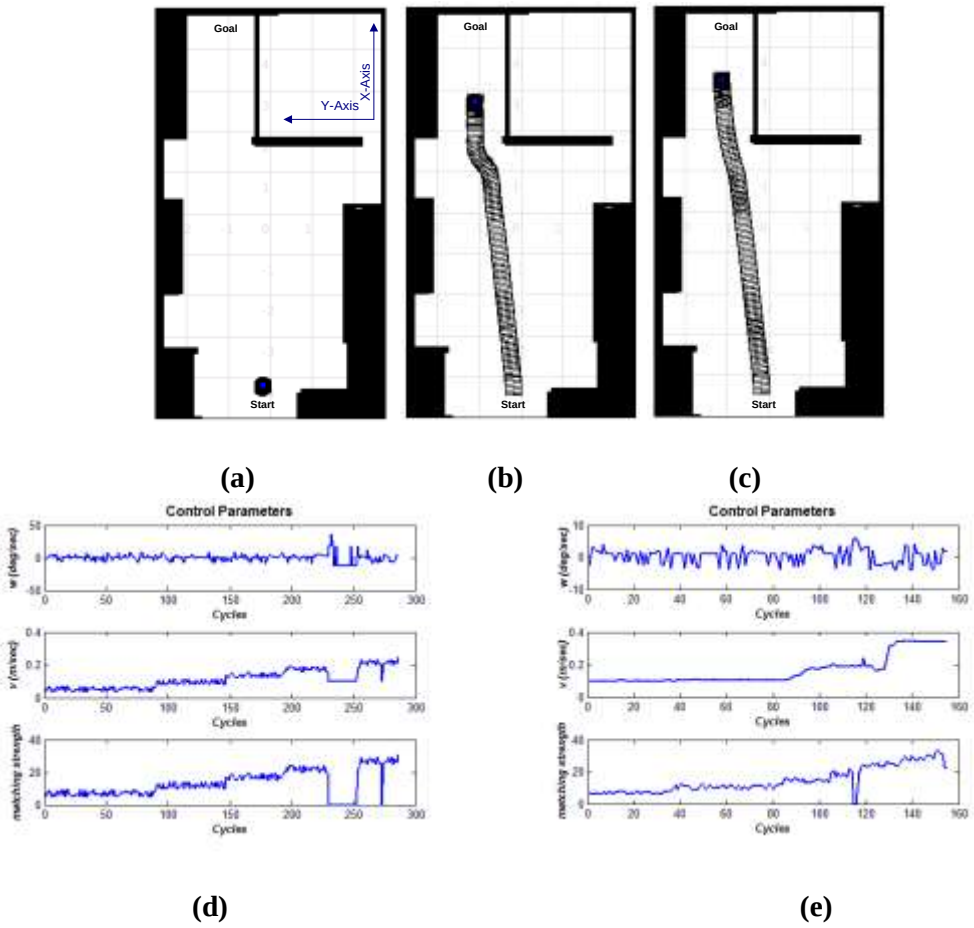


Figure 7.19: Experimental results for PS3, (a) The scenario, (b) Estimated Trajectory with NUS , (c) Estimated Trajectory with INUS, (d) Control Parameters for INUS, (e) Control Parameters for INUS

Table 7.13: Performance measures for PS4

Methods	*$\Delta\Omega$ (deg/s^2)*	*t_s(s)*	*d_t(m)*	*Error(m)*	*v_a(m/s)*	*Collision*	*Search (deg)*
NUS	*7.17*	*57.4*	*7.7 [8.22]*	*1.23*	*0.134*	*No*	*15*
INUS	*5.32*	*45.5*	*7.79*	*0.61*	*0.171*	*No*	*15*

It can be observed that the decrease in matching strength during this final stage does not influence overall performance due to the design of the fuzzy inference system, as discussed in Chapter 5. Table 7.13 presents the performance results utilising both

methods, showing that the INUS navigation method has superior performance compared to NUS and confirming that the INUS offers smoother, and more consistent motion throughout the navigation. The experimental results obtained from the preliminary scenarios have demonstrated that, when employing the INUS method, the robot's performance was enhanced in achieving the desired goals. The following section introduces more complex scenarios.

7.3.1.2 Complex test results

This section describes the more complex scenarios examined, where different obstacles are placed along the robot's path so as to influence its ability to reach its goal. The robot's ability to negotiate successive goals is also evaluated where the robot needs to detect each goal. Six different scenarios were designed to simulate situations that the robot may experience when navigating in an indoor environment. The successive scenarios involve increased levels of complexity.

Complex Scenario 1 (CS1) – requires the robot to navigate towards its goal whilst an external obstacle is located in its path, where the initial heading angle (θ) = 180, as shown in Figure 7.20 (a).

Complex Scenario 2 (CS2) –the robot has to negotiate two external obstacles located in its path, as illustrated in Figure 7.20 (b).

Complex Scenario 3 (CS3) – requires the robot to navigate towards its goal with three obstacle are located in its path, as shown in Figure 7.20 (c).

Complex Scenario 4 (CS4) – requires the robot to negotiate and pass through a narrow gap towards its goal, as depicted in Figure 7.20 (d).

Complex Scenario 5 (CS5) – represents a task that demonstrates the robot's ability to avoid obstacles of two different sizes, as shown in Figure 7.20 (e).

Complex Scenario 6 (CS6) – represents a task that demonstrates the robot's ability to navigate towards successive images, as shown in Figure 7.20 (f).

Table 7.14 displays the starting and goal positions in each scenario, with each starting set to (0.0,0.0)

Table 7.14: Definition of complex scenarios

Scenario	*Start (x,y)*	*Goal (x,y)*
CS1	*(0.0,0.0)*	*(8.0,0.0)*
CS2	*(0.0,0.0)*	*(8.0,0.0)*
CS3	*(0.0,0.0)*	*(8.0,0.0)*
CS4	*(0.0,0.0)*	*(5.0,-1.0)*
CS5	*(0.0,0.0)*	*(6.0,-1.0)*
CS6	*(0.0,0.0)*	*(7.0,1.0;7.0,0,0;7.0,0.0)*

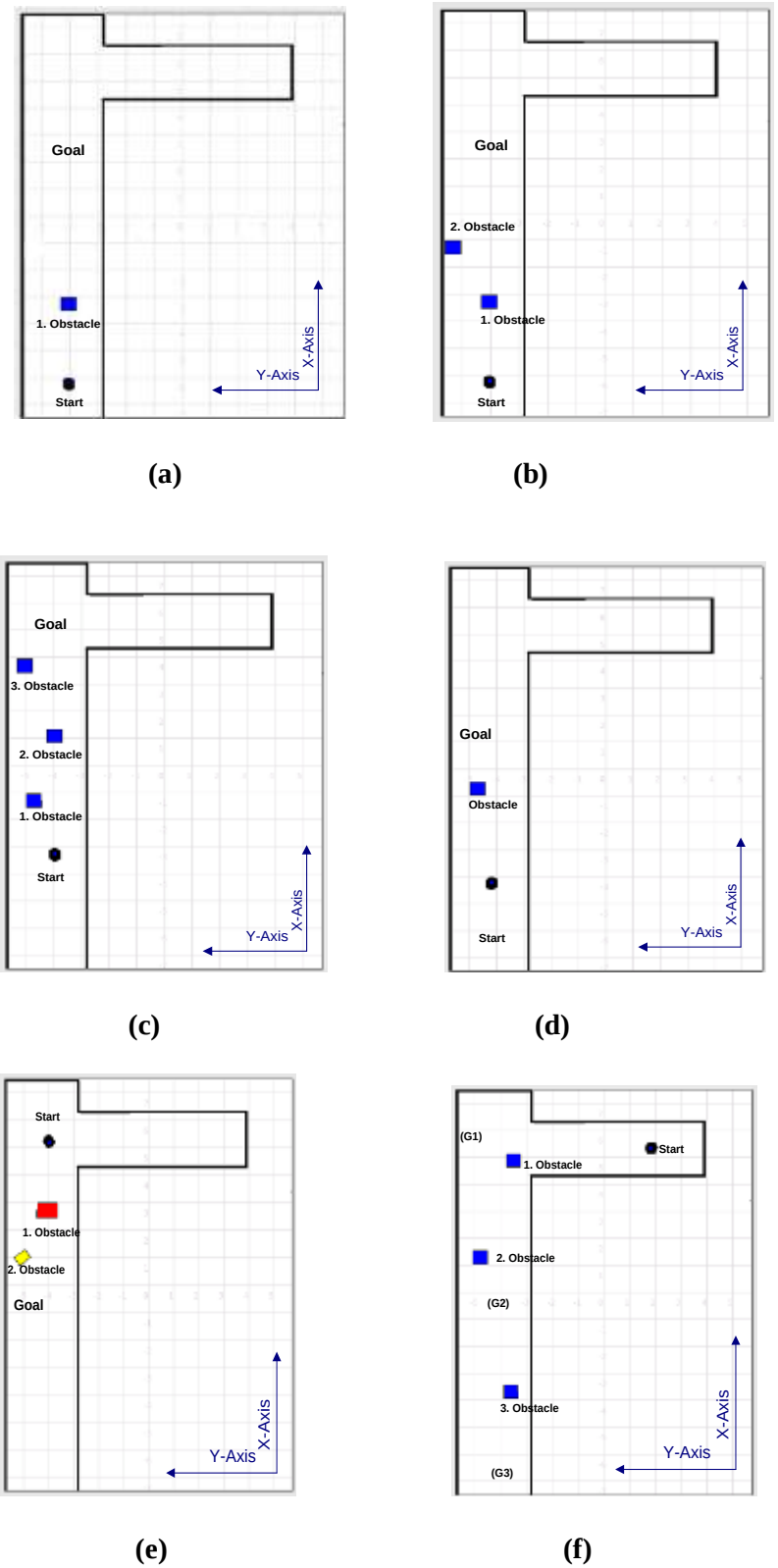
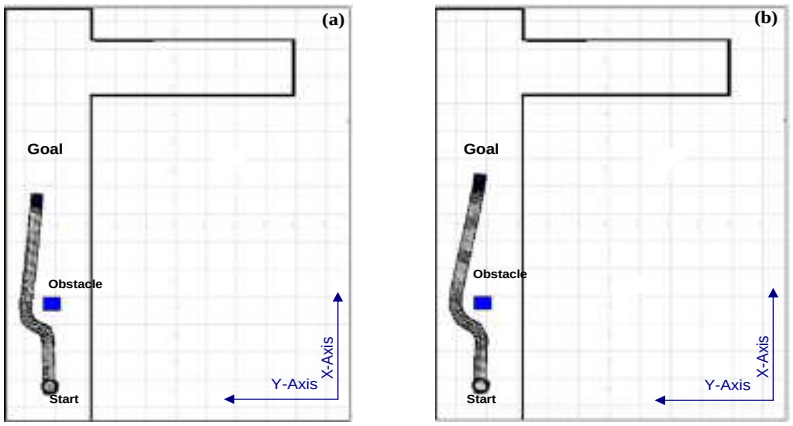
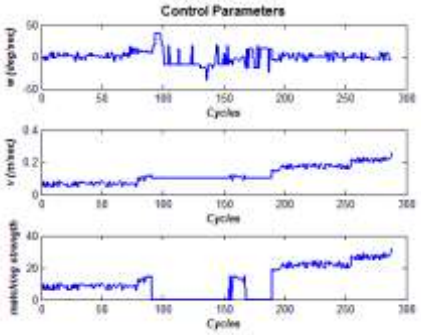


Figure 7.20: Complex scenarious 1 – 6 , (a) CS1 - heads in reverse direction with one obstacle, (b) CS2- two obstacles, (c) CS3 - three obstacles, (d) CS4 – in narrow passage, (e) CS5 - different obstacles (f) CS6 - multible goals with three obstacles

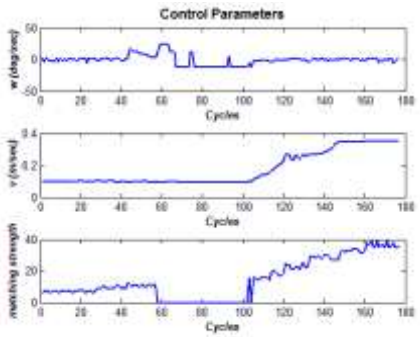
Complex scenario 1

The scenario demonstrates the robot's ability to negotiate with the goal, which is located out of the field of view of the robot. In addition, it needs to avoid an external obstacle which is positioned so as to obstruct the robot's path.





(c)



(d)

Figure 7.21: Experimental results for CS1, (a) estimated trajectory with NUS , (b) estimated trajectory with INUS, (d) control parameters for NUS, (e) control parameters for INUS

Figure 7.21 (a) illustrates the trajectory of the robot using the NUS to avoid the obstacle. The robot initially starts searching for the goal along its initial heading direction, which ends in failure. Thus, according to the characteristics of its search behaviour, the robot performs a 180° turn to change its search direction (see Chapter 4). Subsequently it detects the goal, and starts approaching it until it senses the obstacle. The robot under NUS is able to avoid the obstacle safely. However repulsive forces from both the obstacle and the wall compel the robot to perform extra manoeuvres to prevent collision, as shown in Figure 7.21 (c). After passing through the gap, the robot continues approaching the goal to complete its task, as shown in Figure 7.21 (a).

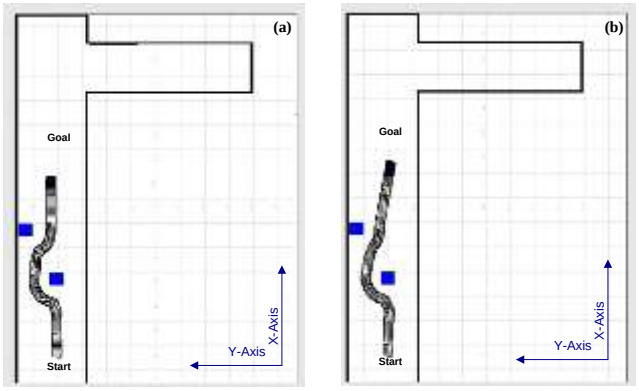
Table 7.15: Performance measures for CS1

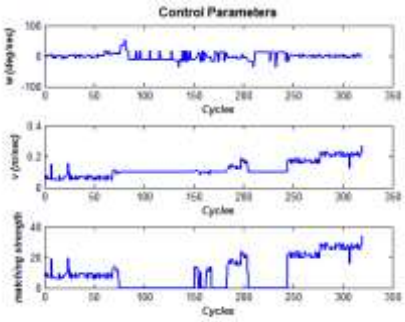
Methods	*$\Delta\Omega$ (deg/s^2)*	*t_s(s)*	*d_t(m)*	*Error(m)*	*v_a(m/s)*	*Collision*	*Search (deg)*
NUS	*8.01*	*63.6*	*8.1 [8.91]*	*1.39*	*0.128*	*No*	*15*
INUS	*6.29*	*51.2*	*8.42*	*0.59*	*0.166*	*No*	*15*

Under INUS, the robot performs the same searching procedure to detect the goal, as shown in Figure 7.21 (b). Nevertheless, its avoidance of the obstacle is smoother. As the obstacle is approached on the right side of the robot, the system is able to adjust its speed in a progressive way and handle the repulsive forces coming from both the wall and the obstacle by using the trained obstacle avoidance technique (see Chapter 5). Once the wall and obstacle disappear from the field of view of the sensors, the system increases the velocity (v) sharply whilst simultaneously decreasing the turning rate (w) as the robot approaches its goal with no obstacles in its field of view, as shown in Figure 7.21 (d). This allows the robot to complete its task. Table 7.15 displays the performance measurements of both algorithms in reaching the goal. The results are quite similar to those of PS4, where the robot's performance with INUS is seen to be more consistent and smoother than when using NUS.

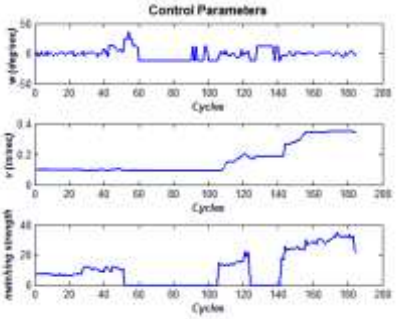
Complex scenario 2

This scenario requires the robot to avoid two obstacles, as shown in Figure 7.20 (b). The robot is able to successfully reach its goal without collisions using both methods, as shown in Figure 7.22 (a) and Figure 7.22 (b) respectively.





(c)



(d)

Figure 7.22: Experimental results for CS2, (a) estimated trajectory with NUS , (b) estimated trajectory with INUS, (d) control parameters for NUS, (e) control parameters for INUS

The INUS control generates smoother motion than NUS which uses a simple method of controlling velocity that employs matching strength as the only input, as previously mentioned. Thus the system progressively increases the speed of the robot until it gets closer the obstacle. Therefore, despite the assignment of an appropriate d_o value, the system may need to perform sharp manoeuvres to prevent collision. For instance, in this case, the robot increases its speed progressively with respect to matching strength until it gets close to the first obstacle, which results in a wider trajectory. In addition, the results for the corresponding avoiding manoeuvre increases the possibility of a frontal collision with the second obstacle. In negotiating the second obstacle, the obstacle avoidance behaviour is invoked, leading to the performance of an avoiding manoeuvre. Subsequently, the robot continues to approach its goal with no obstacle in its field of view, and reaches the goal. The corresponding control parameters are shown in Figure 7.22 (c). Again the INUS, illustrated in Figure 7.22 (b), allows the robot to maintain smoother and faster navigation. This is because this method is able to make more sensitive adjustments leading to a more robust heading direction based on its calibrated camera system. Under the INUS, when the robot initially perceives the first obstacle, the system progressively adjusts its speed as part of the collision avoidance manoeuvre, and the trained avoidance technique generates appropriate values of angular velocity (w) to compel the robot to move at a safe distance from the obstacle.

