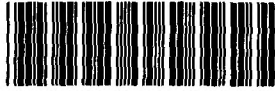


BOOK NO: 1839070



Bound by
Abbey
Bookbinding Co.
105 Cathays Terrace, Cardiff CF24 4HU
South Wales, U.K. Tel: (029) 2039 5882
www.bookbindersuk.com



**NOT TO BE
TAKEN AWAY**

Hybrid control architecture for navigation of autonomous mobile robots

A Thesis Submitted to the University of Wales for the Degree of

Doctor of Philosophy

By

Alexandros Mouzakis,
Mechatronics Research Centre
University of Wales College, Newport
August 2002

I dedicate this Thesis
to my father Elissaios and my mother Zaira,
whose love and support
I could not do without



Declarations

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed  (candidate)

Date 14 August 2002

STATEMENT 1

This thesis is a result of my own investigations, except where otherwise is stated.

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed  (candidate)

Date 14 August 2002

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed  (candidate)

Date 14 August 2002

Acknowledgements

I would like to express my sincere thanks to my project supervisor, Professor Geoff Roberts for his invaluable advice, guidance, motivation and most of all for his support throughout the course of this research.

Special thanks are due to Mandi Stewart for all her support and help throughout my research, without whom none of this would be possible. Thank you for your love, understanding and patience during this tense period.

Thanks are due to all my colleagues in the Mechatronics Research Centre, especially I would like to thank Dr Ioannis Akkizidis, Dr Anastasios Diamantopoulos, Mr Edin Omerdic, Dr Antonio Zirilli, Dr Liren Wang and Mrs Fangmin Shi for their advice, kind help, support and friendship. I would also like to thank the secretary of the Mechatronics Research Centre Mrs Jane John for her help with my many queries during the course of this research.

I express my gratitude to the academic staff of the School of Computing & Engineering of the University of Wales College, Newport for their help, support and advice in the research component of my project, especially I would like to thank Dr Neil Rothwell Hughes and Mr Eric Llewellyn.

I would like to thank the staff of the Allt-yr-yn Campus library, who were always happy to help me find all the papers and books I needed during my research. I would especially like to thank Ms Dawne Leatherdale, Mrs Bronwen Stone and Ms Jenifer Coleman for their kind help and support.

Thanks are due to the University of Wales College, Newport for the bursary that was granted to me during the three years of my research. Also I would like to thank Pirelli Communication Cables Ltd in Newport for the kind provision of financial support from the beginning until the end of my research. I would especially like to thank Mr Tim Moss, Mr Glyn Sutton, Mr Dave Jones and Mr Dave Nichols.

I would like to thank Merlin Systems Corporation Ltd and the University of Plymouth for all the help and support regarding hardware and software problems with the MIABOT V2 mobile robots. I would especially like to thank Dr Mark Norman, Dr Guido Bugmann and Mr Paul Robinson.

Last but certainly not least special thanks are due to my family, my mother Zaira, my father Elissaios, my sister Veta, and my brothers Spiros and Stamatis for their endless support, love and encouragement during all my studies.

Summary

This thesis is concerned with the development, design and implementation of a novel hybrid multi-agent orientated control architecture for navigation of multiple autonomous mobile robots operating in an unknown and unstructured environment populated by static and/or dynamic obstacles.

The proposed hybrid control architecture is modular and draws its design from competitive tasks architecture, production rules architecture, connectionist architecture, dynamic system architecture, multi-agent architecture and subsumption architecture. The reasoning of the control architecture is both deliberative and reactive. The proposed reactive behaviours are modelled using fuzzy logic, neural networks and hybrid behavioural encoding incorporating stateflow-fuzzy logic and stateflow-neural networks. The deliberative system is comprised of finite state machines. The processing is achieved in a centralised and/or decentralised manner using the proposed controller-agent concept from the field of multi-agent systems. The framework of the control architecture is suitable for adaptation in single and multiple robot navigation.

The control architecture has been implemented in MATLAB/Simulink using the full non-linear model of the MIABOT V2 mobile robots. It is evaluated incrementally in order to verify its overall control performance and the performance of each subsystem. Results show that the control architecture's modularity, distribution, reactivity and behaviour-based structure provided the overall control system with robustness in all cases of navigation tasks utilising either single or multiple mobile robots. Furthermore the results obtained show the effectiveness of the control architecture in navigation tasks involving up to five mobile robots operating in unknown static and dynamic environments. The results demonstrate that the control strategy chosen for navigation of multiple mobile robots is efficient and also established the robustness of the control system architecture against the desired requirements, such as supervision, decision-making and co-ordination of internal control structures (subsystems). The autonomous mobile robots were exposed to a complex and highly dynamic environment and successfully achieved every control objective. Their trajectories were smooth despite the interaction between several behaviours and the presence of unexpected static and dynamic obstacles.

The main contributions of this thesis are: development of a novel hybrid multi-agent based control architecture called CAROS; novel approach for identification of direction of moving obstacles (other robots) using finite state machines; novel approach for behavioural encoding using hybrid solutions such as stateflow-fuzzy and stateflow-neural for autonomous robot navigation; proposed a design methodology for developing integrated solutions for autonomous mobile robotic systems and classification of the main design methodology (properties) of control systems architectures for autonomous mobile robots. Less significance contributions are: literature survey on approaches/methods related to the development of intelligent control architectures for navigation of multiple autonomous mobile robots; modelling of MIABOT V2 mobile robots; comparison between PI, fuzzy and neural controllers and algorithmic methodology for discovery of fuzzy/neural local models from observation data; identification of the relationship of the most important requirements/properties of control architecture versus the main control architecture specifications using the Quality Function Deployment tool; modular approach for modelling and evaluation of three types of sensor and sensor sensitivity.

Table of Contents

Declarations	i
Acknowledgements	ii
Summary	iii
Table of Contents	iv
List of Figures	xiv
List of Tables	xxii
Notations and Symbols	xxv
Abbreviations	xxix

CHAPTER 1. Introduction, Motivation and Overview of the Thesis.....	1-1
1.1 Introduction.....	1-1
1.2 Motivation of the thesis.....	1-2
1.3 Application areas	1-4
1.4 Aim and objectives of the research	1-5
1.5 Challenges and problems in mobile robot navigation.....	1-6
1.6 Overview of the thesis.....	1-8
1.7 The main contributions of the thesis	1-11
1.8 Summary.....	1-13
1.9 References.....	1-13

CHAPTER 2. Literature Review of Related Work.....	2-1
2.1 Introduction.....	2-1
2.2 Artificial life	2-2
2.3 Co-operative robots.....	2-3
2.4 Control of multiple robots (multi-agent robotics)	2-4
2.5 Behaviour-based control	2-5
2.6 Distributed artificial intelligence and distributed systems.....	2-9
2.7 Intelligent control and its approaches.....	2-12
2.7.1 Modelling, identification and control of dynamic systems	2-14
2.8 Mobile robot navigation using methodologies from intelligent control.....	2-17
2.8.1 Mobile robot navigation using fuzzy logic	2-18
2.8.1.1 The 1990s: The peak of fuzzy logic in mobile robot navigation	2-19
2.8.1.2 Early 2000: Fuzzy logic in mobile robot navigation continues to grow	2-22
2.8.2 Mobile robot navigation using neural networks.....	2-24
2.8.2.1 The 1990s: The beginning of neural networks in mobile robot navigation.....	2-25
2.8.2.2 Early 2000: More challenges remain.....	2-27
2.9 Discussion.....	2-29
2.10 Summary.....	2-31
2.11 References.....	2-32
 CHAPTER 3. Research Methodology	 3-1
3.1 Introduction.....	3-1
3.2 Fuzzy logic systems (FLS).....	3-6
3.2.1 What is fuzzy logic	3-8
3.2.2 Structure of fuzzy logic	3-8
3.2.2.1 Fuzzy sets.....	3-8

3.2.2.2 Universe	3-9
3.2.2.3 Membership functions.....	3-9
3.2.2.4 Fuzzy sets operations	3-11
3.2.3 Design of fuzzy logic controller	3-13
3.2.3.1 Pre-processing.....	3-13
3.2.3.2 Fuzzification	3-14
3.2.3.3 Rule base.....	3-14
3.2.3.4 Inference engine.....	3-15
3.2.3.5 Defuzzification.....	3-15
3.2.3.6 Post-processing	3-16
3.2.4 Types of fuzzy models.....	3-16
3.2.4.1 Mamdani model	3-17
3.2.4.2 Takagi-Sugeno model	3-19
3.3 Artificial neural networks (ANN)	3-20
3.3.1 Biological neuron	3-20
3.3.2 Model of artificial neuron.....	3-21
3.3.2.1 Types of activation function.....	3-23
3.3.3 ANN architectures.....	3-25
3.3.3.1 Feedforward neural networks (FNN)	3-26
3.3.3.2 Recurrent neural networks (RNN).....	3-28
3.3.4 Training of feedforward multilayer perceptron NN	3-29
3.4 Clustering.....	3-30
3.4.1 Definitions and notations in cluster analysis.....	3-31
3.4.1.1 The data used in cluster analysis	3-31
3.4.1.2 Definition of clusters.....	3-32
3.4.2 Clustering algorithms (off-line).....	3-33
3.4.2.1 Fuzzy C-means clustering	3-33

3.4.2.2 Mountain clustering method.....	3-34
3.4.2.3 Subtractive clustering.....	3-37
3.5 Design of control systems architectures for autonomous mobile robots.....	3-39
3.5.1 Definitions used in design of control system architectures for autonomous mobile robots.....	3-40
3.5.2 Classification of control architectures.....	3-41
3.5.2.1 Deliberative architectures.....	3-41
3.5.2.2 Reactive architectures	3-42
3.5.2.3 Hybrid architectures	3-42
3.5.3 Main key issues of the design of control systems architectures for multiple autonomous mobile robots.....	3-43
3.5.3.1 Centralised, decentralised or hybrid control.....	3-43
3.5.3.2 Heterogeneous or homogeneous robots.....	3-44
3.5.3.3 Co-operation with or without communication.....	3-44
3.5.3.4 Making robots that work as a team.....	3-45
3.5.3.5 Multiple robots path planning	3-45
3.5.3.6 Learning	3-45
3.5.4 Properties of control architectures	3-46
3.6 Multi-agent systems (MAS) in control engineering	3-48
3.6.1 Autonomous agents	3-49
3.6.2 Multi-agent systems.....	3-50
3.6.3 Agent control architectures.....	3-51
3.7 Stateflow design tool based on finite state machines theory	3-52
3.7.1 Finite state machines (FSM).....	3-52
3.7.2 Stateflow	3-53
3.8 Discussion.....	3-55
3.9 Summary.....	3-56
3.10 References.....	3-57

CHAPTER 4. Modelling of Miabot V2 Mobile Robot	4-1
4.1 Introduction.....	4-1
4.2 Main features of MIABOT V2 mobile robot.....	4-4
4.2.1 Main components	4-5
4.2.2 Drive train	4-5
4.2.3 DC motors	4-6
4.3 Establishing the kinematic model	4-6
4.4 Establishing the dynamic model	4-9
4.5 Linearised model.....	4-18
4.6 Results	4-21
4.7 Discussion.....	4-23
4.8 Summary.....	4-24
4.9 References.....	4-25
CHAPTER 5. Control, Robust Stability Analysis And Discovery Of Fuzzy/Neural Local Models	5-1
5.1 Introduction.....	5-1
5.2 Controlling the mobile robot.....	5-4
5.2.1 Selection of speed controller	5-4
5.2.2 Design requirements.....	5-5
5.3 PI controller	5-6
5.3.1 Controller tuning	5-7
5.3.2 Results of tuning.....	5-9
5.4 Fuzzy controller using PI controller as a teacher.....	5-11
5.4.1 Selection of input-output data.....	5-12

5.4.2	Subtractive clustering to define the number of rules.....	5-13
5.4.3	Fuzzy controller testing and verification	5-14
5.5	Neural controller using PI controller as a teacher.....	5-16
5.5.1	Training the neural controller	5-16
5.5.2	Neural controller testing and verification	5-22
5.6	Comparison between the different controllers.....	5-23
5.7	Robust stability testing of MIABOT V2 closed-loop.....	5-27
5.7.1	The application	5-27
5.7.2	Verification of the closed-loop robust stability.....	5-30
5.8	Discussion.....	5-32
5.9	Summary.....	5-35

CHAPTER 6. Hybrid Control Architecture For Navigation Of Multiple Autonomous

Mobile Robots.....	6-1
6.1 Introduction.....	6-1
6.2 Why hybrid architecture?.....	6-2
6.3 Overview of CAROS	6-4
6.3.1 Sensory data	6-6
6.3.2 Action co-ordinator mechanism (ACM).....	6-6
6.3.3 Decision-making mechanism (DMM)	6-6
6.3.4 Speed and orientation adaptive mechanism (SOAM)	6-8
6.3.5 Mobile robot and environment	6-8
6.4 Control strategy.....	6-9
6.4.1 Global control strategy	6-9
6.4.2 Local control strategy	6-11
6.5 Design and implementation methodology.....	6-12

