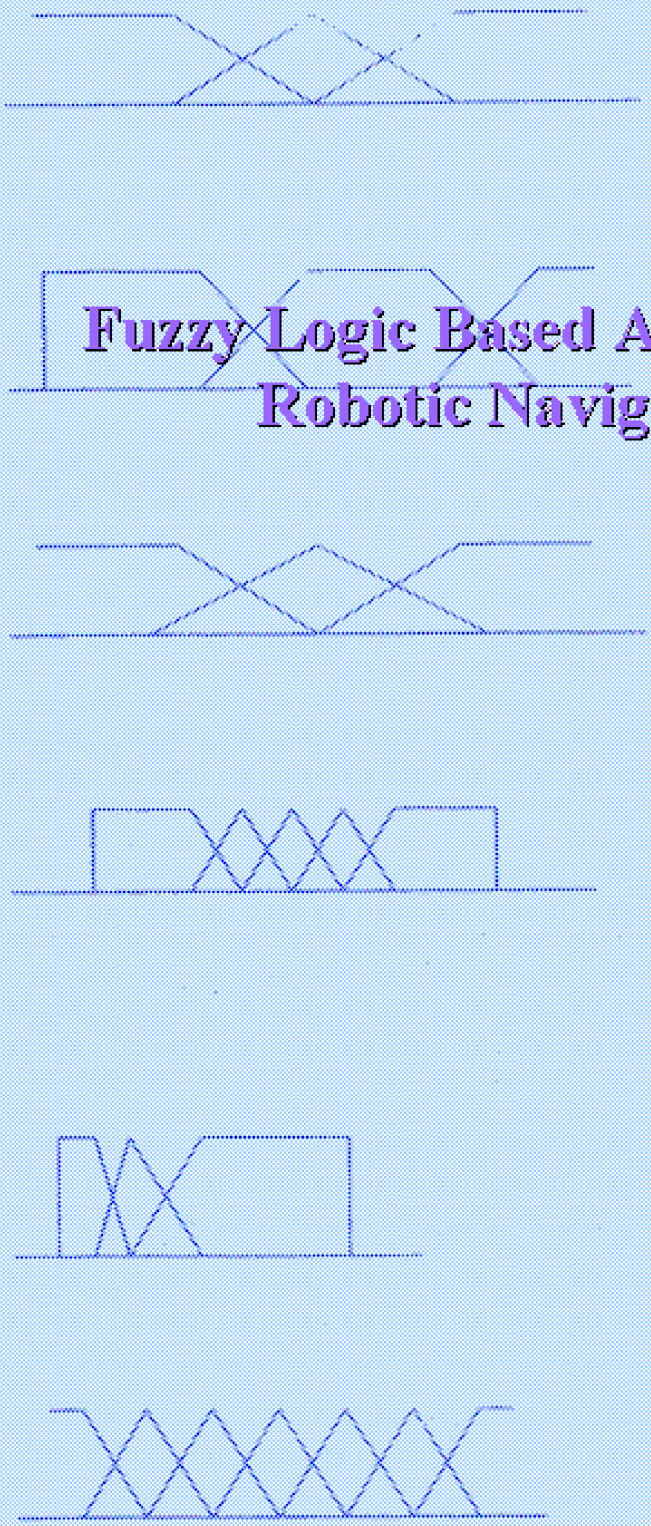
University of Luton

Campbell Hooper

Fuzzy Logic Based Autonomous Robotic Navigation



Acknowledgements

Firstly, I would like to thank Dr. Osei Adjei for his feedback and guidance.
To my family and friends new and old for their patience and advice, especially to Ester for the sacrifice of her shiny new computer. And to Lauren for her continuous support.

Abstract

The working title for this project is 'Fuzzy Logic Based Autonomous Robotic Navigation'. Although much work has been done on this subject, and recently a lot of interest has been shown; there is as yet no central resource and information is to be found in many diverse engineering fields. Important aspects can be found in Artificial Intelligence, Automation, Robotics, Engineering and Manufacturing. Consequently complete and detailed information is rarely presented in one document, hence it has been the goal of this project to demonstrate a complete, and truly autonomous robotic system.

Many of the specifications are found to be encouraging the use of simple and intuitive techniques, the idea being that people from any of the diverse backgrounds mentioned, can tailor and advance the work presented. A generalistic approach has been taken and can be seen throughout.

The robotic system is to be suitable for insertion into unmodified indoor environments, specifically offices and warehouses, therefore no artificial landmarks or pathways shall be given.

Given the aim of developing intelligent systems, no prior map information will be available; and hence dynamic learning of the environment will be demonstrated.

The sensory input for this system is to utilise only simple and well known devices, this means that no Global Positioning Systems (GPS) and similar tracking systems, or vision based approaches can be implemented.

In the view of true autonomy, it is necessary to pertain to the ability to re-localise from any known or unknown position. Autonomy also demands attempts to recover without assistance from unforeseen situations, including simple changes in the environment such as the movement of furniture or doors opening and closing.

In the interest of safe navigation, techniques should be shown for avoiding both static and moving obstacles.

Table of Contents

2

Table of Figures

1 Introduction

The aim of this project is; within the scope defined, to provide a detailed investigation of the general requirements, potential benefits and common troubles of the techniques necessary to build and implement the methods shown on an autonomous robot. Problem areas should be isolated and solutions shown in a clear manner, ideally anyone wishing to pursue a similar project can easily understand the underlying requirements.

Autonomous navigation is a diverse field of research, centralisation of information and research is desirable. It is anticipated that this project should bring this complex field into view for undergraduates and hobbyists, providing a template that can later be used and improved. Treating each topic as separate and interchangeable modules would be ideal.

An outline of the necessary considerations would include the special requirements of physical design, microprocessor devices including the code design and optimisation of complex tasks, sensor systems and representation strategies.

This project arises for many reasons, mainly because of the growing potential benefits of such systems. Industry relies both financially and physically on robotics, there is then a lot of investment into applicable research as the advantages can be pronounced.

Morality and legislation no longer allow us to send people into hazardous environments, as technology advances so do the hazardous environments it creates, these factors can limit industrial and scientific growth.

The ability to maintain complex systems in hazardous environments that demand there is no human interaction, means that such systems can then be develop. Deep underwater research is an example of such an environment that is currently beyond our grasp, yet may hold many secrets and advances.

Humans have for many years dreamed of labour saving devices to remove the burden of everyday common tasks, autonomous robotics has a place in this dream world. Hollywood and the general media, are ever eager to remind us and keep the dream alive.

The computer games industry is one of the most demanding applications for artificial intelligence techniques, lifelike challenges and interaction is in integral part of many computer games. Considering the applicability to games environments, interfacing with other code then becomes an important aspect.

Computers are every day becoming smaller and requiring less power, currently much work in robotics is limited by such advances. Rapidly technology is catching up with the demands of vision processing and intensive on-line computation, putting the realm of autonomous robotics firmly on the 'pocket watch technology' list.

Ultimately the goal of autonomous navigation is to demonstrate a robot that interacts with its environment, including people, in a believable and seamless manner. Consequently toward this goal, much of the work presented here as artificial intelligence research draws its observations from the natural world, cockroaches, hamsters and other animals, and consequently exhibit very naturalistic tendencies.

> For both animals and mobile robots, the ability to self-localize and plan paths, or what is known as navigation, is of utmost importance. A navigation system requires not only reactive behaviour, but also the ability to interpret sensory information with respect to global goals, intelligent interaction with the environment (e.g. active sensing), and recovery from errors. [1]

1.1 Overview of Project

Chapter three presents an introduction to fuzzy logic, included in this chapter is an overview of the construction and evaluation of fuzzy rules. Also presented is a discussion of defuzzification methods.

The aim of this chapter is to introduce the reader to fuzzy logic and establish a perspective for the rest of the work.

Chapter four reviews some of the literature available, assisting the reader to find further information on the subject.

Chapter five details the investigations undertaken in developing the fuzzy inference engine and autonomous robot. Types of sensor systems are considered and generalisations drawn. Common techniques used to represent maps are shown with the associated benefits

[1] Garcia-Alegre, M.C. Ribeiro, A. Cañas, J.M. "A Cartographer Robot: Merging and Interpreting Maps In Unknown Environments" page 405

and difficulties explained. The implementation of fuzzy behaviours is outlined along with path planning and localisation techniques.

Chapter Six presents the work done and the methods by which it has been achieved. The implementation of code and OpenGL structures features heavily in this chapter and the simulation of a physical system is carefully considered.

Sections form the previous chapter are reviewed and revisited, the development of the fuzzy rule base and fuzzy inference system is examined along with the methods and reasoning behind them. The fuzzy section concludes with the chosen defuzzification method being shown in detail and a discussion presented on the flexibility of the final fuzzy system.

Map building and exploration techniques are covered with an emphasis on development by observation, this section includes methods that have been implemented for optimisation. Also a novel path planning method is introduced, again the emphasis is on building simplicity from observation.

In finalisation of the development, an assessment of the robustness of the system is given, general trouble areas are identified and solutions presented. Further discussions are given on the fuzzy behaviours and their interaction, emphasis is given to the complexity of both the problem and solution.

Chapter Seven presents some experimentation with the robot system in typical situations, the results are shown and explained. The common problem areas previously identified are covered and presented in a visual manner. Some statistics for solutions to these standard tasks are given and there is discussion of the applicability of this system to equivalent ‘real world’ problems. Comparisons are drawn to actual implementations and extrapolations can be made to the expected results of the theoretical physical robot.

Chapter Eight presents a comparison to work presented elsewhere, this includes alternative combinations of fuzzy behaviours and control structures. A look at the methods often relied on for solutions to such common problems, emphasises the effects of differences in approaches taken.

Comparisons are drawn to equivalent work on areas including, dynamically updating maps and path planning methods. Alternative solutions to problems such as the interpretation and representation of map data and neuro or neuro-fuzzy systems are considered.

Chapter Nine presents some conclusions drawn from the work achieved, the failures and successes are evaluated with reference to comparative evidence from other sources. The benefits of this work and of the approaches taken are also discussed in comparison to other research.

Successes of incomplete sections of the project are estimated based on the results of the project as a whole.

The appropriateness of the chosen sensor systems and of the map representation strategies are discussed, a look back at development reveals where early oversights have lead to restricted effectiveness in parts.

Functionality of the system and experimentation repeatability is discussed, some future suggestions put forward, and some conclusions are made on enhancements that would be beneficial to this system.

In **chapter Ten**, the general feeling for robotics and autonomous robotics in general is discussed, the applications now and of future trends are considered.

Suggestions for future amendments aimed at furthering this work have been made, methods for achieving these changes are outlined briefly. These methods for expanding functionality include, matching of partial map sections and an add on to provide consideration for holonomic constraints.

The final techniques required for true autonomy are considered, such as the potential for implementing the system presented with a micro-controller CPU system on a real robot. Finally, the benefits of neuro and adaptive techniques are raised, and some areas envisaged to require careful consideration are mentioned.

2 Introduction to Fuzzy Logic

2.1 Conventional and nth Value Logic

Conventional Boolean logic recognises two states, either 0 or 1, representing TRUE and FALSE respectively. No intermediate levels or compromises between the two states exist.

The Logical operators AND, OR, NOT form well recognised methods for manipulating this type of logic, it is this that almost all computer systems are built upon. The result of such operations always has guaranteed results and hence is entirely dependable.

This principle can be extended to include a third truth value, one which represents an indeterminate level of truth or indecision. This three valued logic has its own valid algebra, equivalent AND`s, OR`s and IMPLICATIONS exist, and exhibit equally valid conclusions.

Once it has been excepted that a three valued logic is a reasonable rational, it is to be expected that an interval or nth value logic is a viable possibility, such a logic would normally have equal divisions,

"defined by the following equally spaced partition: having equally spaced partition:
0 = 0/n-1, 1/n-1, 2/n-1, …, n-2/n-1, n-1/n-1 = 1
Each of these truth values describes a *degree of truth.*"[2]

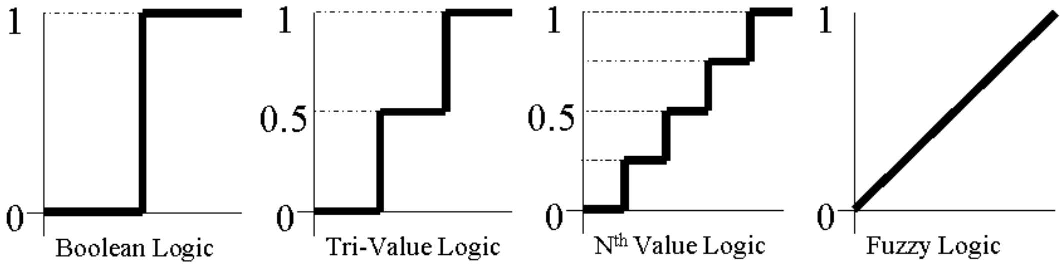


Figure 2.1-1 Boolean to fuzzy truths

Fuzzy logic is the final extension on this principle, this is where any intermediate state can be represented. This is somewhat similar to analogue electronics and numbers of the 'floating point' type.

[2] Chen,C. Pham, T.T. "Introduction to Fuzzy Sets, Fuzzy Logic and Fuzzy Control Systems" page 66

2.2 Fuzzy Memberships

A fuzzy membership is a two dimensional relation defined within the minimum and maximum of a range, starting and ending with a value of zero and having at least one point between these which equals one.

It is possible to define an interval type membership function, which has its equation defined as a series of values store in an array, this technique allows fast evaluation and convenient storage of data.

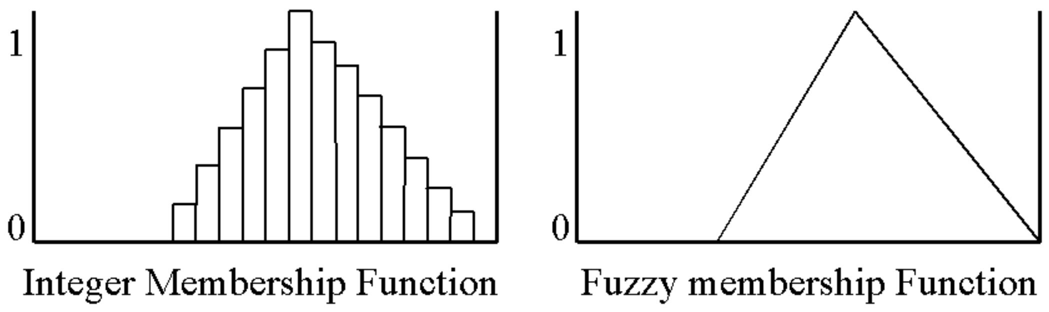


Figure 2.2-1 Membership function styles

By specifying the minimum and maximum values for the relationship and a point between where the relation becomes equal to 1, a triangular analogue function can be described. The advantage of this specification is that they require just 3 memory spaces.

The main disadvantage is that triangular membership functions require

$$Y = mX + c \qquad \text{Equation 1}$$

type calculations to be performed.

To promote flexibility and extensibility, the membership itself should be described by a function or object within a membership object, allowing it to be interchangeable with other compatible functions. Such alternative functions could then be tailored to evaluate functions which closer reflect the desired response.

As only the three essential point are specified, bell curves and windowed function can be defined on top of the triangular membership, the centre point of the function determined by the position defined to be equal to 1.

2.3 Fuzzy Relations

Fuzzy relations consist of two or more fuzzy memberships. An example could be the control voltage to a motor, or the pressure in a pneumatic line. Given such a relation the following memberships could exist within the relations limits, Very_Low, Low, Med, High, Very_High.

One of the advantages of fuzzy logic is being able to specify memberships in such linguistic and natural terms, it is intuitive that Very_High is an extreme condition whilst low is moderate. The exact numerical value may not be of great importance and hence may not be defined or even known.

Such a relation can be shown as,

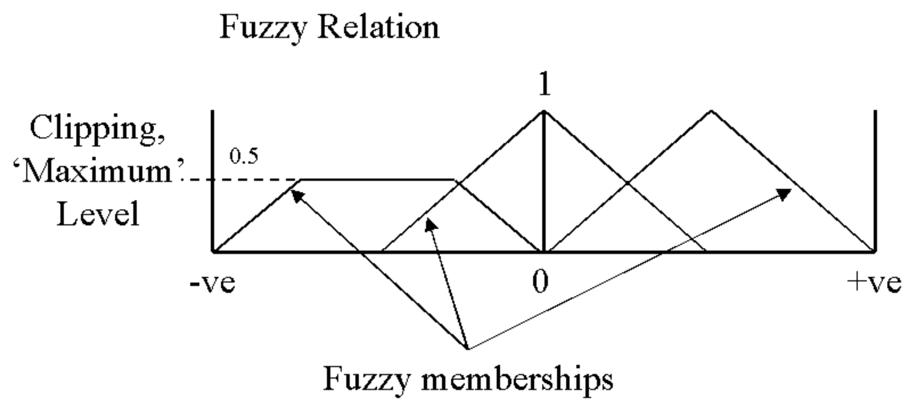


Figure 2.3-1 Membership clipping

It should be noticed that each membership within a relation can be individually clipped to a maximum level.

When developing fuzzy systems, consideration should be given to the use of Fuzzy Hedges.

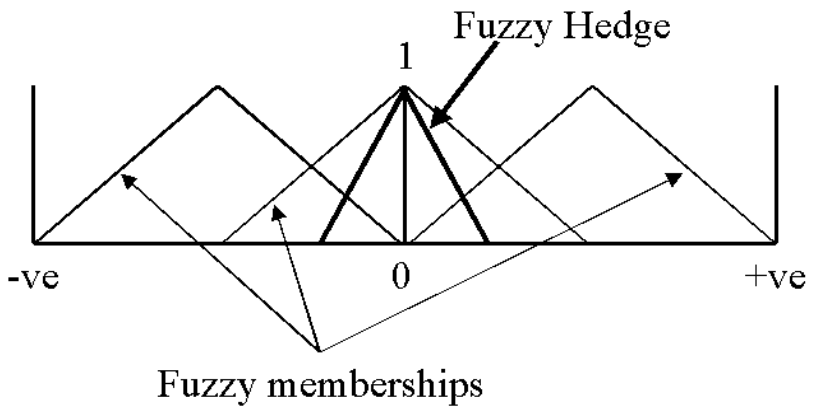


Figure 2.3-2 Fuzzy Hedges

As standard relations may vary in strength, it can occur that they all evaluate to zero. In this situation there would be no bias in either direction. Such a situation can lead to unpredictable and therefore undesirable results.

A standard method for avoiding this ambiguity is to specify a fuzzy hedge, this is a relation that will always evaluate to a maximum value. When a relation is evaluated, the hedge's influence ensures that a known result will be always obtained even in the absence of all other persuasions.

2.4 Fuzzy Operators

Conventional Algebra provides the standard Boolean logic operators AND and OR, these rules provide a sound basis for building reliable systems and can be shown by the truth tables,

AND		
Inputs		Output
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

OR		
Inputs		Output
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Figure 2.4-1 Conventional logical operators

The equivalent two fuzzy operators are described as the maximum and minimum functions OR and AND respectively,

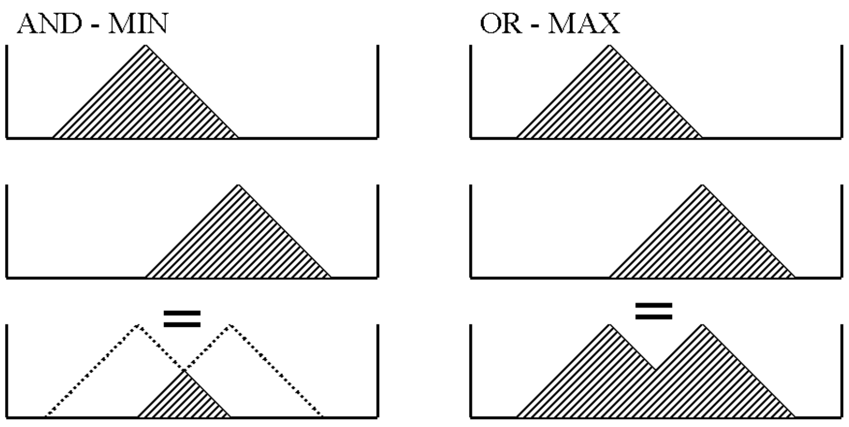


Figure 2.4-2 Fuzzy operators

Like their conventional logic equivalents, these operators form the basis for the fuzzy manipulation.

2.5 Fuzzy Rules

Fuzzy rules provide a method for describing the interaction between two fuzzy relations. These then form the basic building blocks that are then used to develop complex fuzzy systems.

> “Complex behaviour generation for artificial systems requires huge implementation efforts that can be partially overcome by decomposing global tasks in simpler well specified behaviours easier to design and to tune independent from each other.”[3]

The output from a fuzzy rule is also a fuzzy relation, it is this relation that will be evaluated to find a final crisp value as the output from a rule.

2.6 Fuzzy Inference

The IF ‘CONDITION’ THEN ‘ACTION’ type of rule is a common command structure within most programming languages. This is commonly used to represent decisions that may imply knowledge about a system.

[3] Aguirre, E. García-Alegre, M. ”A FUZZY SAFE FOLLOW WALL BEHAVIOUR FUSING SIMPLER FUZZY BEHAVIOURS” page87

A Fuzzy Rule is the equivalent to an IF THEN rule, when several fuzzy rules are brought together in a fashion where the output from one rule becomes the input to one or more other rules, the resulting network can be referred to as a fuzzy inference system.

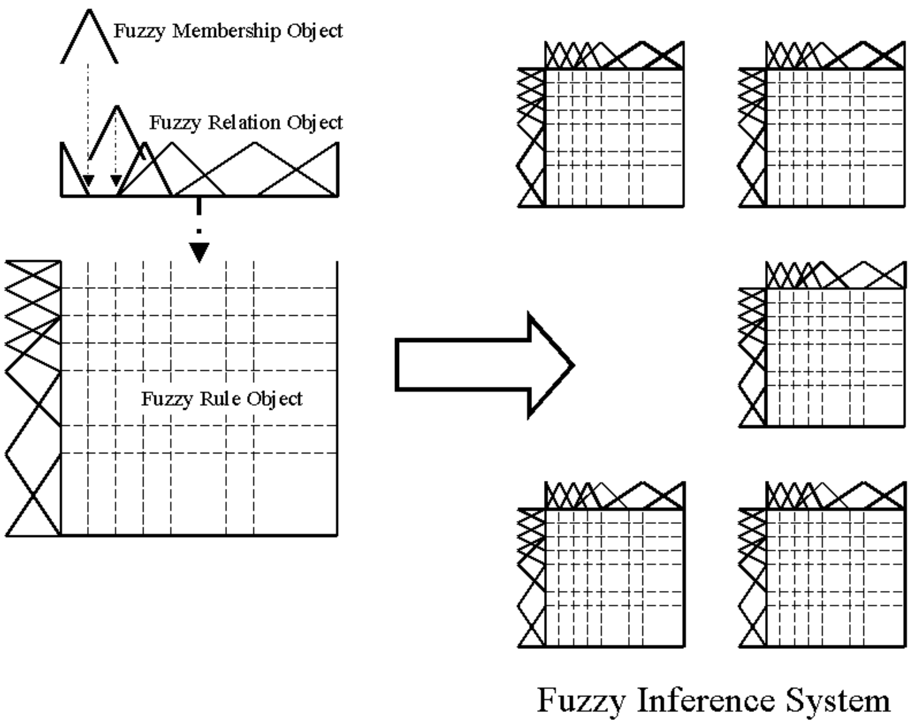


Figure 2.6-1 Fuzzy Inference System

It is anticipated that fuzzy rules will be described for the following areas.