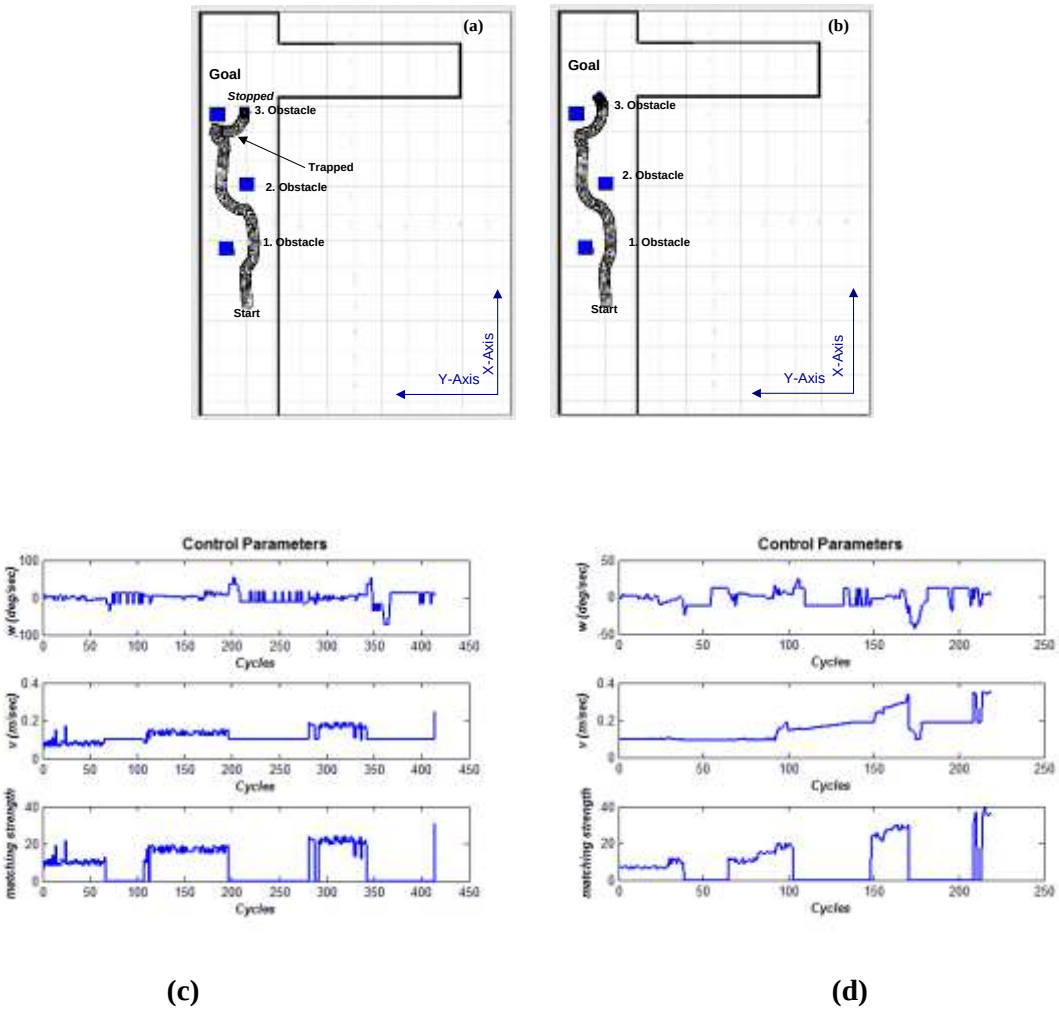
After the first obstacle and the wall disappear from the robot's view, its velocity increases progressively as illustrated in Figure 7.22 (d) until it perceives the second obstacle on its left, which is passed safely and smoothly. The robot then proceeds to complete its task successfully. Table 7.16 presents the results of the performance measurement tests regarding this scenario which reveal that the robot performs better using the INUS.

Table 7.16: Performance measures for CS2

Methods	*$\Delta\Omega$ (deg/s^2)*	*t_s(s)*	*d_t(m)*	*Error(m)*	*v_a(m/s)*	*Collision*	*Search (deg)*
NUS	*9.24*	*70.7*	*8.6 [9.2]*	*1.34*	*0.124*	*No*	*0*
INUS	*6.83*	*57.2*	*8.9*	*0.73*	*0.160*	*No*	*0*

Complex scenario 3

This scenario represents a more cluttered environment involving three external obstacles. The robot is required to navigate from its start point to its goal with obstacles positioned so as to obstruct its path.



(c) **(d)**

Figure 7.23: Experimental results for CS3, (a) estimated trajectory with NUS , (b) estimated trajectory with INUS, (d) control parameters for NUS, (e) control parameters for INUS

The robot running under the NUS fails twice in this complex experiment by colliding with obstacles. The NUS trajectory shown in Figure 7.23 (a) displays a successful result where the goal is attained without colliding. The first obstacle is passed successfully although false matches are processed during navigation, as shown in Figure 7.23 (c)

where extreme points can easily be observed. The system then moves towards the goal until it perceives the second obstacle on its right side that appears within the range of d_o, which it avoids successfully. The final goal for the robot using the NUS algorithm is having to avoid the third obstacle located on its left side. However, the goal is obstructed by the obstacle which results in the robot moving close to it and pushed towards the wall by strong repulsive forces. In negotiating the wall, the robot rotates sharply clockwise towards the open space, which results in an unclear trajectory. Besides this, the robot loses sight of the goal and stops. The search behaviour is then reactivated, the goal is detected and the scenario is completed successfully. Two repeated failures of the NUS method occurred in this part of the experiment where the robot was unable to overcome the trap situation. The control parameters of the method are illustrated in Figure 7.23 (c) which displays the corresponding results along the estimated trajectory. Velocity, *v*, is rather low during navigation, and the change in *w* is quite high, resulting in inconsistent and non-smooth robot motion. However, when employing the INUS method, the robot is able to attain its goal successfully as shown in Figure 7.23 (b). It begins moving towards the goal, during which time the system does not increase the *v* as the robot moves towards the first obstacle. As the robot perceives the first obstacle, it is avoided safely with a smooth manoeuvre. As soon as the obstacle disappears from the robot's view, values of *v* increase sharply, which also shows that both the robot reaches the obstacle-free space and the distance between the goal and the robot reduces. When the robot perceives the second obstacle the system decreases *v* in order to engage with it, and the INUS generates a similar avoiding manoeuvre to that of the NUS method. However under the NUS the robot presents a wider path as compared to when employing the INUS. The final part of the task is to negotiate with the third obstacle which compels the robot operating the NUS method to select the wrong path. With INUS, however, the robot continues moving toward the goal until it senses the third obstacle. In negotiating the third obstacle, the robot makes a sharp manoeuvre towards the trap which, however, is compensated for successively and the robot achieves its escape from the trapped position. It then engages its goal again and completes its mission. Figure 7.23 (d) reveals the corresponding control parameters when the robot is able to navigate the goal whilst avoiding external obstacles in a similar manner to the previous experiments. The only exception appears with the trap

position, where the robot oscillates for a while, resulting in a higher change in *w* and decrease in average velocity (*v*). Overall, the robot under INUS succeeds in avoiding all obstacles with smoother and more consistent manoeuvres than when using the NUS. Furthermore, NUS leads to unsuccessful navigation on two occasions and requires further searching to complete the task. Table 7.17 summarises the performance measures for this complex scenario (time spent on the searching procedure is excluded). As with previous results, the robot's performance is better with INUS than when using the NUS.

Table 7.17: Performance measures for CS3

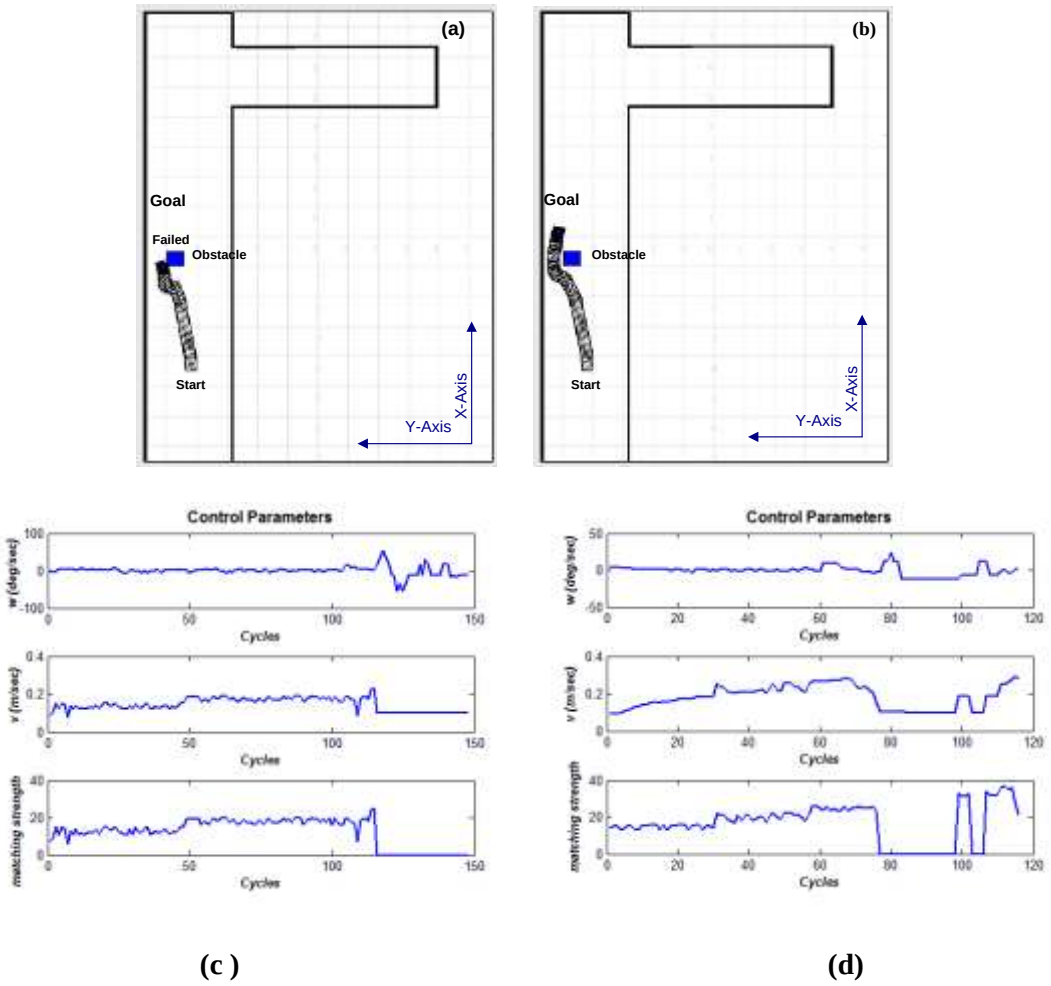
Methods	*$\Delta\Omega$ (deg/s^2)*	*t_s(s)*	*d_t(m)*	*Error(m)*	*v_a(m/s)*	*Collision*	*Serach(deg)*
NUS	*14.4*	*91.4*	*10.4 [10.9]*	*1.37*	*0.114*	*2 times*	*30*
INUS	*9.74*	*66.7*	*9.76*	*0.84*	*0.149*	*No*	*0*

Complex scenario 4

This scenario requires the robot to pass along a narrow path towards the goal. When employing NUS method it is not able to complete the task and collides with the obstacle, as illustrated in Figure 7.24 (a), and corresponding control parameters are shown in Figure 7.24 (c). Under NUS it perceives the obstacle on its right side whilst heading towards the goal, and then performs consecutive left and right manoeuvres due to the repulsive forces generated by the obstacle and the wall respectively. However, it then becomes jammed and collides with the obstacle.

When running under INUS the robot is able to attain its goal successfully, as shown in Figure 7.24 (b). The robot initially rotates clockwise to engage with the goal. Once the goal is perceived, the system progressively increases speed until the robot approaches the pass. It then reduces speed and, despite the narrow gap and repulsive forces generated by the obstacle and the wall, it is able to escape from the trap without collision. After this it continues moving toward its goal to complete the task. Unlike with the NUS method, in this case the robot using the INUS method benefits from the

intelligent speed control and the trained avoidance technique, and is able to pass the gap to safely reach its goal.



(c) **(d)**

Figure 7.24: Experimental results for CS4, (a) estimated trajectory with NUS , (b) estimated trajectory with INUS, (d) control parameters for NUS, (e) control parameters for INUS

Table 7.18: Performance measures for CS4 (Failure for NUS)

Methods	*ΔΩ (deg/s²)*	*t_s(s)*	*d_t(m)*	*Error(m)*	*v(m/s)*	*Collision*	*Search(deg)*
INUS	*5.42*	*32.0*	*5.7*	*0.88*	*0.174*	*No*	*15*

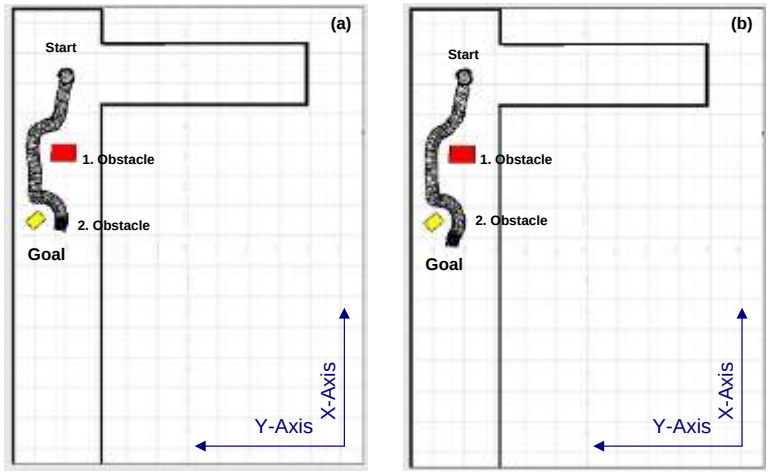
The enhanced obstacle avoidance technique essentially provides smoother angular velocity (*w)* values for narrow paths and the robot is able to negotiate the wall, facilitating safer and smoother avoidance manoeuvres. Figure 7.24 (d) presents the control parameters with INUS which are consistent with the estimated trajectory. Table

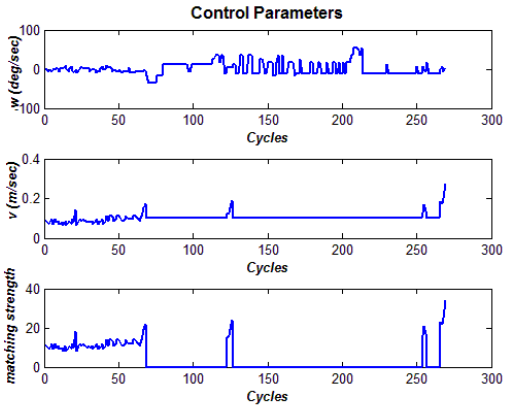
7.18 summarises the performance measures for this trap scenario, and indicates that NUS is unsuccessful and consequently the performance measures associated with this case are indeterminate. However, under the INUS, successful navigation is achieved by the robot.

Complex scenario 5

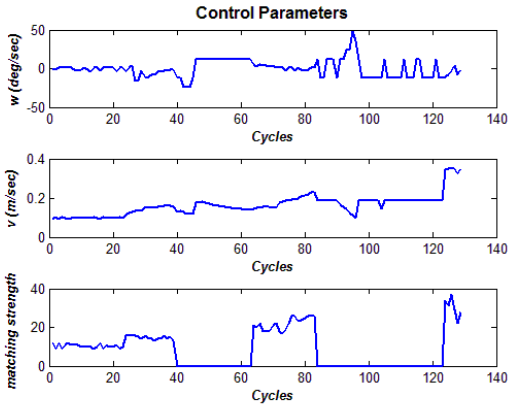
This scenario demonstrates the robot's ability to avoid obstacles of two different sizes positioned so as to obstruct the robot's path. One of these is a rectangular box of 420 mm x 330 mm, and other is a rectangular box having dimensions of 400 mm x 900 mm. In order to increase the challenge inherent in the scenario, the second obstacle is positioned across the path. Figure 7.25 (a) illustrates the trajectory of the robot under the NUS trying to avoid the obstacles. The robot initially starts searching in order to detect the position of the goal, then turns its direction of travel to where it senses the larger obstacle. In negotiating the obstacle, the value of *w* is increased progressively as the robot perceives the obstacle on its right side. After passing the first obstacle, the robot is compelled to pass along the narrow path between the first obstacle and the wall, forcing it to produce a non-smooth trajectory. When the sensors perceive the second obstacle, which is positioned irregularly, the robot performs another avoiding manoeuvre to prevent collision and complete the task. The control parameters of the corresponding method are illustrated in Figure 7.25 (c) in which it can be observed that the robot essentially utilizes obstacle avoidance behaviour during the navigation due to the repulsive forces generated by the obstacles and the wall, and which results in a wider trajectory when using the NUS.

Figure 7.25 (b) displays the trajectory of the robot under the INUS method. After detecting the goal's position, the robot rotates towards it. It then moves towards the goal smoothly until it perceives the first obstacle, whereupon it successfully avoids it and keeps moving towards path between the first obstacle and the wall.





(c)



(d)

Figure 7.25: Experimental results for CS4, (a) estimated trajectory with NUS , (b) estimated trajectory with INUS, (d) control parameters for NUS, (e) control parameters for INUS

With INUS the robot is able to pass along the narrow passage smoothly, and thus the obstacle avoidance behaviour of this method leads to an oscillation-free navigation until the second obstacle is detected. The robot subsequently avoids the second obstacle safely and keeps moving towards the goal. The control parameters of this algorithm are shown in Figure 7.25 (d). In addition, Table 7.19 illustrates the performance assessment of both methods where the overall performance of the NUS method decreases dramatically, especially in terms of v_a and t_s. On the other hand, the INUS method performs with a level of performance somewhat similar to those of previous experiments.

Table 7.19: Performance measures for CS5

Methods	*$\Delta\Omega$ (deg/s^2)*	*t_s(s)*	*d_t(m)*	*Error(m)*	*v_a(m/s)*	*Collision*	*Serach(deg)*
NUS	*13.27*	*61.7*	*6.9 [7.4]*	*1.286*	*0.109*	*1 times*	*-45*
INUS	*8.49*	*40.5*	*6.56*	*0.77*	*0.162*	*No*	*-45*

Complex scenario 6

NUS and INUS were designed primarily to overcome local navigation problems using a vision based approach. Scenario 6, however, is designed to evaluate the performance of these methods with a global navigation problem, in which waypoints are placed at locations so that each goal is successively negotiated, as shown in Figure 7.26. The scenario is illustrated in Figure 7.20 (f). It has three goals placed along the corridor for the robot to navigate around. When the robot reaches its first goal, it starts to search for the second goal, which is then followed by achieving the third goal. The robot must attain all goals in the correct order so as to complete the task successfully. The robot, however, fails on two occasions when using NUS, colliding with the second obstacle which then obstructs the third goal. Figure 7.27 (a) illustrates the trajectory of the robot under the NUS which is eventually able to negotiate the goals successfully.