6.6	Low-level control.....	6-12
6.7	Modelling sensors and sensor sensitivity	6-13
6.7.1	Infrared sensors	6-15
6.7.2	Ultrasonic sensors.....	6-15
6.7.3	No-shape sensors	6-16
6.8	Robot localisation	6-17
6.9	Analysis of SOAM.....	6-18
6.10	Organising SOAM using local controllers by means of agents	6-18
6.10.1	Co-ordination of local controller-agents using stateflow	6-26
6.11	Modelling the robot(s) behaviours	6-28
6.12	Analysis of <i>go_to</i> robot behaviour	6-29
6.13	Analysis of <i>avoid_static</i> robot behaviour	6-33
6.13.1	Static obstacle avoidance using fuzzy controller	6-35
6.13.2	Static obstacle avoidance using neural controller	6-39
6.14	Analysis of <i>avoid_dynamic</i> robot behaviour	6-43
6.14.1	Identification of direction of neighbour robot (moving object).....	6-45
6.14.2	Dynamic obstacle avoidance using hybrid stateflow-fuzzy controller	6-49
6.14.3	Dynamic obstacle avoidance using hybrid stateflow-neural controller	6-54
6.15	Analysis of <i>avoid_trap</i> robot behaviour.....	6-58
6.16	Co-ordination and parallel switching of robot behaviours using stateflow.....	6-60
6.17	Discussion.....	6-65
6.18	Summary.....	6-68
6.19	References.....	6-70
CHAPTER 7. Evaluation Of The Control Architecture.....		7-1
7.1	Introduction	7-1
7.2	Simulation setup	7-2

7.1	Introduction	7-1
7.2	Simulation setup	7-2
7.3	Division of results	7-2
7.3.1	Part I: Results	7-4
7.3.1.1	Group A1: Results.....	7-4
7.3.1.2	Group A2: Results.....	7-7
7.3.1.3	Group A3: Results.....	7-8
7.3.1.4	Group A4: Results.....	7-10
7.3.1.5	Part I: Outcome of results.....	7-14
7.3.2	Part II: Results	7-26
7.3.2.1	Group B1: Results.....	7-26
7.3.2.2	Group B2: Results.....	7-28
7.3.2.3	Part II: Outcome of results	7-32
7.3.3	Part III: Results.....	7-40
7.3.3.1	Group C1: Results.....	7-41
7.3.3.2	Group C2: Results.....	7-42
7.3.3.3	Group C3: Results.....	7-44
7.3.3.4	Group C4: Results.....	7-45
7.3.3.5	Group C5: Results.....	7-46
7.3.3.6	Group C6: Results.....	7-47
7.3.3.7	Part III: Outcome of results.....	7-48
7.4	Discussion.....	7-56
7.5	Summary.....	7-57

CHAPTER 8. Review, Conclusions And Future Work8-1

8.1	Introduction.....	8-1
8.2	Review of the thesis.....	8-2
8.3	Discussion and conclusions of the research	8-10
8.4	The main contributions of the thesis	8-17
8.5	Suggestions for future work	8-22

APPENDIX A. MIABOT V2 Mobile Robot A-1

A.1	Introduction.....	A-1
A.2	Main features	A-1
A.3	Drive train.....	A-2
A.4	DC motors.....	A-3
A.5	MIABOT control board	A-4
A.6	The software	A-6
A.7	PC transmitter	A-6
A.8	References.....	A-8

APPENDIX B. SIMULINK Block Diagrams B-1

B.1	Introduction.....	B-1
-----	-------------------	-----

APPENDIX C. Conventional Control Design..... C-1

C.1	The basic control system structure	C-1
C.1.1	Description of the input and output	C-1
C.2	Linear and non-linear control systems	C-2
C.3	PID Control.....	C-3
C.4	References.....	C-6

APPENDIX D. Constrained Optimisation Using The Non-Linear Control Design Tool. D-1

D.1 Introduction.....	D-1
D.2 Overview	D-1
D.3 Constrained optimisation	D-2
D.4 Sequential Quadratic Programming (SQP).....	D-3
D.5 Solving the optimisation problem using the non-linear control design tool	D-4
D.6 References.....	D-6

APPENDIX E. Robust Stability Analysis For Interval Polynomialsbased On Parametric**Approach..... E-1**

E.1 Introduction.....	E-1
E.2 Description of uncertain structure	E-1
E.3 Value sets and zero exclusion condition	E-3
E.4 Kharitonov's theorem	E-4
E.5 Robust stability testing via graphics.....	E-6
E.6 References.....	E-7

APPENDIX F. Training Algorithms F-1

F.1 Introduction.....	F-1
F.1.1 Error definitions	F-1
F.1.2 Backpropagation to update the parameters	F-3
F.1.3 The Levenberg-Marquardt algorithm.....	F-4
F.1.4 References	F-5

APPENDIX G. Published Work..... G-1

G.1 Introduction.....	G-1
-----------------------	-----

List of Figures

Chapter 1

Figure 1-1 Schematic outline of the thesis	1-11
--	------

Chapter 2

Figure 2-1 Traditional functional decomposition.....	2-6
Figure 2-2 Behaviour-based decomposition	2-7
Figure 2-3 Origin and sub-fields of distributed artificial intelligence	2-10
Figure 2-4 Techniques employed in intelligent control (after (Harris, 1994))	2-13
Figure 2-5 Methodologies used in intelligent control (after (Harris, 1994))	2-14

Chapter 3

Figure 3-1 The sets <i>more less old</i> , <i>very young</i> , and <i>not very young</i> are derived from <i>young</i> and <i>old</i>	3-9
Figure 3-2 Examples of membership functions: (a) triangular-shaped (b) trapezoidal-shaped.....	3-10
Figure 3-3 Structure of fuzzy controller	3-13
Figure 3-4 Biological neuron.....	3-21
Figure 3-5 Model of a neuron	3-22
Figure 3-6 Common activation functions	3-23
Figure 3-7 Feedforward neural network	3-27
Figure 3-8 Recurrent neural network.....	3-28
Figure 3-9 Clustering based on distance: (a) Interpretable by human, (b) ambiguous to human.....	3-31

Figure 3-10 An agent as entity (physical or virtual) that senses, thinks and acts in some environment in order to achieve its goal.....	3-50
---	------

Figure 3-11 An example of 2-state stateflow diagram based on FSM theory	3-54
---	------

Chapter 4

Figure 4-1 MIABOT V2 mobile robots	4-2
--	-----

Figure 4-2 An exploded view of the mobile robot	4-5
---	-----

Figure 4-3 Schematic layout of the mobile robot.....	4-8
--	-----

Figure 4-4 Approximation of slippage factor due to robot linear acceleration.....	4-13
---	------

Figure 4-5 Approximation of slippage factor due to robot angular acceleration	4-13
---	------

Figure 4-6 Real-time experiments to obtain and verify the accuracy of the open-loop model.....	4-16
--	------

Figure 4-7 Robot's input/output open-loop dynamic model in Simulink.....	4-17
--	------

Figure 4-8 Submodel of robot's input/output open-loop dynamic model	4-17
---	------

Figure 4-9 Comparison of real and simulated robot trajectory (RMSE=1.51)	4-22
--	------

Figure 4-10 Comparison of non-linear and linear response for the left wheel linear velocity under perturbations of control inputs (RMSE=0.05).....	4-22
--	------

Figure 4-11 Comparison of non-linear and linear response for the right wheel linear velocity under perturbations of control inputs (RMSE=0.03).....	4-23
---	------

Chapter 5

Figure 5-1 Closed-loop robot model with speed controller.....	5-5
---	-----

Figure 5-2 Proportional Integral MIMO speed controller with eight tuneable variables.....	5-7
---	-----

Figure 5-3 Constraint windows with initial controller gains before optimisation for left (a) and right (b) response.....	5-8
--	-----

Figure 5-4 Constraint windows with final controller gains after optimisation for left (a) and right (b) response	5-9
--	-----

Figure 5-5 Comparison of uncompensated compensated and reference response (PI control). (a) Step input for the right wheel (b) Step input for the left wheel.....	5-10
Figure 5-6 Control effort for the right (a) and left (b) motor (PI control).	5-11
Figure 5-7 Input-output observations for data collection	5-12
Figure 5-8 Schematic layout of fuzzy controller with delayed inputs and outputs.....	5-13
Figure 5-9 Comparison between the reference and actual response (fuzzy control). (a) Step input for the right wheel (b) Step input for the left wheel.....	5-15
Figure 5-10 Control effort for the right (a) and left (b) motor (fuzzy control).	5-15
Figure 5-11 Schematic layout of neural controller with delayed inputs and outputs.....	5-16
Figure 5-12 Neural network training flow chart	5-17
Figure 5-13 Structure of dynamic NN using PI controller as a teacher.....	5-19
Figure 5-14 Comparison between the reference and actual response (neural control). (a) Step input for the right wheel (b) Step input for the left wheel.....	5-23
Figure 5-15 Control effort of the left and right motor respectively (neural control)	5-23
Figure 5-16 Subsystem of speed controller (PI, FL and NN controller)	5-24
Figure 5-17 Mask block for selection of speed controller.....	5-24
Figure 5-18 Performance of PI control	5-26
Figure 5-19 Performance of fuzzy control	5-26
Figure 5-20 Performance of neural control	5-26
Figure 5-21 Kharitonov rectangles for the controlled closed-loop control system.....	5-31
Figure 5-22 A plot of $H(\omega)$ versus ω	5-32

Chapter 6

Figure 6-1 QFD analysis of control architecture specifications versus requirements/properties..	6-3
Figure 6-2 Hybrid approach.....	6-4
Figure 6-3 Structure of CAROS	6-5

Figure 6-4 Control strategy for robot(s) navigation	6-9
Figure 6-5 Robot(s) global strategy	6-10
Figure 6-6 Phase B of local control strategy	6-11
Figure 6-7 Dialog boxes used in low-level control unit. (a) Selection of speed controller (PI, fuzzy or neural) (b) Initial value of state vector for the robot(s) (c) Initial value for robot dynamics and speed controller gains	6-13
Figure 6-8 Schematic layout of sensor arrangement	6-14
Figure 6-9 Dialog box for environment/sensor parameters and gauges used to read sensor sensitivity	6-16
Figure 6-10 Infrared sensor sensitivity	6-17
Figure 6-11 Ultrasonic sensor sensitivity (0.7 Slope)	6-17
Figure 6-12 No-shape sensor sensitivity	6-17
Figure 6-13 Dialog boxes used to set initial values in robot(s) navigation.....	6-20
Figure 6-14 Schematic view of mobile robot navigation when enters Phase B.....	6-25
Figure 6-15 Schematic view of virtual point two calculation.....	6-25
Figure 6-16 Schematic view of virtual point one calculation.....	6-25
Figure 6-17 Supervisor-like co-ordination object.	6-27
Figure 6-18 Schematic layout of <i>go_to</i> robot behaviour.....	6-30
Figure 6-19 Schematic layout of possible robot orientation.....	6-30
Figure 6-20 Function defined in the first and third quadrant (shaded regions)	6-31
Figure 6-21 Dialog box for scaling factor selection used in <i>go_to</i> robot behaviour.....	6-32
Figure 6-22 Shape of hyperbolic tangent functions describing the output of <i>go_to</i> behaviour with $K = 0.2$ and $S = 0.02$ scaling factors.....	6-32
Figure 6-23 Subsystem of <i>avoid_static</i> robot behaviour.....	6-34
Figure 6-24 Dialog box used in <i>avoid_static</i> robot behaviour selection	6-35
Figure 6-25 Schematic layout of <i>avoid_static</i> robot behaviour with fuzzy encoding	6-35

Figure 6-26 Fuzzy membership functions used for the input variables of <i>avoid_static</i> robot behaviour.....	6-37
Figure 6-27 Fuzzy membership functions used for the output variables of <i>avoid_static</i> robot behaviour.....	6-37
Figure 6-28 Values of δV_{LD}^2 and δV_{RD}^2 when rule 7 is activated	6-38
Figure 6-29 ANN learning using fuzzy model as a teacher.	6-39
Figure 6-30 Structure of static NN used in <i>avoid_static</i> robot behaviour	6-41
Figure 6-31 Schematic layout of a hybrid <i>avoid_dynamic</i> robot behaviour.....	6-44
Figure 6-32 Dialog box used in <i>avoid_dynamic</i> robot behaviour selection	6-44
Figure 6-33 Internal structure of <i>avoid_dynamic</i> robot behaviour	6-45
Figure 6-34 Schematic layout of possible positions (states) of moving obstacle	6-46
Figure 6-35 Possible directions of moving object when entering the robot's sensing area	6-47
Figure 6-36 Identification of direction of moving obstacle using stateflow	6-48
Figure 6-37 Fuzzy membership functions used for the dSL, dSC and dSR input variable in <i>avoid_dynamic</i> robot behaviour	6-51
Figure 6-38 Fuzzy membership function used for T input variable of <i>avoid_dynamic</i> robot behaviour.....	6-51
Figure 6-39 Values of δV_{LD}^3 and δV_{RD}^3 when rule 30 is activated	6-54
Figure 6-40 Structure of static NN used in hybrid stateflow-neural <i>avoid_dynamic</i> robot behaviour.....	6-56
Figure 6-41 Schematic layout of <i>avoid_trap</i> robot behaviour.....	6-59
Figure 6-42 Schematic layout of the action co-ordinator mechanism	6-63
Figure 6-43 Global state identification mechanism	6-64