- Target Tracking behaviour
- Obstacle avoidance behaviour
- Motor control signals
- Map Maintenance

2.7 Defuzzification

The final output from the inference system is represented by a crisp, non-fuzzy number. This value may directly control a process or provide legible readout for an operator. The process by which this value is determined is referred to as defuzzification. It is, as the name suggests, the opposite of converting a real crisp number into a fuzzy relationship.

There are two types of defuzzification method commonly used, the Centre of Mass or Centroid method, and the Maximum method.

The centroid method evaluates the whole range to find the centre-of-mass for the whole relation, this process can be computationally expensive.

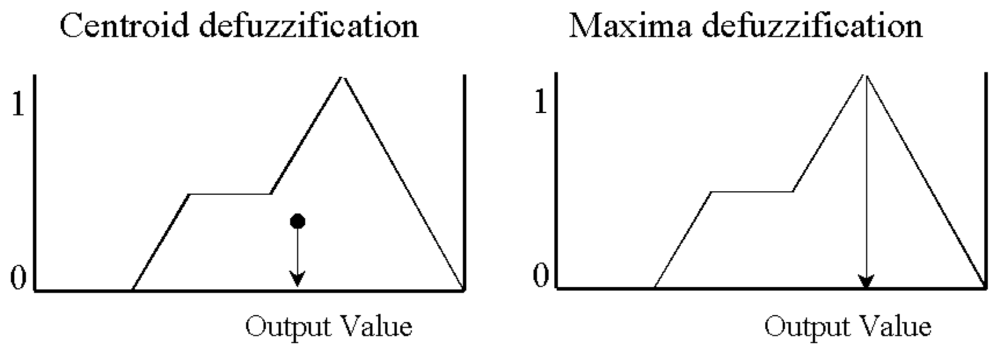


Figure 2.7-1 Defuzzification Methods

The Maxima method requires far less calculation, however given a plateau can result in unpredictable outputs.

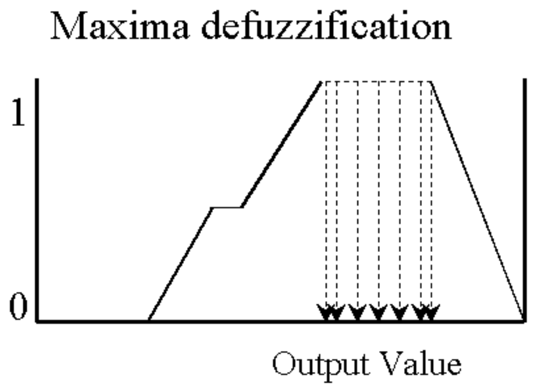


Figure 2.7-2 Maxima Ambiguity

Several points can concurrently evaluate to the same maximum value, hence the output may change rapidly and unpredictably between these values.

The centroid method may suffer from indecision resulting from conflicting desires. When the centre of mass is found of the following relationship, the output can represent no action, or zero, ignoring strong desires on each side.

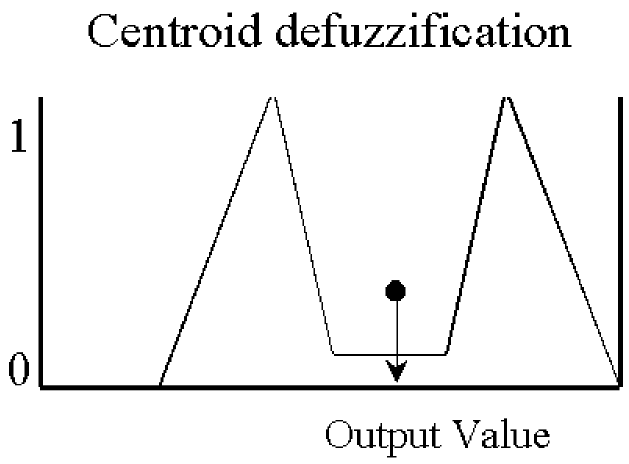


Figure 2.7-3 Centroid Confliction

3 Literature Review

3.1 "*Fuzzy Logic Techniques for Autonomous Vehicle Navigation*"

Dimiter Driankov
Alessandro Saffiotti ed.

The text opens with a definition of autonomous navigation and an introduction to the application of fuzzy logic as an automotive technique, then building a basic structure and argument for the implementation of fuzzy systems.

The first section 'A Reminder on Fuzzy Logic' is a fairly comprehensive and complete review of relevant techniques. In introducing and comparing different methods to achieve a basic fuzzy framework this section leaves the reader feeling confident and wizened.

The section 'Design of Individual Behaviours' has two major themes, the design of static and adaptive behaviours. Each example explains the reasoning behind and requirement for such behaviours. An overview of the problem and general discussion on solution is based on actually implemented systems. The reader is rewarded with a 'voice of experience' type knowledge and a feeling that truly useful information has been conveyed. Conclusions are drawn in an analytical manor and well supported by references. The examples on adaptive control have managed to convey an understanding of inherently complex systems without extensive use of mathematical proofs, this may partially be due to appropriate and ample diagrams.

Part III 'Co-ordination of Behaviours' Builds on the work presented in the previous examples, echoes of traditional layered robotic design can be seen throughout. Pitfalls within fuzzy behaviour systems are exposed and some novel solutions presented in detail.

The concept of blending behaviours has been well covered with a view on the difficulty of contradicting and mutually exclusive behaviours. This section provides detailed functions and rule bases with comprehensive diagrams, these shortcut the transition from conventional layered robotic control by adapting familiar techniques to fuzzy methods. Several detailed examples are given in the section 'Mapping the Environment', these are accompanied by a good explanation of the problem situations. The topics include, fusion of sensor information to improve reliability, the use of robot swarm technology and identification of objects from raw sensor data. These difficult topics are well presented at a level that would be appealing to both hobbyist and professional.

The final section in this text ‘Layer Integration’ is aimed toward providing more humanistic behaviours, the concept of reasoning is presented with a focus on truly autonomous robotics. This section functions as an overview on the more taxing problems associated with the actual implementation of autonomous robots, consideration is given for the interaction with human’s and other robots. Recovering from unforeseen situations is addressed regarding the desire for service robots to behave in a fashion that would be inconspicuous to viewing public. A real focus is given to the integration of helper robots into everyday life, with evidence presented from currently employed systems.

3.2 "*Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems*"

Guanrong Chen
Trung Tat Pham

A collection and extension of course notes from a fuzzy logic module at Huston university, this book begins with a review of both simple and more complex classical set theory as an introduction to chapter 1 'Fuzzy Set Theory'. A brief introduction to fuzzy sets and memberships precedes a detailed and through investigation of interval operations and arithmetic. Fuzzy sets is revisited in the last section of this chapter, IV OPERATIONS ON FUZZY SETS, many intuitive illustrations and clear examples relates previous details of interval theory to the subject of fuzzy memberships, subsets and arithmetic. The fuzzy operators maximum and minimum are introduced for working with fuzzy sets along with the alpha-cut theory. This chapter concludes with a diverse problem section which should appeal to people from many backgrounds.

Chapter 2 of *Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems* has been devoted to guiding the reader through the many types of logic. From a formal definition of classical logic and Boolean algebra, it is explained how the same theory can be extended to include n^{th} valued logic. A clear progression is detailed from this to the idea of fuzzy logic as a true logical methodology. Readers from computer science backgrounds will appreciate the comparisons between traditional IF THEN rule bases to the equivalent fuzzy implications.

Attention is drawn to the development of fuzzy systems from real physical processes. Problem areas addressed include the issues of system analysis and techniques for obtaining a description or system model. Within this it is clearly shown why fuzzy techniques may be preferred to conventional mathematical representation schemes. Also highlighted are the advantages and suitability with regard to the selection of particular fuzzy methods, each with stability descriptions and achieved without the rigorous mathematics usually associated with this type of proof.

These techniques have been related in chapter 4 to conventional control systems, a brief history explains why and when control systems where formally described.
The basic Programmable Logic Controller (PLC) is explained and comparisons drawn to fuzzy control, these ideas are then expanded into several worked examples, with the results presented. These examples provide an impressively readable and concise argument.

The subject of Proportional-Integral-Derivative (PID) Controllers is separately presented in chapter 5, this subject is extremely relevant in terms of industry and has been covered here extensively. The reader is first introduced to the basic building blocks of PID controllers and then talked through simple system design. The concept of fuzzy PID controllers has been presented in much the same way; allowing for comparison and later a detailed stability analysis.

Adaptive fuzzy systems are presented as a variation of the fuzzy systems previously introduced, outlining operational requirements and mechanisms for adjusting the system. A detailed evaluation of each traditional method is provided, the origins and reasoning behind each explained along with a more detailed explanation of the behaviours. Chapter 6 concludes with an overview of sub-optimal systems and a few brief examples.

The final chapter presents five applications of fuzzy logic. These examples cover well the content of this text, and emphasise how development of fuzzy systems can be achieved. Each example is highly appropriate and presented with background subject information and a full evaluation of the system.

3.3 "Artificial Intelligence A Modern Approach"

Stuart Russell

Peter Norvig

A Modern Approach opens with a description and loose definition of AI. Means of identifying or evaluating AI are discussed, including the classing Turing and Chinese Room tests.

The concept of 'Intelligent agents' is introduced and defined, focus is given to the need for AI including the 'combinational complexity explosion' problem.

Attention is drawn to the requirement of describing real world environments in a formal manner. These premises are extended to general problem solving and searching strategies. A detailed review of search strategies investigates breadth and depth first searches, the information presented thus far is then applied to explain informed searches. This section concludes by describing game playing such as chess and chequers, for example, as a search problem.

In the section entitled, 'Knowledge and Reasoning', a simple agent and an environment are developed and LISP code is given. Later, forward and backwards chaining is added.

There is a lengthy section describing logics. From first principles this section builds to be a complete tutorial for the underlying logic drawn upon later within *AI: A Modern Approach*.

This basic logic is used to identify logical atoms, then from these atoms to build complex sentences representing the logic of real situations.

The concept of deductive reasoning is introduced and leads to a fairly complete section on knowledge bases, with good background information and in the same clear style which is found throughout. The categorisation of objects using a structured representation is discussed and concludes with a clever scheme for representing time.

More intelligent reasoning is covered in the 'Acting logically' section; contextual information is added to the concepts previously built, with practical examples to reinforce the text. Practical solutions to representing plans and planning methods are given.

In view of this, new and more expressive logical operators are described and these used to illustrate conditional planning.

Uncertainty and reasoning introduces chance and Bayes rules, an apt subject worth the consideration given. The value of information is discussed and the premesis that results assist the transition from simple to complex decision making.

The subject is contextualised by an interesting view on the life span of an agent and a rational agent is devised.

A collection of methods for learning is presented. This section includes an historical look at the development of ANN's and some more obscure subjects such as adaptive CBR's.

The issues of knowledge as it relates to learning are covered; careful consideration is given to putting information in context. From here it is shown that rules can be extracted from the hypothesis defined.

There is a full and complete section on natural language processing. A detailed grammar is developed within a formal language. An equally detailed section is presented on the subject of perception. Image processing techniques are covered extensively. 3D viewing systems are considered and result in a practical example of a driving system, actually used on open highways.

The field of robotics is described in terms of tools, sensors and the techniques used for navigation and motion of these tools.

Concluding with a section devoted to airing others views that makes for good reading, this book truly is a complete text on the subject presented in a formal way without extensive mathematics. A credit to any bookshelf.

3.4 *"Starting with Microsoft Visual C++"*

Kris Jamsa

Starting with Microsoft Visual C++ is a complete guide to Microsoft Visual C++, it is varied enough to present a guide to installation and using the product. There is an initial walk through explanation of the commonly used features, and this is rounded off by with a presentation of how to use Visual C++ from a command line interface.

This book is presented with a lesson plan structure with topics clearly separated; it has been divided into three sections of increasing difficulty. The first section starts with a guide to producing your first C++ program that would equally serve as a guide to producing your first ever program.

The general C++ introduction gives way to a more specifically Visual C++ oriented approach, and lesson 19 is dedicated to highlighting specific windows programming techniques.

Section two guides the reader through the 'nuts and bolts' of graphical drawing, quickly building to a template that is used to implement and illustrate the more advanced C++ methods introduced in this section. A strong graphical thread runs through the descriptions of inheritance, scope and access within classes of drawable objects.

Time is devoted to the subject of interfacing with the windows environment with a lesson on understanding the resources of a program and the control of windows devices. These sections strike a fine balance between C++ and Visual C++, such that they serve equally those new to the field and those making the transition from old style C++. This is assisted by the development of a standard windows program template that includes methods for handling windows messages.

In this section there is an introduction to structures as objects, this is important as it is the basis for a more detailed look at anonymous unions and functions in structures.

Every lesson ends with a 'What you must know' section that gives a quick review of the lesson covering the main topics and some self test questions are then presented. Throughout the text there are 'Success hints', as a pre-emptive strike these highlight common problems that usually have to be found out the hard way.

The general page quickens toward the end of section 2, lessons 24 to 27 are devoted to a detailed and comprehensive coverage of the more difficult CLASS topics, including

Inheritance, polymorphism and also abstract and virtual classes. These topics are handled well, with a gentle introduction and a firm grasp on the topics.

Sections 28 to 30 delve deeper into windows programming, they cover program resources and the use of dialogue boxes, section 31 then describes the ‘Bells and Whistles’ of windows programming, buttons, lists, scrollbars and styles that add functionality to windows programs and allow the development of useful applications.

Section three the finer points of C++ are acknowledged, the underlying structure is first raised in a section that covers memory management. Clear examples and good explanation allow an understandable technical discussion of memory that leads to a review of pointer that is necessary to work with the memory heap. Finally some groundwork for Dynamic Memory Access (DMA) is shown, detailed in places.

Windows issues are revisited with a look a keyboard focus and the processing of both character and non-character keyboard input. A similar review of mouse input is given.

The classic file stream, I/O Stream, is reviewed, however the focus falls mainly on the new style CARCHIVE streaming for window objects. An extensive chapter follows this section on random I/O file access, consisting mainly of example code.

Exception handling is introduced gently and also covers the more difficult yet essential issues. Overloaded functions and functions with variable argument lists are also considered.

A well-rounded book for the beginner of those transferring from old style C++.

4 Project Design

4.1 A View Toward Hardware, Robot Design

The intention for this project is to show a collection of methods that are suitable and applicable to a real robotic vehicle. It is not intended to implement these theories in hardware, however it should be shown that the simulation approximates realistic physics and physical methods.

The motion of the vehicle must be determined from the velocities of each wheel. This kind of backwards odometry has the advantage that it allows perfect position estimation to be assumed or the strength of any localization routine to be tested.

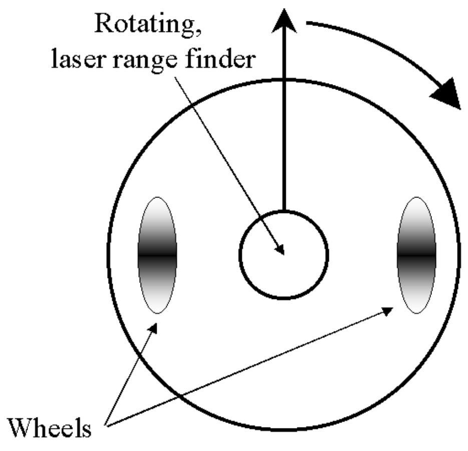


Figure 4.1-1 Robot General Description

4.1.1 Sensor System

A variety of sensor systems have been used on mobile robots, often [4] the fusion of two or more types of sensor helps to provide reliable and accurate data.

When using sensors such as sonar on a real robot, it must be considered that some pre-processing is required [5]. The chosen method will affect the readings obtained, this is because the modeling of such sensors is a complex topic and has many important considerations, for example; sonar systems tend to suffer from spurious echoes.

[4] Driankov, D. Saffiotti, A. Ed. “Fuzzy Logic Techniques for Autonomous Vehicle Navigation” page 266-271
[5] Driankov, D. Ed. “Fuzzy Logic Techniques for Autonomous Vehicle Navigation” page 263-266

This topic is worth careful consideration when implementing a robotic system, however does not contribute significantly to the research area defined by this project, hence a generalized model is to be assumed.

It is possible to simplify the physical model by using a Laser range finder such as the device used in [6] and [7]. These devices maintain a known ranging accuracy of less than ±5mm, also implementations have proven that 0.5 degree rotational increments can be achieved. The maximum range of such a device is approximately 50 meters.[8]

Because of these known resolutions it can be determined that the accuracies in the simulation are comparable to those of the real device, hence realistic for immediate implementation.

4.1.2 Steering System,

The design of the robot will effect the way in which it can maneuver, a phenomenon described as the Holonomic nature of the vehicle.

> The main difference between collision-free path planning for an omnidirectional robot and car-like robot relies on existing nonholonomic constraints. The effect of these constraints is to limit the local mobility of the robot, though not restricting in the large the accessibility of the whole configuration space. [9]

Although the effects on path planning are pronounced, theories exist for converting any direct path into a path that is realistic for the vehicle.[10],[11]

Turning on the spot, tank type maneuvering, allows us to describe the general methods behind path planning and navigation whilst assuming a simplified physical model, enabling us to develop systems without the concerns described in great detail elsewhere.

[6] Puttkamer, E. Weiss, G. Edlinger, T. "Localization and On-Line Map Building for an Autonomous Mobile Robot" page 37
[7] Einsele, T. "Real-Time Self-Localization in Unknown Indoor Environments using Panorama Laser Range Finde" page 697
[8] Limon Marruedo D. Gomez Ortega J. "Neural Mobile Robot Navigation Based On A 2D Laser Range Sensor" *IFAC* Intelligent Autonomous Vehicles" page 319
[9] Kreczmer, B. "PATH PLANNING FOR A CAR-LIKE ROBOT APPLYING MULTI-LAYER GRID MAP OF THE ENVIRONMENT" page 117
[10] Siméon, T. Leroy, S. Laumond, J.P. "Computing good holonomic collision-free paths to steer non-holonomic robots" page 10004
[11] Scheuer, A. Fraichard, Th. "Continuous-Curvature Path Planning for Car-Like Vehicles" page 997

4.2 Map Building

There are two main types of map representation scheme, Grid based and Vector based.

Grid based maps divide an area into regular segments, vector maps refer to positions as an angle from a specific reference direction, and a distance along this angle from a known origin.

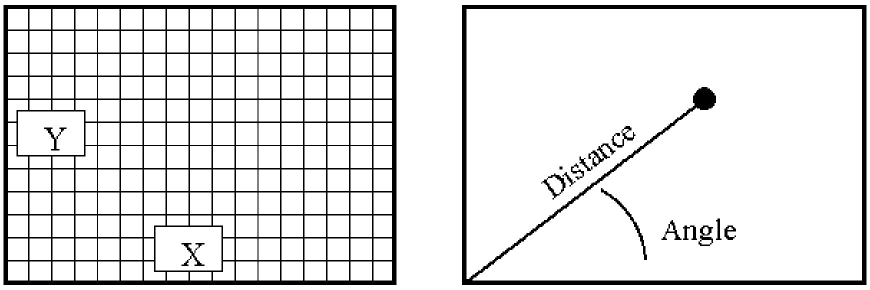


Figure 4.2-1 Map Types

Grid based maps have the advantage that they are representative of Cartesian co-ordinate systems like VDU's. However storage memory is required for every position within the space, whether it is occupied or vacant. A map of 100 by 100 regions would require 10,000 storage locations.

The resolution of such maps is critical, low resolution can result in loss of data and accuracy. Excessively high resolutions require significantly larger storage memory.

Vector maps can be more memory efficient, largely by only specifying the occupied points. Reasonable saving in memory are achieved, particularly on sparsely populated maps. The representation of such maps may not be so clear to operators and programmers.

4.2.1 Fuzzy Belief Map

A map covering the local area of the robot serves the purpose of collating and representing the raw data acquired by the panoramic laser range finder.

Each range measurement produces a vector co-ordinate from the robots position to the object struck by the laser. The task is to convert the sets of polar co-ordinates generated by the panoramic scans of the laser, into the equivalent Cartesian co-ordinates of the local map.

It is intended to represent this map using fuzzy techniques, a level is to be established for the belief in each space being occupied as well as a belief level in each space being clear from obstruction.

Contradictions between these two maps will add to the information conveyed and not detract as might be first assumed.

4.2.2 Grid Based Map

A grid map of the whole area must be generated, this map is to store generalised information about obstacle positions over the whole search space.

As the local fuzzy map is the decoding mechanism and accumulator for raw sensor data, it is this information that should be used to build the world map.

4.2.3 Node Map

In order to navigate through a known area, it is possible to make use of prior information to devise suitable routes.

Given the stated situation for this project is that the robot is to be suitable for insertion with no prior information, it is required that a suitable strategy for building and maintaining a set of known positions is implemented.

As exploration of the environment progresses, newly found areas can be recorded by identifying their relation to surrounding features or other known positions.

It will be necessary to describe rules for linking these known positions to each other, it is intended that a simple representation scheme be used to devise a set of known points, or Nodes.

An important observation is that the known points can be represented as either X and Y co-ordinates or vector co-ordinates, but must be relative to the original starting point of the robot.

When first activated, the robot will have no reference position of any kind, it is necessary to have at least one known point on the map at all times, the position of the robot.

The robots position is far more useful when used in reference to somewhere else, such as the position relative to surrounding obstacles on the real world map. However, the initial position of the robot will be unknown, therefore it is not possible to utilise real world co-ordinates as a reference. Despite this; as the robot moves and the current position changes, a convenient; and potentially very accurately known reference location forms where the robot was previously.

In this way even if a current location is not known, it is still possible to return to the previously known position. From this premise it is possible to begin building a reliable interpretation of the robots surroundings.

Some techniques allow markers to be used as navigation way-posts. Other methods insist that the initial position is explicitly stated, within the scope of this investigation, the initial position must be unknown.

As a response to the unknown starting position it is crucial that the co-ordinates are stored with reference to each others positions and not absolute world co-ordinates.   There must be clear distinction made between the Real-World co-ordinates and the co-ordinates of nodes.

A scheme is proposed that uses two simple rules to govern the creation of new nodes. The first rule states that new nodes shall be able to freely link to existing nodes given that they fall within a predefined distance and are visible within the line of sight.

The second rule provides a restriction in the complexity of the Links that form the Node map. Any node may only form one link to another node in each of the eight compass directions, N,NE,E,SE,S,SW,W,NW.

4.3 Obstacle Avoidance

4.3.1 Fuzzy Reasoning

Simple fuzzy behaviours can be amalgamated [12], to perform reasoned actions taking into account of many factors.

Wall Following and Obstacle Avoidance are examples of simple behaviours. A behaviour can be described in linguistic terms, for example an obstacle avoidance behaviour may be described as,

IF ‘left obstacle distance’ IS ‘very close’ THEN ‘turn is right’

AND

IF ‘right obstacle distance’ IS ‘very close’ THEN ‘turn is left’

Although in conventional logic these two rules would be mutually exclusive, turn could not be left and right, fuzzy logic allows us to evaluate both simultaneously to equate to ‘turn is left AND right’ or in fuzzy terms,

F(turn) IS F(left) AND F(right)

This then forms the basics of one behaviour or trait, it is intended that by developing a set of behaviours and by fusing them together, suitable strategies for local obstacle avoidance can be generated.