Figure 7.26: Each goal leads its successor

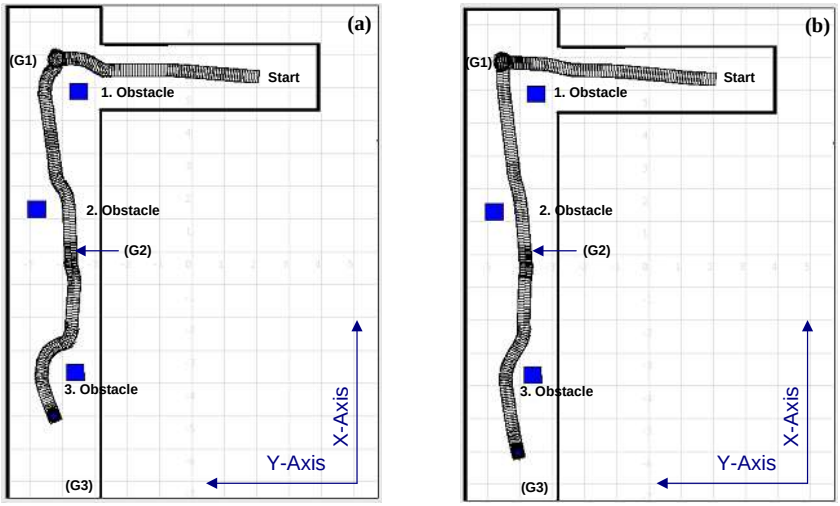


Figure 7.27: Experimental results for CS5, (a) estimated trajectory with NUS , (b) estimated rajectory with INUS, (d) control parameters for NUS, (e) control parameters for INUS

The robot starts its navigation by searching for the first goal which is detected after panning the camera clockwise. The robot then rotates to its right to engage with the goal. The robot moves towards the goal until it perceives the first obstacle, whereupon the robot avoids it and continues moving towards its goal. It then reaches the goal and completes its first task, whereupon the robot starts searching for the second goal, subsequently turning counter-clockwise. While it attempts to move in the direction of this goal, it is influenced by the repulsive forces generated by the first obstacle, the first obstacle appears within the range of d_o, invoking the obstacle avoidance behaviour of the robot which pushes the robot away from its starting position. The robot then localizes itself towards the goal again and continues moving until perceiving the second obstacle. In negotiating this obstacle, the robot avoids it and orients itself towards the direction of the goal. The robot then proceeds to complete its second task successfully.

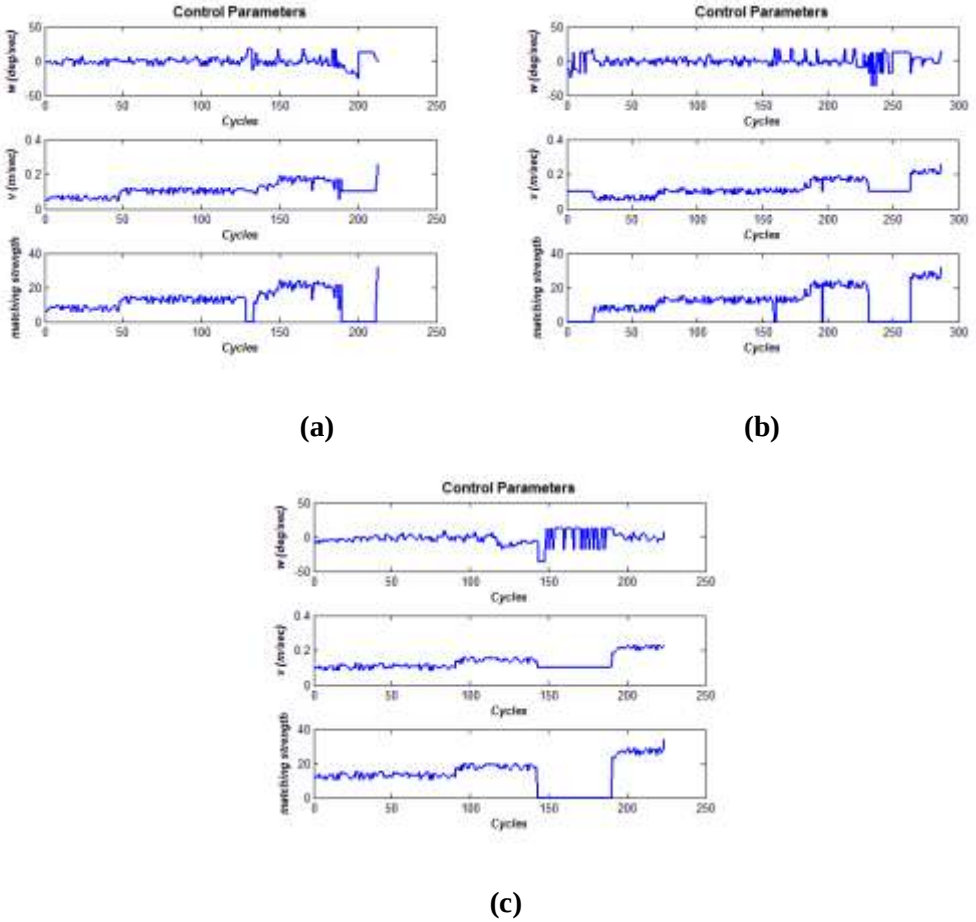
The final goal is eventually detected and the robot moves towards it until the third obstacle is perceived on its right. Having detected the obstacle in its path, the robot performs a sharp avoidance manoeuvre to avoid collision. After the robot successfully avoids the obstacle, it resumes its path and continues moving to achieve its goal. The corresponding control parameters for each goal are illustrated in Figure 7.28. The only criteria for stopping in the NUS method is matching strength. Any mismatching can stop the robot before the goal position is reached which is one of the main disadvantages of the method. The last part of this test is an excellent example of this situation in that the robot is halted away from the goal regarding the matching based stopping criteria.

Figure 7.27 (b) displays the trajectory of the robot when negotiating the scenario using INUS. The results of the corresponding control parameters are shown in Figure 7.29. After detecting the first goal the robot turns clockwise to engage it. It then moves towards its goal. The robot successfully passes between the obstacle and the wall towards its goal. Unlike with the NUS method, the robot with the INUS method does not negotiate the obstacle, resulting in a smaller *Error* parameter than when using the NUS method. Consequently, a smaller *Error* result for the reaching the first goal prevents any influence of the first obstacle during the search attempt for the second goal.

The robot starts searching for the second goal and then rotates counter-clockwise to its left to engage it. Once engaged, it heads towards the goal until it perceives the second obstacle on its right where INUS maintains a safe distance from second obstacle and providing a robust and smooth trajectory towards the corresponding goal. The robot then proceeds to complete its task successfully; the control parameters of the corresponding goal are illustrated in Figure 7.29 (b). For the final goal, the trajectory results are similar to those for the NUS method except that the robot's trajectory is more erratic. Having detected the final goal on its initial heading direction, the robot starts moving towards the goal. Since the estimated distance between the goal and the robot is reduced, the value of v is increased. The robot keeps moving until it perceives the third obstacle which obstructs the goal. As the robot senses the obstacle in its path, v is

decreased by the system in order to provide a safe obstacle avoidance manoeuvre. After the obstacle disappears from the robot's field of view, the robot localizes itself towards the goal and then proceeds to complete its task successfully, as shown in Figure 7.29 (c).

The results show that the robot performs better when using the INUS. The performance measurements displayed in Table 7.20 indicates that using the NUS method yields somewhat higher values of d_t, t_s $\Delta\Omega$ and *Error* and lower values of v_a than when employing the INUS, resulting in a longer travel distance and navigation, inconsistent motion, higher position error and slower speed.



(a) **(b)**

(c)

Figure 7.28: Control parameters for NUS of CS5, (a) Control parameters for Goal A , (b) Control parameters for Goal B, (c) Control parameters for Goal C

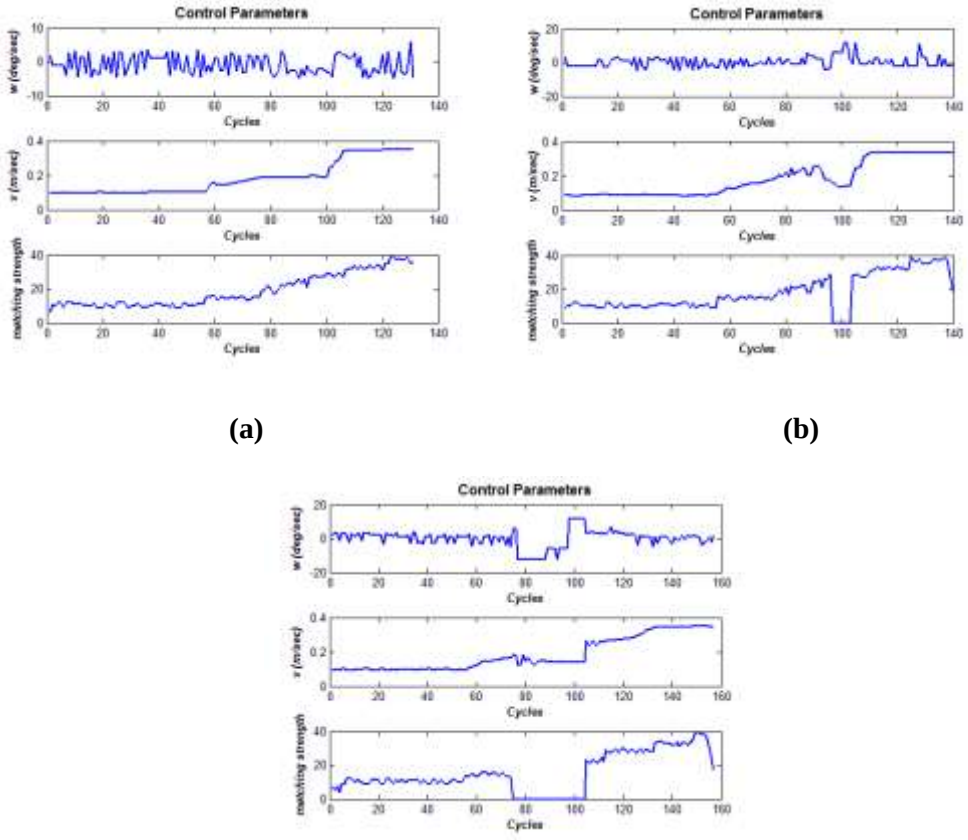


Figure 7.29: Control parameters for INUS of CS5, (a) Control parameters for Goal A , (b) Control parameters for Goal B, (c) Control parameters for Goal C

Table 7.20: Performance measures for Complex 6

Methods	*ΔΩ (deg/s²)*	*t_s(s)*	*d_t(m)*	*Error*	*v_a(m/s)*	*Collision*	*Search(deg)*
NUS	*10. 47*	*161.0*	*21.73 [23.57]*	*(1.513)*	*0.137*	*2 times*	*(-15.90,0)*
INUS	*6.33*	*124.5*	*21.7*	*(0.56)*	*0.175*	*1 times*	*(-15,90,0)*

7.3.2 Performance Analysis

The quantitative results of the experiments are presented in Tables 7.10 - 7.13 for the preliminary scenarios and Tables 7.15 - 7.20 for the complex scenarios, and the evaluation data obtained from the repeated experiments are summarised. The following discussion briefly analyses the performance of each method:

Navigation Using SIFT (NUS) - the preliminary navigation test results reveal that for the PS1-PS4 scenarios, the robot employing the NUS method was able to complete its mission without colliding with obstacles. For the CS1 and CS2 scenarios the robot was also able to complete the tasks successfully, but the performance was somewhat poorer for the more complex scenarios as compared with INUS in terms of the d_t, t_s, $\Delta\Omega$, *Error*, *speed* and *Collision* parameters. For the CS4 scenario, under NUS the robot failed to negotiate the narrow gap. Since velocity control in NUS is adjusted according to matching strength, this allowed the robot to move closer to the obstacle with an unstable velocity, and which in turn led to an increased possibility of collision in the complex scenarios such as CS3, CS4, CS5 and CS6. Having a less accurate control algorithm provides an inconsistent angular velocity with high values of $\Delta\Omega$ and a higher travel distance d_t.

Navigation is deemed less successful, if the robot navigates with a high *Error* value. The criterion for bringing the robot to a stop involves comparing the current matching strength with a previously determined threshold value (see Table 7.7). Despite the preliminary experiments to obtain an appropriate threshold value for the stopping criteria, the variable lighting conditions in the environment and the distortion produced in the images due to the motion of the robot may produce inconsistent matching results, thus increasing the value of the *Error* parameter. For instance, in scenario CS6 (c), the robot completed its mission with a high *Error* due to stopping early based on matching strength. For all scenarios, the NUS method generated higher values of the *Error* parameter than when using INUS. This is because the INUS method utilizes a more reliable stopping procedure, which estimates the distance to the goal using matched key features. As previously mentioned, the robot was unable to navigate out of the trap in CS4, although it was successful when using INUS.

CS3 presents a challenging task for this NUS method and the robot failed to complete the on two occasions. Furthermore searching behaviour in the successful attempt was enabled once more by the system having lost sight of the target after avoiding the third obstacle which obstructed the path of the robot during its navigation. The overall performance of this scenario in terms of the values of t_s, d_t, $\Delta\Omega$ and *Error* parameters

are higher than when using the INUS method. Once again, this produced a greater navigation time and travel distance, and an inconsistent and less smooth trajectory compared to the INUS. CS5 also presented a challenging scenario which involves two different sizes of obstacle obstructing the path. Despite the robot failing once, it was able to complete the task on four occasions. However, the results indicate that the robot produced small values of v_a and high *Error*, resulting in the increased navigation time and decreased precision respectively.

Intelligent Navigation Using SIFT (INUS) - the navigation tests demonstrated that INUS performed better in all test scenarios than NUS for all parameters, for the reasons previously discussed. Table 7.21 highlights the percentage improvement in performance of the INUS over the NUS for the Pioneer robot.

Table 7.21: Performance improvement of the INUS over the NUS

Scneario No	*shorter time*	*shorterl distance*	*consistent motion*	*safer navigation*	*faster speed*
PS1	***%22***	***%3***	***%38***	***%0***	***%45***
PS2	***%20***	***%2***	***%23***	***%0***	***%42***
PS3	***%31***	***%10***	***%37***	***%0***	***%47***
PS4	***%24***	***% 5***	***%26***	***%0***	***%27***
CS1	***%20***	***%6***	***%21***	***%0***	***%29***
CS2	***%19***	***%5***	***%30***	***%0***	***%28***
CS3	***%28***	***%12***	***%33***	***%40***	***%30***
CS4					
CS5	***%34***	***%12***	***%33***	***%20***	***%47***
CS6	***%37***	***%28***	***%39***	***%20***	***%32***

% improvement in CS4 could not be determined since the robot using the NUS failed to reach its goal

The INUS navigation method consistently offers better overall performance compared to NUS in terms of safety and success, lower travel distance and time, consistent motion commands, and a smooth trajectory. The robot with the INUS method was able to pass through the narrow gaps, as illustrated in CS4. This is a scenario in which the NUS method completely failed. Furthermore, the INUS navigation method consistently performed somewhat better than when using NUS in CS3, CS5 and CS6 in terms of

safer navigation. In order to evaluate the limits of the robot using INUS, one additional test including a global navigation problem was designed and performed. The robot was required to complete a task moving from the corridor area outside of the laboratory to the main entrance of the building.

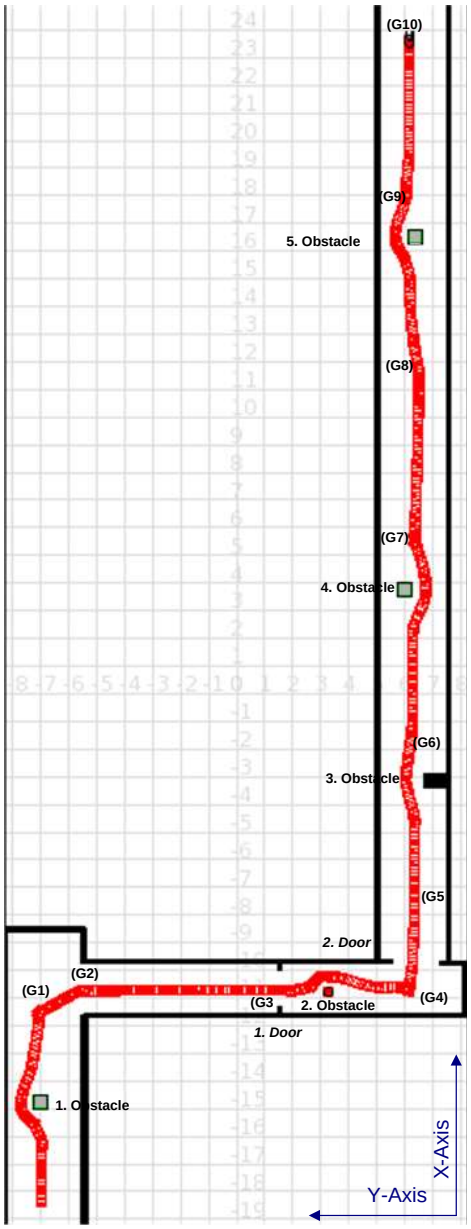


Figure 7.29: Estimated trajectory with INUS for CS7

The robot was required to avoid deliberately positioned obstacles and to detect the corresponding goals in order to reach the final destination. The main aim of this

experiment was to complete the task without colliding with any walls or obstacles. The experiment was repeated six times to evaluate the sustainability of the system. The results revealed that the robot under INUS was able to complete the task four times, and more details of the scenarios are included in Appendix P. Failures occured at the first and second doors. Despite the challenge of conducting such an axperiment, the robot achieved four times to complete the required task without experiencing any collision. However, the results suggest that the proposed system needs a more powerful localization technique for such a complex problem, which should be incorporated into a map based strategy to allow the robot to complete tasks in large and complex environments such as hospitals.

Overall, the navigation results for the robot using INUS are very promising, since they confirm that the proposed navigation system functions as intended. The results also demonstrate that the INUS method is able to overcome problems likely to be associated with vision based mapless navigation on a real mobile robot platform functioning in a real world scenario, without encountering serious problems such as collisions. The robot can complete its missions in a robust, speedy and smooth manner. On the other hand, the NUS navigation system can indeed be implemented on different mobile platforms, and is able to negotiate new environments much more easily than is the case when implementing the INUS navigation system. This is because of the simplicity and flexible characteristics of the NUS method.

7.4 Summary

This chapter is divided into two parts. The first part presented the navigation test results using a mobile robot platform employing conventional vision based obstacle avoidance and the proposed hybrid vision based obstacle avoidance system. A comprehensive series of experiments was conducted, and the results demonstrate that the proposed Hybrid System produced better obstacle avoidance performance compared to the conventional Optical Flow method.

The second part of the chapter revealed the performance of the proposed SIFT based navigation algorithms implemented on a mobile robot platform. The preliminary experiments focused on the basic achievements of each algorithm in different scenarios. The navigation results for complex scenarios focused on more advanced situations that a real robot might encounter in indoor environments. The results from all scenarios demonstrate that the *Intelligent Navigation using SIFT* produced better navigation performance compared to the *Navigation using SIFT* algorithm. The INUS has been shown to provide smoother trajectories and more successful navigation on different situations. The tests also confirm that using the proposed navigation algorithms the robot can easily navigate in different indoor environments with minor modifications to complete their tasks successfully.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

This chapter presents a summary of the research, including the achievements of the study along with a discussion of future work that could be undertaken to enhance the performance of vision based mobile robot navigation using mapless strategies.

8.1 Conclusions

The first hypothesis of this research asserts that "it is possible to develop a vision based obstacle method, using a single monocular vision camera as the only sensor to allow a mobile robot to navigate safely". To test this hypothesis, a novel navigation system is proposed, and an attempt is made to realise this system using the Player architecture so as to give the capability of it being integrated into a mobile robot platform. This has been confirmed as follows.