Chapter 7

Figure 7-1 Results A1	7-16
-----------------------------	------

Figure 7-2 Results A1	7-17
Figure 7-3 Results A1	7-18
Figure 7-4 Results A2	7-19
Figure 7-5 Results A3: (a) Infrared (b) Ultrasonic (c) No-shape	7-20
Figure 7-6 Results A3: (a) Infrared (b) Ultrasonic (c) No-shape	7-21
Figure 7-7 Results A3: (a) Infrared (b) Ultrasonic (c) No-shape	7-22
Figure 7-8 Results A4: (a) Fuzzy controller (b) Neural controller	7-23
Figure 7-9 Results A4: (a) Fuzzy controller (b) Neural controller	7-23
Figure 7-10 Results A4: (a) Fuzzy controller (b) Neural controller	7-24
Figure 7-11 Results A4: (a) Fuzzy controller (b) Neural controller	7-24
Figure 7-12 Results A4: (a) Fuzzy controller (b) Neural controller	7-25
Figure 7-13 Results A4: (a) Fuzzy controller (b) Neural controller	7-25
Figure 7-14 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural	7-34
Figure 7-15 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural	7-34
Figure 7-16 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural	7-35
Figure 7-17 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural	7-35
Figure 7-18 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural	7-36
Figure 7-19 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural	7-36
Figure 7-20 Results B2	7-37
Figure 7-21 Results B2	7-38
Figure 7-22 Results B2	7-39
Figure 7-23 Results C1	7-50
Figure 7-24 Results C2	7-51
Figure 7-25 Results C3	7-52
Figure 7-26 Results C4	7-53
Figure 7-27 Results C5	7-54
Figure 7-28 Results C6	7-55

Appendix A

Figure A-1 Drive train overview	A-3
Figure A-2 MIABOT's DC motor overview	A-3
Figure A-3 Control board overview.....	A-5
Figure A-4 PC transmitter unit overview.....	A-8

Appendix B

Figure B-1 Implementation of CAROS in MATLAB/Simulink-based simulator with one robot...	B-1
Figure B-2 Implementation of CAROS in MATLAB/Simulink-based simulator with two robots (one robot is driven by Human).....	B-2
Figure B-3 Implementation of CAROS in MATLAB/Simulink-based simulator with two robots.	B-2
Figure B-4 Implementation of CAROS in MATLAB/Simulink-based simulator with three robots	B-3
Figure B-5 Implementation of CAROS in MATLAB/Simulink-based simulator with four robots.	B-3
Figure B-6 Implementation of CAROS in MATLAB/Simulink-based simulator with five robots.	B-4
Figure B-7 Gauges used to read sensor sensitivity, robot speed and both global and internal states	B-4
Figure B-8 Controller-agent <i>find_VP1</i>	B-5
Figure B-9 Controller-agent <i>find_VP2</i>	B-5
Figure B-10 Controller-agent <i>approach_target</i>	B-5
Figure B-11 Controller-agent <i>approach_VP1</i>	B-5
Figure B-12 Controller-agent <i>approach_VP2</i>	B-6

Figure B-13 Controller-agent <i>get_target</i>	B-6
Figure B-14 Local controller <i>adjust_speed</i>	B-6
Figure B-15 Modelling parameters for mobile robot sensors and environment	B-7
Figure B-16 Global view of speed and orientation adaptive mechanism	B-8
Figure B-17 Global view of co-ordination between robot behaviours and action co-ordinator mechanism	B-8

Appendix C

Figure C-1 Input-output block diagram of a control system	C-1
Figure C-2 Description of input output of a general control system	C-2
Figure C-3 Conventional PID Controller	C-5

Appendix D

Figure D-1 Constraint window	D-6
------------------------------------	-----

Appendix E

Figure E-1 The Kharitonov's rectangular for $\omega_0 \geq 0$	E-5
---	-----

List of Tables

Chapter 3

Table 1 Main agent control architectures (after (Ferber, 1999))	3-52
---	------

Chapter 5

Table 1 Comparison of performance and execution time of PI, fuzzy and neural controller ..	5-27
--	------

Chapter 6

Table 6-1 Main characteristics of CAROS	6-5
Table 6-2 Variables used for robot navigation	6-10
Table 6-3 Controller-agents used in CAROS and SOAM for robot navigation	6-19
Table 6-4 Behaviours used in CAROS for robot navigation	6-29
Table 6-5 Parameters of fuzzy membership functions for <i>avoid_static</i> input variables	6-37
Table 6-6 Parameters of fuzzy membership functions for <i>avoid_static</i> output variables	6-37
Table 6-7 List of fuzzy rules incorporated in fuzzy <i>avoid_static</i> robot behaviour	6-38
Table 6-8 States used to identify the direction of moving obstacle	6-46
Table 6-9 Transitions used for the identification of direction of moving obstacle	6-49
Table 6-10 Parameters of fuzzy membership functions of dSL, dSC and dSR for <i>avoid_dynamic</i> input variables	6-51
Table 6-11 Parameters of fuzzy membership functions of T for <i>avoid_dynamic</i> input variable	6-52
Table 6-12 Singletons used of δV_{LD}^3 and δV_{RD}^3 for <i>avoid_dynamic</i> output variables	6-52

Table 6-13 List of fuzzy rules used in hybrid stateflow-fuzzy <i>avoid_dynamic</i> robot behaviour to model “ <i>traffic rules</i> ”	6-54
Table 6-14 List of fuzzy rules used in <i>avoid_trap</i> robot behaviour	6-60
Table 6-15 Conditions of the transitions used in global state identification mechanism	6-64
Table 6-16 Proposed control architectures and their architectural specifications.....	6-67

Chapter 7

Table 7-1 Division of results	7-3
Table 7-2 Division of results (Part I)	7-4
Table 7-3 Initial, target co-ordinates and desirable target angle for the mobile robot	7-5
Table 7-4 Comparison of performance criteria between three different speed controllers.....	7-6
Table 7-5 Parameters used in robot navigation in Group A2	7-7
Table 7-6 Comparison between different types of sensor sensitivity	7-10
Table 7-7 Comparison between fuzzy and neural static obstacle avoidance behaviour	7-13
Table 7-8 Division of results (Part II).....	7-26
Table 7-9 Comparison between stateflow-fuzzy and stateflow-neural hybrid dynamic obstacle avoidance behavioural encoding	7-28
Table 7-10 Collision avoidance result of a mixed environment containing both static and moving obstacles.....	7-32
Table 7-11 Division of results (Part III).....	7-40
Table 7-12 Subsystems used in all mobile robots in Part III.....	7-40
Table 7-13 Collision avoidance result of an environment containing three mobile robots.....	7-41
Table 7-14 Collision avoidance result of an environment containing four mobile robots.....	7-43
Table 7-15 Collision avoidance result of an environment containing five mobile robots	7-45
Table 7-16 Collision avoidance result of a mixed environment containing static obstacles and three mobile robots.....	7-46

Table 7-17 Collision avoidance result of a mixed environment containing static obstacles and four mobile robots.....	7-47
--	------

Table 7-18 Collision avoidance result of a mixed environment containing static obstacles and five mobile robots.....	7-48
--	------

Appendix A

Table A-1 Specifications of MIABOT's control board	A-5
--	-----

Table A-2 Command coded.....	A-7
------------------------------	-----

Notations and Symbols

Optimisation

\mathbf{x}	Variables in an optimisation problem
\mathbf{x}_l	Vector of lower bounds
\mathbf{x}_u	Vector of upper bounds
$\mathbf{g}(\mathbf{x})$	Vector of the constraint bound error
ω	Vector of weights on the constrains
γ	Scalar (imposes an element of slackness)
$f(\mathbf{x})$	Objective function
\mathbf{x}^*	Local minimiser or local solution
$\lambda_i \quad i = 1, 2, \dots$	Lagrange Multipliers
$C_i(\mathbf{x}) \quad i = 1, 2, \dots$	Constraint functions
∇	First derivative operator (elements $\partial / \partial x_i$)
\mathcal{R}^n	n-dimensional space
Σ	Summation

Interval polynomial analysis

Q	Uncertainty bounding set
\mathbf{q}	Vector of uncertain parameters
q_i^-, q_i^+	Upper and lower bound of each element of \mathbf{q}
$p(s, \mathbf{q})$	Uncertainty polynomial
$G(s, \mathbf{q})$	Uncertain plant
$F(s, Q)$	Family of plants with given uncertainty bounding set Q
$n(s, Q)$	Family of numerators
$d(s, Q)$	Family of denominators
$a_i(\mathbf{q})$	Affine linear uncertainty structure of uncertain polynomial $p(s, \mathbf{q})$

ω	Frequency
ω_c	Cut-off Fixed frequency
$H(\omega)$	Frequency sweeping function
$K_i(s)$	Kharitonov's polynomials

Fuzzy logic systems

$\mu(x)$	Membership function of x
A, B, C, D	Fuzzy sets
X	Universe of discourse
$\mu_{A \cap B}(x)$	Intersection (AND) of fuzzy sets A and B
$\mu_{A \cup B}(x)$	Union (OR) of fuzzy sets A and B
$\mu_{\bar{A}}(x)$	Complement of fuzzy set A
\hat{t}	Triangular norm or t-norm
\hat{s}	Triangular s-norm
\in	Member of a set
\notin	Non-member of a set
\subset	Subset
\supset	Implication
\wedge	Logical AND
\vee	Logical OR
\forall	Universal quantifier
\otimes	Cartesian product or space
\emptyset	Empty or null set
$[0,1]$	Closed interval
$(0,1)$	Open interval
A_i	Fuzzy set for i^{th} input universe
u, z	Crisp output (after defuzzification), control action (in closed-loop)
a_i	Degree of true of a rule's premise
μ_i	Firing strength of the i^{th} fuzzy rule

Artificial neural networks

K	Neuron
u_k	Output of the summing junction (used in the model of neuron)
b_k	Bias
$\varphi(\cdot)$	Activation function
y_k	Output signal of the neuron
x_1, x_2, \dots, x_m	Input signals
$w_{k1}, w_{k2}, \dots, w_{km}$	Synaptic weights of the neuron

Clustering

\mathcal{R}^n	n-dimensional space
Ψ_k	Observation vector
N	Set of observations
Ψ	Data matrix (or pattern)
n	Number of data points
c	Number of clusters
x_k	k^{th} data point, next cluster centre
v_i	i^{th} cluster centre
μ_{ik}	Degree of membership of the k^{th} data point in the i^{th} cluster
m	Weighting exponent
$H^m(u)$	Height of mountain function at point u
σ	Determines the height as well the smoothness of a mountain function
ζ	Positive constant
D_i	Density measure at data point x_i
r_a	Neighbourhood radius
r_b	Neighbourhood radius usually $r_b = 1.5r_a$
x_c	Centre of group
ε^{up}	Threshold above which a point is selected as a centre
$\varepsilon^{\text{down}}$	Threshold below which a point is definitely rejected
d	Distance

Finite state machines

M	Finite state machine
S	Finite state space
I	Finite input space
O	Finite output space
T	Transition relation
R	Set of reset states
r	Unique reset state
Δ	Next state relation
Λ	Output relation
δ	Next state function
λ	Output function
p	Present state
n	Next state

Abbreviations

ACM	Action Co-ordinator Mechanism
ACTRESS	Actor-Based Robot and Equipment Synthetic System
AI	Artificial Intelligence
ALIFE	Artificial Life
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Networks
BFGS	Broyden Fletcher Goldfard Shanno
BP	Back Propagation
CAROS	Co-operative Autonomous Robotic Systems
CEBOT	Cellular Robotic System
DAI	Distributed Artificial Intelligence
DC	Direct Current
DDM	Decision Making Mechanism
DPS	Distributed Problem Solving
EPF	Electrostatic Potential Field
FCM	Fuzzy C-Means
FIS	Fuzzy Inference Systems
FKCN	Fuzzy Kohonen Clustering Network
FL	Fuzzy Logic
FLS	Fuzzy Logic Systems
FNN	Feedforward Neural Networks
FSM	Finite State Machines
GSIM	Global State Identification Mechanism

IAE	Integral Absolute Error
ISE	Integral Square Error
ITAE	Integral Time Absolute Error
KBS	Knowledge Based Systems
KT	Kuhn-Tucker
LP	Linear Programming
LTI	Linear Time Invariant
MAS	Multi-Agent Systems
MATLAB	Matrix Laboratory
MENN	Modified Elman Neural Networks
MIABOT	Merlin Intelligent Autonomous Robots
MIMO	Multiple-Input Multiple-Output
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
NCD	Non-linear Control Design
NLS	Non-Linear Switch
NLTI	Non-Linear Time Invariant
PD	Proportional Derivative
PI	Proportional Integral
PID	Proportional Integral Derivative
ODF	Ordinary Differential Equations
QFD	Quality Function Deployment
QP	Quadratic Programming
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks

SISO	Single-Input Single-Output
SOAM	Speed Orientation Adaptive Mechanism
SQP	Sequential Quadratic Programming
TS	Takagi-Sugeno
VP	Virtual Point
WENN	Wave Expansion neural Network

1

Introduction, Motivation and Overview of the Thesis

1.1 Introduction

This thesis is concerned with the development of modelling and identification techniques for the design and implementation of a novel hybrid multi-agent oriented control architecture for navigation of multiple autonomous mobile robots operating in an unknown and unstructured environment populated by static and/or dynamic obstacles.

Autonomous control and navigation of mobile robotic vehicles are fundamental enabling technologies for automation in a variety of operating domains ranging from industrial environments to remote planetary surfaces. The engineering problem to be solved generally consists of achieving real-time sensor-based motion control among obstacles in the environment while performing useful tasks throughout its accessible regions. In many instances, mobile robots are required to do so using limited resources (e.g., power computation, sensors etc.) that are resident onboard the vehicle.