[12] M. Benreguieg, H. Maaref, C. Barret “Fusion of Fuzzy Agents for the Reactive Navigation of a Mobile Robot” page 388

4.3.2 *Static Obstacles*

The approach suggested,

"To fuse the drive speed recommendations, the output speed used is simply the minimum of the goal seeking value and the obstacle avoidance value."[13]

This can be interpreted as meaning that if the potential to avoid an obstacle is greater that the potential to achieve a target, then ignore the target and just avoid the obstacle, and vice versa.

This principle can be adapted to suit implementation with fuzzy rules. It is expected that this should provide some flexibility over the exact behaviour and lead to more natural reasoning, showing a balanced response not just a hard and fast rule.

4.3.3 *Mobile Obstacles*

It is possible to identify moving obstacles by the finding the points which have changed state between the previous and most recent laser scans. 'Clusters of each scan's new unmatched points, may correspond to moving obstacles.'[14]

The data from the laser range scanner provides and instantaneous description of the surrounding area. Comparing the most recent scan data with data previously stored in the fuzzy based local map, will identify areas of conflicting information. It should be possible to describe this method using a distance function evaluated on each cell of the local map.

As the fuzzy map provides an incremental belief factor, it is expected that some smoothing of the data shall occur. If the learning rate of the fuzzy map is too low, very fast moving obstacles may not be detectable. A rapid learning rate may increase the occurrences of incorrectly identified movement, as a result of noisy environmental conditions.

4.4 Path Planning

Methods such as the A* algorithm [15] can guarantee an optimal solution to route planning problems, and whilst it is always desirable to find an optimal solution, it may not however; be straightforward to define what is considered to be optimal.

[13] Driankov, D. Ed. "Fuzzy Logic Techniques for Autonomous Vehicle Navigation" P192 (2001)

[14] Mandow, A. Munoz, V.F. Ferandez, R. Garcia-Cerezo, A. "Dynamic Speed Planning for Safe Navigation" P235

[15] Russell, S. Norvig, P. "Artificial Intelligence a Modern Approach" P96-101

Just finding the route with the shortest distance travelled may not imply the optimal route. It may be necessary to consider many factors, the number of sharp corners or the proximity of obstacles may be far more critical factors in the route plan than the exact distance travelled.

This effect can be observed in most journey planning software, one of the questions asked when planning a route will be whether you wish to avoid any particular category of road.

A route planning system that finds any reasonable and suitable solution could be equally effective as more computationally expensive A* style planners.

4.5 Localisation

4.5.1 *Global*

The task of a global localisation method is to identify where on a known map the robot is currently located, even when it has been unexpectedly moved.

> A robot is at an unknown position in an indoor-environment and has to do a complete relocalisation, that is, it has to enumerate all positions that it might be located at. This problem occurs if, for example, the robot 'wakes up' after a break-down(e.g. a power failure or maintenance works) and the possibility exists that it was moved meanwhile [16]

In any circumstance where the robot becomes disorientated or restarts in a new location, a fresh grid map of surrounding obstacle data should be constructed. This map should be built upon as the robot moves and explores from the new location.

It is then possible to compare the new map to the main map previously explored and stored, such a comparison may require using a two dimensional correlation type method. Regions with strong correlation indicate that there is strong similarity measure between the new map and this position on the main map.

4.5.2 *Local*

Although odometry has unbounded errors which makes it unsuitable for extended measurement of position, as part of a larger system an estimate of the distance travelled can be useful in accurately and persistently determining the robots current location.

[16] Karch, O. Noltemeier, H. "Robot Localization – Theory and Practice" page 850

Given a known map of the environment and a suitable sensor system, a comparison between a stored data set and current sensor information is achievable. It is possible to interpret this as an estimate to the likelihood of the robot being at the position it believes it is.

In extension to the afore mentioned system, it seems feasible that a comparison between any position on the known map and current sensor data can be drawn.

This premise extends to Markov style localisation,

> The basic idea of Markov localization is to maintain a probability density over the whole state space of the robot within its environment. Markov localization assigns to each possible pose in the (x,y,θ)-space of the robot in the given environment the probability that the robot is at that particular position and has the corresponding orientation. [17]

By continuously evaluating the area surrounding the robot, a local area belief in position may be determined. Given a strong correlation it is possible for the position to be firmly determined, and if necessary adjusted.

The size of this localisation map should be dynamic, when the belief in the current position is strong this map should be small, the size of the map should increase as the strength of the belief reduces. Such increases in size result in many more calculations, however there is a requirement to increase the chance of including the correct area in the search.

This problem is greatly simplified with the addition of an accurate compass, the sensor data can then be formatted and analysed with a known orientation. The effect of this is to reduce the computational requirements by a factor equal to the implemented angular resolution.

4.6 Control Structure

The control structure undoubtedly requires a distinct separation between the Strategic and Tactical methods, as with MARCO [18] control architecture. This technique divides the control system into two layers, one for implementing basic actions and one for sequencing these actions.

[17] Fox, D. Burgard, W. Thrun, S. "Markov Localization for Reliable Robot Navigation and People Detection" page 3
[18] Qiu, J. "Task Sequencing for Mobile Robot Navigation" page 180

It is intended that a Low Level, Control Layer implement the simplest behaviours, behaviours such as Acquire Target and Avoid Object.

A High Level task sequencing layer may then provide temporal or time based control, including route planning and localisation. An excellent example of this type of control structure is [19]

This control structure allows each layer to specialise in simple tasks without the concerns of global problems. Communication between the layers must be considered carefully, control should be structured and not segmented.

[19] Qiu, J. “Task Sequencing for Mobile Robot Navigation” page 179

5 Implementation

5.1 OpenGL and Windows Programming

5.1.1 Idlefunc()

This function is an OpenGL callback routine, it repeatedly executes whilst the system is not busy updating the associated window.

All repetitive elements of the code are called from within this function, there are five primary functions that are executed with each iteration.

- ROBOT::RUNINFER() Processes the fuzzy reasoning elements.
- ROBOT::IMPLEMENT() Drives the robot according to the results of the fuzzy operations.
- ROBOT::MAKESCAN() Forces an immediate panoramic scan of the surroundings with the laser.
- ROBOT::UPDATEMAP() Updates the fuzzy belief map with the data gained from the previous panoramic scan.
- ROBOT::BUILDNODES() Assesses the robots current position and updates the node map accordingly.

5.1.2 Mousefunc() and Motionfunc()

These two functions share the task of processing mouse input. If the left mouse button is clicked and dragged on either the robot or the goal, the respective object will be dragged with the pointer.

By clicking and dragging with the left mouse button elsewhere in the window walls are drawn, the right mouse button erases walls as it is dragged. These walls are drawn by directly altering the Bool-map objects data structure.

5.2 A View Toward Hardware, Robot Design

5.2.1 Sensor System

The use of any compass would be problematic in the presence of stray electromechanical energies or magnetic fields. Despite modern EMC standards, an office environment would be a difficult place to obtain reliable compass readings.
Extremely inaccurate sensor data would render useless many methods for navigation and localization.

A potential solution to this problem is to build a map of unreliable compass areas, such a chart of errors could be developed with local algorithms for the correction of local errors.

5.2.2 Steering

The following segmentation algorithm describes the simulated motion of the robot. The algorithm takes as input parameters, the drive potential supplied to each wheel. These values are determined by the fuzzy speed and steering rules, and are directly related to the distance traveled by each wheel.

If it is assumed that a finite resolution can be applied to the turning momentum of the vehicle, the actual momentum can be described by calculating the distance each wheel travels in each segment.

A segment takes the form as shown below.

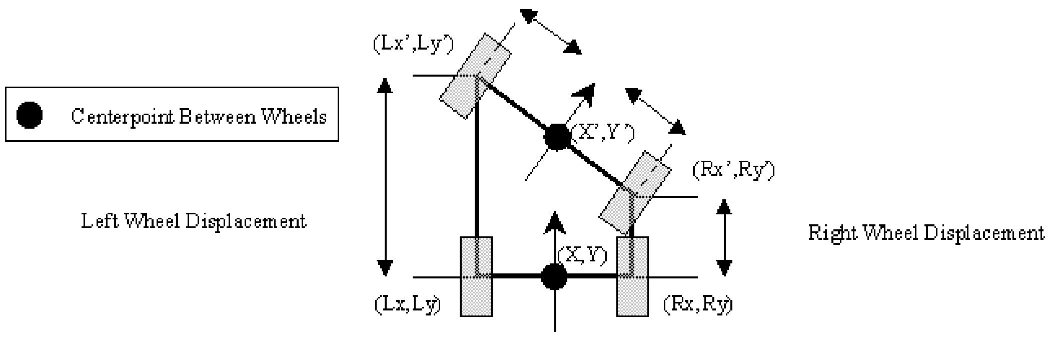


Figure 5.2-1 Motion Assumption

A succession of these segments will chain into an approximation of the curved path of the robot.

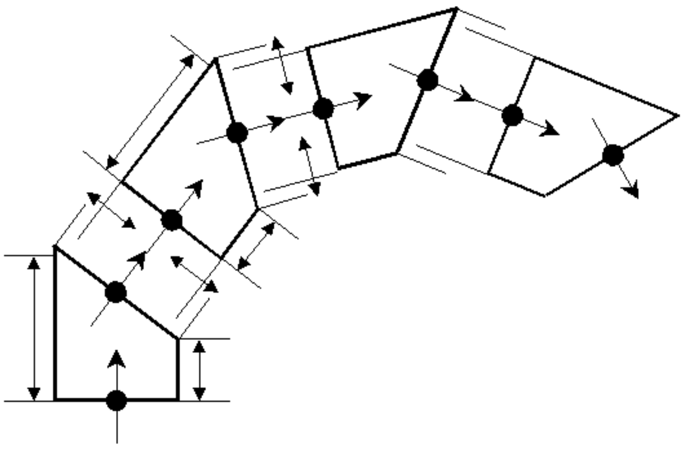


Figure 5.2-2 Motion Segmentation

As the length of these segments tends toward zero the position error is reduced. It is not feasible to implement this algorithm with minute increments, therefore there will be a certain degree of error.

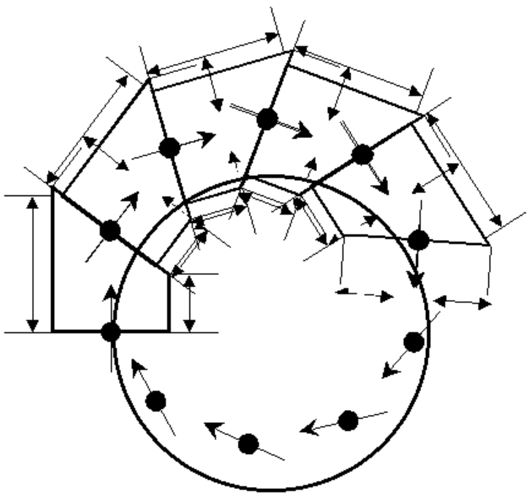


Figure 5.2-3 Displacement Error

The overshoot observed in this diagram is due to the initial assumption that the robot moves directly forward in each stage, the turning momentum would realistically be applied in a continuously curved path.

5.2.3 Acceleration

From the equation of Newton's second law,

$$\text{Force} = \text{Mass} * \text{Acceleration} \qquad \text{Equation 2}$$

The force required to accelerate an object is directly proportional to its mass. Therefore it is not physically possible to instantaneously accelerate or turn any mass, in the interest of modeling a real physical system, it is desirable to consider the mass of the vehicle.

In approximation to the effect of the vehicles mass, feedback from current velocities effects the calculation of the next velocities.

Component of the previous values for turning momentum and velocity are worked back into the new values, currently fixed at a 50/50 ratio. This means that the previous action influence the output as much as the current calculations.

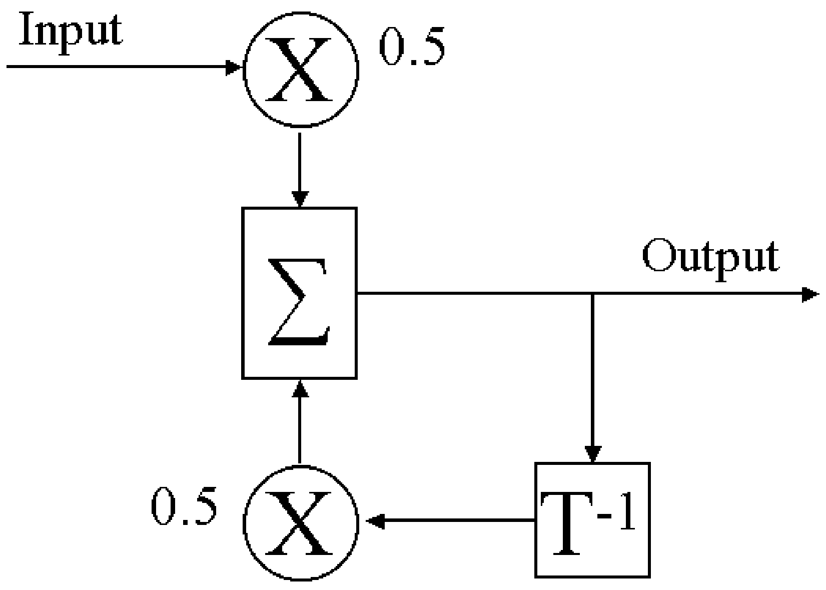


Figure 5.2-4 Recurrent Formula

As this formula is recurrent, previous actions are never completely forgotten, though the influence will be minor after just a few iterations ½ ¼ …2^{-n}

The effect this inclusion has is to smooth the rate of change in rotation and acceleration, when stopping suddenly some overrun is incurred.

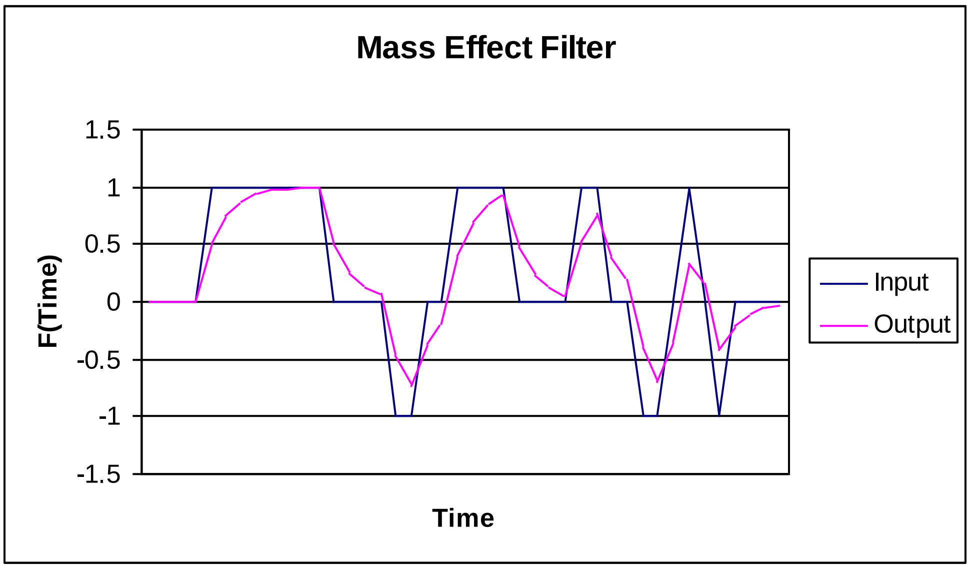


Figure 5.2-5 Filtered Motion Response

The larger the component derived for momentum the more likely collision with static obstacles is, extending the sensor range and reducing the maximum speed are both solutions that eliminate these types of overrun collision. Currently a high top speed and short sensor distance often bring the robot close to obstacles, this is deemed acceptable and provides some clear indications as to the limits of the current setup.

Another solution considered is the implementation of the breaking rule,

IF an obstacle is close and being approached very fast THEN breaking is advisable

Such a system would slow the vehicle much faster that just reducing the power applied to the wheels, resulting in higher maximum safe speeds.

5.2.4 True Bool Map

'In Visual C++ 5.0 and later, **bool** is implemented as a built-in type with a size of 1 **byte**.'[20] This can be shown with the call **sizeof(bool)** which equates to 1 byte. This means that a 100 by 100 grid map takes 10 Kb of memory.

It is possible to manipulate each bit of each byte of memory individually. Implementing the bool-map with a binary bitmap system would require 10,000 bits / 8 = 1.250 Kb, this type of optimisation could mean the difference between a feasible implementation and an unsuccessful one.

[20] Visual C++ 5.0 MSDN help

5.3 Fuzzy Logic

5.3.1 Fuzzy Rules

A fuzzy rule is described as a two dimensional relationship between two fuzzy input relations and one fuzzy output relation, the relationship is implemented via a matrix of state spaces.

Fuzzy rules are defined by attaching pointers to the relevant relations and matrix data structures to an INFER object, as described in **INFER.CPP**.

INFER::EXE() takes two inputs, both floating point values. Each input corresponds to one of the input fuzzy relations, the rule is evaluated by calling this method and it is these values that are used to evaluate the rule.

The defuzzification method of the output relation is called automatically and the resulting floating point number is the returned value of EXE().

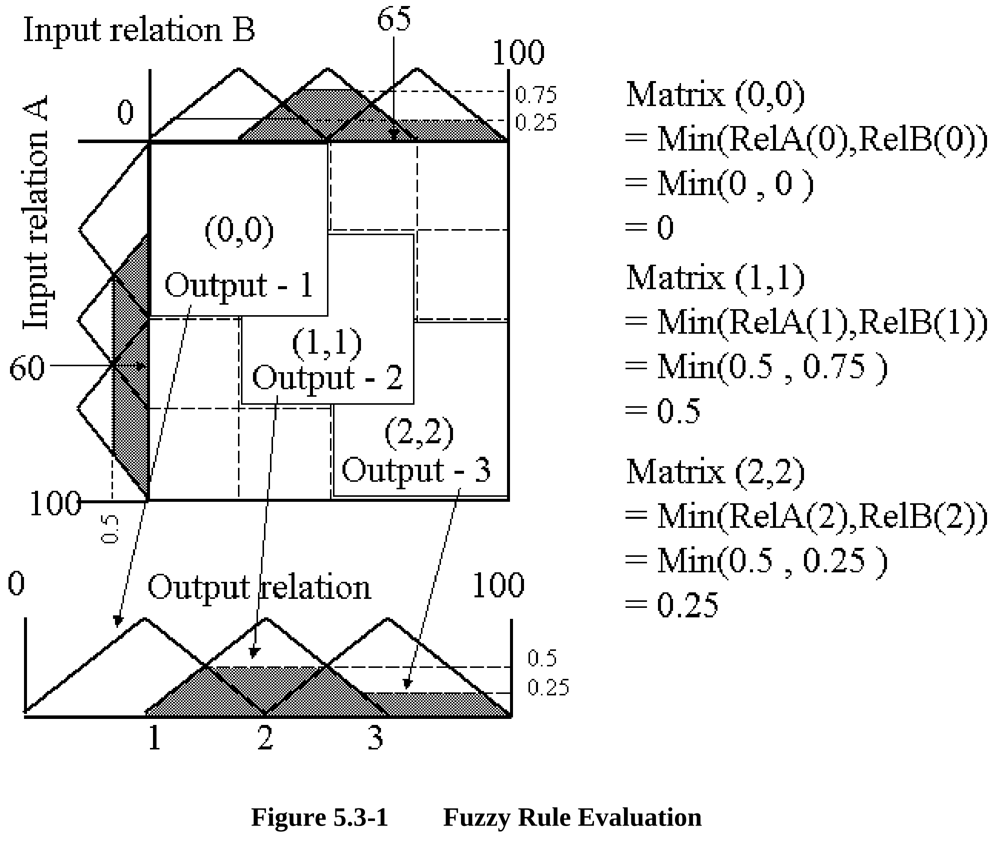


Figure 5.3-1 Fuzzy Rule Evaluation

Each element of the matrix grid is evaluated in turn. The corresponding memberships in both input relations are evaluated with the given input values, the minimum (AND) of these values is then computed. This minimum is compared with the maximum of previous values obtained for this membership, this running maximum is stored in the clipping limit of the output relations membership function that is associated with the matrix element being evaluated. If the new value is greater (OR) then the previous maximum, it supersedes the previous value.

5.3.2 Fuzzy Rule Base

There are just two fuzzy behaviours in the final implementation,

- **Target Tracking behaviour** consists of the rules,
  Direction error Vs Distance to target -> Target tracking
  Target direction error Vs Vehicle speed -> Vehicle Speed
  Target distance Vs Maximum vehicle Speed -> vehicle speed
- **Obstacle avoidance behaviour** consists of the rules,
  Left distance Vs Right distance -> Obstacle avoidance
  Minimum distance Vs Average distance -> Obstacle avoidance
  Steering angle Vs Obstacle Proximity -> Maximum vehicle speed
  Target tracking Vs Obstacle avoidance -> Steering control

There are two further fuzzy rule base sections,

- **Motor control signals** converts the rule base outputs to crisp signals suitable for driving the wheel motor controllers,
  Steering direction Vs Vehicle speed -> Left Wheel speed
  Steering direction Vs Vehicle speed -> Right Wheel speed
- **Map Maintenance** is a fuzzy rule for the representation of map data,
  Belief in wall Vs Belief in Clear -> Grey scale map change

5.3.3 Fuzzy Inference

A network of chained rules has been built by passing the defuzzified output from one rule as an input to another rule. The whole rule base is stored in an array of INFER objects in the only MAKEINFERENCE object ever created, **MAKEINFERENCES::BUILD()** is called to initialise this array and **MAKEINFERENCES::EXE()** is called to run the fuzzy inference that is fully described within this method.

5.3.4 Defuzzification

The defuzzification method achieves a centre of mass evaluation, it uses an integer based method, evaluated by the weighted sum formula.

$$\text{Weighted sum} = \frac{\sum x * y(x)}{\sum x} \qquad \text{Equation 3}$$

Where, x is an integer in N

and Y(x) is the fuzzy function of x

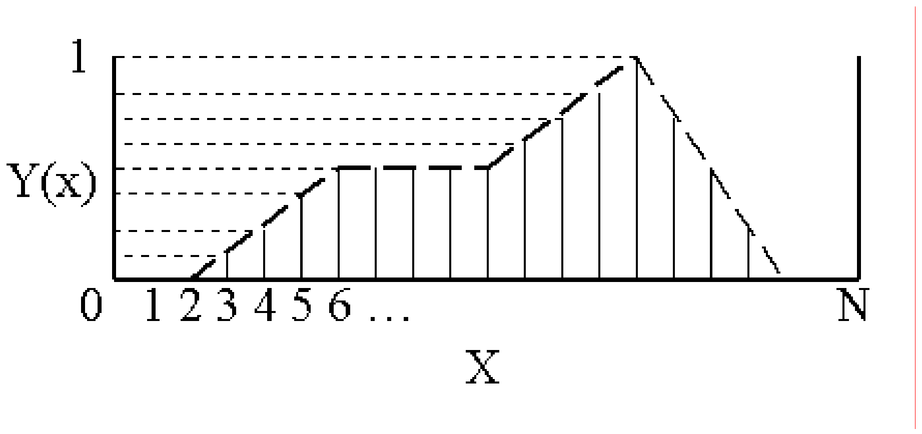


Figure 5.3-2 Integer Fuzzy Range

This methods resolution is defined on an integer base, relations possessing a very narrow range must be scaled to a more suitable scale prior to deffuzification. 0-10 would be a recommended absolute limit for functional defuzzification, however the resolution would be poor and the separation between memberships may be lost, narrow membership functions may be missed completely.