The computational complexity of the vision algorithms and the cost of the sensors are the most critical aspects for real time applications. Monocular vision is a good cost effective solution and is able to support vision based navigation and obstacle avoidance. The study set out to develop reactive obstacle avoidance using a single monocular vision camera as the only sensor. The system integrates a high performance appearance-based obstacle detection method into an optical flow based navigation system. The main motivation behind this approach was the observation of limitations in conventional methods in terms of accuracy and efficiency. For instance, optical flow methods suffer from illumination problems and are sensitive to noise and distortion [Contreras, 2007]. On the other hand, appearance-based methods are highly sensitive to floor imperfections, as well as to the physical structure of the terrain. To overcome these problems, a hybrid architecture was adopted in the proposed control system, which combines optical flow and an appearance-based method. The first step of the proposed

architecture was to design an optical flow based system. Accordingly, a multi-scale version of Horn and Schunk's [1981] optical flow estimation algorithm was employed to calculate flow vectors. The conventional balance strategy [Duchon et al., 1998; Temizer, 2001] was utilised as the control law to adjust the steering direction of the robot. The method was integrated into a behavioural architecture inspired by the subsumption architecture [Brooks, 1986]. Subsumption architecture selects the behaviour according to a defined priority. This was employed to provide a possible solution to the high-level coordination of different behaviours. The next step of the proposed hybrid architecture focussed on the design of an appropriate appearance-based obstacle detection system in HSV colour space. According to the algorithm, the image is divided into clusters and each of the clusters is compared with a template image illustrating a free path. From the results of the comparison, corresponding clusters are allocated as either free or occupied in constructing a binary map.

The hybrid architecture was designed and implemented to run both methods simultaneously, and is able to combine the results of each method using an arbitration mechanism. The proposed strategy successfully fused two different vision based obstacle avoidance methods using this arbitration mechanism in order to permit a safer obstacle avoidance system. The Microsoft Robotics Studio (MRS) environment provided an ideal simulation tool and was found to be particularly useful during the early stages of this work. Accordingly, to establish the adequacy of the design of the obstacle avoidance system, a series of experiments were conducted in this simulation tool. The results demonstrate the characteristics of the proposed architecture and the results prove that its performance is somewhat better than the conventional optical flow based architecture. The results provided encouragement to conduct real world experiments in order to evaluate the capability of the system. Accordingly it has been rigorously evaluated and tested on the Pioneer 3D-X mobile robot in real conditions. Experimental comparison with the conventional Optical Flow Based method (OFB) demonstrated that the robot employing the Hybrid (FT) technique performed better than the conventional technique, as expected. It was also found that the FT technique is able to negotiate and avoid walls and doors, benefiting from the results of the OFB technique utilizing frontal optic flow to estimate the so-called time-to-contact before a frontal

collision is likely to occur. In addition, the FT avoids lateral obstacles in a more smooth and robust manner than when using the conventional OFB technique. Despite this important contribution to solving the vision based obstacle avoidance problem, the method is still vulnerable to lighting conditions and floor imperfections.

The second part of this research focused on goal based navigation architectures using the monocular vision as the primary sensor. The second hypothesis of is that "it is possible to develop vision based mapless navigation using a behaviour based framework to allow a robot to safely complete its tasks in a robust and smooth manner". The mapless navigation technique and methodologies developed resemble human behaviour more than other approaches. Humans are able to position themselves in an absolute way, and reach a goal position with notable accuracy by repeating "a look at the target and move" type of strategy [DeSouza and Kak, 2002]. Visual tracking is a way of mimicking this human behaviour when considering real-time applications in mobile robot navigation problem. After extracting robust interest points, which are related to necessary movement commands, using a mode of operation called visual servoing.

The SIFT algorithm was introduced by Lowe [2004], being one of the strongest feature extraction algorithms. SIFT features are demonstrably invariant to translation, scaling and rotation in images. They are also highly distinguishable from one another and somewhat invariant to illumination changes. These properties make them suitable for the purposes of visual based navigation, where a particular image feature changes in appearance due to changes in image position between current and snapshot locations. The conventional SIFT algorithm is not suitable for real time applications due to its high dimensionality and computational complexity. One of the significant contributions of this study was made by integrating the conventional SIFT algorithm into a monocular vision based navigation system. Accordingly, an accelerated version of the algorithm was adapted for the proposed navigation system which exhibits a higher degree of parallelism, resulting in a performance gain of around 50%. The improved feature extraction technique was employed by a novel and simple control strategy in which the technique requires a single forward-looking camera with no calibration. The algorithm is entirely qualitative and does not require a map or any fundamental matrix to calculate

the control variables. To increase the functionality of the technique, it was integrated into a behaviour based mobile robot navigation system which is capable of extracting and interpreting data and realising a task in an indoor environment without human intervention. However, it is difficult to formulate reactive behaviour quantitatively. In this case, a subsumption architecture was applied to the system which selects behaviour according to a defined hierarchy of priorities via arbitration [Brooks, 1986]. Several behaviours were defined for the navigation system. One of these is obstacle avoidance behaviour, which employs a 2D laser range finder device to detect obstacles and provides a simple but efficient avoidance manoeuvre. The search behaviour, allows the robot to search for a goal in the environment by enabling the pan and zoom features of the camera. A series of successful experiments was designed and tested in the Microsoft Robotics Studio (MRS) environment so as to assess the performance of the proposed system.

In addition to this SIFT based navigation architecture, another major contribution was in the design and implementation of an intelligent SIFT based navigation system. This utilizes the same behavioural architecture but was amended by soft computing techniques. To build an intelligent control system able to control reactive behaviors in unknown environment is challenging for real-time applications. Accordingly, in the proposed navigation architecture several components of the soft computing field, including neural networks, cluster analysis and fuzzy logic, were applied to enhance the overall performance of the system. The first enhancement was achieved by eliminating mismatched features according to scale parameters. To remove those features, a K-means clustering algorithm was employed, resulting in relatively good outcomes in terms of the accuracy and efficiency of matched points. Feature extraction is often followed by a correspondence search which is employed to estimate the next possible heading direction in this work. However establishing and maintaining the correspondence is not easy. Artificial neural networks can be used to represent complex nonlinear functions by training a connected network. To estimate the heading direction, a multi-layered feed-forward network was trained. Furthermore a robust distance estimation technique was proposed for any single monocular vision camera which has a zoom capability. It has been previously proven that there is a strong relationship

between the quotients of the keys' scale parameter and its distance [Sjöö et al., 2009]. According to this approach a multi-layered feed-forward neural network was designed, which has the scale parameter and zoom factor as input and then yields the approximate physical distance to the goal. Another key enhancement was carried out by employing a neural network technique to compensate for the previously defined weaknesses of the obstacle avoidance behaviour. To do this, a number of predetermined paths were designed, and the network trained under the guidance of a human user. The enhancement in avoidance behaviour was validated during the experiments. Estimation of the linear velocity is also a challenging task, especially in vision based navigation systems, but it is essential for both smoother and safer navigation. Fuzzy logic is an important technique for handling control problems that are difficult to analyse with qualitative techniques, and fuzzy rule based systems appear to be one of the best models to represent expert knowledge and in handling uncertainty. In this work, the control of velocity depending on the robot's position and current environmental conditions was achieved using a fuzzy inference system where the fuzzy rules were used to dynamically generate velocity. The results of the simulations revealed that the intelligent strategy allows the robot to successfully complete its goal without experiencing any collision.

Navigation using SIFT (NUS) and Intelligent Navigation using SIFT (INUS) methods were rigorously evaluated and tested on the Pioneer 3D-X mobile robot under real conditions. The experimental results for the robot using INUS were very promising, and demonstrated that the robot can complete its missions in a robust and smooth manner. Under NUS the robot succeeded in completing the preliminary tasks and several of the complex scenarios, although performance was somewhat poorer for more complex scenarios as compared with INUS. The experimental comparison demonstrated that the robot employing the INUS performed better in terms of safe and successful navigation, shorter travel distance and navigation time, smoother trajectory and consistent motion.

8.2 Summary of Achievements

The work described in this dissertation includes the following seven achievements, namely:

- The development of an optical flow based navigation architecture, employing a multi scale optical flow estimation technique and the design of a hybrid vision based obstacle avoidance system integrating an appearance-based obstacle detection system into this navigation architecture.

- Developing a control algorithm which uses only the distribution of matched features obtained from the Scale Invariant Feature Transform (SIFT) algorithm to steer the robot to all desired goals. As well as this, a set of suitable behaviours were developed where the data from different sensors are accepted as inputs, and some flexibility is provided to enable the robot to undertake its tasks.

- The design of an intelligent navigation framework which employs soft computing techniques to control the robot in a more robust and safer manner. The framework has a reactive architecture where each layer implements a particular goal and higher layers are increasingly abstract.

- A novel distance estimation technique was developed and integrated into this proposed intelligent framework to acquire depth information under monocular vision using the scale parameters of each pair of matched key points. The steering parameters of the control scheme are estimated by employing an artificial neural network.

- The development of a fuzzy inference system to represent the linear velocity of the system. Fuzzy rules were built depending on movement analysis related to conditions in the environment through which the robot navigates.

- Implementing, configuring and testing the proposed navigation systems in the simulation environment provided by Microsoft Robotic Studio software to ensure its reliability in terms of safe and reliable navigation.

- Evaluating the navigation systems developed by conducting rigorous experiments on the mobile robot platform. Test results for the hybrid obstacle avoidance system reveal that the combination of the two conventional obstacle avoidance methods has performed better than the conventional optical flow method. For the SIFT based architectures, the INUS method is able to fulfil the given tasks by leading the robot from its current position, avoiding obstacles, navigating through the environment, and successfully reaching its desired goals in a robust and smooth manner.

8.3 Recommendations for Future Work

The hybrid vision based obstacle avoidance system developed here is more flexible and reliable than the conventional obstacle avoidance architectures, as shown in this thesis. The proposed hybrid system principally combines the results of two detection methods and adapts the result into a conventional control mechanism which is called 'balance strategy' [Duchon et al., 1998; Temizer, 2001]. An extension to the control architecture of the system would be to integrate the balance strategy and possibly some other control laws into the controller in a hierarchical structure which also includes a rational arbitration module that will evaluate the output of different control laws so as to make the overall decision. Furthermore different optical flow and appearance-based obstacle detection algorithms can be applied to the proposed architecture. In spite of the many advantages of the Horn-Schunk optical flow estimation method, which utilizes partial derivatives of the image and/or the flow field sought and higher-order partial derivatives, it is still sensitive to noise. Therefore, in order to eliminate noise, hybrid methods can be replaced by the current method combining the Horn-Schunk method or other general variational methods with appropriate local optical flow methods. The Lucas-Kanade method may be a good alternative for local methods, and includes image

patches and an affine model for the flow field [Lucas and Kanade,1981; Ohnishi and Imiya, 2007].

Despite the increase in overall performance, the proposed vision based obstacle detection technique is still unable to compete with range-finder based obstacle avoidance techniques. This is due to the complexity of vision based systems and their sensitivity to lighting and environmental conditions. Thus, the integration of this hybrid system with other sensors will be able to compensate for their respective shortcomings and provide a significant contribution to solving obstacle avoidance problems. Having a mobile robot navigation system using vision as the only sensor both for preventing collision and tracking targets is the ultimate goal and is still an ongoing process, especially for applications in cluttered environments. The proposed hybrid collision avoidance system is flexible enough to be integrated into different navigation architectures. Consequently the proposed hybrid system can be incorporated into a goal-oriented navigation strategy so as to overcome complex navigation problems; however, obstacle avoidance is a key issue, and difficult to combine with vision based navigation systems. For instance, detecting obstacles relies on the robot's attention being directed along its path, while the best localisation information is obtained from distinctive invariant features extracted from the target. These contradictory requirements must be resolved. One solution to this problem may be to employ a two-camera system, one of which detects obstacles while the second tracks the goal.

The proposed SIFT based navigation systems are reliable and able to successfully overcome local navigation problems. The INUS method, in particular, is capable of manoeuvring effectively in its environment while avoiding collisions. Unlike most research in vision based navigation, the proposed navigation algorithms have been demonstrated to operate successfully in both realistic simulations and 'live' robotic trials. However, as the system is executed for a global navigation problem which requires the sequential achievement of multiple goals, a lot of processing time is consumed by the search procedure. This is because the robot does not have any prior knowledge about the goals and needs to search for each of them in order to localize itself. Adding a learning capability to the searching procedure of the navigation system

is an essential requirement for a fully autonomous robot, enabling it to overcome global navigation problems in a more consistent and speedy manner. In order to address this problem, a learning paradigm is recommended according to which the correspondence between each goal and its successive positions should be taught by the system so as to eliminate or minimize searching. For example, once the robot reaches its first goal it will already be dynamically localized towards the next goal without having to enable the searching module. Some examples of learning paradigms applied to different aspects of navigation systems are learning with fuzzy control, reinforcement learning, supervised learning , and the genetic algorithms [Cang et al., 2003; Manikas et al., 2007; Nattharith, 2010; Daoyi et al., 2012].

The main focus of this study has been on designing mapless navigation strategy, however, the developed systems can be incorporated into other navigation strategies in order to provide a fully integrated autonomous system. For instance, the integration of the proposed SIFT-based navigation methods into a map based hybrid navigation architecture, combining deliberative and reactive control based upon an appropriate arbitration technique, may be sufficiently flexible to allow the robot to carry out tasks in large and complex indoor environments such as hospitals. As the IWARD robots combine several sensors providing highly accurate data, they may be ready for such further development using combined strategies.

8.4 Publications

The author has published several papers on his work, as follows:

- Guzel, M.S. and Bicker, R., "*Optical flow based system design for mobile robots*", *Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference*, pp.545-550, 28-30, June 2010.
- Guzel, M.S. and Bicker, R., "*Vision based obstacle avoidance techniques*", *Recent Advances in Mobile Robotics*,"Dr. Andon Topalov (Ed.), ISBN: 978-953-307-909-7, pp. 83-108, 2011.

- Guzel, M.S. and Bicker, R., "*A behaviour-based architecture for Mapless Navigation Using Vision*" , *International Journal of Advanced Robotic Systems,* vol 9, pp. 1-13, 2012.

REFERENCES

ActivMedia (2010) *Pioneer 3 Operations Manual, MobileRobots Inc.* Available at: http://www.ist.tugraz.at/_attach/Publish/Kmr06/pioneer-robot.pdf.

Alper, Y., Omar, J. and Mubarak, S. (2006) 'Object tracking: A survey', *ACM Comput. Surv.*, 38, (4), pp. 13.

Althaus, P. (2003) *Indoor Navigation for Mobile Robots: Control and Representations.* PhD thesis.

Argyros, A. A., Bekris, K. E. and Orphanoudakis, S. C. (2001) 'Robot homing based on corner tracking in a sequence of panoramic images', *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on.* 2001. pp. 3-10.

Argyros, A. A. and Bergholm, F. (1999) 'Combining Central and Peripheral Vision for Reactive Robot Navigation', *In Proceedings of Computer Vision Pattern Recognition Conference (CVPR'99). Fort Collins.* pp. 8-17.

Arkin, R. (1987) 'Motor schema based navigation for a mobile robot: An approach to programming by behavior', *Robotics and Automation. Proceedings. 1987 IEEE International Conference on.* Mar 1987. pp. 264-271.

Aslantas, V. and Pham, D. T. (2007) 'Depth from automatic defocusing', *Optics Express*, 15, (3), pp. 1011-1023.

Atcheson, B., Heidrich, W. and Ihrke, I. (2009) 'An evaluation of optical flow algorithms for background oriented schlieren imaging', *Experiments in Fluids*, 46, (3), pp. 467-476.

AXIS (2011) *VAPIX version 3*. Available at: www.axis.com.

Barney, B. (2012) *OpenMP*. Available at: https://computing.llnl.gov/tutorials/openMP.

Barron, J. L., Fleet, D. J. and Beauchemin, S. S. (1994) 'Performance of optical flow techniques', *International Journal of Computer Vision*, 12, (1), pp. 43-77.

Bay, H., Tuytelaars, T. and Van Gool, L. (2006) 'SURF: Speeded Up Robust Features Computer Vision – ECCV 2006', in Leonardis, A., Bischof, H. and Pinz, A.(eds). Vol. 3951 Springer Berlin / Heidelberg, pp. 404-417.

Bernardino, A. and Santos-Victor, J. (1998) 'Visual behaviours for binocular tracking', *Robotics and Autonomous Systems*, 25, (3–4), pp. 137-146.

Bogacki, P. (2005) 'HINGES - An illustration of Gauss-Jordan reduction', *Journal of Online Mathematics and its Applications*.

Bonin-Font, F., Ortiz, A. and Oliver, G. (2008) 'Visual Navigation for Mobile Robots: A Survey', *Journal of Intelligent and Robotic Systems*, pp. 263-296.

Brett, R. F., William, H. W., Selim, T. and Leslie Pack, K. (2003) 'A Dynamical Model of Visually-Guided Steering, Obstacle Avoidance, and Route Selection', *Int. J. Comput. Vision*, 54, (1-3), pp. 13-34.

Brooks, R. A. (1986) 'Robust Layered Control System for a mobile robot', *IEEE journal of robotics and automation*, RA-2, (1), pp. 14-23.

Brooks, R. A. (1990) 'Elephants Don't Play Chess', *Robotics and Autonomus Systems* 6, pp. 3-15.

Cai, H., Li, K., Liu, M. and Song, P. (2010) 'Fast camera calibration of stereo vision system using BP neural networks', *5th International Symposium on Advanced Optical Manufacturing and Testing Technologies.* Dalian, pp. 7-16.

Canny, J. (1986) 'A Computational Approach to Edge Detection', *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8, (6), pp. 679-698.

Cantoni, V., Lombardi, L., Porta, M. and Vallone, U. (2001) 'Qualitative Estimation of Depth in Monocular Vision', *In Proceedings of the 4th International Workshop on Visual Form.* Springer-Verlag, pp. 1-10.

Chaari, A., Lelandais, S., Montagne, C. and Ahmed, M. B. (2008) 'Global interior robot localisation by a colour content image retrieval system', *Eurasip Journal on Advances in Signal Processing*, 2008.

Chao, Z., Yucheng, W. and Tieniu, T. (2003) 'Mobile robot self-localization based on global visual appearance features', *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on.* Sept. 2003. pp. 1271-1276. .

Chi, K. H. and Lee, M. F. R. (2011) 'Obstacle avoidance in mobile robot using neural network', *International Conference on Consumer Electronics, Communications and Networks, CECNet 2011* XianNing, pp. 5082-5085.

Chrislb. (2005) *ArtificialNeuronModel* WikiBooks.

Contreras, E. B. (2007) 'A biologically inspired solution for an evolved simulated agent', *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference.* London, pp. 206-213.