The remainder of this chapter is organised as follows: Section 1.2 highlights the main motivations of the thesis for the research work carried out. Potential application areas for the results of this thesis are given in section 1.3. The aim and objectives of the research carried out is indicated in section 1.4. Section 1.5 presents the main challenges and problems in mobile robot navigation with a brief discussion of how they have been approached in this thesis. An overview of the thesis is presented in section 1.6. Section 1.7 presents a brief summary of the main contributions of the research described in this thesis. A summary of the chapter is presented in section 1.8. A list of references is given at the end of the chapter.

1.2 Motivation of the thesis

Clearly there are many motivations for working on mobile robots. The mobile robots, sometimes called autonomous guided vehicles (AGVs), can be found nowadays in factories, storage areas, universities, hospitals, nuclear plants, homes and even on Mars for planetary exploration. Several robots working together provide an advantageous solution with respect to a single-robot system. Advantages include: robustness, scalability, large range of possible tasks, greater efficiency, parallel execution, ease of development, lower economic cost and problem solving.

Theoretically, multiple robots should be able to accomplish any task that a single robot can. However, since multiple robots can cover more ground than a single robot, there are tasks that multiple robots can accomplish that a single robot cannot. Multiple robots may also offer performance benefits. For instance, (Balch and Arkin, 1994) show how multiple robots provide speedup for foraging and similar tasks. Furthermore, implementation of multiple robots offers robustness, for example, if one robot malfunctions or is destroyed, the others can continue the task. This is a prime advantage for military robots, which may be under attack from enemy forces or damaged while cleaning fields of landmines.

A single robot system for a particular task may need to be more complicated than each of the individual robots in a multi-robot system for the same task. A single robot system will need to handle all aspects of the task, while a multi-robot system can divide the subtasks among the robots, requiring each robot to only know how to accomplish its subtask. Therefore, a multi-robot system is often less expensive and easier to develop than an equivalent single-robot system.

Multi-robot systems are inspired and have several similarities with multi-agent systems (MAS) but are not exactly the same. Multi-robot systems have to interact with the real world, which is difficult to model. According to (Fukuda and Ueyama, 1994), the behaviour of a robot depends on its perceptions and on the physical constraints from the environment and other robots. Therefore, there is a huge motivation for working on mobile robots as the challenges are not small or easy to overcome. The design of these systems involves knowledge from different areas, such as artificial intelligence, control engineering, electronics, mechanics and others. Very often, the application domain is not structured involving a large number of interactions, such as static and moving obstacles (other robots). When many robots work together in a confined space, one of the biggest challenges is the control of individuals in order to avoid interference and resource sharing problems. Another major challenge is the maintenance of the multi-robot system working in good conditions. A single robot is hard to maintain. Many robots are even harder.

According to (Arkin and Balch, 1998) another motivation for working on mobile robots is to shed some light on human cognition. Such work may be considered good if it provides an explanation of certain observed cognitive behaviour, for example dealing with conflicts in reasoning or goals, and more importantly if it leads to testable predictions whose results are not known in advance but which turn out to be as predicted.

1.3 Application areas

The direct application for the results of this thesis is for robotics education, further research and also for certain non-academic areas. According to (Altenburg, 1995) the appropriate use of robots is best recognised where the task or environment is life threatening or for other reasons inaccessible to people. Therefore, there is an increased interest in multiple autonomous mobile robot systems due to their large applicability to various tasks. In the following brief outline of application areas, which may involve operation of multiple mobile robots is given. This range of applications is by no means exhaustive and appropriate references for these ideas are included.

According to (Boutros, 1994) military land mines kill and maim thousands of people every month around the world, most of these people are civilians. Countries where civil wars have persisted for many years are covered with thousands of unmapped and uncovered land mines. Simple, inexpensive robots could detect and detonate these land mines, possibly sacrificing parts or all of themselves in the process.

The treatment of hazardous materials spills is a real problem. Although every precaution is taken to prevent accidental release of these materials, accidents do happen. The containment and clean up of such materials is usually very costly. A multi-robot system equipped with appropriate actuators could replace human workers in this task. This would certainly reduce the threat to human life, and furthermore the robot could operate 24 hours a day.

The idea of sending and establishing a robotic presence on other planets has been suggested since the beginning of the space age. Several researchers such as (Flynn *et al*, 1989), (Miller, 1990) and (Steels, 1990) have proposed that a self-replicating system of mining and construction robots could be set in place in other planets. This application domain can be also extended for planetary surface exploration.

The application of multiple robots to ocean floor exploration has been proposed by (Lipkin, 1995). The vast riches of the ocean floor can be explored and cultivated using a system of multiple mobile robots. Underwater vehicles are already being used for surveillance, explorations and salvage operations.

Certainly there are many more applications of multiple robots to mention such as agriculture, medicine, ethological research and applications involving micro-robotics.

1.4 Aim and objectives of the research

The aim of the research is to develop new hybrid control architecture for navigation of multiple autonomous mobile robots operating in an unknown and unstructured environment populated by static and/or dynamic obstacles.

The objectives of the research are to:

- Conduct a literature survey on approaches/methods related to the development of intelligent control architectures for navigation of multiple autonomous mobile robots.
- Propose a design methodology for developing integrated solutions for autonomous mobile robotic systems.
- Classification and discussion of the main design methodology of control systems architectures for autonomous mobile robots.
- Familiarisation with existing control and learning algorithms and investigation of intelligent control methods.
- Investigation and suggestions for merging multi-agent systems and control engineering.
- Familiarisation and modelling of MIABOT V2¹ mobile robots.

¹ MIABOT is trademark of Merlin Systems Corporation Ltd. Copyright 1999. All rights reserved.

- Discovery of fuzzy-neural local models from observation data using clustering techniques and supervised learning.
- Optimisation and tuning of control parameters using both classical and intelligent control theories.
- Investigation of robust stability testing of closed-loop systems under uncertainty in robot's dynamics.
- Prediction and estimation of dynamic objects using linear and non-linear models.
- Design and implementation of the hybrid proposed control architecture for navigation of multiple autonomous mobile robots.
- Evaluate the proposed control architecture and compare it with existing architectures.
- Test the proposed architecture and analyse the results.

1.5 Challenges and problems in mobile robot navigation

Navigation is a vital issue in the research of autonomous mobile robots. The navigation of an autonomous mobile robot may be considered as a task of determining a collision free path that enables the robot to travel through an environment populated with obstacles from an initial configuration to a target configuration, where configuration here refers to the spatial co-ordinate and the heading angle of the robot. The ultimate goal of research in mobile robot systems is to develop control strategies and architectures, which support autonomous operations in an unknown or partially known environment.

Several methods for controlling mobile robot systems have been put forward. They can be subdivided into two main parts: global planning and local control. The former is usually conducted off-line and is based on the complete knowledge of the environment and the robot. It enables generation of collision-free paths, which are assumed to be executed correctly. The knowledge about the system and the environment is originated either from the modelling through *a priori* knowledge or from the perception through a sensory system. Many attempts such as geometric

algorithms (Janet *et al*, 1997), potential fields methods (Khatib, 1986), (Hwang and Ahuja, 1992) and (Guldner and Utkin, 1995), as well as other heuristic or approximating approaches (Brooks, 1983), (Lozano-Perez, 1983) and (Diamantopoulos *et al*, 2000) for solving this problem have been reported. As a pre-specified environment is required for these methods to plan the path, they fail when the environment is not fully known.

The local control or behavioural strategies, also known as the obstacle avoidance methods, are more efficient in autonomous mobile robot navigation in an unknown or partially known environment as has been demonstrated by (Brooks, 1986) and (Arkin, 1998). Such strategies do not require a global map of the environment but utilises on-line sensory information to tackle the uncertainty. Considering the up-to-date status of the robot and the relationships with its environment the robot motion decision is made. The main advantage of this approach is the ability to handle changing aspects of the environment because the structural modelling of the environment is not necessary. Usually the behaviour strategies are well-suited for real-time implementation, although they suffer from a deadlock problems since the high level planning is no longer available.

Advanced mobile robotic systems operating in uncertain dynamic environments, combine information from several sensory sources needed to acknowledge the dynamics of the robot. Prior knowledge of the domain may be incomplete, and reasoning must be deliberative in nature and fast enough to respond to unexpected events. Also, the information gained via sensory subsystems is often incomplete, inaccurate and uncertain. To operate correctly in this type of environment, planning systems must be reactive, taking into account, information about current state. Thus advanced mobile robotic systems architectures must combine deliberative planning with reactive sensor driven operations.

In this thesis a novel hybrid multi-agent oriented control architecture for navigation of multiple autonomous mobile robots operating in an unknown and unstructured environment populated by static and/or dynamic obstacles is proposed. The control architecture integrates deliberative planning and reactive control with attention focused on the design and co-ordination of robot behaviours. The complex behaviour is generated by combining simpler behaviours. This complex behaviour in the robot results from the interaction between the goals, the internal and global states and the environment itself. Fuzzy logic, neural networks and hybrid solutions form the basis of the behaviours in order to generate more complex observable robot behaviour.

1.6 Overview of the thesis

The thesis is organised as follows:

Chapter one is this introduction. It presents the motivations for conducting the research, as well as an overview of the thesis.

Chapter two discusses research related to this thesis. The literature review of related work based on co-operative robotics and control of multiple mobile robots (multi-agent robotics) is presented. Background information of the birth and origin of the behaviour-based control is given. The role of distributed artificial intelligence and distributed systems in the development of control architectures for multiple autonomous mobile robots is presented. The field of artificial life is introduced with several numbers of contributions. The origin of intelligent control including its approaches is illustrated. Recent research in modelling, identification and control of dynamic systems using methodologies from intelligent control are reviewed. Robot navigation using intelligent control methods is described. In particular, the chapter reviews robot navigation techniques up to date, focusing on those utilising fuzzy logic and neural networks.

Chapter three proposes, justifies and presents the main methodology adopted for the research work carried out in this thesis. This has been broken down into nine basic steps. Each step is presented individually focusing on design/modelling issues followed by a discussion of advantages or disadvantages when the particular method is adopted. The nine steps include: conventional control design, constrained optimisation using the non-linear control design tool, parametric approach, fuzzy systems, neural networks, clustering, design of control system architecture, multi-agent systems and finite state machines.

Chapter four presents the modelling of MIABOT V2 mobile robot. The robot is small in size measuring 8cm^3 and is steered and driven by differential drive utilising two DC motors enabling robot's speed up to 1.2m/s . The full non-linear dynamic model of the robot is established as a complete description of its dynamics. The robot's dynamic model is used in chapter six and seven for the design and evaluation of the proposed control architecture.

Chapter five presents control and robust stability analysis of the MIABOT V2 mobile robot and discovery of fuzzy-neural local models from observation data. The robot's closed-loop system is tested under uncertainty in robot dynamics. An algorithmic methodology is presented for discovery of fuzzy-neural local models from observation data using clustering and supervised learning.

Chapter six presents the development of a novel hybrid multi-agent based control architecture called CAROS (Co-operative Autonomous RObotic Systems) for navigation of multiple autonomous mobile robots in unknown static and/or dynamic environment. The proposed architecture takes the advantages of various control structures integrating them in a way that results in an overall increase in synergy.

Chapter seven evaluates the proposed control architecture. The chapter investigates the validity of the control architecture when applied to the problem of navigation of single/multiple autonomous mobile robots. Results are presented to show the effectiveness of the proposed architecture.

Chapter eight reviews the thesis, draws some conclusions from the work presented, summarises the contributions of this research and makes recommendations for further work.

The appendices describe lower level details of this work. Appendix A presents the MIABOT V2 mobile robot. The main block diagrams designed in Simulink² for the modelling and implementation of the CAROS control architecture are given in Appendix B. Appendix C gives a brief overview of the conventional control design. Constrained optimisation using the non-linear design tool is described in Appendix D. Appendix E presents definitions and theorems related to robust stability testing based on interval polynomials. The backpropagation algorithm for training multi-layer feedforward neural networks is presented in Appendix F. Appendix G contains copies of the publications that have been produced during the course of the research described in the thesis.

Figure 1-1 depicts a schematic outline of how the thesis is organised.

² SIMULINK is registered trademark of the Math Works, Inc. Copyright 1990-2000. All rights reserved.