All the relations presented in this work are created with an integer defuzzification in mind, the normal ranges are 0 -> 100 and –50 to +50.

5.3.5 Integration

The fuzzy rules are objects, they have a callable execution method EXE(). It is easy to use these objects within any section of code that can refer to the INFER class.

In ROBOT::FTOGMAP() a fuzzy rule is used to evaluate and implement the change between fuzzy local map data, and greyscale global map data.

One of the strengths of this code structure is this ability to use fuzzy rules as logic bricks, they behave much the same as the math functions, power() or factorial().

This flexibility enables good integration between fuzzy and conventional logical systems, assisting the rapid development of robust and flexible code.

5.4 Map Building

5.4.1 Node Map

The distance rule governing the creation of new nodes is specified in MAPCONTROLLER::NODERULES().

The distance restriction is set to 8 < distance < 20, this keeps a finite resolution between nodes while allowing flexibility over the exact link distance.

The maximum distance limit exists to prevents exceptionally long links forming, this distance should always be within the laser scan distance, as other techniques rely on linked nodes being within the laser range.

The second condition ISVISABLE() fails whenever an object interrupts the line of sight between nodes, in this way links may only be formed to visible nodes.

The structure for each node is detailed in NODE.H,

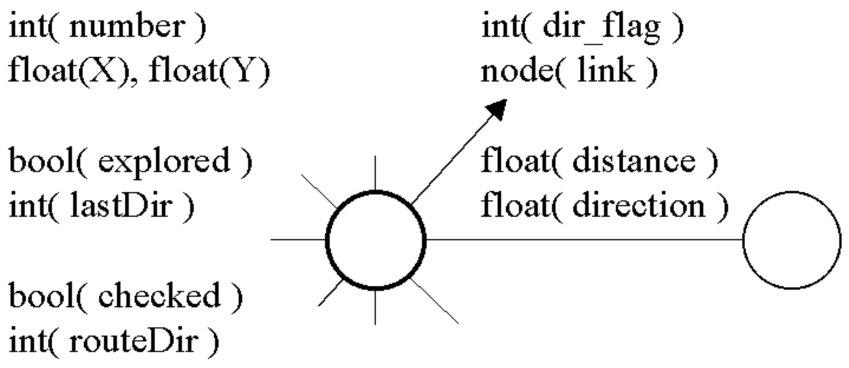


Figure 5.4-1 Node Map Data

There are several sections to this data structure, a unique ID number and (x,y) co-ordinates provide the basic information.

The **explored** and **lastDir** variables are used in map building, **explored** is set to true whenever a node has been fully explored, **lastDir** is used to continue exploring a node from the position where it was last left incomplete.

The variables **checked** and **routeDir** are used in route planning and following, the **routeDir** stores the direction toward the target of the route and **checked** is a flag used when constructing a route.

An instance of the other data exists for each link of a node, stored in an array of size eight, this is one for each of the compass directions and hence potential links.

The **dir_flag** is a representation of the links that exist from a node, the first element of the array corresponds to north, the elements run clockwise from there so that the third element represents east. This variable can take one of the following values,

- 1 – A link exists in this direction
- 0 – No link exists, or has failed to be created
- -1 – Exploring in this direction did not create a link
- -2 – Exploration in this direction failed

The **link** is an array of pointers to other nodes, if a link is identified in the dir_flag array, the pointer to the node that is linked to in this direction, can be found in the corresponding element of the link array.

The **distance** and **direction** variables store the vectors of links to other nodes, potentially this information can be used to further the work done on route finding, though it is currently unused.

5.4.2 EXPLORE Mode

When a node is reached with the intention of creating new nodes and links, the **LastDir** variable is evaluated as a starting direction to explore from.

Originally, when a node was created; the **LastDir** direction defaulted to north, this resulted in the robot always exploring to the top of the map first. More importantly the region that was known to the robot became very strung out, this implied that none of the explored zone was known very well.

By randomising the **LastDir** direction for each node upon its creation, the region is explored in a chaotic fashion. The average action taken, exhibits a tendency to explore the current area first. The strength of this method is found when conjoined with an edging algorithm that explores the most viable nodes first.

The ASSESSNODE() routine starts from the LastDir-1 direction, this is assessed for its suitability for exploration, if the chances of forming a new link are good a goal position is set, else the next direction clockwise is evaluated.

The directions are evaluated with the NODEMAP::ASSESSDIR function, this uses a partial scan of the surroundings to find an estimate of the distance to obstacles.

The average distance of a simple compass direction is found from the scan data obtained in this direction. Given that 32 measurement are taken in a complete panoramic scan and there are 8 possible directions, this means that (32/8=4) data points are evaluated to find the average distance.

The target position is plotted at the average distance away and along the vector of the longest distance. The actual position is recalled a little to keep the target away from walls, essential this only occurs when a wall is parallel to the robot. This position then becomes the new EXPAND target.

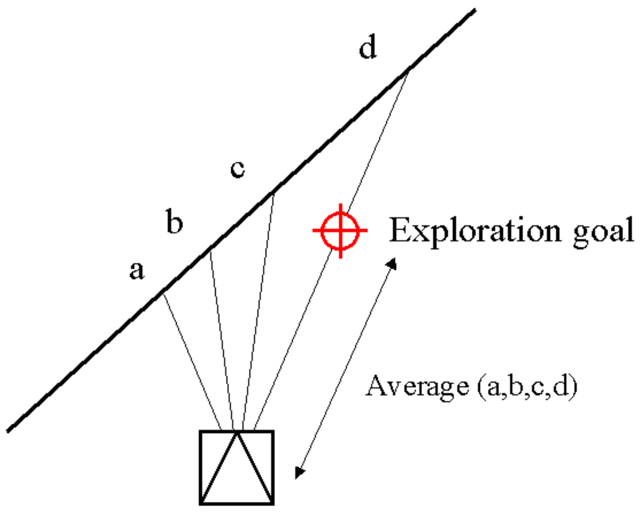


Figure 5.4-2 Exploration Target

For the data set: a= 8, b= 10, c= 12, d= 16

The average distance is the mean average of all readings,

= (8+10+12+16) / 4 Equation 4

= 46 / 4

= <u>11.5</u>

Along the vector which intercepts a wall much later at 16 units distance.

This goal must conform to a minimum distance. If the average is too short it can be assumed that there is an obstacle to close to be able to form a new node, hence there is no advantage to evaluating this direction.

Assuming that there is a valid target generated, the robot then proceeds toward this location. One of two things should then occur,

a) The robot reaches the location and no link has been formed. In this case the direction flag of that node is set to (-1) reflecting the unsuccessfulness of this direction. This can occur if there are many nodes close already. In this case the map will develop as the robot assesses other node in the area.

b) As the robot approaches the local target a link is formed in the target direction. This has then been a successful and productive exploration. The appropriate flag for the nodes are set and the robot then finds the nearest suitable node to continue exploration, the sub-goal is then set to that node and the robot proceeds there.

5.4.3 Optimisation with an Edging Algorithm

The basic strategy will eventually explore all possible positions on a map, however as the number of nodes is increased, the time taken to fully explore every node will increase in an exponential fashion.

By observing an exploration it will be noticed that the majority of successful explorations will occur along the edges of the current node map.

The technique presented here has been inspired by the capillary action of water, when presented with an edge, some water molecules will no longer be completely surrounded by other water molecules. These molecules find themselves with spare bonding potential, this potential is shared among the surrounding molecules, hence molecules within this area will be attracted more strongly.

This principle has been adopted for identifying nodes at the edge of the known map. By selecting nodes with the fewest links created, or failed to be created, a rough impression of edge nodes is possible. It is by exploring these nodes first that exploration around the edge of the known map is achieved.

Specifically this method currently finds nodes with most potential (unexplored) links then find closest of these. This then becomes the next node to be explored.

An alternative method finds a subset of nodes, those that are the closest, then the one with the most potential link directions is selected from this subset. As this method would return the closest best node, time spent travelling between nodes could be reduced.

It is anticipated that a X-Breed of the two methods would provide optimal exploration within the bounds of this technique.

5.4.4 TARGET Mode

Finding a target that is not visible from any node has been achieved by modifying the exploration strategy. Instead of randomly exploring the local area, a bias is set to explore nodes closest to the target, this is shown in NODEMAP::FINDNEXT().

The direction of expansion **LastDir** is set toward the target, consequently the expansion of the node map occurs in the direction of the target.

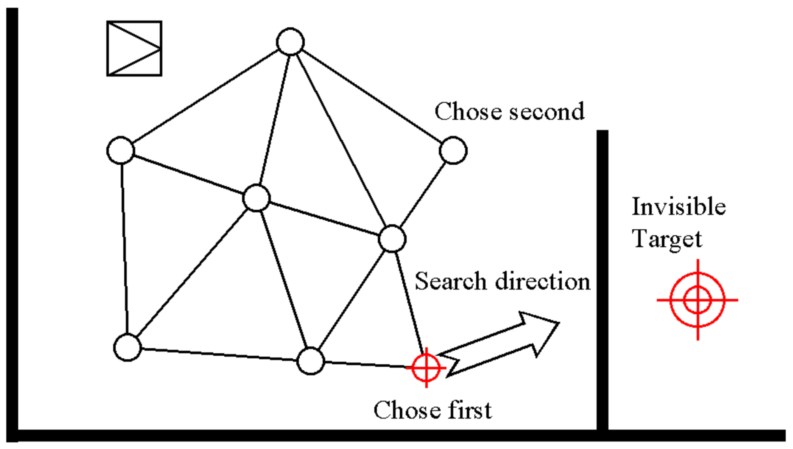


Figure 5.4-3 Hidden Target

In the absence of a visible node at the endpoint, the closest non-visible point is selected.

5.4.5 Fuzzy Map

The ROBOT::UPDATEMAP() routine is called after each laser scan, it both converts raw sensor data into Cartesian co-ordinates, and updates the fuzzy maps according to the following two rules:

- The distance returned from the laser for each direction equates to the position of an obstacle, the result of each hit is to increase by a fixed amount; the corresponding grid co-ordinate of the fuzzy 'belief in Wall' obstacle map.
- The second rule is an extension of the first. Points in-between the robot and an obstacle cannot be occupied, else the laser would not penetrate. All the corresponding intermediate positions on the fuzzy 'belief in clear space' map are adjusted by a fixed amount to reflect this.

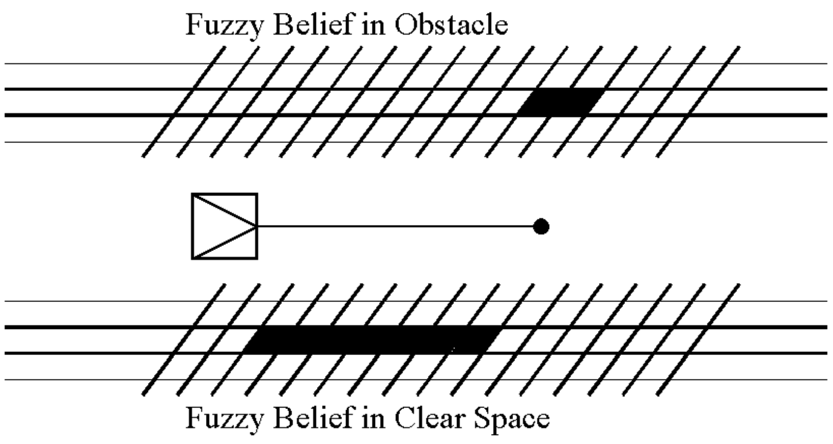


Figure 5.4-4 Fuzzy Local Map

The amount each position is adjusted by on each successive scan is determined by the following formula.

Current value += (100 – Current value) / 5 Equation 5

Given a low starting value this formula results in an initially quick change but increasingly slow change as the belief is reinforced to higher values. Consequently the change in belief between 0.20 and 0.25 represents far less reinforcement than the change of 0.90 to 0.95.

These values only ever increase, data is accumulated as the robot scans its environment, and is evaluated as the robot moves. The accumulated scan data is converted into a grey scale belief level map by the function ROBOT::FTOGMAP() This function uses the fuzzy relation shown below for the actual conversion,

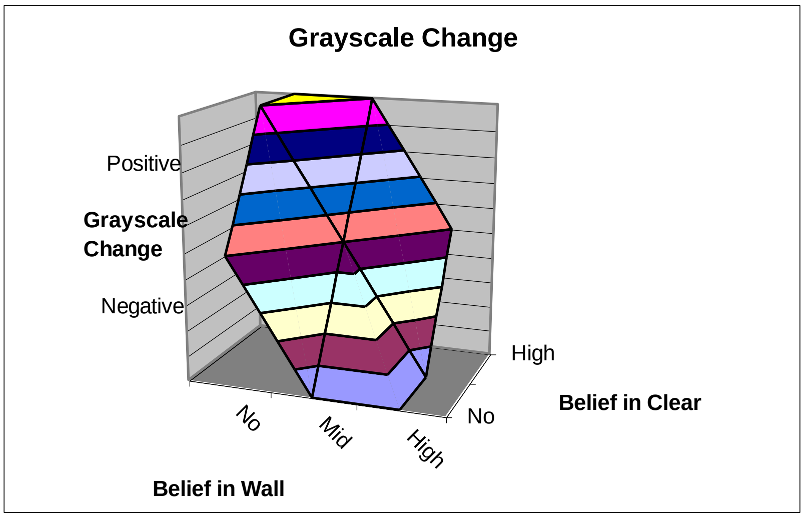


Figure 5.4-5 Greyscale Map Change

The fuzzy map has been fixed at a size such that the laser range never exceeds its bounds. As the fuzzy map is square and the laser scan circular, the corners are redundant, this occurs simply because is easier to represent rectangular arrays in computer code.

5.5 Obstacle Avoidance

5.5.1 Fuzzy Min Vs Fuzzy Average

The obstacle avoidance fuzzy rules are based on the following assumptions,

- In the absence of any obstacles, a direct route to the target can be taken.
- If all obstacles are far away and present no immediate danger, slight avoidance of visible obstructions to the desired path is advisable.
- As obstacles become close, evasive manoeuvres must be taken.
- Spurious range readings can be ignored so long as if they do represent a real obstacle, there would be no immediate danger.
- Spurious range readings may be indicative of a real obstacle and hence if the potential exists for danger, these readings must be treated as true.

With these guidelines in mind the rule base has been constructed to allow general obstacle avoidance to be derived from the averages of sensor data in the given directions. However, if the minimum distance in any direction suggests that there may be an obstruction which could endanger the robot, another rule enhances the avoidance behaviour.

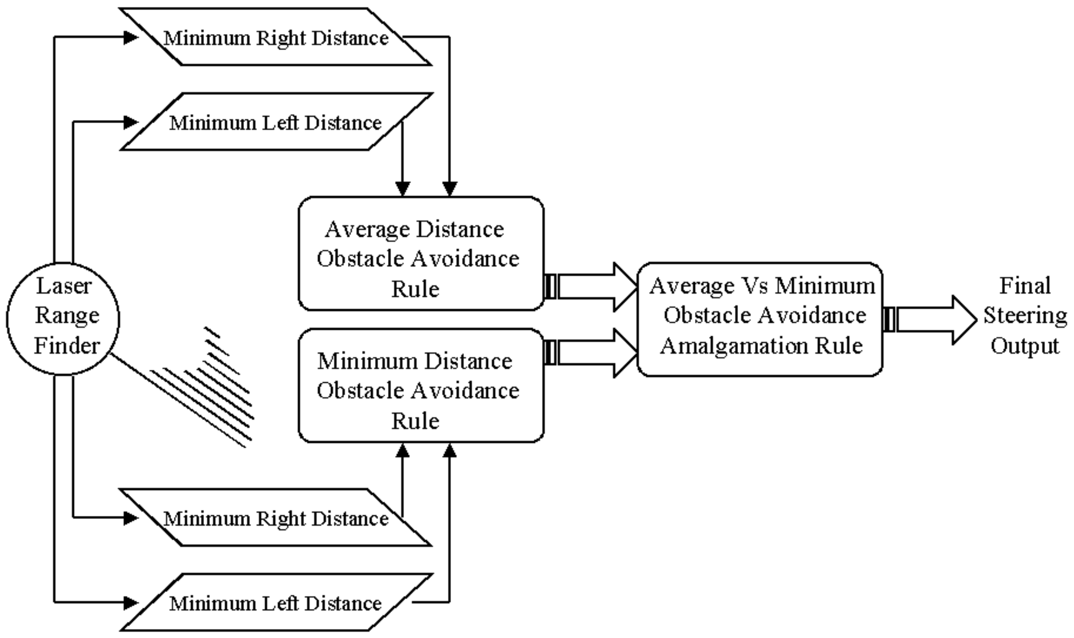

Minimum Right Distance
Minimum Left Distance
Average Distance Obstacle Avoidance Rule
Laser Range Finder
Minimum Distance Obstacle Avoidance Rule
Average Vs Minimum Obstacle Avoidance Amalgamation Rule
Final Steering Output
Minimum Right Distance
Minimum Left Distance

Figure 5.5-1 Obstacle Avoidance Rules

5.5.2 *Fuzzy Symmetric Indecision*

When presented with an obstacle dead ahead and target beyond this, the fuzzy rule base can suggest equally; the solutions of turning right and of turning left to avoid the obstacle.

This situation occurs partly because of fuzzy logic's tendency to aggregate solutions, which is ironically one of the features that makes fuzzy logic so appealing and flexible. A common situation is,

> Human beings who suddenly meet one another at a door or hallway often find themselves in a symmetric indecision dilemma. [21]

Without intervention the robot is going to either hit the obstacle or stop in front of it in a state of indecision.

5.5.3 *Frustration*

In the presence of an obstacle the standard avoidance rules normally play out their role. If facing a wall and a slight turn to the left or right is exhibited this action will have a positive reinforcement of itself. A slight turn to the left will bring the left side around to face open space and the right side toward the wall.

Positive feedback can be a strong instinct, however all rules are prone to exceptions and this is no different. On occasions the robot may be presented with a difficult avoidance challenge that is failed to be solved with the basic obstacle avoidance rules. Symmetric indecision is often linked closely to this type of trouble.

In response to a conflict in rules a frustration level may be determined which can influence the behaviour of the robot.[22]

If basic methods are persistently unable to rectify the situation, the level of frustration will build until it reaches a trigger point. At this time the behaviour dramatically changes, switching from a balance between target finding and obstacle avoidance to a purely obstacle avoidance behaviour. In this mode the target direction is ignored, this behaviour is intended to result in the robot driving immediately into a clear space by retreating from all obstacles.

This is equivalent in the humans in a doorway example of one person deciding that they shall move to the left, even if the other person mimics this behaviour, keep moving left and clear a path to the right.

[21] Driankov, D. Ed. "Fuzzy Logic Techniques for Autonomous Vehicle Navigation" page 197
[22] Driankov, D. Ed. "Fuzzy Logic Techniques for Autonomous Vehicle Navigation" page 194

By sticking to a chosen behaviour the nature of the problem is changed, the current implementation holds on to this frustration behaviour for ten cycles, after this the behaviour is returned to normal.

If the problem remains and frustration persists the final resort is to eventually declare the task at hand to be impossible. At the very least the solution is not going to be found given the current circumstances and may benefit from a change in tack. Once the current task is skipped and a new and different goal will be established.

5.5.4 Moving Obstacles

In order to implement the method for identification of moving obstacles suggested in the development stage, it is necessary to define some intermediate grid based maps. The conflicting regions within the fuzzy map should be copied to a separate grid layer the same size as the fuzzy map.

This map when compared to another generated from the previous scan would identify the change of position identified as moving obstacles. The changes within local regions could then be averaged from one layer to the other, producing vectors for the motion of identified objects. This method is similar to identifying the centre of mass of moving objects, yet addresses the issue of multiple moving objects or objects with unusual shapes.

> Clusters of each scan's new unmatched points, may correspond to moving obstacles. These are represented by their center of gravity in order to simplify the problem, and most important, to minamize the uncertainty about the actual size, shape and volume of detected obstacles. Each of these centers is then compared with the ones from the 5 latest scan intervals in order to update the trajectories of mobile objects. Particularly, the latest 3 centers of an obstacle are considered to fix a straight line as its estimated trajectory. [23]

Subtracting from these vectors, the vector representing the vehicles speed and direction equates all vectors for stationary obstacles to zero. Leaving just the vectors for moving obstacles identified. These vectors then may be mapped onto the belief map to provide a predicted position for the moving obstacle relative to the vehicle speed.

As the belief map updates the obstacle map these objects should be identified in there predicted positions, by this method the standard obstacle avoidance techniques can be used for moving obstacles, requiring little mathematical processing.

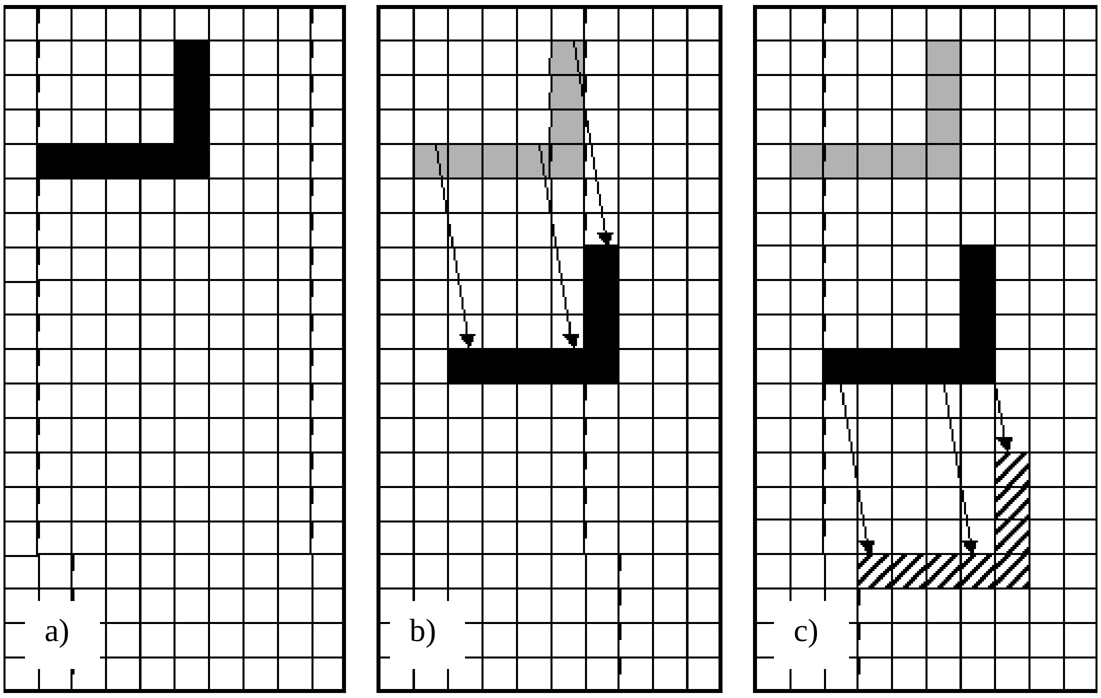


Figure 5.5-2 Mobile Object

a) First identified b) Second position provides movement vectors c) Vectors are extrapolated to predict next position

5.6 Path Planning

A Depth first search can be generated by using a recursive algorithm, this is a function that calls itself. Such an algorithm first finds the closest visible nodes to both the starting and ending positions, a route is then established between the two.