Daniel, L. M., Toal, D. and Flanagan, C. (1999) *A Fuzzy Logic Based Navigation System for a Mobile Robot.* .

De Oliveira Caldeira, E. M., Schneebeli, H. J. A. and Sarcinelli-Filho, M. (2007) 'An optical flow-based sensing system for reactive mobile robot navigation', *Controle y Automacao,* 18, (3), pp. 265-277.

Deans, M. C. (2005) *Bearings-Only Localization and Mapping.* PhD thesis. Mellon University.

Dellaert, F., Fox, D., Burgard, W. and Thrun, S. (1999) 'Monte Carlo localization for mobile robots', *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on.* 1999. pp. 1322-1328 vol.2.

DeSouza, G. N. and Kak, A. C. (2002) 'Vision for mobile robot navigation: A survey', *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 24, (2), pp. 237-267.

Driankov, D. (1987) 'Inference with a single fuzzy conditional proposition', *Fuzzy Sets and Systems,* 24, (1), pp. 51-63.

Driankov, D., Hellendoorn, H. and Reinfrank, M. (1993) *An introduction to fuzzy control.* Springer-Verlag New York, Inc.

Duchon, P. A., Warren, H. W. and Kaelbling, P. L. (1994) 'Ecological Robotics ', *Adaptive Behavior,* pp. 1-30.

Duchon, P. A., Warren, H. W. and Kaelbling, P. L. (1998) 'Ecological Robotics', *Adaptive Behavior* 6:3/4, (Special Issue on Biologically Inspired Models of Spatial Navigation), pp. 473-507.

Duda, R. O. and Hart, P. E. (1972) 'Use of the Hough transformation to detect lines and curves in pictures', *Communications of the ACM, ,* 15, (1), pp. 11-15.

Eric, R., Maxime, L., Michel, D. and Jean-Marc, L. (2007) 'Monocular Vision for Mobile Robot Localization and Autonomous Navigation', *Int. J. Comput. Vision,* 74, (3), pp. 237-260.

F. Vassallo, R., Schneebeli, H. J. and Santos-Victor, J. (2000) 'Visual servoing and appearance for navigation', *Robotics and Autonomous Systems,* 31, (1), pp. 87-97.

Fasola, J. and Veloso, M. (2006) 'Real-time object detection using segmented and grayscale images', *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.* 15-19 May 2006. pp. 4088-4093.

Fazl-Ersi, E. and Tsotsos, J. K. (2009) 'Region classification for robust floor detection in indoor environments', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 5627 LNCS Halifax, NS: pp 717-726.

Fu, L. M. (1994) *Neural Networks in Computer Intelligence.* New York, NY, USA: McGraw-Hill, Inc.

Gat, E. (1995) 'Towards principled experimental study of autonomous mobile robots', *Autonomous Robots,* 2, pp. 179-189.

Gaussier, P., Joulain, C., Zrehen, S., Banquet, J. P. and Revel, A. (1997) *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on.* 7-11 Sep 1997.

Gerkey, B. P., Vaughan, R. T., Stoy, K., Howard, A., Sukhatme, G. S. and Mataric, M. J. (2001a) 'Most valuable player: a robot device server for distributed control', *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on.* 2001. pp. 1226-1231.

Gerkey, B. P., Vaughan, R. T., Stoy, K., Howard, A., Sukhatme, G. S. and Mataric, M. J. (2001b) 'Most valuable player: a robot device server for distributed control', *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on.* 2001. pp. 1226-1231 vol.3.

Player C++ Client Library Version 1.5 Reference Manuel. [Online]. Available at: htt://playerstage.sourceforge.net (Accessed: May 2010).

Godjevac, J. (1997) *Neuro-Fuzzy Controllers Design and Application.* Lausanne: Polytechniques et Universitaires Romandes.

Gonzalez, R. C. and Woods, E. R. (2002) *Digital Image Processing.* Prentice Hall.

Green, B. (2002) Advanced *Edge Detection Tutorial*. Available at: http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/index.html

Guerrero, J. J., Martinez-Cantin, R. and Sagüés, C. (2005) 'Visual map-less navigation based on homographies', *Journal of Robotic Systems,* 22, (10), pp. 569-581.

Guzel, M. S. and Bicker, R. (2010) 'Optical flow based system design for mobile robots', *2010 IEEE Conference on Robotics, Automation and Mechatronics, RAM 2010.* Singapore, pp. 545-550.

Guzel, M. S. and Bicker, R. (2011) 'Vision Based Obstacle Avoidance Techniques', in Topalov, A. V.(ed), *Recent Advances in Mobile Robotics.* Intech, pp. 1-26.

Hagan, T. M., Demuth, B. H. and Beale, M. (1996) *Neural Network Design.* PWS Publishing Company.

Hao, L. and Yang, S. X. (2003) 'A behavior-based mobile robot with a visual landmark-recognition system', *Mechatronics, IEEE/ASME Transactions on,* 8, (3), pp. 390-400.

Harb, M., Abielmona, R. and Petriu, E. (2009) 'Speed control of a mobile robot using neural networks and fuzzy logic', *Neural Networks, 2009. IJCNN 2009. International Joint Conference on.* 14-19 June 2009. pp. 1115-1121.

Hellström, T. (2011) 'Biological Foundations of Robot Behavior', Department of Computing Science Umeå University., pp. 4-6.

Hoffmann, F. (2003) 'An Overview on Soft Computing in Behavior Based Robotics Fuzzy Sets and Systems ', in Bilgiç, T., De Baets, B. and Kaynak, O.(eds) *IFSA 2003.* Vol. 2715 Springer Berlin / Heidelberg, pp. 544-551.

Hongli, D., Wei, Z., Mortensen, E., Dietterich, T. and Shapiro, L. (2007) *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on.* 17-22 June 2007.

Horn, B. K. P. and Schunck, B. G. (1981) 'Determining optical flow', *Artificial Intelligence,* 17, (1-3), pp. 185-203.

Huq, R., Mann, G. K. I. and Gosine, R. G. (2008) 'Mobile robot navigation using motor schema and fuzzy context dependent behavior modulation', *Applied Soft Computing,* 8, (1), pp. 422-436.

Hutchinson, S., Hager, G. D. and Corke, P. I. (1996) 'A tutorial on visual servo control', *Robotics and Automation, IEEE Transactions on,* 12, (5), pp. 651-670.

Jackson, J. (2007) 'Microsoft robotics studio: A technical introduction', *Robotics & Automation Magazine, IEEE,* 14, (4), pp. 82-87.

Jae Kyu, S., Ho Gi, J., Gen, L., Seung-In, N. and Jaihie, K. (2011) 'Background Compensation for Pan-Tilt-Zoom Cameras Using 1-D Feature Matching and Outlier Rejection', *Circuits and Systems for Video Technology, IEEE Transactions on,* 21, (3), pp. 371-377.

Janglova, D. (2004a) 'Neural Networks in Mobile Robot Motion', *International Journal of Advanced Robotic Systems,* 1, (1), pp. 15-22.

Janglova, D. (2004b) 'Neural Networks in Mobile Robot Motion', *Inernational Journal of Advanced Robotic Systems,* 1, (1), pp. 15-22.

Jarvis, R. A. (1983) 'A Perspective on Range Finding Techniques for Computer Vision', *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* PAMI-5, (2), pp. 122-139.

Jeff, M., Ashutosh, S. and Andrew, Y. N. (2005) 'High speed obstacle avoidance using monocular vision and reinforcement learning', *Proceedings of the 22nd international conference on Machine learning.* Bonn, Germany, ACM, pp. 593-600.

Junghee, J. and Choongwon, K. (1999) 'Robust camera calibration using neural network', *TENCON 99. Proceedings of the IEEE Region 10 Conference.* 1999. pp. 694-697 vol.1.

Khatib, O. (1985) 'Real-time obstacle avoidance for manipulators and mobile robots', *Robotics and Automation. Proceedings. 1985 IEEE International Conference on.* Mar 1985. pp. 500-505.

Kidono, K., Miura, J. and Shirai, Y. (2002) 'Autonomous Visual Navigation of a Mobile Robot Using a Human-Guided Experience.', *Robotics and Autonomous Systems,* 40, (2), pp. 124-132.

Kim, D. and Nevatia, R. (1998) 'Recognition and localization of generic objects for indoor navigation using functionality', *Image and Vision Computing,* 16, (11), pp. 729-743.

Kim, D. and Nevatia, R. (1999) 'Symbolic Navigation with a Generic Map', *Autonomous Robots,* 6, (1), pp. 69-88.

Kim, S. and Oh, S. (2007) 'Hybrid Position and Image Based Visual Servoing for mobile robots', *Journal of Intelligent and Fuzzy Systems,* 18, (1), pp. 73-82.

Kiwon, P. and Nian, Z. (2007) 'Behavior-Based Autonomous Robot Navigation on Challenging Terrain: A Dual Fuzzy Logic Approach', *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on.* 1-5 April 2007. pp. 239-244.

Koh, C. K., Kim, S. J. and Cho, S. H. (1994) 'A position estimation system for mobile robots using a monocular image of a 3-D landmark', *Robotica,* 12, pp. 431-444.

Konar, A. (2000) *Artificial Intelligence and Soft Computing.* CRC Press.

Kosaka, A. and Kak, A. (1992) *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on.* 7-10 Jul 1992.

Kröse, B. J. A., Dev, A. and Groen, F. C. A. (2000) 'Heading direction of a mobile robot from the optical flow', *Image and Vision Computing,* 18, (5), pp. 415-424.

Li Guo, J. and Li Guang, R. (2011) 'Camera calibration for monocular vision system based on Harris corner extraction and neural network', *Consumer Electronics,*

Communications and Networks (CECNet), 2011 International Conference on. 16-18 April 2011. pp. 1-4.

Li, H. and Yang, S. X. (2003) 'A behavior-based mobile robot with a visual landmark-recognition system', *Mechatronics, IEEE/ASME Transactions on,* 8, (3), pp. 390-400.

Lindeberg, T. (1994) 'Scale-space theory: A basic tool for analysing structures at different scales', *Journal of Applied Statistics* 21, (2), pp. 225-270.

Lorigo, L. M., Brooks, R. A. and Grimsou, W. E. L. (1997) 'Visually-guided obstacle avoidance in unstructured environments', *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on.* 7-11 Sep 1997. pp. 373-379 vol.1.

Lowe, D. G. (1999) 'Object recognition from local scale-invariant features', *Proceedings of the IEEE International Conference on Computer Vision.* Kerkyra, Greece, IEEE, pp. 1150-1157.

Lowe, D. G. (2004) 'Distinctive image features from scale-invariant keypoints', *International Journal of Computer Vision,* 60, (2), pp. 91-110.

Lucas, B. D. and Kanade, T. (1981) 'An iterative image registration technique with an application to stereo vision', *IJCAI81,* pp. 674–679.

Lynch, M. B., Dagli, C. H. and Vallenki, M. (1999) 'Use of feedforward neural networks for machine vision calibration', *International Journal of Production Economics,* 60, pp. 479-489.

Maire, M., R. (2009) *Contour Detection and Image Segmentation.* Electrical Engineering and Computer Sciences University of California at Berkeley (UCB/EECS-2009-129).

Mataric, M. J. (1992) 'Integration of representation into goal-driven behavior-based robots', *Robotics and Automation, IEEE Transactions on,* 8, (3), pp. 304-312.

Matsumoto, Y., Inaba, M. and Inoue, H. (1996) 'Visual navigation using view-sequenced route representation', *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on.* 22-28 Apr 1996. pp. 83-88 vol.1.

McCarthy, C. and Barnes, N. (2004) 'Performance of optical flow techniques for indoor navigation with a mobile robot', *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on.* 26 April-1 May 2004. pp. 5093-5098 Vol.5.

Meng, M. and Kak, A. C. (1993) 'NEURO-NAV: a neural network based architecture for vision-guided mobile robot navigation using non-metrical models of the environment', *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on.* 2-6 May 1993. pp. 750-757 vol.2.

Mikolajczyk, K. and Schmid, C. (2005) 'A performance evaluation of local descriptors', *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 27, (10), pp. 1615-1630.

Moravec, H. P. (1983) 'The Stanford Cart and The CMU Rover', *Proceedings of the IEEE.* pp. 872-884.

Murphy, R. M. (2000) *Introduction to AI Robotics.* Cambridge, Massachusetts London, England: MIT Press.

Nattharith, P. (2010) *Mobile Robot Navigation using a Behavioural Control Strategy.* PhD thesis. Newcastle University.

Nattharith, P. and Bicker, R. (2009) 'Mobile robot navigation using a behavioural strategy', *Proceedings of the 11th IASTED International Conference on Control and Applications, CA 2009.* pp. 143-148.

Nilsson, N. J. (1984) *Shakey the Robot.* Artifical Intelligence Center, Computer Science and Techonlogy Division, Standford Research Institute.

Nixon, M. and Aguado, S. A. (2008) *Feature Extraction & Image Processing for Computer Vision.*

OpenMP (2011) *The OpenMP® API specification for parallel programming* Available at: http://openmp.org/wp/.

Passino, M. K. and Yurkovich, S. (1998) *Fuzzy Control*. [Online]. Available at: (Accessed: http://www.e-booksdirectory.com/details.php?ebook=2284).

Peng, J. (2004) *Extraction of Salient Features from Sensory-Motor Sequences for Mobile Robot Navigation.* PhD thesis. Vanderbilt University.

Pons, J. S., W, H., bner, Dahmen, H. and Mallot, H. A. (2007) 'Vision-based robot homing in dynamic environments', *Proceedings of the 13th IASTED International Conference on Robotics and Applications.* Germany, ACTA Press, pp. 293-298.

Reynolds, C. (1999) 'Steering Behaviors For Autonomous Characters', *Game Developers.* San Jose Ca., pp. 1-8.

Ross, T. J. and Hoboken, N. J. (2004) *Fuzzy Logic with engineering applications* Hoboken, NJ:Wiley.

Rutherford, M. (2009) *Navy seeks 'kamikazae' robots to clear mines*. Available at: http://news.cnet.com/8301-1369_3.9752976-42.html.

Saitoh, T., Tada, N. and Konishi, R. (2009) 'Indoor mobile robot navigation by central following based on monocular vision', *IEEJ Transactions on Electronics, Information and Systems,* 129, (8), pp. 1576-1584.

Schiele, B. and Crowley, J. L. (1996) 'Probabilistic object recognition using multidimensional receptive field histograms', *Pattern Recognition, 1996., Proceedings of the 13th International Conference on.* 25-29 Aug 1996. pp. 50-54. .

Se, S., Lowe, D. and Little, J. (2001) *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on.* 2001.

Sim, R. and Little, J. J. (2009) 'Autonomous vision-based robotic exploration and mapping using hybrid maps and particle filters', *Image and Vision Computing,* 27, (1-2), pp. 167-177.

Sjöö, K., Lopez, G. D., Paul, C., Jensfelt, P. and Kragic, D. (2009) 'Object search and localization for an indoor mobile robot', *Journal of Computing and Information Technology,* 17, (1), pp. 1-12.

Souhila, K. and Karim, A. (2007) 'Optical flow based robot obstacle avoidance', *International Journal of Advanced Robotic Systems,* 4, (1), pp. 13-16.

Sourceforge (2011) *Fast SIFT Image Features Library*. Available at: http://sourceforge.net/projects/libsift/develop.

Szabo, S., Coombs, D., Herman, M., Camus, T. and Liu, H. (1996) 'A Real-time Computer Vision Platform for Mobile Robot Applications', *Real-Time Imaging*, 2, (5), pp. 315-327.

Szelisnski, R. (2010) *Computer Vision: Algorithms and Applications*.3nd ed: Springer.

Szenher, D. M. (2008) *Visual Homing in Dynamic Indoor Enviorenments.* PhD thesis. University of Edinburgh.

Tao, Y., Xia, Y., Xu, T. and Cai, X. (2010) 'Research Progress of the Scale Invariant Feature Transform (SIFT) Descriptors', *Journal of Convergence Information Technology*, 5, (1), pp. 116-121.

Temizer, S. (2001) *Optical Flow Based Local Navigation.* Master thesis. Massachusetts Institute of Technology.

Thorpe, C. E. (1984) *An Analysis of Interest Operators for FIDO*. Departmant of Computer Science and The Robotics Institute, Carnegie-Mellon University.

Tinne, T. and Krystian, M. (2008) 'Local invariant feature detectors: a survey', *Found. Trends. Comput. Graph. Vis.*, 3, (3), pp. 177-280.

Toal, D., Daniel, F. C. and Strunz, B. (1995) *Subsumption Control of a Mobile Robot.*

Tomasi, C. and Kanade, T. (1991) 'Detection and Tracking of Point Features ', *International Journal of Computer Vision*, 1, pp. 1-30.

Trahanias, P. E., Velissaris, S. and Garavelos, T. (1997) *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on.* 7-11 Sep 1997.

Trieu, H. T., Nguyen, H. T. and Willey, K. (2008) 'Advanced obstacle avoidance for a laser based wheelchair using optimised Bayesian neural networks', *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2008, pp. 3463-3466.

Tschumperlé, D., A., A., Assemlal, H., Barra, V. and Blei, R. (1999) *CIMG Library* Available at: http://cimg.sourceforge.net/.

Tuytelaars, T. and Mikolajczyk , K. (2008) 'Local invariant feature detectors: a survey', *Foundations and Trends in Computer Graphics and Vision* 3, (3), pp. 177--280.

Ulrich, I. and Nourbakhsh, I. (2000) *Proceedings of the AAAI National Conference on Artificial Intelligence.*

Wei, L., Farrell, J. A., Shuo, P. and Arrieta, R. M. (2006) 'Moth-inspired chemical plume tracing on an autonomous underwater vehicle', *Robotics, IEEE Transactions on*, 22, (2), pp. 292-307.

Weinstock, R. (2008) *Calculus of Variations with Applications to Physics and Engineering.* Dover.

Wen-Chia, L. and Chin-Hsing, C. (2009) *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference on.* 12-14 Sept. 2009.

Wikipedia (2011) *General Atomics MQ-9 Reaper.* Available at: http://en.wikipedia.org/wiki/MQ-9Reaper.

Witkin, A. P. (1983) 'Scale-space filtering *', 8th Int. Joint Conf. Art. Intell.* Karlsruhe, Germany. , pp. 1019-1022.

Xiong, J. L., Xia, J., Xu, X. and Tian, Z. (2010) in. Vol. 29-32 Changsha: pp 2692-2697.

Zadeh, L. A. (1965) 'Fuzzy sets', *Information and Control*, 8, (3), pp. 338-353.

Zadeh, L. A. (1968) 'Fuzzy algorithms', *Information and Control*, 12, (2), pp. 94-102.

Zhang, Z. (2000) 'A flexible new technique for camera calibration', *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22, (11), pp. 1330-1334.

Zhichao, C. and Birchfield, S. T. (2006) 'Qualitative vision-based mobile robot navigation', *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.* 15-19 May 2006. pp. 2686-2692.