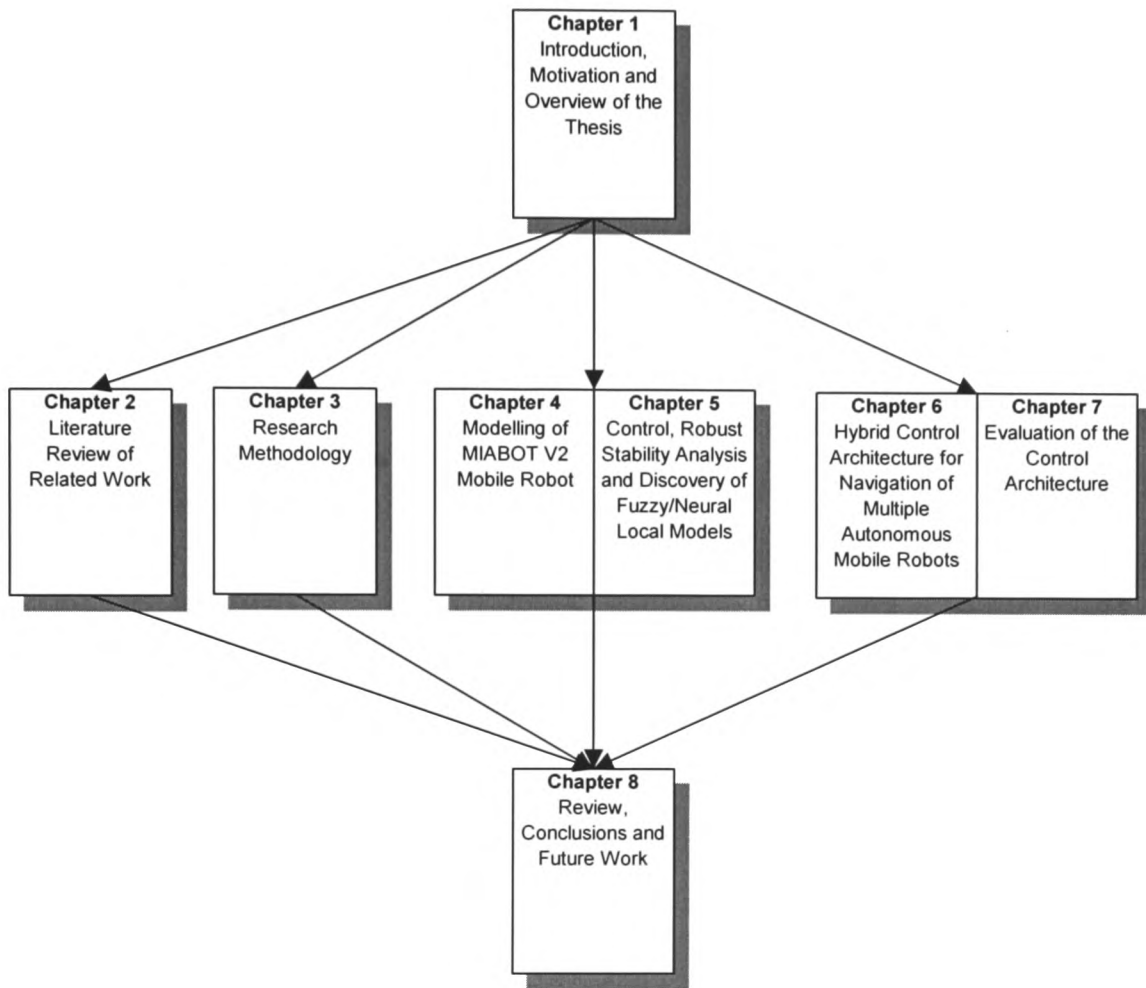


Figure 1-1 Schematic outline of the thesis

1.7 The main contributions of the thesis

Here is a brief summary of the main contributions of the research described in this thesis, these are discussed in more detail in chapter eight, section 8.4.

1. Development of a novel hybrid multi-agent based control architecture called CAROS for navigation of multiple autonomous mobile robots operating in an unknown and unstructured environment populated by static and/or dynamic obstacles.

2. Novel approach for identification of direction of moving obstacles (other robots) using finite state machines.
3. Novel approach for behavioural encoding using hybrid solutions such as fuzzy/stateflow³ and neural/stateflow for autonomous robot navigation.
4. Proposed a design methodology for developing integrated solutions for autonomous mobile robotic systems and classification of the main design methodology (properties) of control systems architectures for autonomous mobile robots.

Less significance contributions are:

1. Literature survey on approaches/methods related to the development of intelligent control architectures for navigation of multiple autonomous mobile robots.
2. Modelling of MIABOT V2 mobile robot. A generic approach for optimisation and identification of parameters of physical components (i.e moments of inertia) conducting experiments and using the non-linear control design tool.
3. Comparison between PI, fuzzy and neural controllers based on their performance criteria (ISE, ITE and ITAE) and execution time. An algorithmic methodology for discovery of fuzzy/neural local models from observation data. Use of parametric robustness analysis approach for robust stability testing of closed-loop control system under uncertainty in robot dynamics (The approach has not been considered previously within the research community of autonomous mobile robots).
4. Identification of the relationship of the most important requirements/properties of control architecture versus the main control architecture specifications using the Quality Function Deployment (QFT) tool.
5. Modular approach for modelling three types of sensor and sensor sensitivity.

³ Stateflow is registered trademark of the Math Works, Inc. Copyright 1997-2000. All rights reserved.

1.8 Summary

This chapter presented an introduction, motivation and overview of the thesis. The motivation of conducting the research described in this thesis was given. Some potential application areas for the results of this thesis were suggested. The list of applications suggested is by no means exhaustive and appropriate references for these ideas were included. The aim and objectives of the research carried out here are stated. Then the main challenges and problems in mobile robot navigation with brief discussion of how they have been approached in this thesis were discussed. An outline of the thesis was presented. A brief summary of the main contributions resulting from the research work described in the thesis was given.

The next chapter presents a literature review of topics related to this thesis.

1.9 References

ALTENBURG, K. R. 1995. *COCOBOTS: Robots, Ants and Randomness*. Master Thesis. North Dakota State University.

ARKIN, R. C. 1998. *Behaviour-Based Robotics*. USA: MIT Press. 0-262-01165-4.

ARKIN, R. C. and BALCH, T. 1998. Co-operative Multiagent Robotic Systems. In: eds. D. KORTENKAMP, R. P. BONASSO, and R. MURPHY. ed. *Artificial Intelligence and Mobile Robots. Case Studies of Successful Robot Systems*. USA: AAAI Press/The MIT Press, pp. 277-296.

BALCH, T. and ARKIN, R. C. 1994. Communication in Reactive Multiagent Robotic Systems. *Autonomous Robots*, **1** (1), pp. 27-52.

BOUTROS, B. G. 1994. The Land Mine Crisis: A Humanitarian Disaster. *Foreign Affairs*, **73** (5), pp. 8-14.

BROOKS, R. A. 1983. Solving the Find-path Problem by Good Representation of Free Space. *IEEE Transactions of Systems, Man and Cybernetics*, **13** (3), pp. 190-197.

BROOKS, R. A. 1986. A Robust Layered control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, **RA-2** (1), pp. 14-23.

DIAMANTOPOULOS, A., ROBERTS, G. N. and HARWOOD, D. J. 2000. *Robot Motion Planning in Environments Populated by Shrinking and Expanding Obstacles*. EUREL, European Advanced Robotics Systems, Masterclass and Conference - Robotics 2000.

FLYNN, A. M., BROOKS, R. A., WELLS III, W. M. and BARRETT, D. S. 1989. Intelligence for Miniature Robots. *Sensors and Actuators*, **20**, pp. 187-196.

FUKUDA, T. and UEYAMA, T. 1994. *Cellular Robotics and Micro Robotic Systems*. USA: World Scientific Publishing. 981-02-1457-X.

GULDNER, J. and UTKIN, V. I. 1995. Sliding Mode Control for Gradient Tracking and Robot navigation Using Artificial Potential Fields. *IEEE Transaction of Robotics Automation*, **11** (2), pp. 247-254.

HWANG, Y. K. and AHUJA, N. 1992. A Potential Field Approach to Path Planning. *IEEE Transactions of Robotics Automation*, **8** (1), pp. 23-32.

JANET, J. A., LUO, R. C. and KAY, M. G. 1997. Autonomous Mobile Robot Motion Planning and Geometric Beacon Collection Using Traversability Vector. *IEEE Transactions of Robotics Automation*, **13** (1), pp. 132-140.

KHATIB, O. 1986. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, **5** (1), pp. 90-98.

LIPKIN, R. 1995. Tracking the Undersea Scent: A Robot Mimics the Lobster's Keen Sense of Smell. *Science News*, **147** (5), pp. 78-79.

LOZANO-PEREZ, T. 1983. Spatial Planning: A Configuration Space Approach. *IEEE Transactions of Computing*, **32** (2), pp. 19-40.

MILLER, D. P. 1990. *Multiple Behaviour-Controlled Micro-Robots for Planetary Surface Missions*. IEEE International Conference on Systems, Man and Cybernetics. pp. 281-292, Los Angeles.

STEELS, L. 1990. Cooperation Between Distributed Agents Through Self-Organisation. *Decentralized A. I.*, pp. 175-196.

2

Literature Review of Related Work

2.1 Introduction

This chapter presents a literature review of topics related to this thesis. Some of the topics discussed here will be mentioned in more detail than others, since they are more closely related to the main subject of this thesis. However, it is impossible to refer to all the contributions from the last two decades or so, therefore, this chapter does not present an exhaustive survey but only gives an indication of the development of techniques/methods related to this thesis. Further literature reviews related to this thesis can be found in the contributions of (Dudek *et al*, 1993), (Harris, 1994), (Bhattacharyya *et al*, 1995), (Everett, 1995), (Borestein *et al*, 1996), (Muller, 1997), (Cao *et al*, 1997), (Jang *et al*, 1997), (Zadeh *et al*, 1997), (Arkin, 1998), (Kortenkamp *et al*, 1998), (Senehi and Kramer, 1998), (Ferber, 1999), (Ridao *et al*, 1999) and (Van Breemen, 2001). Note that some parts of chapters four and five are concerned with modelling and control of a mobile robot. Brief surveys of related contributions in these fields are provided at the beginning of each of these two chapters.

The remainder of this chapter is organised as follows: Section 2.2 introduces the field of artificial life with a number of related contribution to the work presented in this thesis. Related research work based on co-operative robot and control of multiple robots is given in sections 2.3 and 2.4. The birth of the behaviour-based control including its advantages/limitations for control of a mobile robot is described in section 2.5. The influence and importance of distributed artificial intelligence and distributed systems related to the development of a multi-robot system is given in section 2.6. Section 2.7 presents the origin of the intelligent control including its main approaches. Modelling, identification and control of dynamic systems using methodologies from intelligent control is also discussed in this section. Section 2.8 presents a literature review of related work based on navigation of mobile robot using methodologies from the intelligent control. In particular the research work to date on mobile robot navigation using both fuzzy and neural techniques is presented in this section. A discussion follows in section 2.9, and the main summary of the chapter is presented in section 2.10. A list of references is given at the end of the chapter.

2.2 Artificial life

The field of Artificial Life (Alife) focuses on bottom-up modelling of various complex systems. This scientific field of study, as defined by (Langton, 1989) is the study of man-made systems that exhibit behaviour characteristics of natural living systems. Alife incorporates a broad, interdisciplinary approach to the study of a variety of phenomena. Disciplines included in the study of Alife are Artificial Intelligence, Computer Science, Mathematics, Control Engineering and Psychology. Knowledge of experimental material from these disciplines is required in order to develop artificial life. These materials and models may include fuzzy logic, neural networks, multi-agent systems, control of autonomous mobile robots, modelling and identification of dynamic systems. These diverse and evolving scientific fields are the source of inspiration for the work undertaken in this thesis. In fact, the new hybrid control architecture described in chapter six as well modelling, control and identification techniques developed in this thesis owe

their design origins to the artificial life. Artificial life relevant to this thesis features contributions based on simulations of multiple autonomous robots, as described by (Maes and Brooks, 1990), (Steels, 1990), (Mataric, 1992), (Noreils, 1993), (Lueth and Laengle, 1994), (Balch and Arkin, 1998), (Arkin and Balch, 1998), (Parker, 1999), (Goldberg and Mataric, 1999) and (Kube and Bonabeau, 2000).

Work in artificial life is related to the work in this thesis in that both are concerned with exploiting the dynamics of local interactions between agents (agent is either a physical unit or software, for instance it can be a single mobile robot or a program representing or encoding a specific task) and the world in order to create complex global behaviours.

2.3 Co-operative robots

As an integrative engineering discipline, robotics has always had to confront technological constraints that limit the domains that can be studied. Co-operative mobile robotics has been subject to these same constraints and tends to be more severe because of the need to cope with multiple mobile robots. Co-operative robotics is a highly interdisciplinary field that offers the opportunity to draw influences from many other domains. Therefore developing systems for co-operative robotics is a very difficult and challenging task. In particular, the assumption that multiple robots (or multi-agent robotics) have the potential to solve problems more efficiently than a single robot has attracted the attention of many researchers in the areas of control engineering, computer science and psychology. According to (Arkin and Balch, 1998) there are several reasons why two or more robots can be better than one. Firstly, *distributed action*: multiple robots can be in many places at the same time (Jung and Zelinsky, 1999). Secondly, *inherent parallelism*: it is quite possible that many applications could be solved much more quickly if the mission could be divided across a number of robots in parallel (Parker, 1998). Thirdly, *simpler is better*: building and using several simple robots, can be easier, cheaper, more flexible and more fault-tolerant than having a single powerful robot for each separate task

(Deneubourg *et al*, 1990). Fourthly, *divide and conquer*: several problems are well suited for decomposition and allocation among multi-robot system (Azarm and Schmidt, 1997). Unfortunately there are also drawbacks, in particular regarding collision avoidance among individual robots, co-ordination and elimination of interference. The degree of difficulty imposed depends heavily upon the task, communication (with or without) and the control strategy chosen.