All routes are planed along paths between previously identified positions, and are represented as a stack of intermediate nodes between the current position and the goal. This stack is generated by the following actions,

- First putting the ending node on a stack of nodes to be evaluated, then for each of the nodes eight possible linking directions,
- Calculate the opposite direction of the link direction and pass this return direction to the node that is linked to. This direction is stored in each linked-to node along with a marker to show it has been evaluated.
- Put each of the nodes linked to, on the stack of nodes to be evaluated.

[23] Mandow, A. Munoz, V.F. Ferandez, R. Garcia-Cerezo, A. "Dynamic Speed Planning for Safe Navigation" page 235

- If the starting node is one of the nodes linked to, then the algorithm has completed its task.
- Otherwise Repeat until the stack of nodes runs empty, at this point all nodes will have been evaluated and no solution will have been found. It is then not possible for this route to succeed.

This algorithm starts at the end and ends at the beginning, in this way with only one pass a reverse route can be ‘left behind’ as the algorithm examines the map. This reverse route leaves markers at each node showing the way to proceed next, the robot simply follows these markers to the goal.

This algorithm produces a depth first search, a strategy that quickly finds the first successful route. It is not possible to guaranteed the optimal route has been found until all the possible routes have been evaluated.

Using a list to store the nodes waiting to be evaluated instead of a stack converts this algorithm to a breadth first search.

As this algorithm runs backwards, if a node has already been evaluated by the algorithm, its return direction will not be altered again in this route finding task. This fact is responsible for one of the characteristics of this route finder, this is that routes are evaluated on a very much first come first serve basis.

The A* method guarantees to find the optimal solution, actually though in certain circumstances the standard A* algorithm can fail to converge. Successful A* routes are achieved by continuously evaluating the potential cost of succeeding in any given direction, the direction with the lowest cost becomes the ‘next best first’ direction.

By dynamically evaluating route segments having the same start and end positions, as routes are created, the winner takes all strategy is acquired. This would mean that if a node had already been examined and again the route finder looks toward it, instead of being ignored the two routes should be evaluated.

Comparison of route sections requires backtracking along each route until a common point of divergence is reached, the distances covered while backtracking form an evaluation function for the two paths. In this way the best route is selected as the progressor and an optimal solution considering the whole map can be guaranteed.

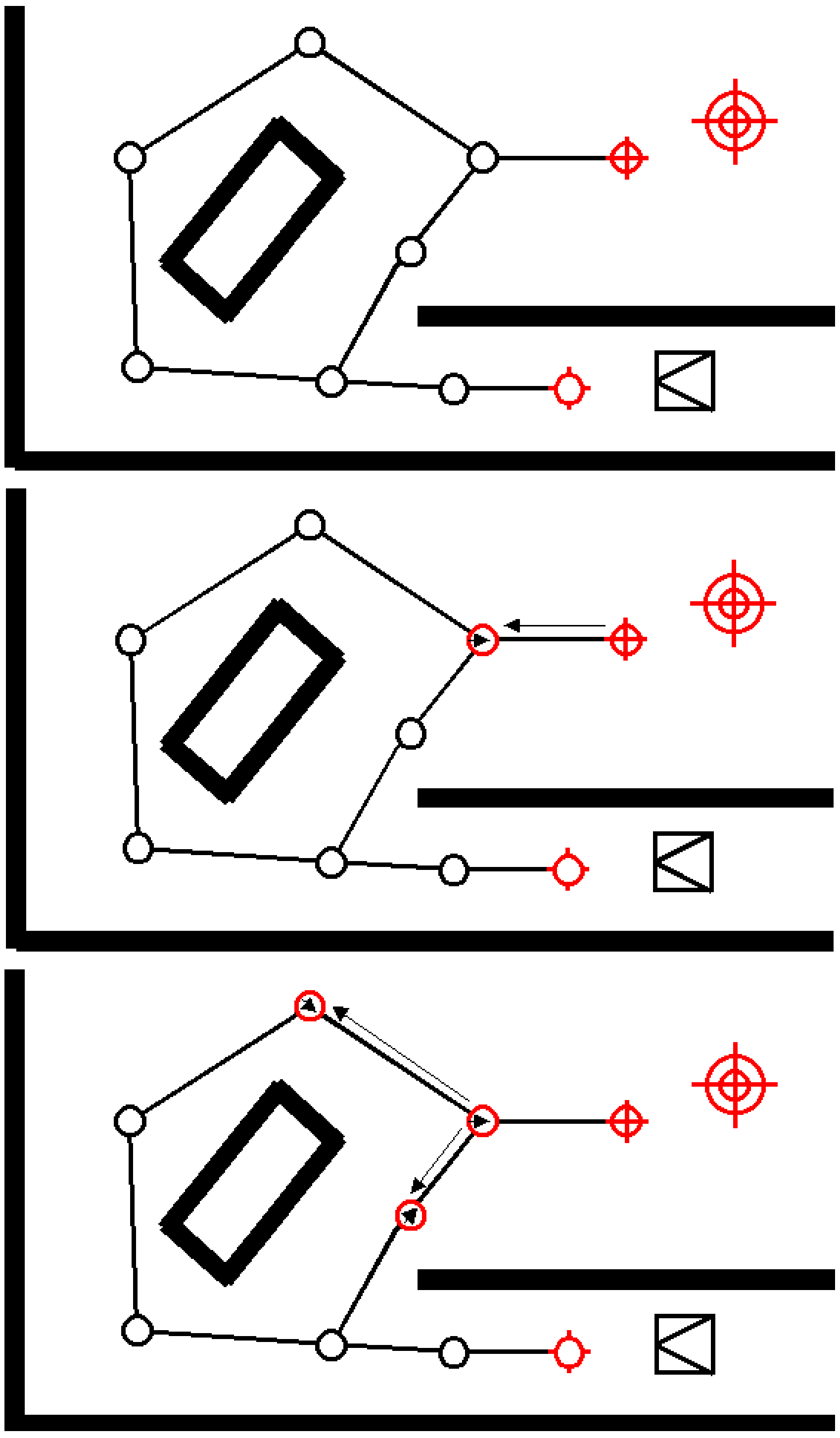


Figure 5.6-1 Route Planning Sequence A

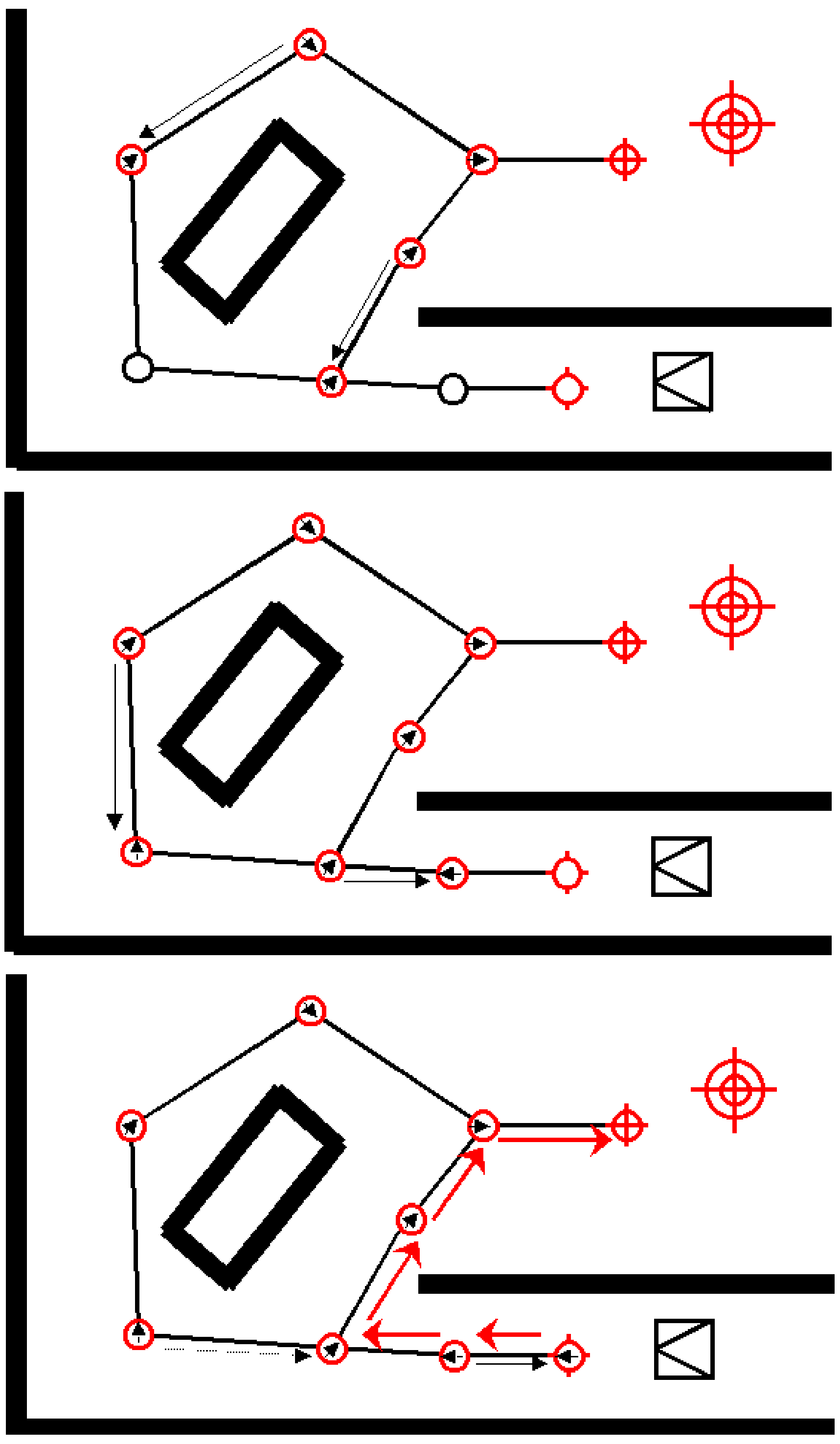


Figure 5.6-2 Route Planning Sequence B

5.7 Layered Control

The control structure has been divided into three distinctly separate functional categories, the first two layers are analogous to the MARCO layered control architecture in which control is divided into planning and execution layers.

- **Layer 1**
  Fuzzy reasoning for obstacle avoidance and target acquisition, the real time and sensor based systems.

- **Layer 2**
  Path planning and following strategies, including map exploration methodologies.

- **Layer 3**
  Robot Functions, either Target Finding or Map Exploration.

5.8 Localisation

5.8.1 Global

If the level of belief in being correctly localised on the map degrades to a point where it is unsafe to continue without taking action, a global localisation strategy must be adopted.

By building a new map of the same nature as the grid type greyscale map from current sensor data, a comparison can be drawn from this new map to all the areas in the old map.

If a strong correlation exists at some position and there is no contradiction or ambiguity it should be reasonable to assume that the new map has been correctly matched with a section of the old map.

The data from a stationary point may not provide enough information to serve as a reliable comparison between maps. Even if a reasonable comparison can be achieved there may be several identical places on the old map, resulting in contradictions and ambiguities.

In these circumstances it is necessary to explore the new area, building a larger argument map as exploration proceeds. This must be continued until all ambiguities are resolved.

If the situation arises that the new map cannot be matched to any part of the old map, and no reasonable exploration will resolve this situation. In these circumstances it must be declared that the robot has been placed in an unknown area of the same map, or on a completely separate map.

Although at this point the robot can continue its activities based on the newly generated map of its surroundings, it should be considered that re-exploring known areas is costly in both time and effort.

There then becomes the requirement to match two partial maps together, or at the least the edges of the existing maps.

5.8.2 Map Matching

A jigsaw puzzle approach would allow continuous evaluation between several pieces of a map. By assessing the interaction of obstacles at the edge of a map, something like a signature or a bar code could be produced and this may assist matching of known segments.

5.9 Fuzzy Inference Overview

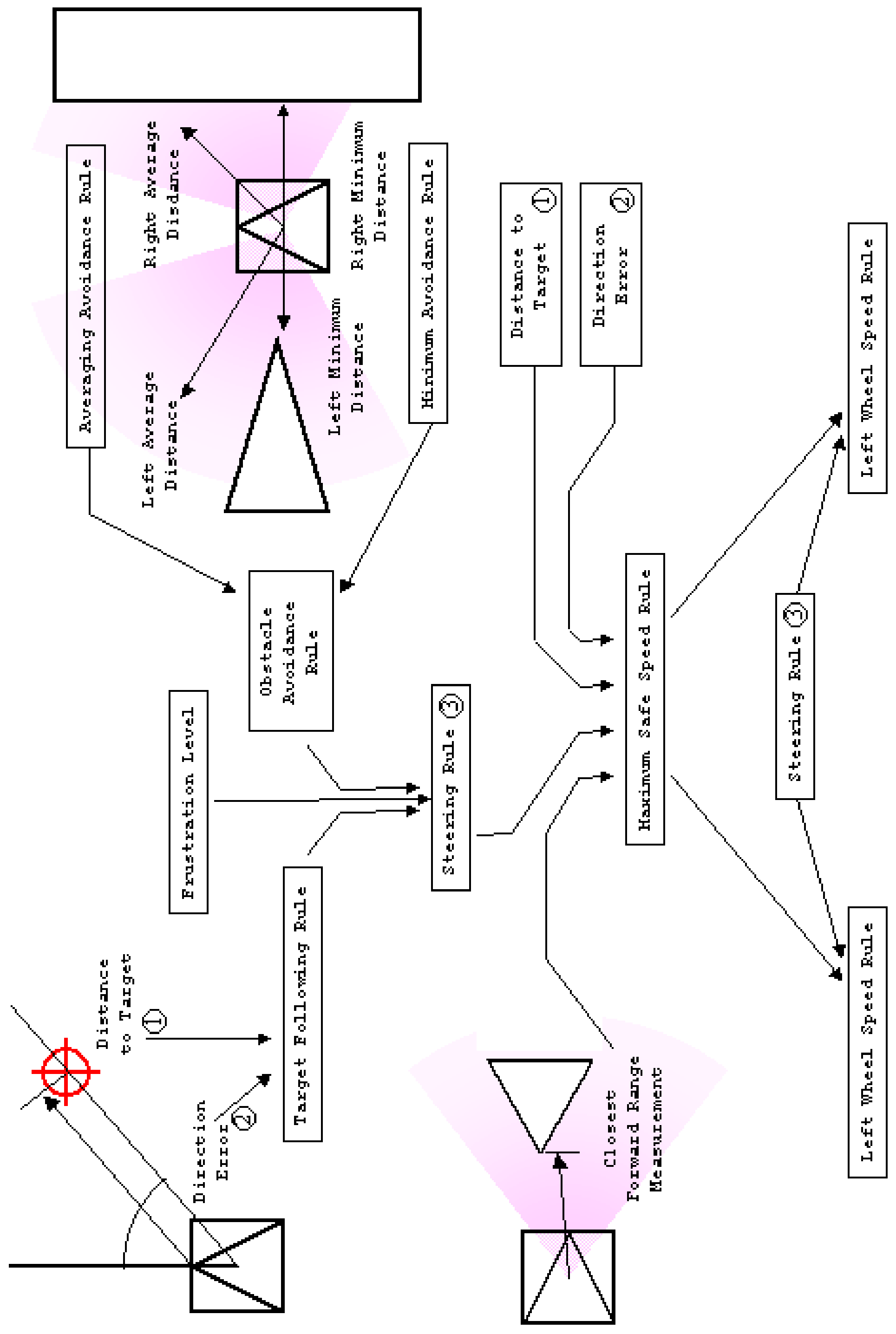


Figure 5.9-1 Fuzzy Inference Structure

6 Experimentation

6.1 The Chair Leg Problem

Small obstacles such as the legs of a chair present a class of problem that this implementation cannot operate around, they obstruct the passage of the robot but are not necessarily interpreted as obstacles.

This class of obstacle cause undesirable effects including,

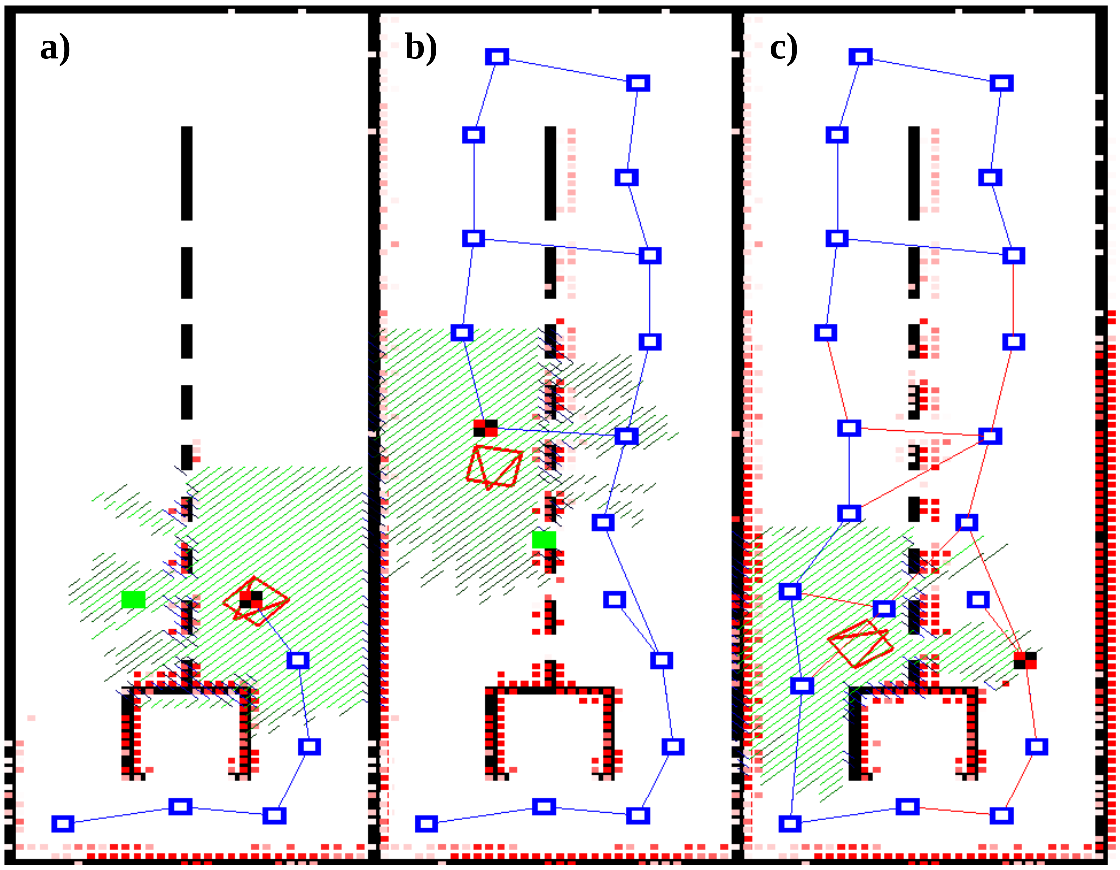


Figure 6.1-1 Small Obstacles

- a) Sub goals may be calculated to be through a narrow gap.
- b) Links can be formed through narrow gaps, line of sight information is used to maintain validity of links.
- c) Goals may be visible through narrow gaps, nullify an otherwise valid route.

The suggested solution to this problem is the addition of an exclusion zone to all observed obstacles, gaps narrower than the robots minimum traversable width should be treated as solid obstacles.

6.2 Map Changes Route Blocked

The first slide depicts a complete and valid map. In order to demonstrate the effect of environment changes that invalidate **Node-Map** data, the goal has been set in a position that allows two different routes to be taken. One of the routes however, has been blocked after being selected. (Slide 2)

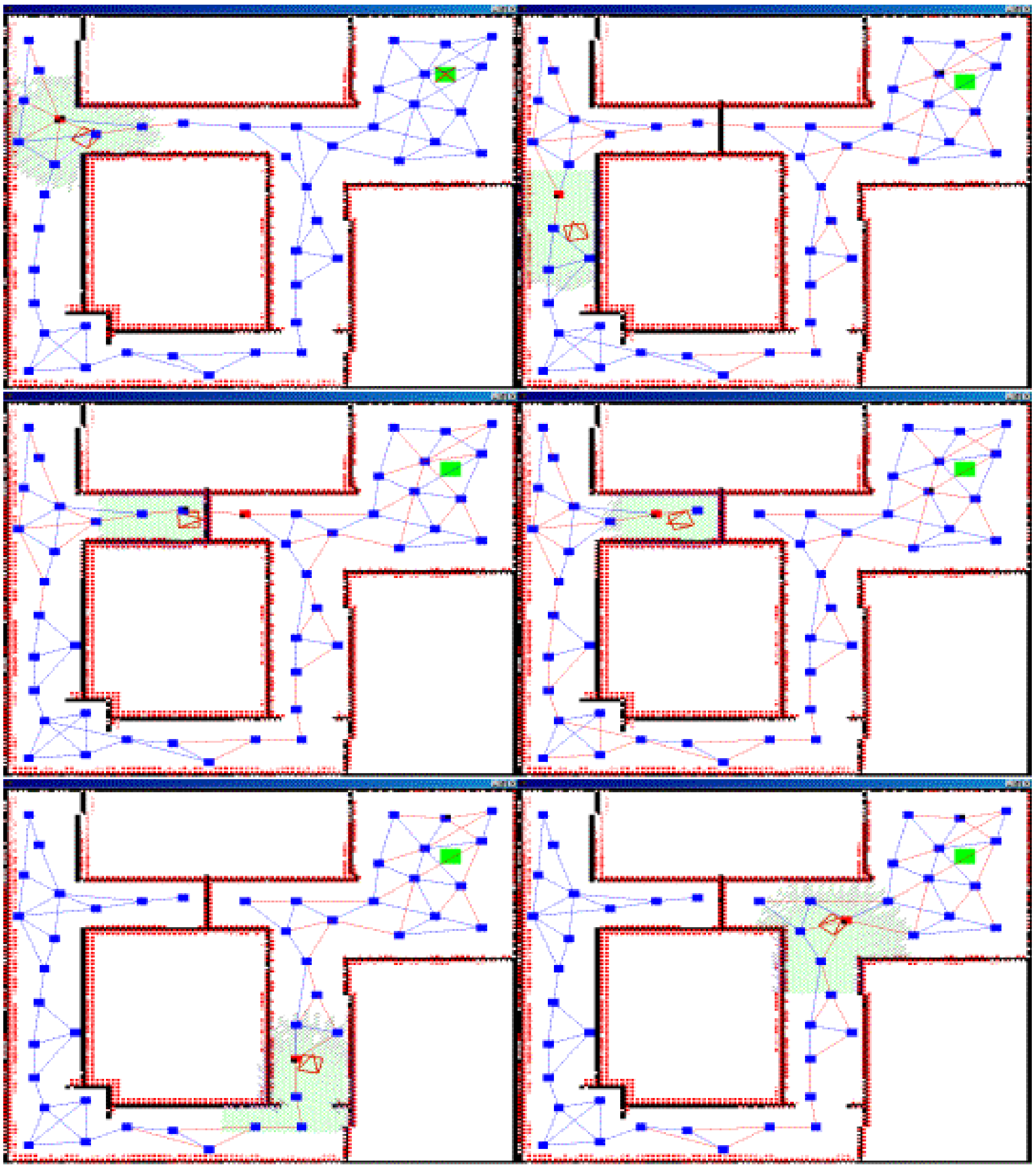


Figure 6.2-1 Route Blocked

An attempt is made to follow the planned path and traverse the invalid link (Slide 3). When this fails, the link is broken (Slide 4), and a new route is planned. The alternate longer route is followed to the goal (Slide 5 & 6).