Zitovai, B. and Flusser, J. (2003) 'Image registration methods: A survey', *Image and Vision Computing*, 21, (11), pp. 977-1000.

Zou, A., Hou, Z., Zhang, L., Tan, M., Wang, J., Liao, X.-F. and Yi, Z. (2005) 'A Neural Network-Based Camera Calibration Method for Mobile Robot Localization Problems ', in. Vol. 3498 Springer Berlin / Heidelberg, pp. 987-987.

APPENDIX A: Optical flow vectors with high resolution

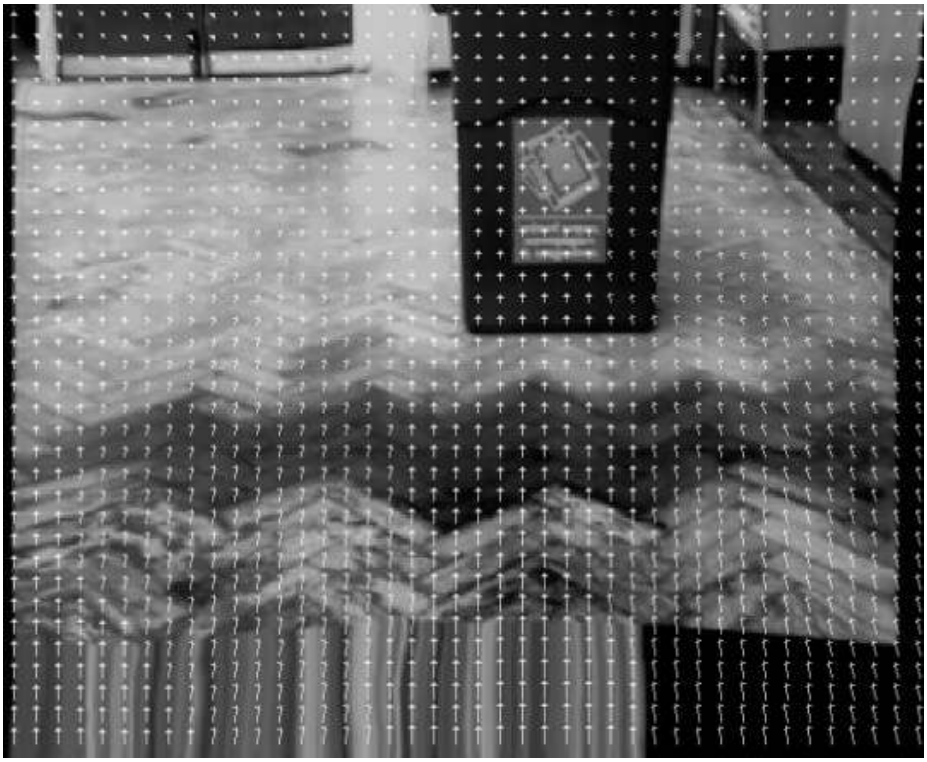


Figure A1: Frame 1 Flow vectors with higher resolution

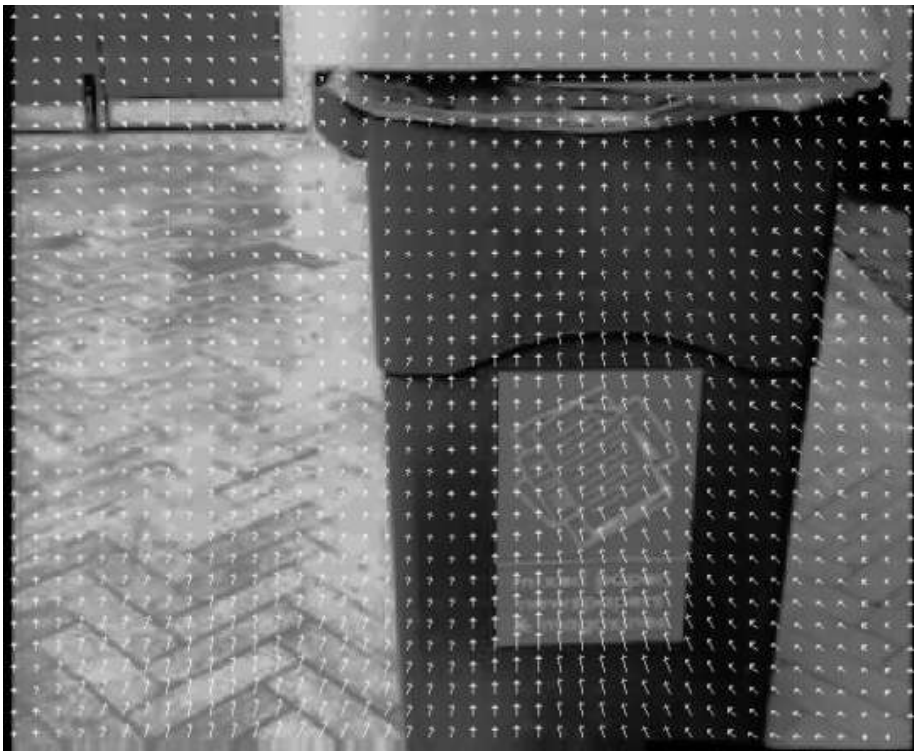


Figure A2: Frame 32 Flow vectors higher resolution

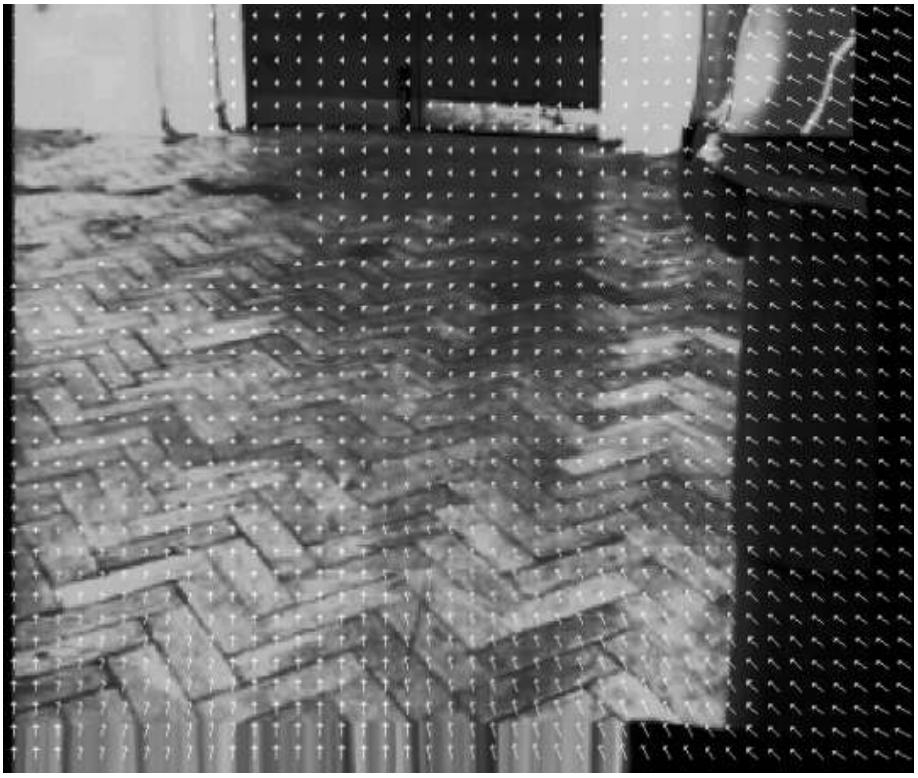


Figure A3: Frame 53 Flow vectors higher resolution

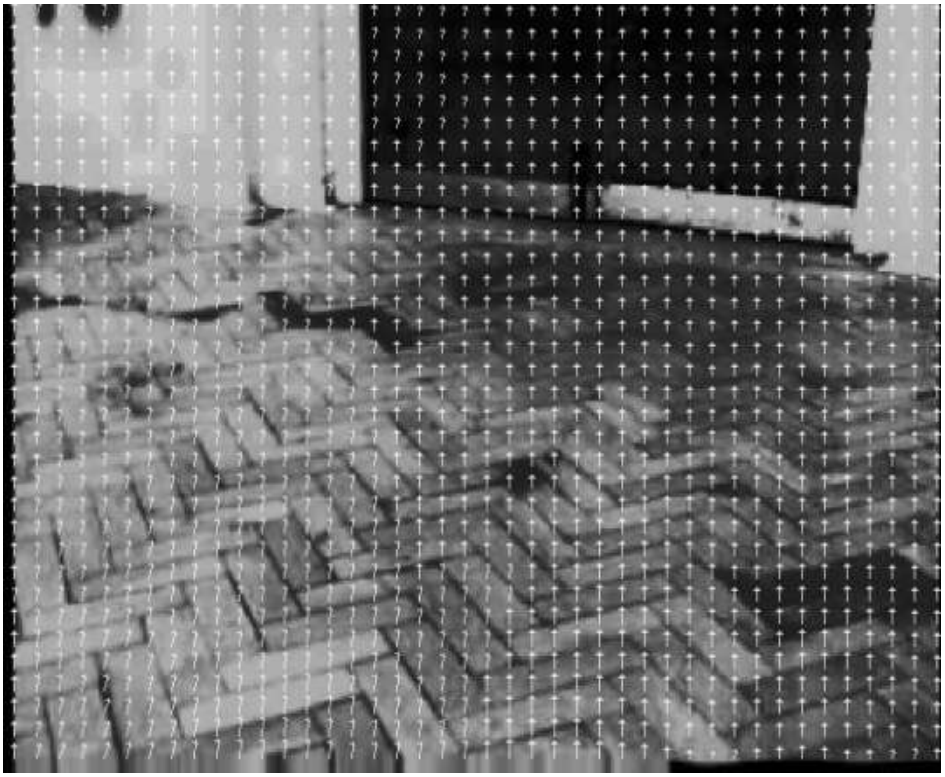


Figure A4: Frame 97 Flow vectors higher resolution

APPENDIX B: Specification of Corobot mobile robot

Dimensions	12"L x 13"W x 10"H
Weight	18 lbs.
Payload	10 lbs.
Maximum Speed	3 feet/second
Battery	13AH NiMh Rechargeable
Battery Life	2-4 hours
Camera	1080p
Encoders	Yes, 138mm/tick
Inputs	8 Digital, 6 Analog
Outputs	8 Digital
Operating System	Ubuntu Linux
Wheel	4 Wheel Drive
Supported Software	Robotic Operating System (Linux)
Motherboard	Dual-Core Itel Atom CPU @ 1.6Ghz 4Gb DDR3 RAM Nvidia ION, Supports Nvidia CUDA Technology USB 2.0 SATA 3.0
Communications	802.11n Wireless, Bluetooth
Hard Drive	160Gb 7200 RPM Hard Drive
Range Finder	Indoor Laser Range Finder (Range 5m)

APPENDIX C: Wander behaviour

An example of wander behaviour is illustrated in Figure C.1. The robot wanders aimlessly around its working environment until it encounters the goal

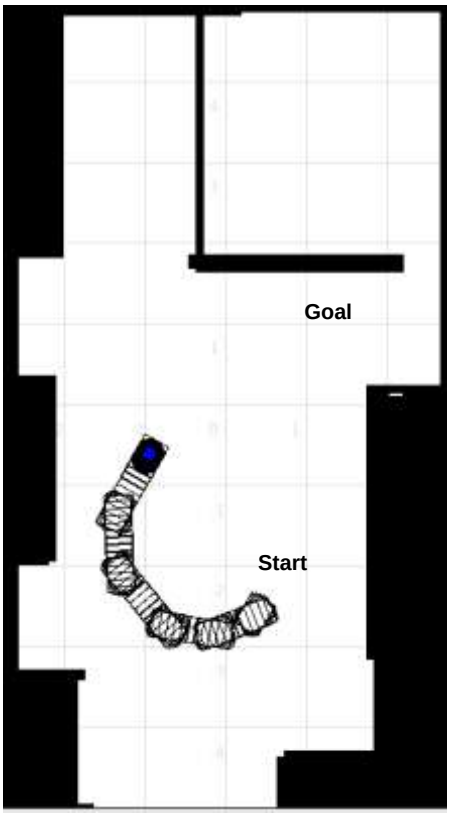


Figure C.1: Wander behavior

APPENDIX D: Back-propagation algorithm

The simplest implementation of backpropagation learning updates the network weights and biases in the direction in which the performance function decreases most rapidly, the negative of the gradient. This is called gradient descent algorithm and one iteration of this algorithm can be written as:

$$x_{k+1} = x_k - \alpha g_k \qquad (D.1)$$

where x_k is a vector of current weights and biases, g_k is the current gradient and α is the learning rate. There are two different ways in which this gradient descent algorithm can be implemented: incremental mode and batch mode. In incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. In batch mode, all the inputs are applied to the network before the weights are updated. However, gradient descent algorithms are slow for practical problems

Levenberg-Marquardt algorithm is an alternative and fast way of training, which was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares. Iteration of this algorithm can be defined as follows:

$$x_{k+1} = x_k - [\mathrm{J}^{\mathrm{T}}\mathrm{J} + \mu\mathrm{I}]^{-1}\mathrm{J}^{\mathrm{T}}\mathrm{e} \qquad (D.2)$$

where the Hessian matrix can be approximated as $\mathrm{H}=\mathrm{J}^{\mathrm{T}}\mathrm{J}$, and the gradient can be computed as $\mathrm{g} =\mathrm{J}^{\mathrm{T}}\mathrm{e}$, J is the Jacobian matrix including first derivatives of network error regarding weights and biases, and e is a vector of network errors. If the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent.

APPENDIX E: Camera calibration with conventional method

For conventional calibration techniques, It is assumed that a three dimensional coordinate system whose origin is at the centre of projection and whose *Z* axis is along the optical axis, as shown in Figure 5A.1 This coordinate system is called the standard coordinate system of the camera. A point 'M*'* on an object with coordinates (*X*,*Y*,*Z*) will be imaged at some point *m* = (*x*, *y*) in the image plane. These coordinates relate to a coordinate system whose origin is at the intersection of the optical axis and the image plane, and whose *x* and *y* axes are parallel to the *X* and *Y* axes. The relationship between the two coordinate systems C(*x*,*y*) and C(*X*,*Y*,*Z*) is given by following equations:

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

(E.1)

where s is a scale parameter, f_x and f_y represents the focal length parameters of the camera along the x and y axes respectively. The next step is to provide a transformation from the three dimensional world coordinates to image pixel coordinates using a matrix 3x4 matrix as shown below:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f_x/w & 0 & u_c & 0 \\ 0 & f_y/h & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

(E.2)

where w and h represent the pixel width and height values respectively, and u_c and v_c represent the centre point of the image, In short hand notation, `this can be illustrated as follows:

$$\bar{u} = P.\bar{M}$$

(E.3)

where $\bar{u}$ represents the homogeneous vector of image pixel coordinates, *P* is the perspective projection matrix, and $\bar{M}$ is the homogeneous vector of world coordinates. Thus, a camera can be considered as a system that performs a linear projective transformation from the projective space P^3 into the projective plane P^2.

Consequently, there are five camera parameters; namely, focal length *f*, pixel width (w), the pixel height (h), the parameter u_c which is the *u* pixel coordinate at the optical centre, and the parameter v_c which is the *v* pixel coordinate at the optical centre. However, only four separable parameters can be solved, since there is an arbitrary scale factor involved in *f* and in the pixel size. Therefore, the ratios σ_u and σ_v are calculated. The parameters σ_u , σ_v , u_c and v_c do not depend on the position and orientation of the camera in space, and are thus called the intrinsic parameters.

The next step is to transform from camera coordinates to world coordinates. The three dimensional world coordinates of a point will generally not be specified in a frame whose origin is at the centre of projection and whose *Z* axis lies along the optical axis. Some other, more convenient, frame will more likely be specified, and then a change of coordinates from this other frame to the standard coordinate system must be included. Thus, the following equation is obtained:

$$\bar{u} = P.K.\bar{M}$$

(E.4)

where *K* is a 4x4 transformation matrix as defined as follows:

$$K = \begin{bmatrix} R & t \\ 0_3^T & 1 \end{bmatrix} \qquad \textit{(E.5)}$$

R represents a 3x3 rotation matrix and sets the camera orientation with respect to the given world frame, and the final column is a homogeneous vector t capturing the camera displacement from the world frame origin. The matrix K has six degrees of freedom, three for the orientation, and three for the translation of the camera. These parameters are known as the extrinsic camera parameters. According to these values, Eq. E.6, comprising both intrinsic and explicit parameters is defined as the camera calibration matrix, and can be expanded as follows:

$$C = \begin{bmatrix} \sigma_u r_1 + u_c r_3 & \sigma_u t_x + u_c t_z \\ \sigma_v r_2 + v_c r_3 & \sigma_v t_y + v_c t_z \\ r_3 & t_z \end{bmatrix}$$

(E.6)

where the vectors , r_1, r_2 and r_3 are the row vectors of the matrix **R**, and **t** = (t_x, t_y, t_z).