2.4 Control of multiple robots (multi-agent robotics)

Research on control of co-operative multiple robots began in the late 1980s. (Fukuda and Nakagawa, 1987), introduced new project in the field of co-operative robotics called CEBOT (Cellular Robotic System). Their work is based on co-ordination among mobile multi-robot systems with emphasis on communication mechanisms, which can be used to support co-ordinated behaviour. About the same time relevant work on co-operative robotics carried out by (Beni, 1988), SWARM (large numbers of homogeneous robots), and (Asama *et al*, 1989), with the ACTRESS (ACTor-based Robot and Equipment Synthetic System), a multi-robot system designed for heterogeneous agents (robots), with focus on communication issues. The robots act independently, but if the need arises, they negotiate with other robots to form a co-operative group to handle the problem. The 1990s decade begins with the works of (Caloud *et al*, 1990), on the GOFER architecture, and (Steels, 1990). The latter used the behaviour-based approach to solve the problem of co-operation between distributed robots. Research activity on co-operative robots increased dramatically with important work by (Deneubourg *et al*, 1990), (Asama *et al*, 1991), (Wang, 1991) and (Arkin, 1992) the latter presents research concerned with sensing, communication, and social organisation for tasks such as foraging. (Mataric, 1992), has developed behaviours for multi-robot system using the subsumption style architecture. (Noreils, 1993), proposed a three-layered control architecture that included a planner level, a control level, and a functional level. Similar work was carried out by (Parker, 1993), (Kube and Zhang, 1993), (Laengle and Lueth, 1994), (Barnes, 1996), (Shim *et al*, 1997), (Yoshida *et al*, 1998) and

(Werger, 1999). There is more work reported in the literature, but the aforementioned is the most significant within the research community. All the above research activity has attempted to solve the same problem (control of multiple robots) by adopting different techniques. The solution to the problem is not straightforward, so many of the researchers prefer to perform simulation rather than physical implementation. Many of the recent attempts to control multiple robots, in contrast to the earlier works, are based on a behaviour-based approach (Brooks, 1986). However, the emphasis on the connection between intelligence and environment is strongly associated with the behaviour-based approach but is not intrinsic to multiple robot systems.

In terms of co-operation and communication, most of the work cited either uses extensive explicit communication and co-operation among the robots, or almost none at all. In systems that are co-operative by design, two or more robots are aware of each other's existence, and can sense and recognise each other directly or through communication. This type of research explores explicit co-operation, usually through the use of direct communication. The other category includes work on implicit co-operation, in which the robots usually do not recognise each other but indirectly co-operate by having identical or at least compatible goals. The research work described in this thesis, falls nearer this end of the spectrum. It is focused on robots (agents) that can discriminate each other from the rest of the world based on a local reactive approach and use this ability as a basis to form global behaviour.

2.5 Behaviour-based control

The real world can be described as a complex and unstructured environment. Robots, which are designed to operate in the real world, must be able to operate in situations which their designers only vaguely envisaged and must have the ability to respond appropriately and quickly to unexpected events. The classical artificial intelligence approach to interacting with an environment is to divide the task into a number of major subsystems as shown in Figure 2-1.

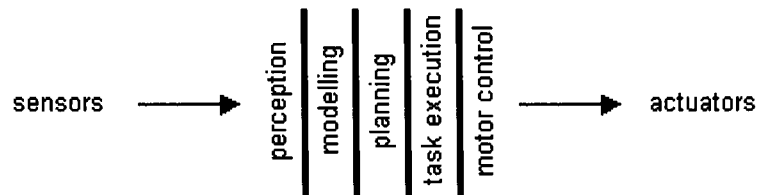


Figure 2-1 Traditional functional decomposition

The perception subsystem handles the sensing devices connected to the robot. The modelling subsystem converts the sensor input into a description of where the robot is in relation to its internal model of the environment. The planning subsystem attempts to work out how it will achieve its goals given the current world state. The task execution subsystem breaks down the plan into detailed motion commands and finally the motor control subsystem causes these commands to be executed. Each of these subsystems is a complex program, and all have to work together perfectly for the robot to operate at all. In particular perception and world modelling subsystems are extremely complex. Currently it is only possible to design such subsystems for structured environments only, as noisy and random environments of the real world overwhelms them. In addition, as the complexity of the environment increases, the time needed to perceive, model and plan about the world increases exponentially. In (Watanabe *et al*, 1992), (Heikkila and Roning, 1992), (Liscano *et al*, 1995) and (Van Brussel, 1995) the disadvantages of the classical artificial intelligence approach are discussed in more detail.

The behaviour-based control is an alternative to the classical artificial intelligence approach. Instead of having a number of complex individual vertical tasks, the behaviour-based approach tackles the control problem by thinking of it as a number of horizontally arranged layers. This paradigm is inspired by biology and fostered by the artificial intelligence community. It has been observed that the number of behaviour patterns of even simple animals exhibits most of

the characteristics in designing artificial autonomous agents. Complex behaviour patterns can often be decomposed into hierarchies of simple behavioural patterns. When individual simple behaviours operate concurrently, new behavioural patterns may emerge. In the subsumption architecture developed by (Brooks, 1986), behaviours are arranged into horizontal layers as shown in Figure 2-2. Each layer provides a degree of performance by adding a new behaviour on top of the previous layers. The new layer overrides some aspects of the low-layer behaviours. Thus each layer is fully capable of controlling the robot by itself. Major advantages of the behaviour-based control are rapid response to environment change, robustness in real worlds, less demanding computation requirements and flexibility.

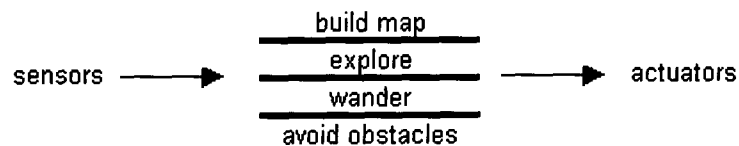


Figure 2-2 Behaviour-based decomposition

Industrial robots usually operate in a more controlled environment than experimental mobile robots. Working in a controlled structured environment scientifically reduces the perception and world modelling sub-elements of the control task as described above. Inclusion of advanced sensory systems such as vision systems does imply an increase in the complexity of control in the areas of perception, interpretation and world modelling. In addition, flexibility is one of the most important attributes of robotics over other forms of automation. This flexibility implies uncertainty in the robot environment. For these reasons, behaviour-based control may therefore have a useful role to play in certain applications of industrial robotics. However, earlier behaviour-based systems have suffered from difficulties in systems modularity, state representation and integration of world models. For instance for a mobile robot to be able to execute tasks, such as transportation, cleaning, assisting disabled persons it is necessary to have

a representation of the overall state and to be able to use knowledge of the environment. As the behaviour-based architectures are usually highly distributed and each behaviour is meant to implement a specific function, representation and sharing of systems states and knowledge have been inconvenient. According to (Hartley and Pipitone, 1991) in control architectures, like subsumption, behaviours usually need to access the internal states of low-level behaviours, which make implementation of behaviours dependent and modularity difficult. According to (Wilson *et al*, 1997) a behaviour-based approach can take away the cognitive bottleneck provided by the classical artificial intelligence, but on the other hand it is difficult to give the robot a specific task. All the behaviours are manually designed to respond to specific stimuli. For a given task, the behaviours must be engineered and combined to provide the complex interactions with the world. Therefore the success of the behaviours depends on the competence of their designer. (Schoppers, 1987) proposed a “universal plan” which generates appropriate actions in unpredictable environments. The activation of parts of the plan depends on the environment. This method relies on extensive knowledge of the reactive components about the analysis of the robot world before running. In (Gat, 1992) a version of the subsumption architecture was used in which higher levels provide information or advice to the lower layers. A sequencer builds up chains of primitive behaviours using advice from a planning system. In (Lyons and Hendriks, 1995) the deliberative component was allowed to adapt the reactive component on-line. Their planner, which has a model of the environment and the knowledge of the reactive components, was used to tune the performance of the reactive agents, which were able to trigger independently of the planner. However, it was shown that this may provide problems in reacting quickly to sudden changes in the environment and still requires knowledge of the real environment along with the problem of the designer determining appropriate behaviours. In this thesis the proposed control architecture (CAROS) is introduced to overcome some shortcomings of the pure behaviour-based architecture, especially where modularity and task execution capability are concerned. The proposed control system is developed

incrementally. The overall control system is straightforward but the system exhibits reactivity and a great degree of autonomy in a completely unknown and unstructured environments.

2.6 Distributed artificial intelligence and distributed systems

The field of distributed artificial intelligence (DAI) deals with multi-agent interactions and concerns itself with the study of distributed systems of intelligent agents. According to (Ferber, 1999) the following lines of reasoning can explain the need of distributed intelligence: *Problems are physically distributed*: complex problems are often physically distributed such as transportation network or traffic management. *Problems are widely distributed and heterogeneous in functional terms*: for instance, a formula one car requires a large number of experts to perfect its design. All these experts integrate their knowledge to try to make the best possible car. *The complexity of problems demands a local point of view*: when problems are too complicated to be solved and analysed as a whole, solutions based on local approaches often allow them to be solved more quickly. *Systems should be adaptive to changes in the environment*: it is widely claimed that regarding system complexity is no longer enough to design efficient and accurate systems, but systems should be able to adapt to changes in the context of operations. Therefore, the field of distributed artificial intelligence is highly relevant to multi-robot and co-operative systems.

According to (Bond and Gasser, 1988) distributed artificial intelligence is defined as the sub-field of artificial intelligence concerned with concurrency in artificial intelligence computations, at many levels. (Rosenschein, 1993) has divided DAI into two sub-fields: Distributed problem solving (DPS) and multi-agent systems (MAS) as shown in Figure 2-3.

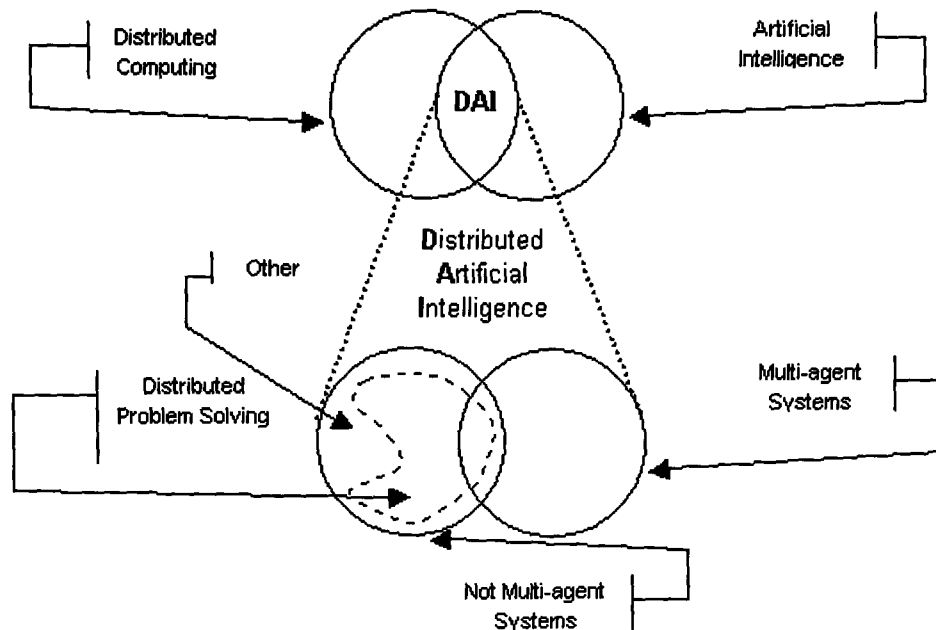


Figure 2-3 Origin and sub-fields of distributed artificial intelligence

(Cao *et al*, 1997) has defined the field of DPS as three possible overlapping phases, problem decomposition (task allocation), subproblem solution and solution synthesis. The first phase has attracted the most researchers in the field of DAI, (Asama *et al*, 1989) and (Ozaki *et al*, 1997). For instance, (Decker and Lesser, 1993) have addressed the task of fast co-ordination and reorganisation of agents on a distributed sensor network with the goal of increasing system performance. DPS deals with centrally designed systems solving global problems using frameworks for co-operative behaviour between willing agents. In contrast, multi-agent systems deals with heterogeneous, not necessarily centrally agents faced with the goal of utility-maximising coexistence using frameworks to enforce co-operation between potentially incompatible agents. The group of these heterogeneous agents can form a collective behaviour with potentially conflicting goals. The earliest works in the field of multi-agent systems are presented in the contributions of (Rosenschein, 1982), (Georgeff, 1983) and (Genesereth *et al*, 1986).

The majority of the research work on multi-agent systems begins in the early 1990s, which seems to be theoretical and in an abstract domain. A common underlining assumption is that although the agents may be selfish, they are rational and highly deliberative. In that period examples of work in MAS includes (Miceli and Cesta, 1993), (Kraus, 1993) and (Durfée *et al*, 1993). The latter present game theory and different approaches of artificial intelligence in order to deal with rational agents. As previously mentioned until that period the majority of the work in MAS is purely theoretical and deals with the difficulty of multi-agent planning and control in abstract environments. In the late 1990s multi-agent systems become new field in control engineering. Researchers started to consider how the approach to conventional control theory could be replaced by a multi-agents system methodology. Chapter three illustrates the main research methodology adopted in this thesis; background information regarding multi-agent systems in control engineering and in agent control architectures is presented in that chapter.