6.3 Exploration of Maze Type Maps

This sequence shows how the exploration strategy works toward the goal in a complex environment. Initially progress is made towards the goal, then as necessary around the terrain.

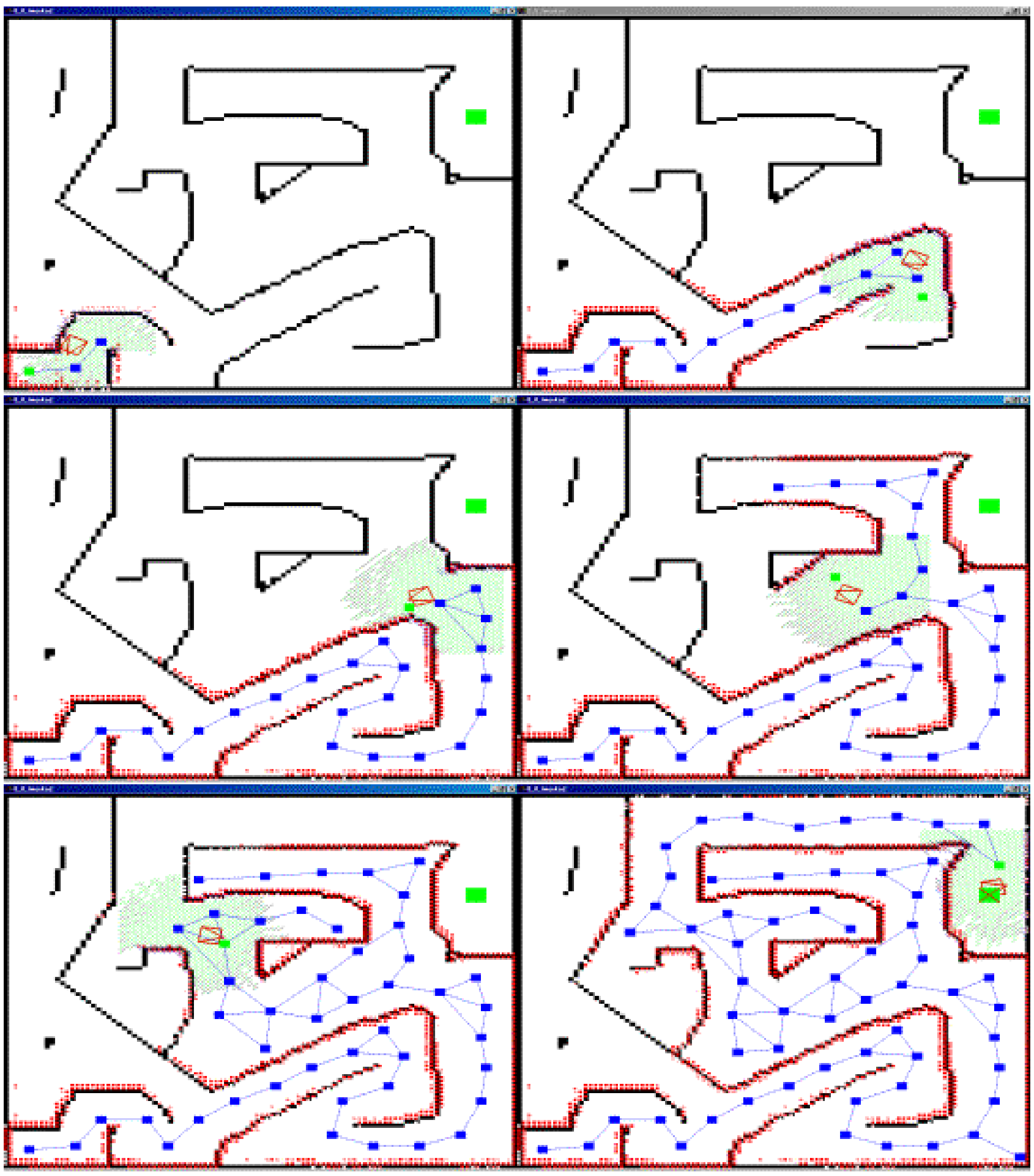


Figure 6.3-1 Maze Map

Although this map has few incorrect paths, it can be seen how the exploration strategy takes the form of a fluid, supplied at the starting point and flowing toward a gravity centre at the goal, filling depressions as it progresses.

The following maze map took approximately 40 minutes to complete, the first 4 slides show the exploration required to overcome a large local minima in the maze. Slides 5 & 6 occur within 2 minutes of slide 4. With 95% of the exploration time taken to overcome the local minima, this is a clear demonstration of the effect that the shape of the search space has.

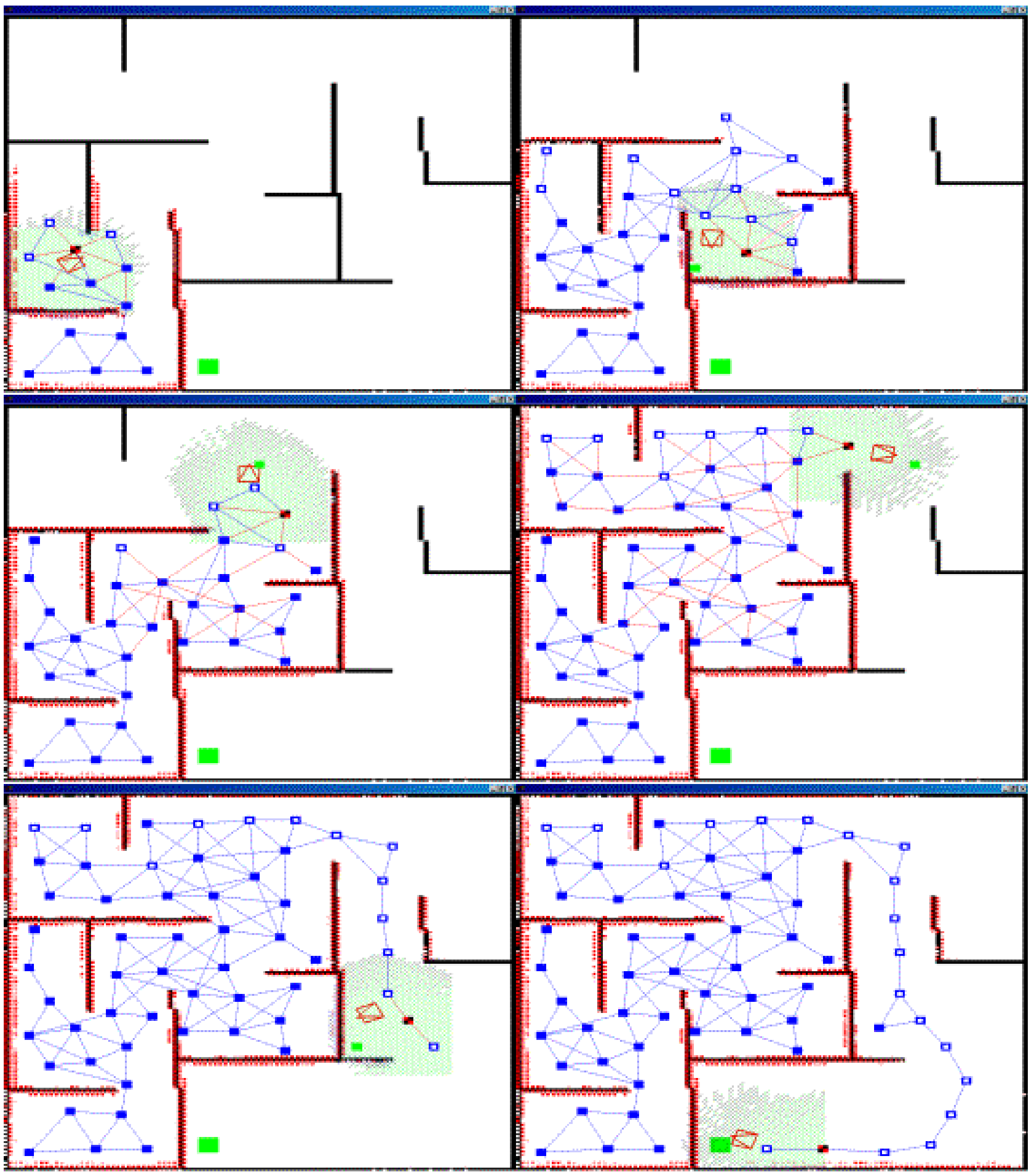


Figure 6.3-2 Maze Map

6.4 Map Exploration of an Office Floor Plan

Ultimately this robot has been designed for insertion into unknown office type environments. The following series of slides shows the progressive exploration of such an environment.

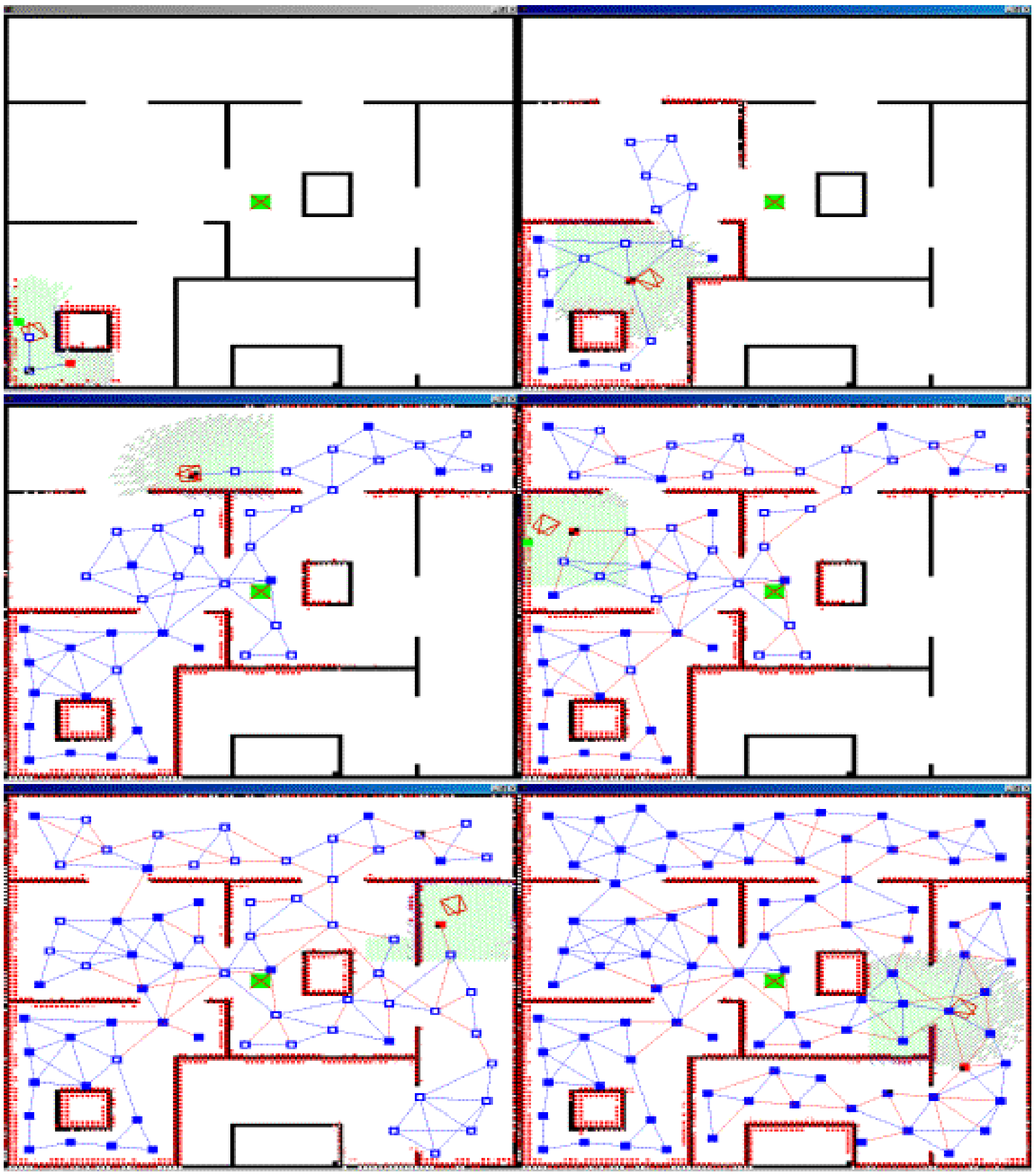


Figure 6.4-1 Office Floor Plan

6.5 Map Exploration of a Warehouse Type Environment

A cluttered environment that restricts motion to simple paths will be explored in a very short time. It took less than half an hour to explore this map to the stage shown in the last slide, it then took a further half an hour to completely explore all the nodes.

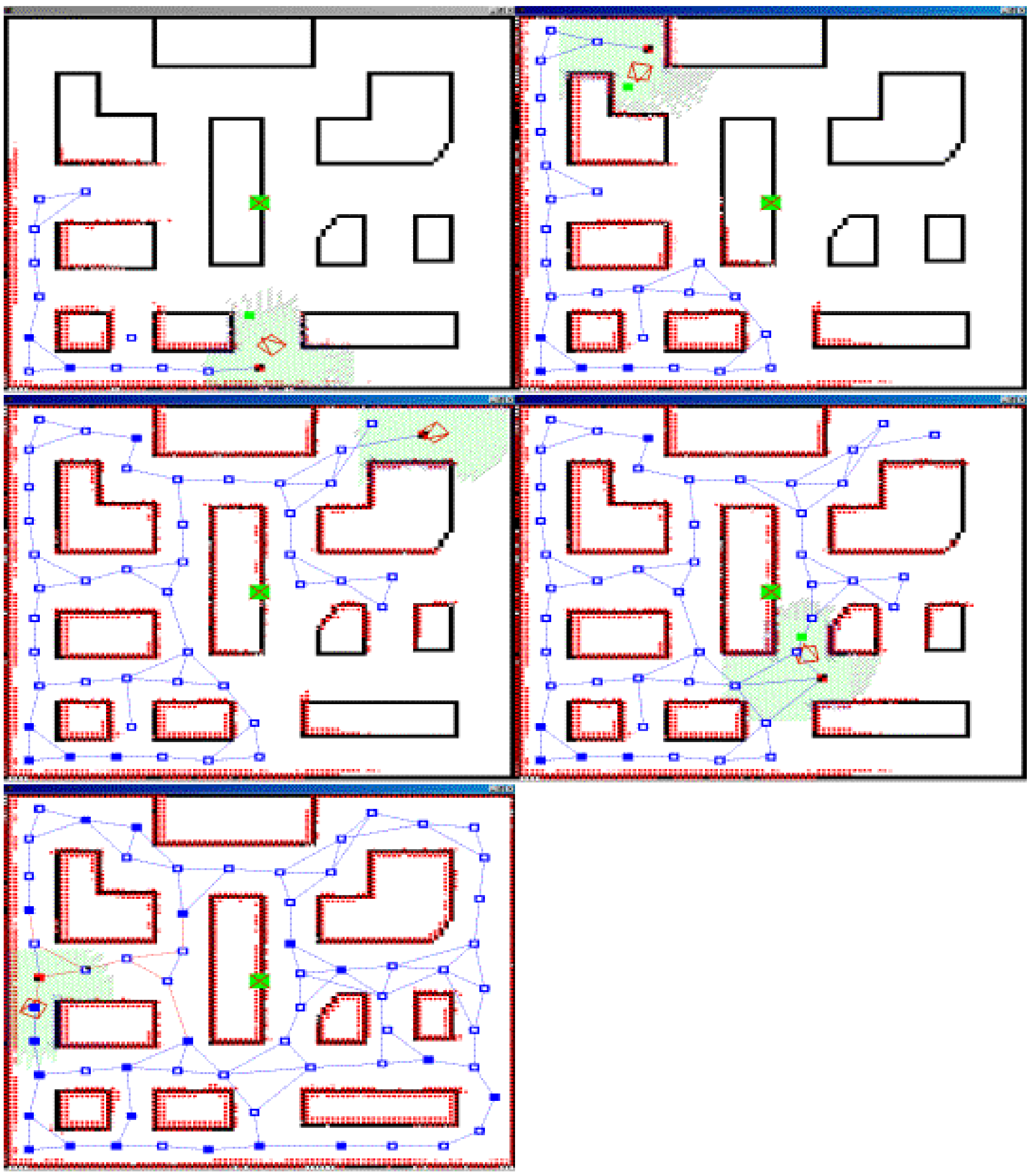


Figure 6.5-1 Warehouse Floor Plan

The quick completion time is due to two main factors;

- Much of the search space is occupied by obstacles, the map is in a sense much smaller than it appears. Restricting the number of nodes that are created correspondingly reduces the number of nodes that must be explored.
- Almost all the nodes link to just two others, this is because all other directions are obscured by obstacles. Forming chains of nodes reduces the number of explorable links and hence the convergence time.

This frame shows the planning of a route through a similar heavily obstructed environment. Routes can be planned from any known position to any other known position. Environments like this may be found in storage areas such as warehouses, the picking of products from storage areas is a common task ideally suited for robotic applications.

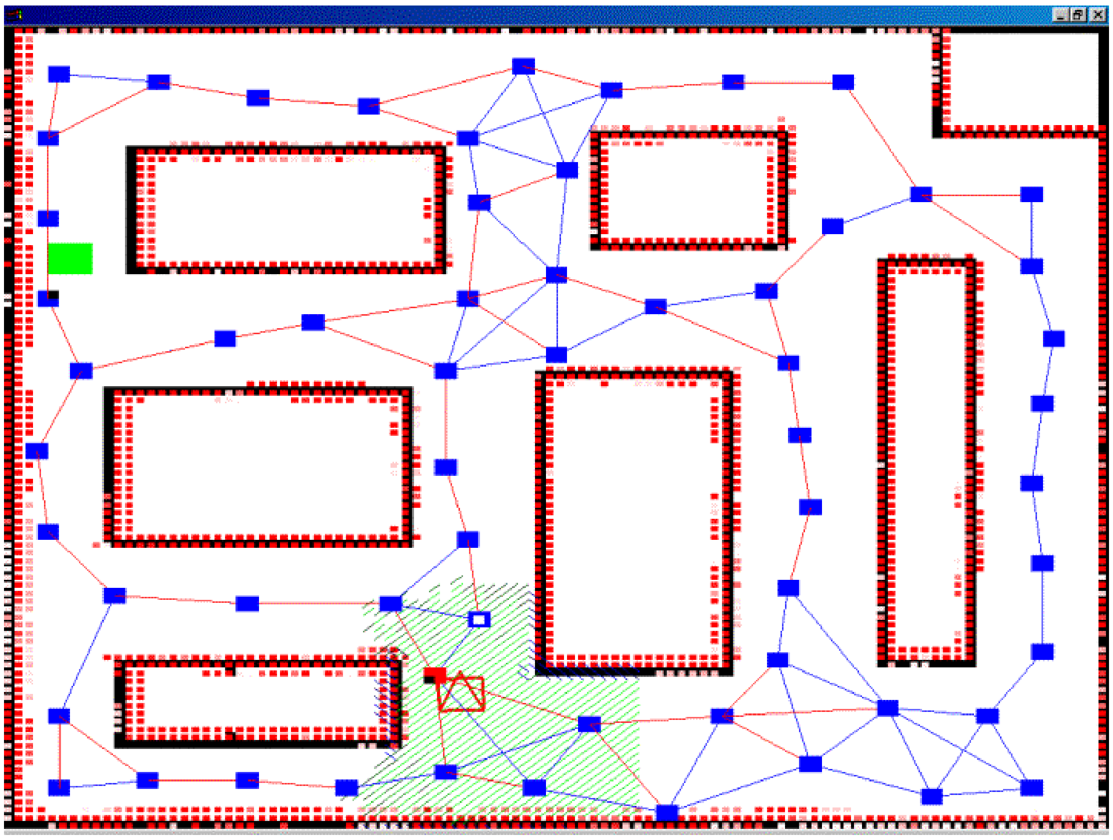


Figure 6.5-2 Warehouse Floor Plan

6.6 Map Exploration of Open Space

This exploration took several hours to develop to this stage, the number of nodes and the potential explorable directions contribute to what is a very long completion time.

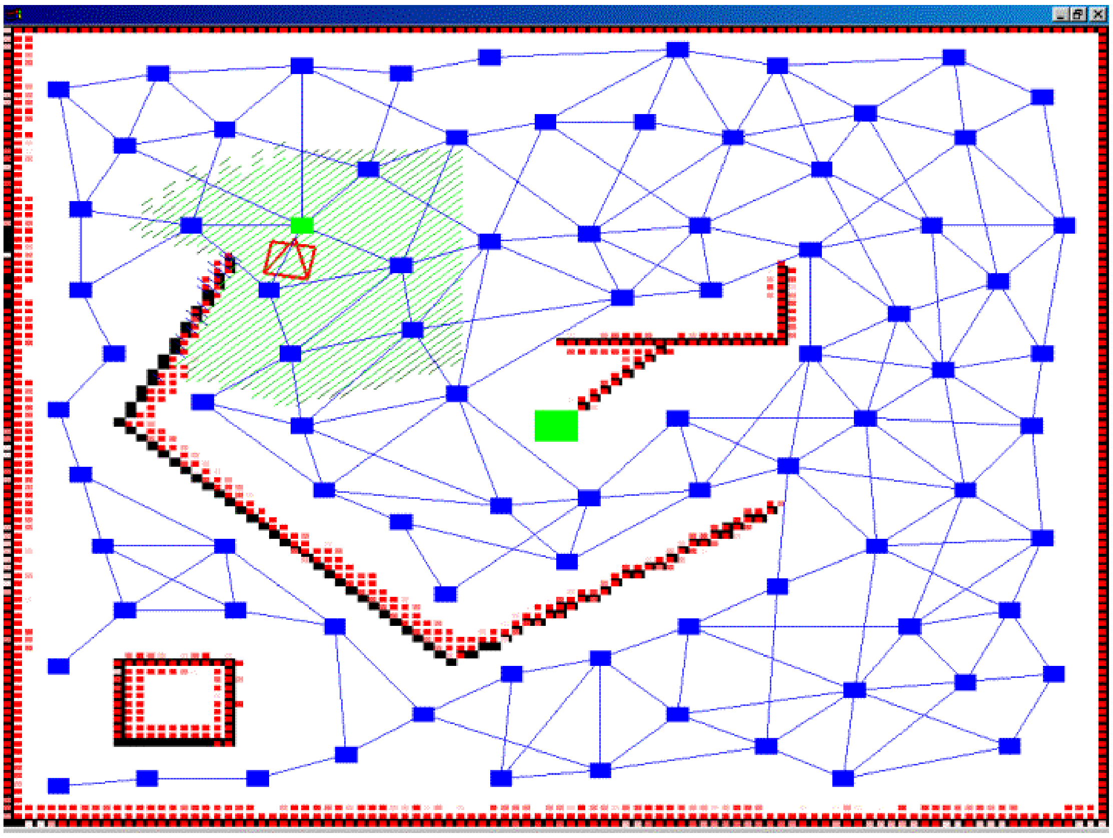


Figure 6.6-1 Open Space

Despite this long convergence time, a good network in the open space has been generated, the node map allows any position within the map area to be reached via a route plan.