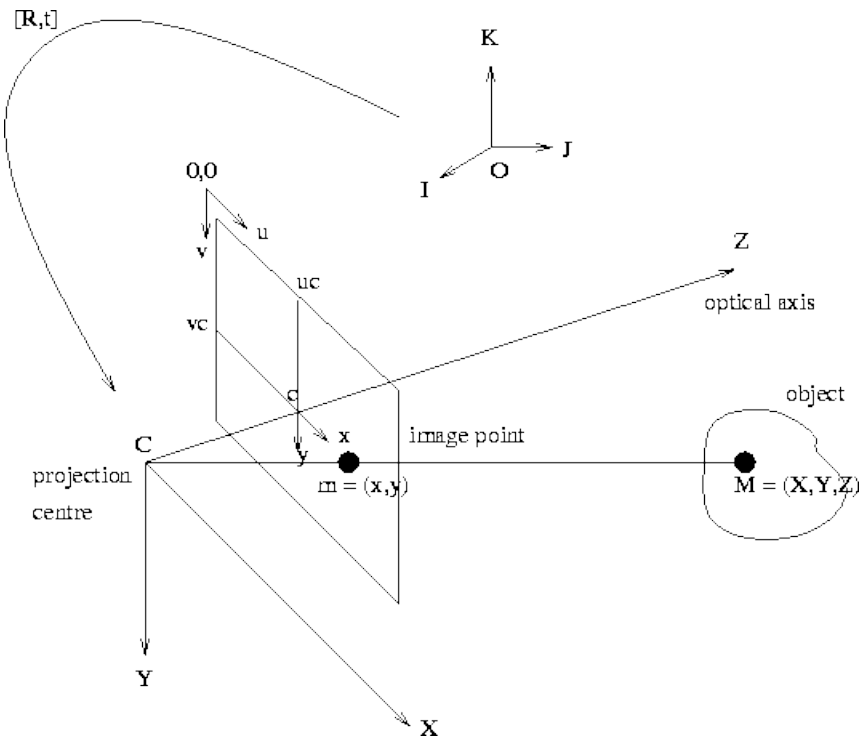


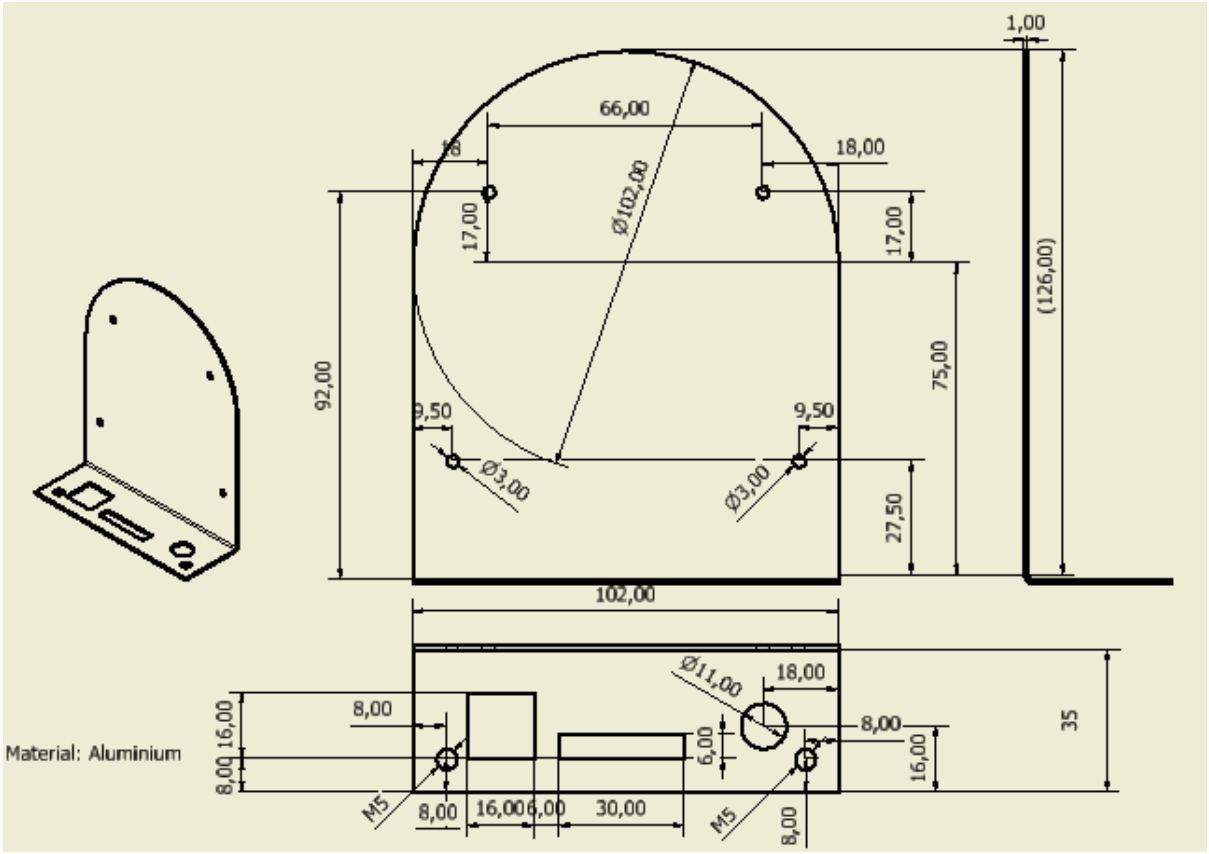
Figure E.1: The coordinate systems involved in camera calibration

APPENDIX F: Specification of Pioneer 3-DX

Specification of the mobile robot base of Pioneer 3-DX	
Length	44.5 cm (44)
Width	40 cm (38)
Height (body)	24.5 cm (22)
Body clearance	6.5 cm (6)
Weight (with min. battery capacity	9 kg
Payload of base platform with included battery	23 kg
Body Construction	1.6 mm painted aluminium
POWER	
Charge	252 watt-hr
Run time, base platform	18-24 hours
Recharge time,	12 hrs
MOBILITY	
Drive	2-wheel drive, plus rear balancing caster
Wheel diam.	19 cm
Wheel width	5 cm
Steering	Differential
Wheel diam.	19 cm
Wheel width	5 cm
Translate max speed (unloaded)	1.6 m/sec
SENSING && MANIPULATION (not requiring onboard computing)	
Front sonar ring	8 included; 1 each side; 6 forward @ 15^{o} intervals
Rear sonar ring	8 optinoal; 1 each side; 6 rear @ 15^{o} intervals
Wheel width	15 cm – 5 m
Std. Position encoders	500 tick encoders
Surveillance option	Yes
IR Sensors	No
Compass option	Yes
Gripper option	Yes
ONBOARD COMPUTING	
Optional onboard computer	Embedded size
Max. no. card & ports	3 PC104+;2 USB; serial
Speaker	Piezo std., opt, high decibel
Laser option	Yes
Gyro option	Yes
Vision	Yes
Speech	Yes
StereoCam Rangefinder option	Yes
ELECTRONICS	
Processor	Hitachi H8S
Sonar inputs	16 max
Custom I/O connections	8-bit external I/O bus w/ up to 16 devices + PC104 I/O

	boards
Communication ports	RS-232 serial ports on microcontroller, 4 RS-232 and 1 Ethernet on optional embedded computter
Wireless Communications options	Radio modern pair without embedded computer; Ethernet station adapter & access point with
Flash Memory	1 Mb
CONTROLS, PORTS AND INDICATORS (side or top panel)	
LCD display	-na-
Reset pushbutton	Warm reboot
Charging	12 VDC charge port & Docking
Joy drive port	Off & opt. onboard
Motors pushbuttons	Single enable/disable
Flash Memory	1 Mb
Serial comm. Ports	9-pin RS232 with Rcv and Xmt LED indicators

APPENDIX G: Platform for the AXIS-213 camera



APPENDIX H: Specification of URG-04LX laser range finder

Specification of the URG-04LX	
Voltage	5.0 V
Current	0.5 A
Detection range	0.02 m to approximately 5.6 m
Scan angle	240^{o}
Scan time	100 ms/scan (10.0 Hz)
Angular Resolution	0.36^{o}
Interface	USB 2.0, RS232
Weight	Approx, 160 gm
Material	Polycarbonate
External dimension	50 mm (W) x 50 mm (D) x 70 mm (H)
Turn angular velocity	360 deg/sec
Turn acceleration	pi/2 rad/s^2
Life	5 years (it changes depending on operating condition)

APPENDIX I: Specification of LinITX 8.4" Touch-Screen

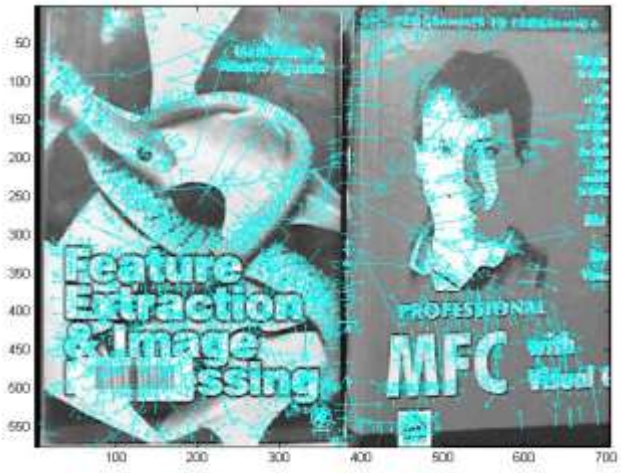
Specification of the URG-04LX	
Resolution	800 (V) x 3 x 600 (H)
Display Size	8-inch LCD
Active Area (mm)	126.0 (H) x 121.5 (V)
Brightness	350cd/m2 (centre)
Colour Configuration	RGB
Back Light	LED
Power Source	DC12-24 V
Power Consumption	600 mA(max)
Operation Temperature	-30^{o}C to +85 ^{o}C
Dimensions	229.6 mm x 162.79 mm x 33.9 mm
Viewing Angle	L/R: 70 T: 50 : 70
Storage Temparuture	-40^{o}C to +95 ^{o}C

Figure I.1 LinITX Plus 8.4 inch VGA touch-screen

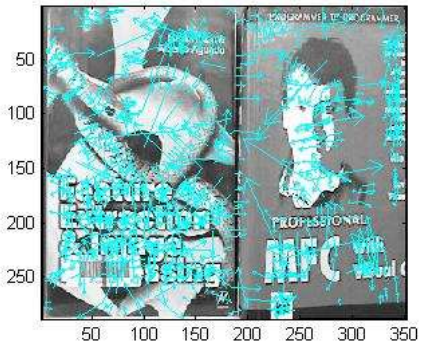
APPENDIX J: Specification of AXIS 213 camera

Specification of the AXIS 213 pan/tilt/zoom camera	
Image sensor	1⁄4” Interlaced CCD
Lens	3.5 – 91 mm, Angle of view, horizontal: 1.7° – 47°
Pan/Tilt/Zoom	20 preset positions Pan: • 170°, 1 – 90°/sec Tilt: -10 – 90°, 1 – 70°/sec Zoom: 26x optical, 12x digital Sequence mode, control queue Supports Windows compatible
VIDEO	
Video Compression	MPEG-4 Part 2 (ISO/IEC 14496-2) Motion JPEG
Resolutions	160x90 to 704x576
Frame Rate	Motion JPEG Up to 30/25 fps at 4 CIF
Image Settings	Compression, backlight compensation, manual IR-cut filter Day/Night, white balance, rotation, color/BW, brightness, noise reduction, exposure control Aspect ratio correction Text and image overlay De-interlace (4CIF Resolution)
GENERAL	
Processors and Memory	ETRAX 100LX, ARTPEC-2, 32 MB RAM, 4 MB Flash
Power	11.5 – 14 V DC, max. 13 W
Dimensions (HxWxD)	130 x 104 x 130 mm (5.1” x 4.1” x 5.1”)

APPENDIX K: Evaluations of goals via SIFT algorithm



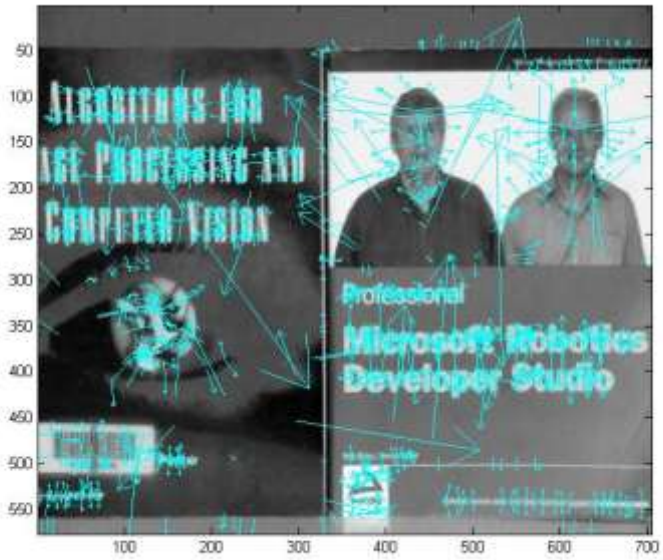
(a)



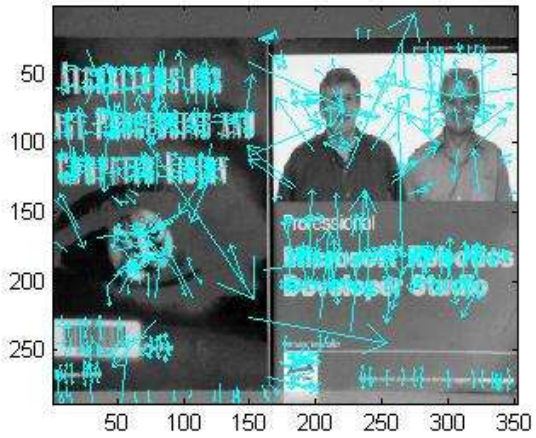
(b)

(c)

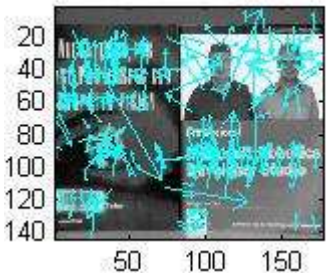
Figure K.1: SIFT features extracted for Goal A in different resolutions, (a) 704x576 [4704], (b) 352x288 [1015], (c) 176x144 [269]



(a)

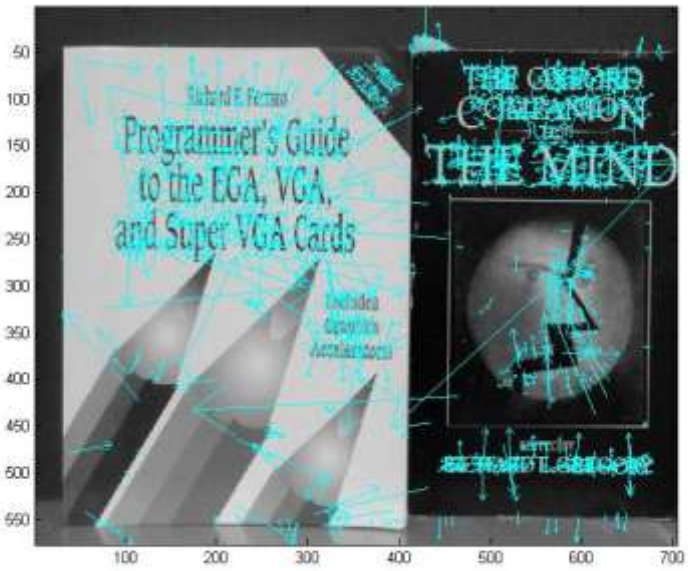


(b)

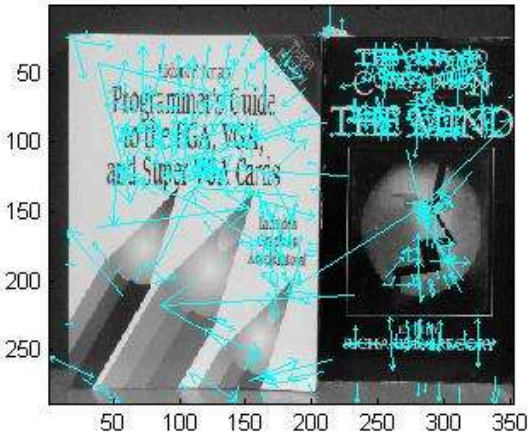


(c)

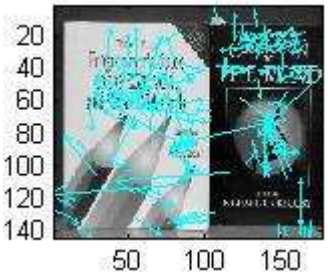
Figure K.2: SIFT features extracted for Goal B in different resolutions, (a) 704x576 [5173], (b) 352x288 [1117], (c) 176x144 [297]



(a)

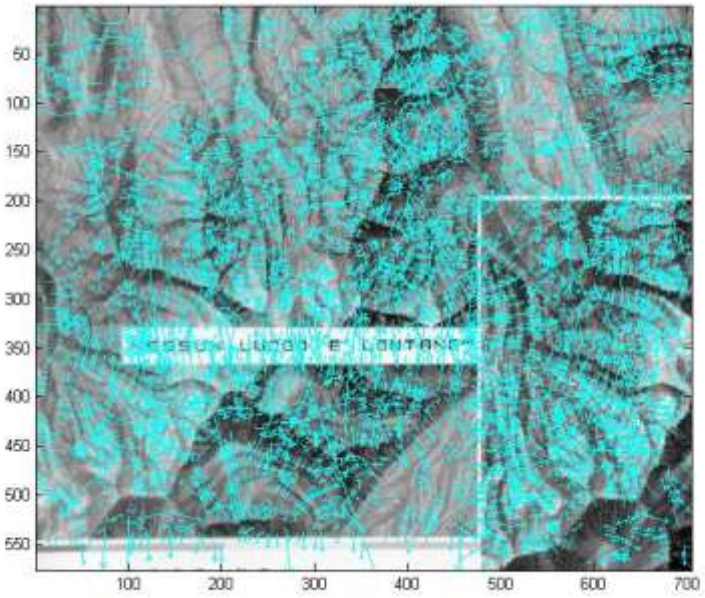


(b)

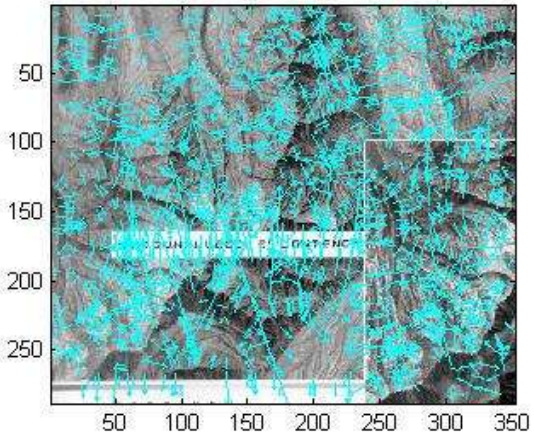


(c)

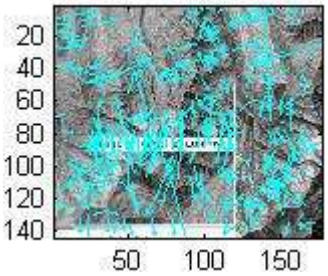
Figure K.3: SIFT features extracted for Goal C in different resolutions, (a) 704x576 [3522], (b) 352x288 [769], (c) 176x144 [213]



(a)



(b)



(c)

Figure K.4: SIFT features extracted for Goal D in different resolutions, (a) 704x576 [6358], (b) 352x288 [1396], (c) 176x144 [375]

Computational performance analysis of these algorithms with respect to the given objects are illustrated in Figure K.5, indicating that the results of the algorithm utilizing the Fast SIFT library extract key features faster than when using conventional SIFT algorithm.

The average overall improvement in terms of percentage reduction in total computational time with given resolutions obtained from the data set shown in Figure K.1, is given as follows: 53% (176x144), 48% (352 x 288) and 51% (704 x 576). The results reveal that the OpenMP based SIFT algorithm with the given system configuration provides a computational performance approximately twice as fast as when the using conventional method. Despite the significant performance enhancement obtained via multiprocessor systems according to the characteristics of the OpenMP API [OpenMP, 2011], the improvement in computational speed with a single processor is adequate to apply the algorithm in real time navigation algorithms.