According to the aforementioned literature review on distributed artificial intelligence it becomes quite clear that any kind of multiple-robot system can be a unique or special case of distributed system. The methodology of the field of distributed systems can therefore be a very good source for solutions and new ideas in multiple-robot systems. For instance (Beni, 1988) describes cellular robotics as a subject that belongs to the field of distributed computing shown in Figure 2-3. However he has accepted that distributed computing can be applied successfully in theoretical bases which means that further progress is vital in order to extent distributed computing capabilities into multiple robot systems. In (Cao *et al*, 1997) an extensive list of references regarding research work associated with the field of distributed computing can be found. Most of these research works focus on deadlock detection, resource allocation, pattern generation and task allocation. In section 2.4 was mentioned that in terms of co-operation and communication, most of the research work on multiple robots lies at two ends of the spectrum (explicit and implicit communication). Research works, which evolve explicit communication, can take an advantage of the techniques used in computer networks. For instance, networking

issues can be applied successfully to multi-robot systems if explicit communication is assumed. Some of the research work in this field include (Weiser, 1993) and (Badrinath *et al*, 1994). As stated by (Cao *et al*, 1997) distributed control is a promising framework for the co-ordination of multiple robots. In an ideal scenario, maximal fault tolerance is possible, modelling of the other agents can be avoided, and each agent can be controlled by a very simple mechanism.

2.7 Intelligent control and its approaches

Intelligent control, as a discipline, has certainly been one of the main growth areas in the field of control systems over the last 5-10 years. Although the topic is relatively new in itself, a number of other research areas, some of them well-established, have effectively been swallowed up under the overall intelligent control umbrella. Intelligent control was first proposed by (Fu, 1971) and was defined as an approach to generate control actions by employing aspects of artificial intelligence, operations research and automatic control systems (Saridis and Valanidis, 1988). In the book of (Harris and Billings, 1985) it is shown that in the 1970s and 1980s flexible control systems were developed that could adapt to plant changes as they occurred. Rapid improvements in computer capabilities allowed for on-line, real-time control where previously it had quite simply not been possible. Initially adaptive, self-tuning controllers were designed, often being based, in some sense, on the classical Kalman filtering techniques of previous years. Since that time adaptation has turned more to learning and control has become more task oriented. According to (Warwick, 1998) and (Roberts, 1999) the term “intelligent control” means a wide variety of things to different people. However, as with any relatively new topic of study it suffers from a considerable amount of hype and terminology abuse. Currently one of the most popular definition of intelligent control is given by (Harris, 1994), who suggest that “intelligent control is defined as an approach to generate control actions by employing aspects of artificial intelligence, operations research, automatic control systems and computer science”. Figure 2-4 illustrates the inter-relationships between the various techniques that are utilised in intelligent control and the functionality and infrastructure that they attempt to incorporate.

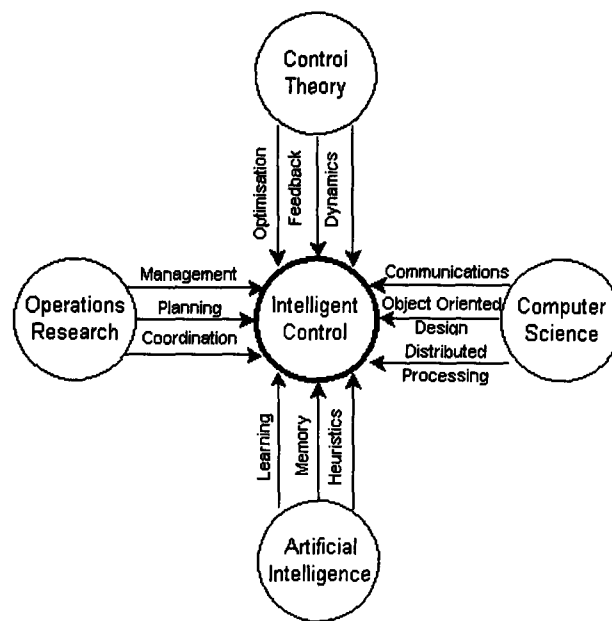


Figure 2-4 Techniques employed in intelligent control (after (Harris, 1994))

(Harris, 1994) also suggests that the main methodologies used in intelligent control are knowledge based systems (KBS), fuzzy logic and neural networks as shown in Figure 2-5. In the work presented in this thesis both fuzzy logic and neural network methodologies are used. The main design methodology of both topics is presented in chapter three.

The concept of fuzzy logic was introduced to model human reasoning by giving definitions to vague terms and allowing several rules in the rule base to interact with varying degrees of belief. It is important to note that it is irrelevant whether or not humans store knowledge in this form, what is important is that fuzzy logic allows the creation of rule base systems with vague terms using interacting rules which have the property of generalisation.

Artificial neural networks (ANN) and neural engineering/computing in the wide sense are among today's most rapidly developing scientific disciplines. ANN are parallel computational models that consist mainly of interconnected adaptive processing units. These networks are

considered fine-grained parallel implementation of non-linear dynamic and static systems. An ANN is an abstract simulation of real nervous system that contains a collection of processing units or processing elements communicating with each other via axon connections. Such a model resembles the axons and dendrites of the nervous system. Because of its self-organising and adaptive nature, the model provides a new parallel and distributed paradigm that has the potential to be more robust and user-friendly than traditional schemes.

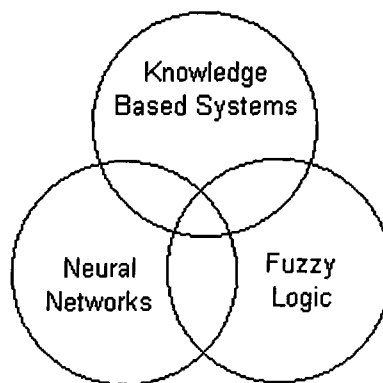


Figure 2-5 Methodologies used in intelligent control (after (Harris, 1994))

2.7.1 Modelling, identification and control of dynamic systems

The establishment of an input-output model for a process is very important in systems engineering. Many deterministic and stochastic methods have been proposed to derive acceptable mathematical models for both continuous-time and discrete-time processes. However, in the modelling of complicated processes, precise mathematical models may fail to give satisfactory results. Some of the modelling, identification and control approaches for dynamic or static systems that been researched during the last two-three decades are based on methodologies that are used in intelligent control (Kichert and Mamdani, 1978), (Sugeno and Kang, 1986), (Yager and Filev, 1994c), (Jang and Sun, 1995), (Hellendoorn and Driankov, 1997) and (Norgaard *et al*, 2000).

Traditionally, modelling is seen as a conjunction of a thorough understanding of the system's nature and behaviour, and a suitable mathematical treatment that leads to a usable model. This approach is usually termed "white box" (physical, mechanistic, first-principle) modelling. In practice, however, when complex and poorly understood systems are considered, the requirement for a good understanding of the physical background of the system proves to be a severe limiting factor. The difficulties that can arise in conventional "white-box" modelling approaches appear from poor understanding of the underlying phenomena, inaccurate values of various process parameters, or from the complexity of the resulting model. A complete understanding of the underlying mechanisms is virtually impossible for a majority of real systems. However, gathering an acceptable degree of knowledge needed for physical modelling may be difficult, time-consuming and an expensive task. Even if the structure of the model is determined, a major problem of obtaining accurate values for the parameters remains. It is the task of system identification to estimate the parameters from data measured. Identification methods have been developed to a mature level, mostly, for linear systems. Most real systems are, however, both non-linear and dynamic and can be approximated by local models.

The accuracy of mathematical models is based on how good are the approximations of the mathematical functions that are used to describe the system's characteristics under study. If the model is not accurate enough, the subsequent steps of analysis, prediction and controller synthesis, cannot be successful. However, there is an obvious trade-off between the necessary accuracy of the model and its complexity. Models should provide information at the most relevant level of precision (abstraction), suppressing unnecessary details when appropriate. If the model is too simple, it cannot properly represent the characteristics of the system and does not serve its purpose. However, the model should not be too complex if it is to be practically useful.

Fuzzy modelling is a method of describing the characteristics of a system using fuzzy rules and is a topic that has been studied extensively in recent years mostly as a problem of function approximation instead of a problem of knowledge acquisition (Takagi and Sugeno, 1985) and (Jang *et al*, 1997). Compared to other “intelligent” modelling techniques (Haykin, 1999), fuzzy systems provide a more transparent representation of the non-linear dynamic systems under study, and can also be given a linguistic interpretation in the form of rules. Moreover, fuzzy sets serve as a smooth interface between qualitative variables involved in the rules and numerical domains of the inputs and outputs of the model (Akkizidis and Roberts, 1998). The rule-based nature of fuzzy models allows the use of information expressed in the form of natural language statements, and makes the models transparent to interpretation and analysis. At the same time, at the computational level, fuzzy models can be regarded as flexible mathematical structures, similar to neural networks, that can approximate a large class of non-linear system to a desired degree of accuracy (Kosko, 1992) and (Wang, 1997). This duality allows qualitative knowledge to be combined with quantitative data. Finally, it can be said that the use of linguistic qualitative terms in the rules can be regarded as a kind of information quantisation. Thus, depending on the number of qualitative values considered, models at different levels of abstraction and accuracy can be developed for a given system. Each of the models may serve a different purpose such as prediction, controller design, monitoring.

Another approach to modelling, identification and control of dynamic systems is to use some sufficiently general “black-box” structures, such as, Artificial Neural Networks (ANN) (Picton, 1998), used as a general function approximator. The modelling problem is then that of obtaining an appropriate structure of the approximator, in order to correctly capture the dynamics and the non-linearity of the system. In “black-box” modelling, the structure of the model is hardly related to the structure of the real system. The identification problem consists of estimating the parameters in the model. If representative system data is available, “black-box” models usually can be developed quite easily, without requiring system-specific knowledge. A severe drawback

of this approach is that the structure and parameters of these types of models usually do not have any physical significance. Additionally such models cannot be used for analysing the internal system's behaviour otherwise than by numerical simulation. Finally, it is neither possible to use prior knowledge to initialise the network, nor can its final state be interpreted in terms of rules.

The drawback of the conventional "white-box" and "black-box" techniques in modelling non-linear system is their trade-off between accuracy and knowledge acquisition as well as that they are based mostly on quantitative mathematical techniques. The weakness of the traditional quantitative techniques to adequately describe complex systems was summarised in the well-known principle of incompatibility, formulated by (Zadeh, 1973). This principle states *that "as the complexity of a system increases, our ability to make precise and yet significant statements about its behaviour diminishes, until a threshold is reached beyond which precision and significance (or relevance) become almost mutually exclusive characteristics"*.

2.8 Mobile robot navigation using methodologies from intelligent control

One of the most difficult challenges in mobile robotics is autonomous navigation in a real world. A real world can change suddenly, so a robot in this environment easily makes wrong estimates of which events can happen and which events can produce unwanted side effects. In order to overcome such uncertain and drastic changes, methodologies from intelligent control have been employed. According to (Chatila, 1995) navigation is in general an incremental process that can be summarised in four main steps: *Environment perception and modelling*: any motion requires a representation of the local environment at least, and often a more global knowledge. *Localisation*: the robot needs to know where it is with respect to its environment and goal. *Motion decision and planning*: the robot has to decide where or which way to go, locally or at the longer term, and possibly compute a trajectory. *Motion execution*: the commands corresponding to the motion decisions are executed by control processes possibly

sensor-based and using environment features. The reminder of this section presents a literature review in the area of mobile robot navigation, focusing on intelligent systems techniques for navigation control. The literature review in this section is closely related with the work presented in chapter six in which robot behaviours are implemented using fuzzy, neural and stateflow for navigation control. Therefore the survey in this section is focused on mobile robot navigation using either fuzzy or neural control.

2.8.1 Mobile robot navigation using fuzzy logic

Fuzzy logic has been successfully used in various knowledge-based systems to control real-time decision-making in the area of command and control in environments where no mathematical model can be applied with efficiency. As previously mentioned, robot navigation in unknown environments is a very complex and difficult task, because the large amount of imprecise and ambiguous information that has to be considered. In section 2.7.1 a discussion was based on how human knowledge can satisfactorily deal with such information in an efficient manner. Therefore human knowledge gained through experience can be modelled to control real-time navigation systems. The knowledge provided by the human can be represented as IF-THEN rules as has been modelled in expert systems where the importance has been shown for an efficient inference mechanism to manipulate the rule base. In this context, in addition to the imprecision and uncertainty of the information perceived from the environment, other sources of imprecision/uncertainty have to be considered such as the reasoning process and rule description. Under these circumstances the quality of the decision is strongly dependent of fuzzy models that will be implemented to control the movements of the robot. In the following, examples of research work relating to fuzzy logic for the navigation of mobile robot(s) are presented.

The use of fuzzy logic in robot navigation begins in late 1980s. (Takeuchi *et al*, 1988) discussed fuzzy control of a mobile robot for obstacle avoidance in an indoor environment. Processing the

floor image in front of the robot, which was obtained with a CCD camera, derived the inputs to the fuzzy controller. Computer simulations modified fuzzy control rules, which were derived on the basis of a human's driving actions. In (Ciliz and Isik, 1989) a rule-based approach for the motion control for an autonomous mobile robot was discussed. The assumed inexactness in world description is represented by fuzzy membership functions, and a state space is discretised into a linguistic vocabulary. Fuzzy motion control rules that have been experimentally derived are then used in fuzzy inference mechanism to give the final control command to robot actuators. The work of (Rosa and Garciaalegre, 1990) follows a similar approach.