6.7 Concave Obstacles

Concave obstacles present a particular problem, unlike convex or flat obstacles it is unlikely when trapped by a concave obstacle; that the situation will ever be resolved accidentally. It is required then to have some supervisory methods that can provide support to the standard navigation techniques.

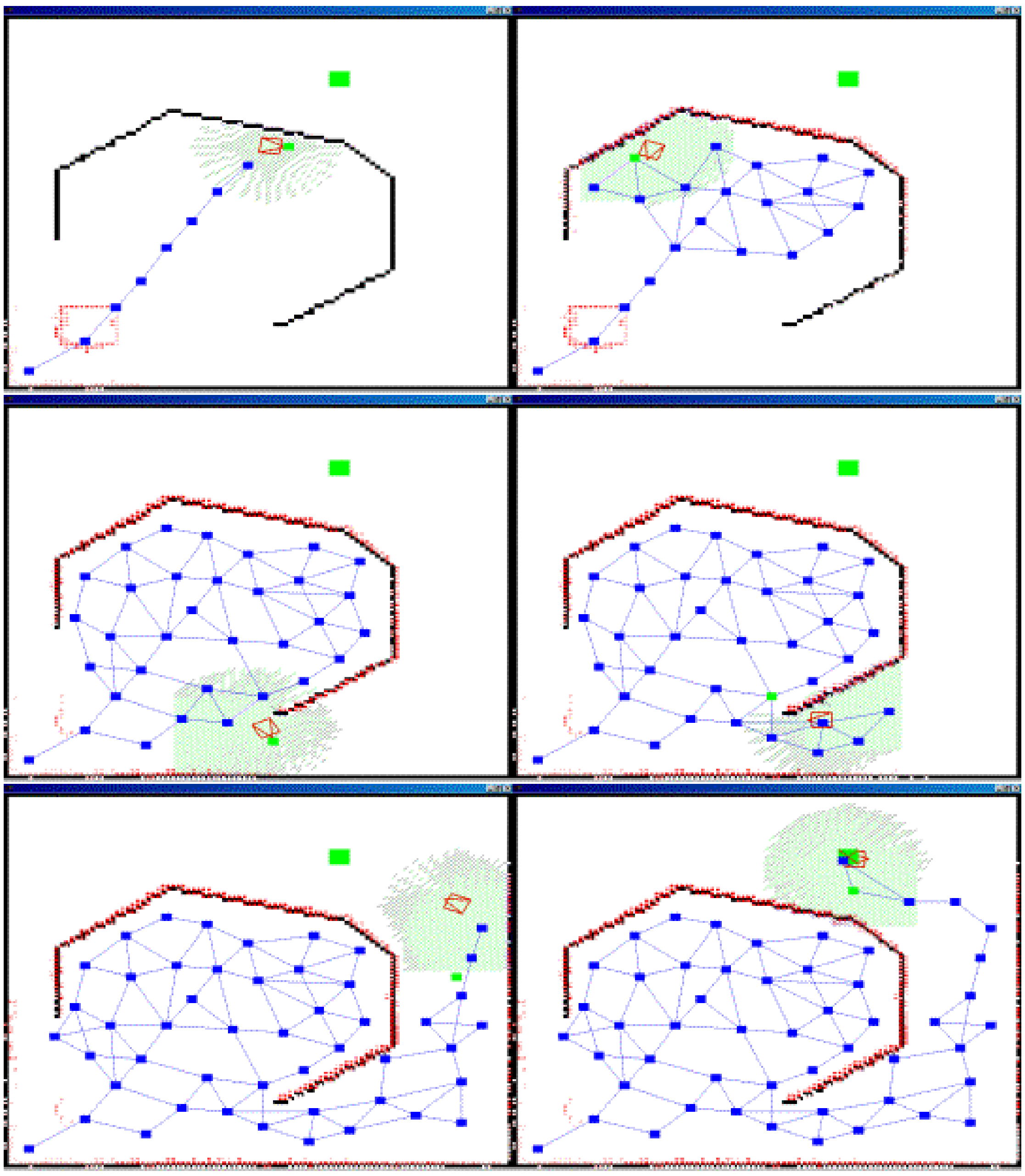


Figure 6.7-1 Concave Obstacle

The strategy employed to overcome this type of obstruction is described fully in chapter ‘implementation’.

- The nearest node is explored in the direction that is closest to being toward the goal. (Frame 1)
- All nodes are explored in turn, closest first. (Frame 2) Eventually most nodes will be completely explored. (Frame 3)

  The map will continue to expand as new positions are created and explored.(Frame 4)
- When a new node is created preference is likely to be shown to this node.(Frame 5)

However, if the only new node fails to connect to the goal or spawn new nodes, the whole procedure will start again from the closest node and working away from the goal eventually re-considering all nodes in the network.

The result is that although a solution is guaranteed it may be some time before it is found, this example took over 3 hours to complete. In response to the length of this experiment a modification to the **Find-Next** node algorithm has been implemented.

When a direction from a node is evaluated for its potential for expansion, the proximity of the generated sub-goal to other nodes in the network is considered. If there is any node within the minimum specified distance between nodes then this direction is marked as unobtainable.

The results of this modification are quite dramatic, a similar experiment to that above took just 20 minuets to complete.

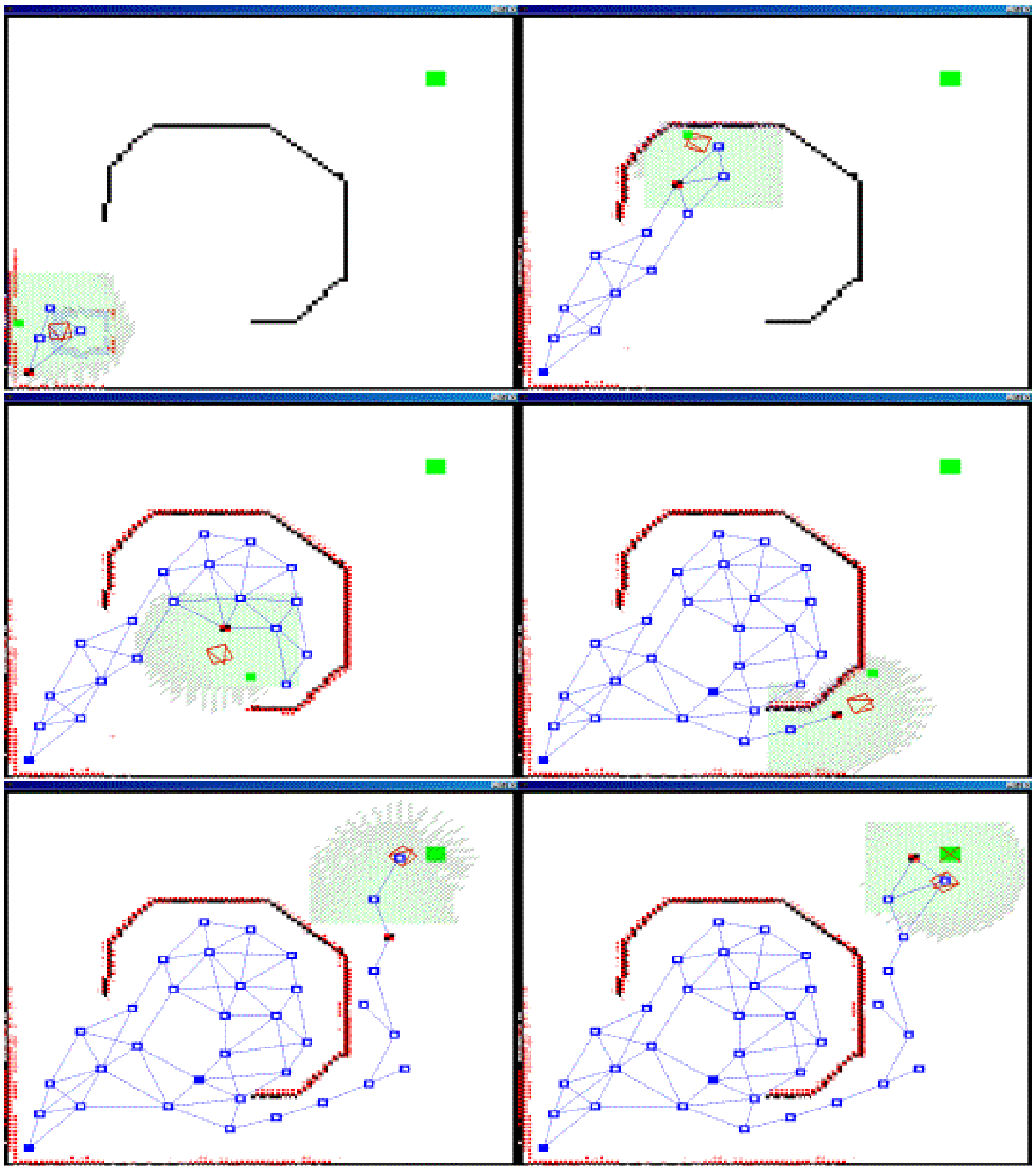


Figure 6.7-2 Concave Obstacle

7 A Comparative Study

7.1 Speed Planning

Several categories of speed constraint have been defined in *Speed planning Method for Mobile Robots Under Motion Constraints* [24] these and their respective implementations are listed here,

- *Mechanical Considerations*. Maximum possible motor speed. –
  The maximum speed for both wheels is set in the Robot object code by the variable **MaxSpeed**.
- *Kinematic Considerations*. The maximum acceleration and turn rates –
  The feedback derived from the previous motion defines the mass effect of the vehicle, this in turn constrains the maximum acceleration/deceleration and turning rate.
- *Dynamic Considerations*. Deflection from desired path –
  As there is no designed path to follow and perfect localisation is assumed, the consideration of dynamic constraints as define above, is unnecessary.
- *Operational Speed Constraints*. The closeness of an obstacle or goal –
  To prevent overshoot, fuzzy rules for obstacle and goal distance have been implemented, these slow the robot as it approaches the goal location.
- *Safety Speed Navigation*. Dependant on working environment –
  Not considered in this implementation.

Speed planning Method for Mobile Robots Under Motion Constraints presents a method for attaching speed markers to a sequence of robot postures along a route.
The reasoning presented is that this provides contextual information about the nature of the environment, an awareness; or memory of certain features and restrictions.

Pro-active strategies could be developed on line, providing more than just pure obstacle avoidance and knowledge about the vehicles maximum safe speeds.

By collation of observations, danger areas could be revealed. It may be discovered that it is advisable to go slowly as you pass doorways as often people walk out unrepentantly.

[24] Munoz V.F., Cruz, A. Ollero, A. Garcia-Cerezo, A. "Speed planning Method for Mobile Robots Under Motion Constraints" page 124

In a similar fashion to that presented in *Speed planning Method for Mobile Robots Under Motion Constraints* it would be possible to attach speed markers to the nodes or linked directions from the nodes. These speed constraints may be considered during a route planning stage to optimise a route or just avoid trouble areas.

7.2 Comparison to Completely Reactive System

A purely reactive system that implements a basic direction change in the presence of obstacles behaviour is presented in *On the sensor-based navigation by changing a direction to follow an encountered obstacle.*[25]

A simple yet effective strategy that as the name suggests, just decides to go either left or right around an encountered obstacle. A decision that is based on the previously taken action, 'This strategy has been adopted by cockroaches living in a natural environment.'[26]

This purely reactive strategy claims that in comparison to potential field methods, 'In the sensor-based navigation, a mobile robot always arrives at its goal.'[27] It may however spend much of its time involved in repetitive looping, consequently much of this work is involved in solving problems with perpetual loops. The task eventually being solved by alternate the choice in direction and not choosing random or constant directions.

> Especially in an uncertain 2D world with complicated shape, a mobile robot selects a very long path because the robot frequently enters into loops, i.e. senseless routes. In truth, passing through many loops is meaningless. [28]

Although the technique shown in this project is similar to the change in direction approach in that it registers the **LastDir** direction and move clockwise, it is not restricted to purely left and right decisions. Most importantly, the landmark positions are not required to be attached to obstacles, consequently the **Node-Map** maps the space around obstacles and not the obstacles themselves.

[25] Hiroshi Noborio, Takashi Yoshioka, and Shoji Tominaga "On the sensor-based navigation by changing a direction to follow an encountered obstacle" page 510
[26] Hiroshi Noborio et al. "On the sensor-based navigation by changing a direction to follow an encountered obstacle" page 510
[27] Hiroshi Noborio et al. "On the sensor-based navigation by changing a direction to follow an encountered obstacle" page 510
[28] Hiroshi Noborio et al. "On the sensor-based navigation by changing a direction to follow an encountered obstacle" Page 510

The exploration strategies presented in this project may result in sporadically high converge times taken to achieve the goal, this may occur given a complexly shaped environment. However the majority of the time is spent in developing meaningful data on the world, and not pointlessly looping around.

Sensor-based navigation describes landmarks as being either,

- A ‘turning decision point (T Point)’ where it was necessary to make a direction decision to avoid an encountered obstacle,
- Or a ‘leave point (L Point)’ where following an obstacle gave way to target acquisition.

These two structures are used as the basis of the navigation system, the information stored at each of the landmarks includes its position and the most promising future direction. These data sets are comparable to the data stored about nodes, with the exception that nodes are of only one type, and therefore convey no contextual information about the route they describe.

Sensor-based navigation is a similar solution to that of *change direction* and again the **Node-Map** implementation is similar. Considering that the **Node-Map** method will create nodes at regular intervals of space, the main identifiable difference between these strategies is the identification of space between obstacles.

T points are created when an obstacle is encountered, the best tangent either left or right is computed and taken.

> The agent starts to follow the concavity, and if it encounters again the previously generated landmark (which means that the taken direction was not correct) it updates the T-point landmark taking the other tangent direction [29]

This method uses a loop detection strategy, if a loop is found to be repeating as if at a local minima, a hill climbing approach is adopted to provide an exit back up the slope it previously descended on.

It is not necessarily sufficient to assess these navigation methods based on a one off run. Although initially this project may suffer from a higher convergence time than those

[29] Piaggio, M. Vercelli, G. Zaccaria R. “A Reactive Sensor-Based System for Solving Navigation Problems of an Autonomous Robot” page 241

compared here, its methods provide great flexibility for a variety of environments and a good modelling capacity; which is beneficial for future explorations.

The real merits of each system must be considered with respect to repeatability, once a space map has been established; navigable routes can be generated, thus avoiding repeatedly becoming trapped by the same obstacle.

7.3 Fuzzy Behaviour Comparison

Navigation of an Autonomous Mobile Robot by Co-ordination of Behaviours [30] presents a simple rule base that comprises of 50 rules formatted in two tables, one for angular velocity and one for linear speed. These two tables represent the control structure that implements a find the centre of collision free space behaviour. It also demonstrates some constraints of the robots turning abilities and allows for stopping once a target has been acquired.

This structure is very similar to that of this projects basic Obstacle Avoidance behaviour. Except that this project subdivides these tasks, using a table of rules to implement each simple consideration. Although this requires more processing and a more complex structure, it does allow for greater flexibility and once setup it allows for independent control of each individual consideration.

This last point is a presented as an advantage; but has its downside too. Changing one specific trait of a fuzzy system can have impacts that where not initially anticipated, whether the robot slows to turn or the turning is restricted at high speeds can have a major impact on the final behaviour.

Careful design with a balance between flexibility and simplicity should provide an optimal result, being both easy to implement and simple to maintain or adjust. The exact methods to achieve this will undoubtedly be specific to the implementation and nature of the task.

7.4 Concave Obstacle

The simple wall follow behaviour fails to escape concave obstacles, it is sufficient to follow around flat or convex shapes; but is prone to fall back into the depression if the walls curl back on themselves. Purely reactive systems often suffer from this trouble.

[30] Benreguieg, M. Maaref, H. Barret C. "Navigation of an Autonomous Mobile Robot by Coordination of Behaviours" page 69

This problem with concave obstacles has in, *Navigation of an Autonomous Mobile Robot by Co-ordination of Behaviours*, [31] been solved by taking a myopic or short-sighted approach. Short-sightedness in this case is not a negative aspect, it suggests that instead of considering all things at once, at all times, when a certain task is being performed this task has priority over other considerations.

By implementing a follow contour of obstacle behaviour, escape from convex obstacles occurs quickly and dynamically. This behaviour has been implemented with a neuro-fuzzy system.

The myopic approach provides a simple yet effective solution to the problem. This project cannot claim the same immediate success at escaping from concavities. Given a narrow obstacle, escape occurs quickly, however concavities enclosing large spaces have a much larger search space, hence result in long convergence times.

In *A Reactive Sensor-Based System for Solving Navigation Problems of an Autonomous Robot*, [32] a potential field method is employed to develop a repulsion from such concavities. This field enables a reverse route to be followed in order to exit the concave obstacle.

The exit strategy must first be authorised by higher level planner. The identification of a local minima is therefore the crucial task, without this recognition stage there would be no impetus to rectify the situation.

These techniques have not been considered in this project implementation, the robot escapes from concavities by systematically exploring the encompassed space. Eventually the exploration flows over the edge of the concavity and around the far side toward the goal.

7.5 Map Changes

Although a lot of attention has been paid to the map building, the map updating is almost completely ignored in navigational research.[33]

Detecting Changes in a Dynamic Environment for Updating its Maps by Using a Mobile Robot presents a similar solution to the implemented methods. It is claimed that the detection of obstacles introduced to the map and the detection of obstacles removed from the map have

[31] Benreguieg, M. “Navigation of an Autonomous Mobile Robot by Coordination of Behaviours” Fig.7 page 72
[32] Piaggio, M. et al. “A Reactive Sensor-Based System for Solving Navigation Problems of an Autonomous Robot” Fig. 5 page 241
[33] Zha, H. Tanaka, K. Hasegawa T. “Detecting Changes in a Dynamic Environment for Updating its Maps by Using a Mobile Robot” page 1729

been treated separately. The comparison between sensor data and the global map reveals newly included obstacles, whilst;

> Different from checking object moving-in, the object removal is found by examining disappearance of segments in G with respect to sensor data. This is because, in general, the removed object will make visible the segments it occludes in G, and hence the map segments can be traced more easily … The segments with no matching points in the sensor data are Labelled as one(sic) that have disappeared and thus indicate object removal.[34]

The same observations form the basis of map change detection in this project, there is however; little or no distinction made between the detection of objects being included and removed. The grid positions corresponding to current sensor data are adjusted to reflect the new set of distance readings, either grid locations are found to be occupied or empty, they are incrementally adjusted toward this state regardless of their current state.

7.6 Voronoi Techniques

The mathematics known as Voronoi techniques can be applied to segment a space by identifying all the points equidistant from a given set of positions, these points then divide the space into distinctly separate regions. *Implementing the GVG in the presence of sharp corners* [35] utilises these boundaries as a road map of the space.

In Reference to the incremental method of constructing a Generalise Voronoi Diagram GVD)

> The basic building block of the GVG is the set of points equidistant to two sets Ci and Cj, such that each point in this set is closer to the objects Ci and Cj than any other object [36]

[34] Zha, H. et al. "Detecting Changes in a Dynamic Environment for Updating its Maps by Using a Mobile Robot" page 1729

[35] Konukseven, I. Choset, H. "Mobile Robot Navigation: Implementing the GVG in the presence of sharp corners" page 1217

[36] Konukseven, I. "Mobile Robot Navigation: Implementing the GVG in the presence of sharp corners" page 1219

Both node map and Voronoi techniques are computationally efficient when compared to grid based approaches, 'Roadmaps are also concise in that they do not require that the entire environment be discretized into a fine resolution of pixels.' [37]

Applying the voronoi techniques to the generation of nodes in the node map could potentially arise at a new system of describing routes, the roadmaps described by voronoi techniques could provide a network of superhighways. These superhighway routes would have very good connectivity and directness through open space.

Identification markers attached to the nodes may be considered when planning routes and preference shown to the superhighway nodes, with the other links forming B-roads that actually guide the robot toward the exact location of the goal.

7.7 Path Planning

7.7.1 Grid Based

The grid map route planner based on the A* method, *Path Planner for a Car-Like Robot Applying Multi-Layer Grid Map of the environment*, [38] is a concise report covering clearly the implementation of route finding algorithm under holonomic constraints. It is based on a grid map representation and presents a multi-layer and shifted grid maps scheme.[39] This technique is too detailed to cover here but certainly worth drawing the reader's attention to.

The obvious demands on memory and computation associated with grid map schemes are somewhat offset by the comment made in reference to the compromise between optimality and demands,

The most interesting result is that despite of the large size of the graph the great reduction of computations can be achieved by increasing value of the coefficient q. However, found

[37] Konukseven, I. "Mobile Robot Navigation: Implementing the GVG in the presence of sharp corners" page 1218

[38] Kreczmer B. "Path Planner for a Car-Like Robot Applying Multi-Layer Grid Map of the environment" page 117

[39] Kreczmer B. "Path Planner for a Car-Like Robot Applying Multi-Layer Grid Map of the environment" page 121

solutions are suboptimal they are still reasonable and they are still very close to the optimal solution. [40]

Even with such optimisation and assuming low computational demands, the backtracking route-finder presented in this project offers a tidy solution at minimal expense, for a map with 50 nodes a maximum search would take 49 iterations. In contrast, to perform one pass of assessing a 10 by 10 grid map would require 100 iterations.

Both the methods require prior information about the environment, the work in *Path Planner for a Car-Like Robot Applying Multi-Layer Grid Map of the environment* does not however require the generation of a map of known positions. Shedding the overheads associated with generating a node-map may allow the implementation of a route finder on very simple systems, relying entirely on neural-network approaches or those without the advantage of the more advanced computational techniques.

7.7.2 Potential Field

A Probability-Based Approach to Model-Based Path Planning, [41] presents a potential field method for route planning. It produces many routes from the start to the goal point, the possible routes covering the extent of the map area.

The advantage in this technique is that it is robust and flexible, the potential field method allows a hill climbing type algorithm to track any near solution.

The backtracking route-finder may be altered, in order to aspire to this robustness. By allowing the algorithm to continue past the point of solution until every node has been assessed, all nodes will point toward the goal.

It would be necessary to attach markers to the nodes such as to provide an estimate reflecting the solution distance. The whole known map would then represent a complex path toward the goal, this robustness could then be exploited to achieve a similar effect to that presented in *A Probability-Based Approach To Model-Based Path Planning.*

Any position within sight of a node may be considered to be within the known map area, it follows that the potential field of this route finder covers the extents of the known map. In case of becoming lost or detached from the desired path, it would be possible to find any node and take up a new start point.

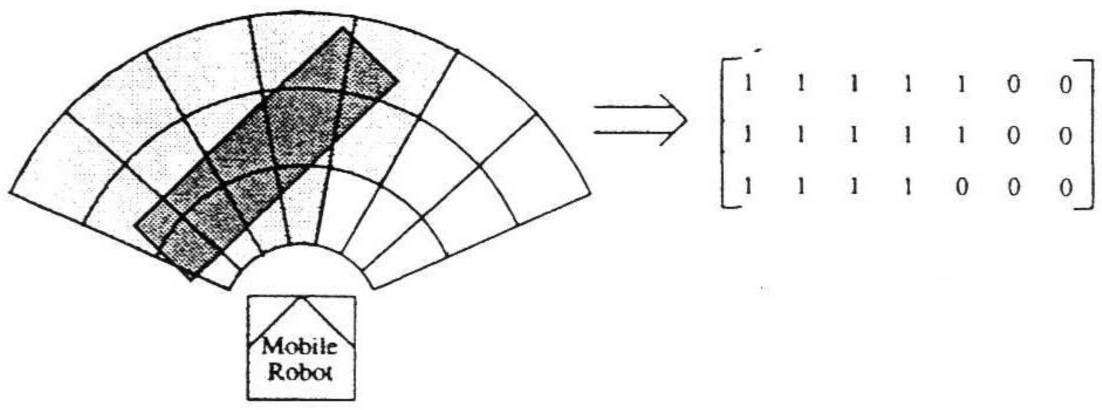
[40] Kreczmer B. “Path Planner for a Car-Like Robot Applying Multi-Layer Grid Map of the environment” page 121

[41] Mantegh, I. Jenkin M. R. M. Goldenberg A. A. A “Probability-Based Approach To Model-Based Path Planning” page 1189

As the position of the robot changed the route may become invalid or un-optimal, in these circumstances it would be required to rebuild the route from the new starting location. Such a route planning method should therefore be updated dynamically, hence the planning demands would increase significantly.