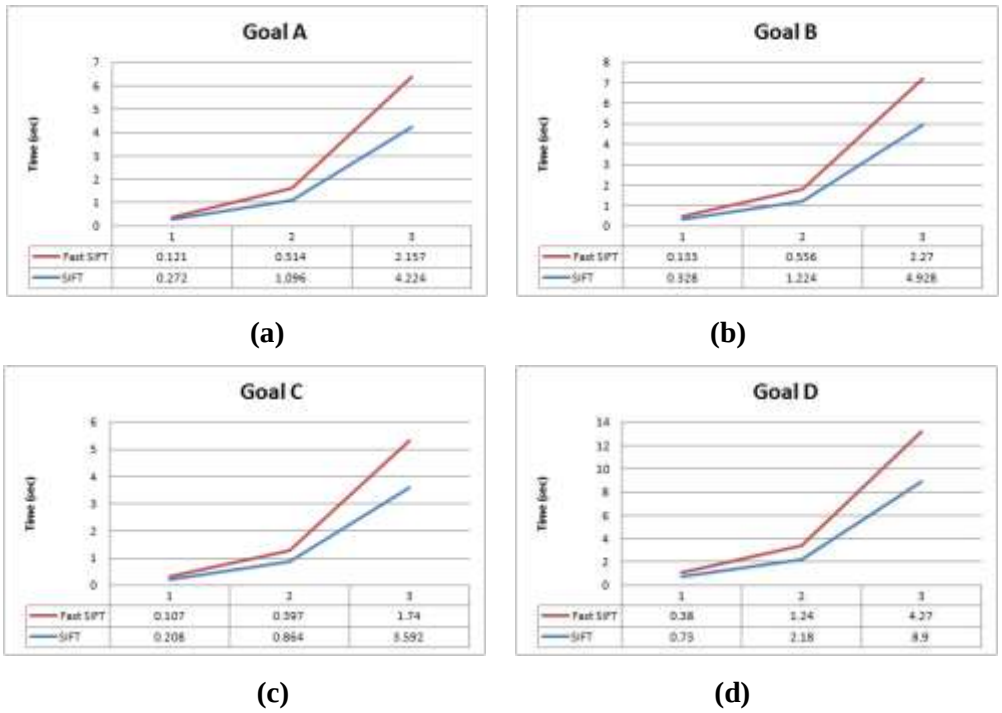
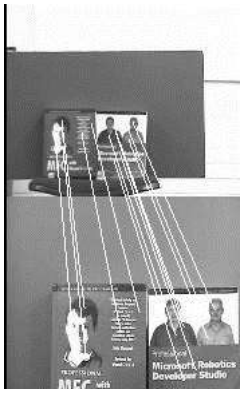


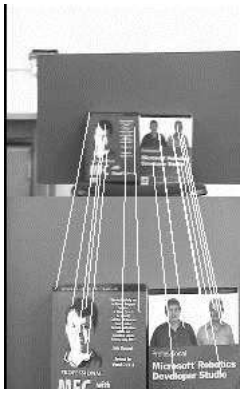
Figure K.5: Comparison of SIFT and FAST SIFT algorithm performance using different goals of different resolutions "(1) 176x144, (2) 352x288, (3) 704x576", (a) Goal A, (b) Goal B, (c) Goal C, (d) Goal D

APPENDIX L: Goal tracking example via calibrated camera

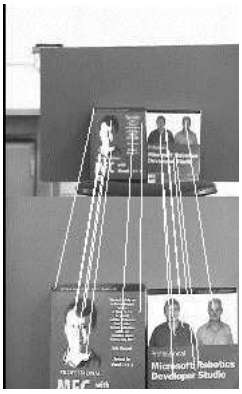
Figure L.1 illustrates a feature tracking example with the proposed calibration system using SIFT features where the goal is always in the field of view, and the similarity between the goal and the current image increases gradually. Each frame includes generated steering angle (w) from the matching.



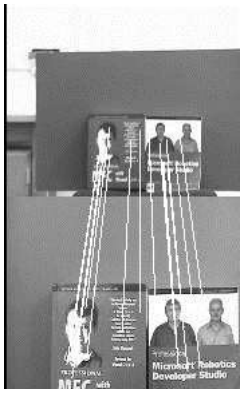
(a) -6^{o}



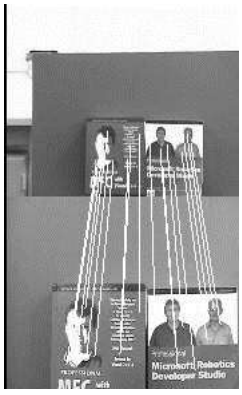
(b) 0^{o}



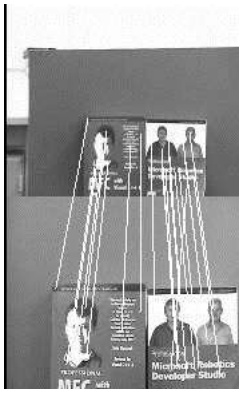
(c) 2^{o}



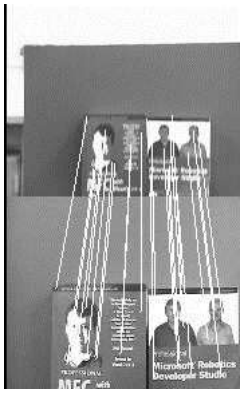
(d) 1 ^{o}



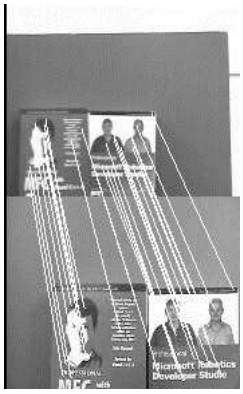
(e) -4^{o}



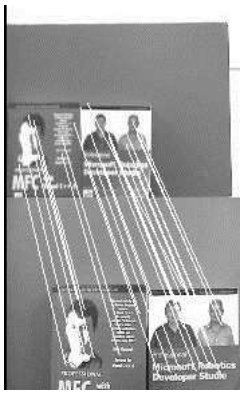
(f) 3^{o}



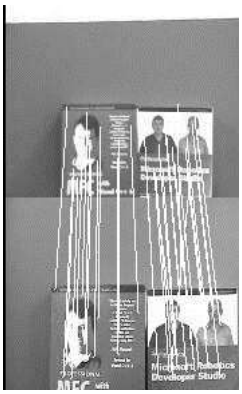
(g) 7^{o}



(h) 2^{o}



(i) -9^{o}



(j) *completed*

Figure L.1: Tracking example via calibrated camera from a to j, including generated steering output

APPENDIX M: The training results for the simulated camera

This appendix includes the results of the training performance for the simulated camera. The specifications of the network used for simulated camera and its training performance are illustrated in Table L.1 and Figure L.1 respectively. The best validation point is at the 55_{th} epoch (iteration) with an error of $1.1083ex10^{-6}$. The output of the network is only the pan angle based on the specifications of the camera.

Table M.1: Basic specifications of the network for heading angle estimation (simulated camera)

Camera Type	*Resolution*	*Data*	*Topology*	*Train*	*Validation*	*Test*
Simulated	***320x240***	***85***	***2-4-1***	***69***	***8***	***8***

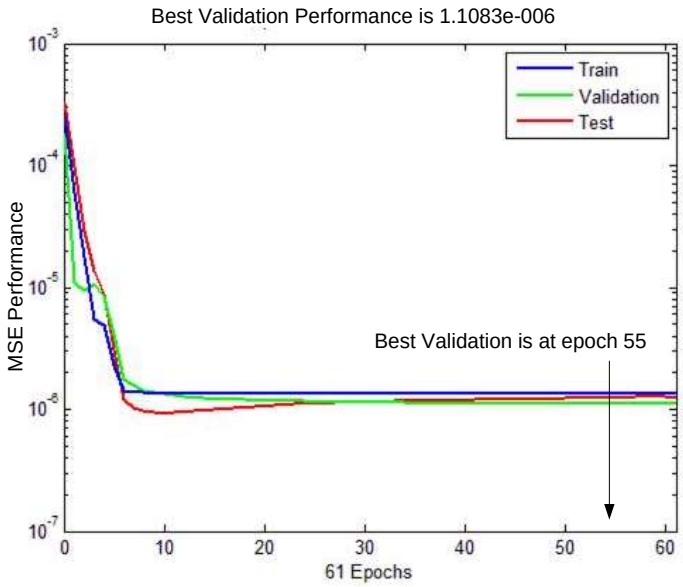


Figure M.1: Training results for the simulated camera

Table M.2: Basic specifications of the network for distance estimation (simulated camera)

Camera Type	*Resolution*	*Data*	*Topology*	*Train*	*Validation*	*Test*
Simulated	***320x240***	***128***	***2-4-1***	***104***	***12***	***12***

The specifications of the network for distance estimation used for simulated camera is shown in Table M.2, as well as the results of training algorithm are shown in Figure M.2. The results reveal that the best validation point is at the 29_{th} epoch (iteration) with an error of 0.00073247

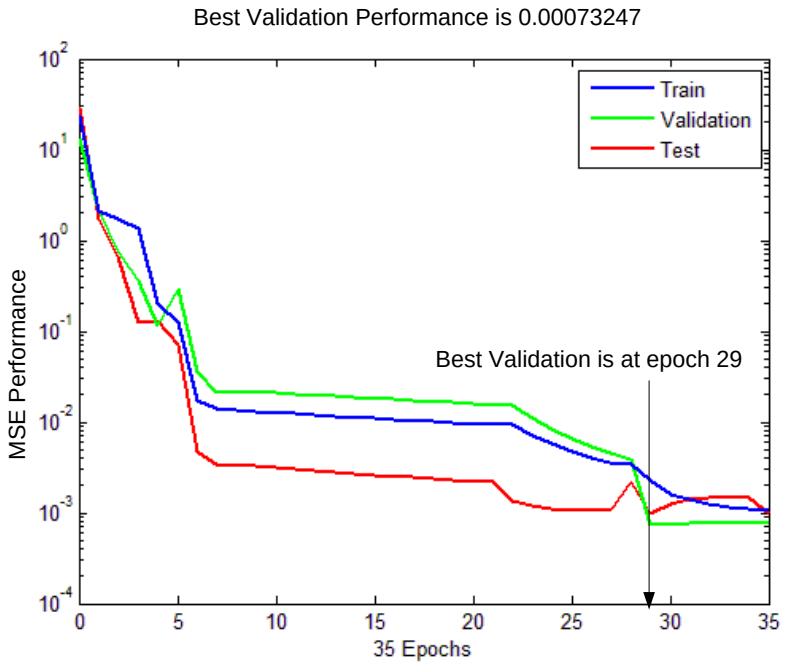
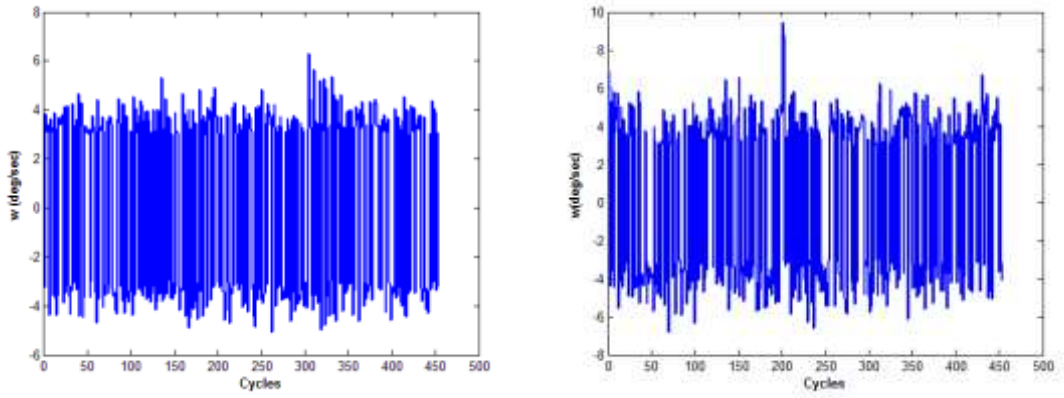


Figure M.2: The training results for distance estimation

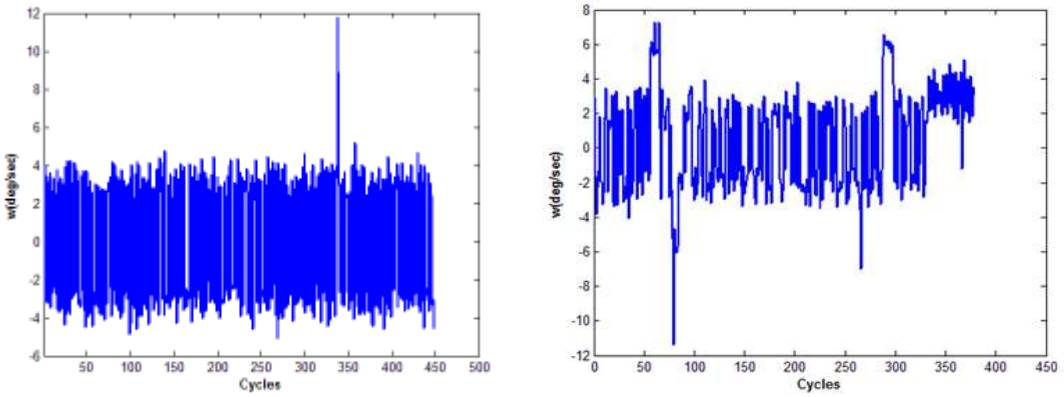
APPENDIX N: Control outputs of output scenarios

During the experiments, the sampling period of the duration of a decision cycle was set at t_c = 150 ms, and each cycle used in these control figures, shown below, averages the consecutive three frames in order to make the data more clear. The constant turns are excluded from the results. . For each decision cycle, the robot is controlled by an updated angular velocity command (w).



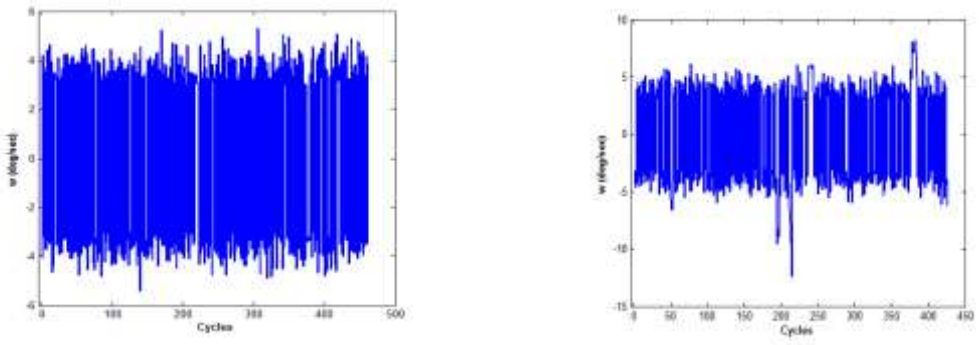
(a) **(b)**

Figure N.1: Control parameters for SC1, (a) FS , (b) OFB



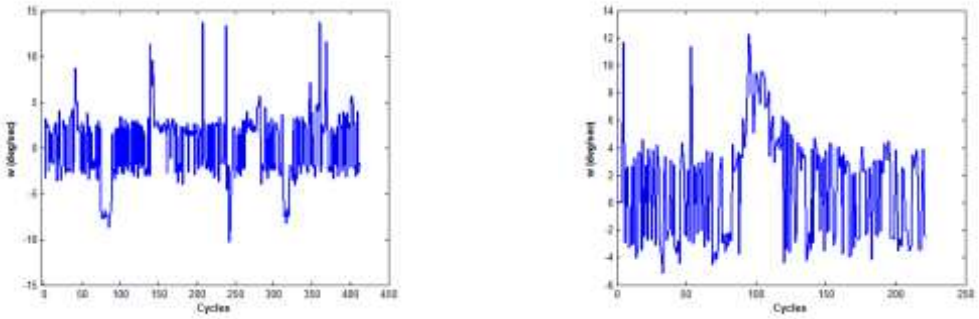
(a) **(b)**

Figure N.2: Control parameters for SC1, (a) FS , (b) OFB



(a) **(b)**

Figure N.3: Control parameters for SC3, (a) FS , (b) OFB



(a) **(b)**

Figure N.4: Control parameters for SC4, (a) FS , (b) OFB

APPENDIX P: Definition of SC7

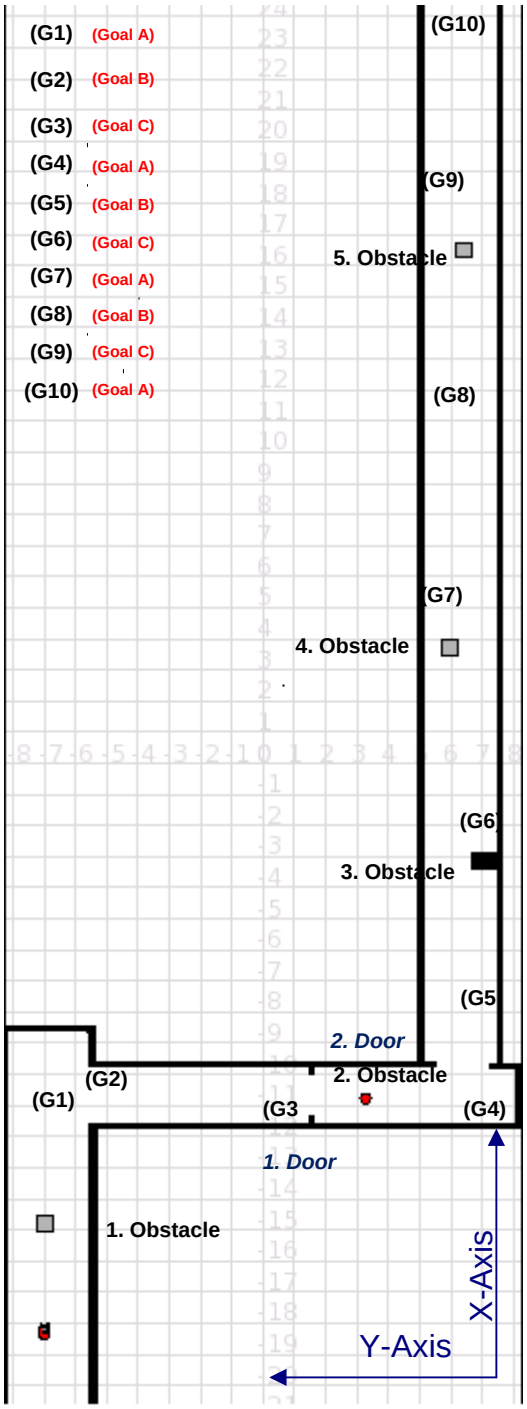


Figure P.1: Definition of CS7

The corridor environment has dimensions of 43.76m x 16.20m, as illustrated in Figure P.1. In this scenario, ten sub goals are defined to navigate the robot along the corridor. In order to

maintain the consistency, the goals, defined in Chapter 6, were utilized. Besides this, to increase the challenge inherent to the scenario, five obstacles are positioned across the path. Three different types of external obstacle were positioned in the environment so as to increases the challenge of the experiment. The main obstacle is a rectangular box having dimensions of 550 mm x 500 mm, one of which is has a 187 mm diameter irregular shape box, and other is a rectangular box having dimensions of 1100 mm x 500 mm which consists of two main obstacles.