7.8 Neuro Vs fuzzy

This image shows the representation scheme used to interpret incoming sensor data for a neuro based reactive sensor setup.[42] It is worth noting that identification of occluded zones is achieved.



$$\Rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

[43]

Figure 7.8-1 Neuro Sensor Setup

The sensor region is sectioned into distinct zones providing a crisp occupancy grid, this grid then is ideal as the input data set to a neural network system.

In comparison this projects fuzzy implementation represents the same data as average distances within the given range of a direction. The distances are then categorised by fuzzy relations, consequently the final representation is very similar to that of this grid based scheme.

There is a certain lack of contextual information with this fuzzy distance and direction scheme, with the neural based system the occupancy grid is viewed as a whole, not as individual directions and distances. This weakness has been overcome by developing fuzzy

[42] Limón Marruedo, D. Gómez Ortega, J. "Neural Mobile Robot Navigation Based On A 2D Laser Range Sensor" page 315-320

[43] Limón Marruedo, D. Gómez Ortega, J. "Neural Mobile Robot Navigation Based On A 2D Laser Range Sensor" Fig. 5 page 317

rules that relate individual distance readings to others in order to restore the contextual information about the whole scene.

The grid method is specifically suitable to neuro systems because of its highly parallel nature. However the drawback to this representation scheme is that it is purely crisp logic, a segment is either occupied or not occupied. A fuzzy approach to this problem would imply far more about the scene, and such a method of segmentation for oncoming obstacles could seriously benefit this system.

Undoubtedly a compromise of the two methods would offer the best solution, drawing on the strengths of both techniques. This could be achieved as either a similar representation in fuzzy, or a neuro-fuzzy interface.

It is possible that the fuzzy methods may simplify the problem by reducing the requirement for neuro training.

8 Conclusions

8.1 The Need for Intuitive Techniques

Traditional implementations of robotic systems usually require both an expert with an in depth knowledge of the system being automated, and an expert in the techniques used to model and represent the system.

Consequently to adjust, modify or implement a robotic system either an expert in computers must learn about the system, or an expert in the system must learn about the techniques used to model such a system.

A real advantage is having the people that actually use the systems being able to configure the working parameters. This results in faster, more reliable implementations, fewer demands on one person and less requirement for outside expertise.

There is clearly then, the requirement of allowing domain experts access to the methods required to implement robotic systems, without the requirement for them to become computer experts or simulation mathematicians.

8.2 Fuzzy in General

It has been shown that fuzzy logic provides an intuitive and flexible method for implementing complex unmodelled systems. It has also been shown that in this way rapid prototyping becomes more successful.

One of the drawbacks to fuzzy systems is that because of its abstract nature it may often require a lot of tweaking before it functions correctly. It may also be extremely difficult to identify the cause of a problem, so it may not be possible to guarantee elimination even if the effects are sufficiently tempered.

Further than this it is difficult to prove the stability of fuzzy systems and therefore careful consideration during the design phase is essential.

The system presented in this project has a fully populated rule base, it therefore has no ungoverned regions of control. Though this strategy does not guarantee that the result will be how intended, it does ensure that the control region is boundless.

8.3 Neural Leaning

The ability to learn from, and adapt to, aspects of the environment is an essential ability for advanced autonomous systems.

Although the general skeleton structure can be intuitively described in a fuzzy rule base, the specific nature of the relations and relationships between them within the rule tables will remain approximated. These relations have a strong impact on how the system performs.

Trial and error methods of describing such relations produce good results that approach an optimal solution. It would however be greatly beneficial to allow the system to recognise and learn from its mistakes, generating precise relations developed from experienced failures and successes.

Within this utopia of learning, it must be recognised that the most difficult tasks are describing the evaluation functions and describing the nature of the tuning mechanism. These tasks then describe the requirements of a neural type learning system.

The benefits of such a system are clear and are similar to the benefits found with fuzzy systems, including rapid development and reliability.

8.4 Localisation

This project presents a system that has no method of localisation, it assumes perfect GPS type localisation. This is contrary to the original project specifications, which stated that direct localisation of this type should not be assumed. However for the completeness of other stages in the project, is has been necessary to bypass the implementation of a working localisation algorithm.

It is duly noted that as per the original specifications, localisation is essential for the implementation of a truly autonomous system. It is hoped that these algorithms will be implemented in future incarnations.

With this in mind, methods have been described that allow and assist the implementation of robot localisation. Consequently, the project should support such a system with minimal modification.

The route following algorithm moves the goal from one node to the next as they are reached, this provides the opportunity to re-localise properly at each stage.
A level of belief must be established and linked to the node creation algorithm, further techniques for updating the actual positions of the nodes should also take into account this localisation belief.

The map building and localisation techniques should be piers, feeding from each others successes and being aware of any failures.

8.5 Completeness

The field of autonomous robotic is as mentioned; extremely large, it encompasses many aspects of diverse fields, from AI to aeronautics.

There is a lot of work to be presented in each of the sections that are required to achieve autonomy, each of these fields present challenges that are both difficult and rewarding.

Having chosen to present a complete autonomous system means that each section is only grazed. Presented here is the groundwork or framework for a much larger system. It is anticipated that this work be continued, hence there is a strong basis for adding modules.

This is why then, that during the research phase of this project, it was noted that papers tend to present a narrow scope of work directed at the task; and assume all aspects that were out of this scope. In this way they present concise and detailed reports on just one topic.

It is hoped that now this project can accumulate methods and progressively answer the question of autonomous navigation.

8.6 Sensor Reliability

The input to this system is a single laser range finder, for more realistic implementations it would not necessarily be sufficient. Obstacles that are higher or lower than the scan height would not be seen, complex representations of the environment may be gained by a combination of many sensor types. Consideration should be given to the following,

- Tactile sensors for touch
- Torque sensing on the motors to identify stall conditions
- Infrared type sensors for close obstacles

The integration of more sensors aimed at providing a robust structure, rules to deal with extra types of information would be required to support these sensor systems. For example, an infrared emission will reflect strongly only when close to an obstacle, hence immediate action should then be taken.

8.7 Strength by Division

There are a lot of observations to be drawn from the world around us, many are applicable to robotics, this is where the majority of the inspiration for this project has been gained, from the natural world. One of the most difficult challenges is to interpret these observations and describe them within computer code.

Humans navigate with ease, particularly in well known areas. In order to achieve this much sensory information is draw upon along with a wealth of past experience.

Like humans, robots can benefit from a combination of many different sensors and reasoning techniques. Each technique has its own characteristic strengths and weaknesses, and the contribution from each expands the system as a whole.

The more resources available to a robotic system the stronger it becomes. This strength can be measured by the quality and complexity of the interaction between the robot and its environment.

This has been clearly demonstrated by the application of the simple navigation techniques presented in this project, each of the simple techniques supports and is supported by the others.

The past experiences of an autonomous system constitute much highly relevant and contextual information which should not be lost, by implementing adaptive techniques these experiences can be incorporated into the structure of the robot.

8.8 Grid Map Pre-Processing

Given that the width of the robot is explicitly stated, the smallest gap that the robot can physically fit through is also known. Any two objects placed closer to each other than the width of the robot will result in an impassable route.

To infer from this, the centre of the robot cannot get to less than half the robots width of any obstacle or wall. All environment features then could be expanded by half the robot width to produce a map of accessible areas, and not just of obstacles.

Originally this method had been considered as a major part of the map processing for obstacle avoidance, however care has been taken to ensure even the smallest detectable obstacles are treated as importantly as the larger ones. Therefore it is presented here as reinforcement to the majority of the map processing.

It would only reinforce the obstacle avoidance routines if the area between such close points were to be filled in as if there is a wall drawn between them. The node map techniques work well for corners and ends of walls, however are ineffective for ‘chair leg’ type obstacles.

8.9 Stack Route

More robust route finding and navigation behaviours could be achieved with the ability to represent complex, stacked routes. If the robot became distanced from the next node on a route, a new route may be created to guide the robot back to the original routes nearest node. This modification could mean that never would the robot be stuck behind a wall by accident, but dynamically re-plan a route to recover from such unexpected events.

The danger is that many routes could be formed, It may become more efficient to fail the current route and re-plan a new one from scratch. This is currently how the system recovers from this type of problem, the trouble is that it takes some time to recognise that something is wrong with the route being followed, and to reset the system. Un-interruptible behaviour is a measure of recovery from unpredicted situations.

8.10 Wall Following Characteristics

The navigational fuzzy techniques are supported well by the node map structure and route planning system, without the nodes map sub-targets the navigation techniques are fairly unproductive and prone to failure.

There is no wall following behaviour, quite often obstruction occurs where avoidable. The gradual encroachment of a wall onto the desired path may be interpreted as an unpassable obstacle. Wall following would eleviate this problem, hence it should be considered with a view toward implementation.

8.11 Robustness

It is essential that autonomous robots are able to recover from unpredictable situations. The frustration method used here is simple yet provides a reliable system for recognising when there is a navigation problem.

This and the ‘3 strikes and you are out’, method for identifying truly impossible tasks work well for continuously re-addressing problems.

To further utilise this on line information, it could influence the validity of links between nodes. If it is persistently difficult to navigate a link then the distance function of that link should be increased. Route-finding may then take this distance into account when evaluating a route.

8.12 Inference Pipework

The inference system is represented by a list of rules, each rule is associated with three relations; each of which having several membership functions. These rules and relations are stored in an arrays and referred to by their index number. The individual rules are chained together as separate blocks whose outputs form the inputs for other rules. Consequently it is very easy to lose track of the whole network.

Defining linguistic names for these relations may help to keep tracks of the state of the network. Ultimately a different system, preferably graphical is required to assist the developer in ‘re-wiring’ the inference engine.

8.13 Suitability

The successes of exploration and navigation within unmodified indoor environments presented here show encouraging and promising results within the limitations shown.

The suggested modifications are required to complete the work, however assuming they have the same levels of success then an interesting and useable template for navigation should develop.

9 Future Work

9.1 Saving of Maps

Currently the starting environment is hard-wired, a map of all obstacles is stored in the code. In order to change the default map it is necessary to edit the code and completely recompile the source. The default map can be edited online, but these changes will be lost once the program stops.

As any experimentation performed is only as valid as its repeatability, the loss of map data is clearly not convenient, to allow more complex tests and experiments to be set up, the functionality to save hand drawn maps is required.

Consideration should be given to saving both environment maps and node maps, 'pickling' the system state part way through a run. This would enable more adequate testing of individual behaviours and traits. Comparisons to other works would then be far more relevant.

9.2 Node Map Maintenance

9.2.1 De-Noding

The number of nodes and ensuing Node-Map complexity are directly related to many time based factors. The planning of routes, finding of difficult targets and updating the map in response to changes in the environment, all take longer the more nodes that exist.

A method for identifying unnecessary nodes and simplifying the node-map by pruning may be included to reduce the final number of nodes on a map. This would then reduce the real time processing requirements.

Consideration must be given however, to detrimental effects on the accuracy and accessibility of the maps.

9.2.2 Repositioning

Currently, although node connections can be broken and remade, the actual positions of the nodes cannot be altered after first creation.

As there is an accuracy limit to the localisation algorithm, when addressing this problem it should be considered that nodes may not be exactly where they indicate they are. This is dependant on the accuracy of localisation at the times of adjustment, the methods must therefore be closely linked to the localisation algorithm’s belief level.

It is anticipated that given time, unreachable nodes would be virtually eliminated and for the map to dynamically change under the influence of the environment and environmental considerations.

9.3 Forward Projection for Localisation and Obstacle Avoidance

Given a known route through a known map, the previously stored obstacle data can be used to ‘predict’ the surroundings that will be found further along the planned route.

This advanced information can be gained by first projecting the robots future position along the route, and then performing a simulated laser scan from this predicted position using the stored obstacle data to generate the distance measures.

Combining this data with that of the local fuzzy map may allow accurate and robust identification of position.

9.3.1 Leapfrogging

Given that forward projection is a reliable technique, a forward goal can be established in the near vicinity along the route and the robot can be guided to this goal by the standard techniques.

Once this position is obtained a new goal can be found, by this means the robot is never in an unknown position, the challenge of localisation should be reduced.

Problems are envisaged when large changes in the map occur, this method may strongly suggest the position being incorrect, not identifying that the map has altered. However if the position is well known and strongly believed, then it is possible that large changes in the map be recognised immediately.

9.4 Potential Field

By marking areas where confusion occurs, it may be possible for subsequent route plans and dynamic navigation to prevent the same situation arising again.

This could be implemented on either a grid or vector based map, either way, the accumulation of results from previous visits to all areas, should form regions of exclusion. Problem areas such as scattered obstacles, concave surfaces and narrow hallways should all collect occurrences and hence be avoided in future.

This behaviour must be tempered by the fact that it must not be allowed to prevent access to any area of the map or restrict the traversability of any route.

9.5 Feature Extraction

It is anticipated that extra data attached to each of the position nodes would help in verification of the robots location. Landmarks like edges and corners can all be utilised to assist accurate position estimation.

By attaching a full panoramic scan data set to each node, the area can be charted as known features and not just as obstacle positions. This type of environmental data may be used with advanced localisation techniques like triangulation.

Corners and edges are usually indicative of rapid changes between distance measures, appropriate methods may provide strong and precise identification of these features. Given careful consideration it should be possible to interpret these local features as a map of landmarks.

The extrapolation of landmarks from sensor information is an interesting challenge. The mathematics behind such interpretation is likely to involve techniques such as correlation.

Given a set of landmarks with known characteristics and positions on a map, the change of orientation to these landmarks in relation to the robots movement may provide accurate identification of position. Such methods are the topic for much future research

9.6 Map Representation Scheme

The 'parti-game multi-resolution approach' presented in *Sensor-Based Learning of Environment Model and Path Planning with a Nomad 200 Mobile Robot,* [44] Shows a tree type structure, the representation of this is similar to fractal based compression algorithms.

The potential exists to represent a map in a grid based form but to compress and store the data with standard graphics compression algorithms. The immediate advantage is reducing demand for storage memory, the main disadvantage may be the requirement for real time decompression.

[44] Araújo, R. Almedia, A. "Sensor-Based Learning of Environment Model and Path Planning with a Nomad 200 Mobile Robot" page 539-544

9.7 Holonomic Adaptation

The path planning methods should be modified to accept functions for holonomic constraint considerations, as mentioned previously, many methods exist for the conversion of any path into one suitable for a robot with holonomic constraints. These techniques may be applied to the existing route planning and navigation techniques to provide suitable paths for any type of robot.

Bibliography

Aguirre, E. García-Alegre, M. "A FUZZY SAFE FOLLOW WALL BEHAVIOUR FUSING SIMPLER FUZZY BEHAVIOURS" page 87-92 *IFAC Intelligent Autonomous Vehicles* (1998) Madrid, Spain

Amar, F. B. "Steering behaviour and control of fast wheeled robots" page1396-1401 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Angel, E. *OpenGL: A Primer* (2002) USA: Addison-Wesley

Araújo, R. Almedia, A. "Sensor-Based Learning of Environment Model and Path Planning with a Nomad 200 Mobile Robot" P539-544 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Benreguieg, M. Maaref, H. Barret C. "Fusion of Fuzzy Agents for the Reactive Navigation of a Mobile Robot" page 388 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Benreguieg, M. Maaref, H. Barret C. "Navigation of an Autonomous Mobile Robot by Coordination of Behaviours" P69-74 *IFAC Intelligent Autonomous Vehicles* Madrid Spain (1998) Pergamon

Castellanos, J. A. Martínez, M. Neira, J. Tardós, J. D. "EXPERIMENTS IN MULTISENSOR MOBILE ROBOT LOCALIZATION AND MAP BUILDING" page 369-374 *IFAC* Madrid, Spain (1998) Pergamon

Chen,C. Pham, T.T. "Introduction to Fuzzy Sets, Fuzzy Logic and Fuzzy Control Systems", page 66 (2001) USA:CRC Press LLC

Chohra, A. Farah, A. Belloucif, M. "Neuro-Fuzzy Expert System E_S_CO_V for the Obstacle Avoidance of Intelligent Autonomous Vehicles(IAV)" page 1706-1713 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Chong, K. S. Kleeman, L. "Indoor Exploration Using a Sonar Sensor Array: A Dual Representation Strategy" page 676-682 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Cires, J. Bertrand, F. Zufiria, J. P. "TWO MODELING APPROACHES FOR NAVIGATION CONTROL OF A NOMAD 200 MOBILE ROBOT" page 363-368 *IFAC Intelligent Autonomous Vehicles* Madrid Spain (1998) Pergamon

Cox, E. D. *Fuzzy Logic for Buisness and Industry* (1995) USA: Charles River Media, Inc.

Driankov, D. Saffiotti, A. Ed. “Fuzzy Logic Techniques for Autonomous Vehicle Navigation” page 266-271 (2001) New York: Physica-Verlag Heidelberg

Einsele, T. “Real-Time Self-Localization in Unknown Indoor Environments using Panorama Laser Range Finde” page 6 IEEE International Conference on Systems, Man and Cybernetics (SMC) (1997) IEEE Press

Fernández, R. Mandow, A. Muñoz, V.F. García-Cerezo, A. “REAL-TIME MOTION CONTROL FOR SAFE NAVIGATION” page 303-308 *IFAC Intelligent Autonomous Vehicles* Madrid Spain (1998) Pergamon

Fox, D. Burgard, W. Thrun, S. “Markov Localization for Reliable Robot Navigation and People Detection” page 3 *Sensor Based Intelligent Robots* (1999*) Berlin:Springer-Verlag Heidelberg*

Garcia-Algre, M. C. Ribeiro, A. Cañas, J. M. “A CARTOGRAPHER ROBOT: MERGING AND INTERPRETING MAPS IN UNKNOWN ENVIRONMENTS” page 405-409 *IFAC Intelligent Autonomous Vehicles* Madrid Spain (1998) Pergamon

Gökkus, L. Erkmen, A. M. Tekinalp, O. “INTERACTING FUZZY MULTIMODEL INTELLIGENT TRACKING SYSTEM FOR SWIFT TARGET MANOEUVRES” page 766-771 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Hiroshi Noborio, Takashi Yoshioka, and Shoji Tominaga “On the sensor-based navigation by changing a direction to follow an encountered obstacle” P510 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Hu, H. Gu, D. “Concurrent Navigation and Map Building for Mobile Robots” page 207-212 *Proceedings of the Annual Chinese Automation Conference in the UK* Sept.19-20th Leicester, England (1998)

Kandel, A. Langhotz, G. *Fuzzy Control Systems* (1994) USA: CRC Press, Inc.

Karch, O. Noltemeier, H. “Robot Localization – Theory and Practice” page 850 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Khatib, M. Siméon, T. “Sensor-Based Motion Planning and Control for the HILARE mobile robot” V8-V9 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Kimoto, K. Thorpe, C. “Map Building with Radar and Motion Sensors for Automated Highway Vehicle Navigation” page 1721-1728 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Konukseven, I. Choset, H. “Mobile Robot Navigation: Implementing the GVG in the presence of sharp corners” page 1217 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Kreczmer B. “Path Planner for a Car-Like Robot Applying Multi-Layer Grid Map of the environment” page 117 *IFAC Intelligent Autonomous Vehicles* Madrid Spain (1998) Pergamon

Limon Marruedo, D. Gomez Ortega J. “Neural Mobile Robot Navigation Based On A 2D Laser Range Sensor” *IFAC* Intelligent Autonomous Vehicles” P319 *IFAC Intelligent Autonomous Vehicles* Madrid Spain (1998) Pergamon

Liu, Z. White, B. Hu, H. “Trajectory Planning and Real-time Control of Soccer Robots” page 77-82 *Proceedings of the Annual Chinese Automation Conference in the UK* Sept. 24-25th Derby, England (1999)

Mandow, A. Munoz, V.F. Ferandez, R. Garcia-Cerezo, A. “Dynamic Speed Planning for Safe Navigation” P235 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Mantegh, I. Jenkin M. R. M. Goldenberg A. A. A “Probability-Based Approach To Model-Based Path Planning” P1189 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Miyake, N. Aono, T. “Position Estimation and Path Control of an Autonomous Land Vehicle” page 690-696 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Munoz V.F., Cruz, A. Ollero, A. Garcia-Cerezo, A. “Speed planning Method for Mobile Robots Under Motion Constraints” P124 *IFAC Intelligent Autonomous Vehicles* Madrid Spain (1998) Pergamon

Paromtchik, I. E. Laugier, C. "Automatic Parallel Parking and Return to Traffic Maneuvers" V-21-V-23 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Piaggio, M. Vercelli, G. Zaccaria R. “A Reactive Sensor-Based System for Solving Navigation Problems of an Autonomous Robot” P241 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Puttkamer, E. Weiss, G. Edlinger, T. “Localization and On-Line Map Building for an Autonomous Mobile Robot” page 37-48 *Sensor Based Intelligent Robots (1999) Berlin:Springer-Verlag Heidelberg*

Qiu, J. “Task Sequencing for Mobile Robot Navigation” P180 *Proceedings of the Annual Chinese Automation Conference in the UK* 19-20th Sept. Leicester, England (1998)

Qui, J. Walters, M. “Learning of Membership Functions of Fuzzy Behaviours for a Mobile Robot Control System” page 772-777 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Rencken, W. D. Feiten, W. Zöllner, R. “Relocalisation by partial Map Matching” page 21-35 *Sensor Based intelligent Robots* (1999) Springer-Verlag Berlin Heidelberg

Russell, S. Norvig, P. “Artificial Intelligence a Modern Approach” P96-101 (1995) Prentice Hall, Inc.

Scheuer, A. Fraichard, Th. “Continuous-Curvature Path Planning for Car-Like Vehicles” page 997 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble*, *France* (1997) IEEE Press

Siméon, T. Leroy, S. Laumond, J.P. “Computing good holonomic collision-free paths to steer nonholonomic robots” page 1004-1009 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble, France* (1997) IEEE Press

Visual C++ 5.0 MSDN help

Wright, R. S. Sweet, M. *OpenGL SuperBible* 2nd Edition (2002) USA: Waite Group Press

Zha, H. Tanaka, K. Hasegawa T. “Detecting Changes in a Dynamic Environment for Updating its Maps by Using a Mobile Robot” P1729 *International Conference on Intelligent Robots and Systems (IROS97) Sept 7-11th Grenoble*, *France* (1997) IEEE Press

Zurada, J. Wright, A. Graham, H. “A Neuro-Fuzzy Approach for Robot System Safety” page 49-64 *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART C:APPLICATIONS AND REVIEWS, VOL.31,NO.1* February (2002) IEEE Press