2.8.1.1 The 1990s: The peak of fuzzy logic in mobile robot navigation

(Martinez *et al*, 1994) have considered a problem, which consists of achieving sensor-based motion control of a mobile robot among obstacles in structured and/or unstructured environments with collision-free motion as the priority. They used fuzzy logic to implement the approximate reasoning necessary for handling the uncertainty inherent in the collision avoidance problem. Navigation of a mobile robot using fuzzy logic has also been discussed by (Li, 1994b), (Li, 1994a) and (Li and He, 1994), they used ultrasonic sensors and fuzzy logic in order to navigate a single mobile robot in an unknown environment. The output from their control scheme is command for the speed control unit of two rear wheels of a mobile robot. They have shown simulation results of the proposed method. In (Lee and Wang, 1994) a fuzzy logic approach was proposed for a single robot navigation. The authors stated that their approach needed few modifications if their proposed algorithm was to be used in navigation of multiple robots. In (Beaufrere and Zeghloul, 1995a) and (Beaufrere and Zeghloul, 1995b) a fuzzy logic controller was developed using information from a few ultrasonic sensors to control the navigation of mobile robot in dynamic environment. Their approach reduces the large quantity and variety of sensors usually used in autonomous vehicles and at the same time handles uncertain and noisy data. The problem with their approach is that they do not control the robot's velocity, which has not been addressed by many researchers. During the same period

the research work of (Lee, 1995), (Czarnecki *et al*, 1995) and (Maaref and Barret, 1995) summarises the contributions of fuzzy logic in robot navigation.

In the late 1990s the research effort in mobile robot navigation using fuzzy logic increased dramatically. Researchers investigated different solutions to solve the navigation problem of the mobile robot, in terms of robot behaviours. The behaviour of a single robot during its movements towards a target, in an open field with simple obstacles, has been studied by (Xu and Tso, 1996). They worked out two types of reaction strategies. The first is based on human experience whereas the second is based on reaction rules. Their technique has only been demonstrated in simulation and was proved to be suitable for the navigation of a single robot, but not multiple mobile robots. (Ramirez-Serrano and Boumedine, 1996) study robot navigation based on a fuzzy logic controller, which was designed in order to deal with the uncertainty and ambiguity of the information the system receives. They used a set of seven ultrasonic sensors in order to perceive the environment for the navigation of a single mobile robot. A similar approach is adopted by (Gasos and Martin, 1996) in which single robot navigation is considered using sensor observations. Their approach is mainly based on indoor environments. The research work of (Miyata *et al*, 1996) considers single robot navigation with a primary goal a parallel parking. Their method cannot be applied in multiple robot navigation and again only simulation results are available.

(Lin and Wang, 1997) proposed a fuzzy approach to collision avoidance for automated guided vehicle navigation. Their work was based in sensor modelling and trap recovery. The first topic is concerned with finding the minimum number of sensor used and their optimal arrangement. The fuzzy approach to robot trap recovery is demonstrated in simulation. (Benreguiieg *et al*, 1997) developed a fuzzy navigator, which integrates heuristics copying of human behaviour. They claim that their approach provides several contributions such as a low sensitivity to erroneous or inaccurate measures. In addition if the inputs of the controllers are normalised, an

effective portability on various platforms is possible. In order to demonstrate the advantage of their approach the same fuzzy navigator is implemented on two different mobile robots (Khepera and RMI). The research work of (Zhang *et al*, 1997) is concerned with fuzzy logic based reactive navigation for a mobile robot operating in an unknown environment. They provide a steering command enabling a mobile robot to avoid collision with obstacles. At that time the contributions of (Castellano *et al*, 1997), (Kam *et al*, 1997), (Jagannathan, 1997), (Oriolo *et al*, 1997) and (Hoffmann and Pfister, 1997) are reported in the literature.

(Godjevac, 1998) used fuzzy logic to implement linguistic rules for navigation of a mobile robot. He demonstrated his approach with a navigation of a single mobile robot in a simple environment. He did not consider multiple robot navigation. (Oriolo *et al*, 1998) present an algorithmic solution method for the problem of autonomous robot motion in unknown environments. Their approach is based on the alternate execution of two fundamental processes: map building and navigation. In the former, range measures are collected through the robot sensors and processed in order to build a local representation of the surrounding area. This representation is then integrated in the global map so far reconstructed by filtering out insufficient or conflicting information. In the navigation phase, A*-based planner generates a local path from the current robot position to the goal. The robot follows the path up to the boundary of the explored area, terminating its motion if unexpected obstacles are encountered. The most peculiar aspects of their method are the use of fuzzy logic for the efficient building and modification of the environment map, and the iterative application of A*, a complete planning algorithm which takes full advantage of local information. Experimental results show the real-time performance of the proposed method, both in static and moderately dynamic environments using a single mobile robot (NOMAD 200). (Gasos and Rosetti, 1999) present a fuzzy-sets based approach to the problem of mobile robot navigation in unknown environments. Fuzzy sets are used to represent the uncertainty that is inherent to the perception of the environment through the robot sensors. This uncertainty is then propagated in the process of

map building so that not only a plausible spatial layout of the environment, but also the confidence on this layout, is obtained. The initial map built by the robot is then used for self localization as it continues navigating in the same environment. The new information collected by the sensors is matched to the initial map and the transformation that brings them together is used to correct and bound the dead-reckoning errors. The approach is illustrated by experiments in office environments. In the paper of (Fukuda and Kubota, 1999) a fuzzy-based intelligent robotic system is presented. The paper, proposed a robotic system with "structured intelligence." The authors focused on a mobile robotic system with a fuzzy controller and proposed a sensory network that allowed the robot to pre-perceive its environment. The effectiveness of the proposed method is demonstrated through computer simulations of collision avoidance and path-planning problems.

2.8.1.2 Early 2000: Fuzzy logic in mobile robot navigation continues to grow

(Song and Sheen, 2000) presents a pattern recognition approach to reactive navigation of a mobile robot. A heuristic fuzzy-neuro network is developed for pattern-mapping between quantised ultrasonic sensory data and velocity commands to the robot. The design goal was to enable an autonomous mobile robot to navigate safely and efficiently to a target position in a previously unknown environment. Useful heuristic rules were combined with the fuzzy Kohonen clustering network (FKCN) to build the desired mapping between perception and motion. (Seraji, 2000) introduced Fuzzy Traversability Index as a new and simple measure for quantifying the ease of traversal of natural terrains by field mobile robots. This index provides a simple means for incorporating the terrain quality data into the robot navigation strategy and is used for terrain-based navigation of field mobile robots. A set of fuzzy navigation rules was developed using the Fuzzy Traversability Index to guide the robot toward the safest and the most traversable terrain. In addition, another set of fuzzy rules was developed to drive the robot from its initial position to a user-specified goal position. These two rule sets were integrated in a two-stage procedure for autonomous robot navigation without a *priori* map-based knowledge

about the environment. Three simulation studies were presented to demonstrate the capability of the mobile robot to reach the goal safely while avoiding impassable terrains. (Xu and Lu, 2000) studied the cause of the limit cycle (the robot wanders indefinitely in a loop in the course of navigation in unknown environment) using fuzzy logic. The main features of the proposed strategy were compared with other approaches for handling the local trapping problems. Efficiency and effectiveness of the proposed approach were verified through simulation and experiments.

(Nanayakkara *et al*, 2001) presents an approach for evolving optimum behaviours for a nonholonomic mobile robot in a class of dynamic environments. A new evolutionary algorithm reflecting some powerful features in the natural evolutionary process to have flexibility to deal with changes in the environment is used to evolve optimum behaviours. Furthermore, a fuzzy set based multi-objective fitness evaluation function is adopted in the evolutionary algorithm. The multi-objective evaluation function is designed so that it allows incorporating complex linguistic features that a human observer would desire in the behaviours of the mobile robot movements. The authors illustrate the effectiveness of the proposed method in simulation. (Mucientes *et al*, 2001) described a fuzzy control system for navigation and obstacle avoidance by a mobile robot. The control system has over 117 rules, which reflects the complexity of the problem to be tackled. The controller has been subjected to an exhaustive validation process and simulation results were shown. (Tsourveloudis *et al*, 2001) proposed an electrostatic potential field (EPF) path planner, which is combined with a two-layered fuzzy logic inference engine and implemented for real-time mobile robot navigation in a 2-D dynamic environment. The environment is first mapped into a resistor network; an electrostatic potential field is then created through current injection into the network. The path of maximum current through the network corresponds to the approximately optimum path in the environment. The first layer of the fuzzy logic inference engine performs sensor fusion from sensor readings into a fuzzy variable, collision, providing information about possible collisions in four directions, front,

back, left, and right. The second layer guarantees collision avoidance with dynamic obstacles while following the trajectory generated by the electrostatic potential field. The proposed approach was experimentally tested using a mobile robot.

In (Tunstel *et al*, 2002) an approach to hierarchical control design and synthesis for the case where the collection of subsystems is comprised of fuzzy logic controllers and fuzzy knowledge-based decision systems is presented. The approach is used to implement hierarchical behaviour-based controllers for autonomous navigation of one or more mobile robots. Theoretical details of the approach are presented, followed by discussions of practical design and implementation issues. (Maaref and Barret, 2002) considered the problem of the navigation of a mobile robot either in an unknown indoor environment or in a partially known one. A navigation method in an unknown environment is based on the combination of elementary behaviours. Most of these behaviours are achieved by means of fuzzy inference systems. In the case of a partially known environment, a hybrid method is used in order to exploit the advantages of global and local navigation strategies. The co-ordination of these strategies is based on a fuzzy inference system by an on-line comparison between the real scene and a memorized one. The planning of the itinerary is achieved by visibility graph and A* algorithm. (Barbera and Skarmeta, 2002) considered the use of fuzzy behaviours in the field of autonomous mobile robots. They address the problem of conflicts between the different behaviours that compete with each other to take control of the robot using learning techniques to efficiently co-ordinate them. They used fuzzy rules to perform such fusion.

2.8.2 Mobile robot navigation using neural networks

Interest in robot navigation using neural networks has increased recently due partly to some significant breakthroughs in research in learning and training algorithms. Advances in computer hardware technology that made neural network implementation faster and more efficient have

also contributed to the progress in research and development in neural networks for mobile robot navigation.

Humans do not employ an accurate spatial environment model to fulfil a predefined navigation task (at least not in terms of a co-ordinate system). They use only some abstract, symbolic world knowledge to plan a route. This symbolic route description defines the necessary actions to be performed at selected, classifiable places in order to pilot a vehicle from a starting point to a destination. In addition, the steering and low-level navigation behaviour of humans, like obstacle avoidance, is based on experience and skills rather than high-level planning procedures. Human travelling performance can be seen as a small set of basic visual guidance activities, which are adapted to the current, specific environment configurations. Neural networks have been shown to be successful in emulating the human behaviour. In the following, some examples of research work relating to neural networks for the navigation of mobile robot(s) are presented.

2.8.2.1 The 1990s: The beginning of neural networks in mobile robot navigation

The use of neural networks in mobile robot navigation begins in middle 1990s with the research works of (Tani and Fukumura, 1994), (Sethi and Yu, 1994) and (Ortega and Camacho, 1994). In (Pal and Kar, 1996) a neural network was trained to navigate a mobile robot in simulation with a finite turning radius. Simulation studies were also carried out by (Kodaira *et al*, 1996) in which an intelligent control algorithm for mobile robot navigation was presented. The feasibility of using neural networks for camera localization and mobile robot control was investigated by (Choi and Oh, 1997). Their technique has been tested and compared through both simulation and real-time experiments and was shown to yield more precise localization than analytic approaches. Furthermore, this neural localisation method is also shown to be directly applicable to the navigation control of an experimental mobile robot along the hallway purely guided by a dark wall strip. (Yun *et al*, 1997) have presented a neural network based path planning

algorithm for a single mobile robot. Their method integrates both global and local path planning and has been demonstrated through simulation on a known environment.

In (Chohra *et al*, 1998) a neural navigation approach essentially based on pattern classification to acquire target localization and obstacle avoidance behaviours was suggested. Simulation results displayed the ability of the neural approach to provide a mobile robot with capability to intelligently navigate in a partially structured environment. (Zhou *et al*, 1998) discussed the development of an associative, neural network as an on-line algorithm to train and control a fire-fighting robot. Learning is externally supervised with encoded target actions. The robot acquires basic navigation skills as well as the ability to detect a fire and to extinguish it. (Floreano and Mondada, 1998) described a methodology for evolving neuro-controllers of autonomous mobile robots without human intervention. The implications, of their methodology in engineering, biology, cognitive science and artificial life are discussed. (Gutiérrez-Osuna *et al*, 1998) used neural network techniques to compute the location of a mobile robot with respect to the obstacles around it. The knowledge about its position helps the robot to navigate in an unknown environment. The method is only suitable for navigation of a single mobile robot. (Quoy *et al*, 1999) present a neural model for the control of a robot, which is based on two structures. The first one enables visual navigation using landmarks for use in unknown and changing environments. The second structure enables building a proximity map of the environment. Using this map, the robot may successfully reach different goals linked to different motivations and solve various types of action selection problems. (Kassim and Kumar Bvk, 1999) discussed the navigation of single robot using neural network techniques. Their paper described a neural network called the wave expansion neural network (WENN) and shows that it is capable of developing a variety of artificial potential fields that are useful for path planning. The discretised environment including information about the target configuration (position and orientations) and the obstacles are applied to the WENN as input. Activity is then propagated in the form of waves throughout the WENN neural field. The research work of (Howard and

kilchen, 1999) also uses neural network techniques in which the mobile robot can plan its path in two degrees of freedom and avoid obstacles in a simple environment, although environmental uncertainty was not considered. In (Rivals, 1999) a piloting module of a four-wheel-drive autonomous vehicle, whose implementation relies entirely on neural network, was introduced. The authors showed how neural networks can be advantageously used for navigation of a single mobile robot. A similar approach was adopted by (Jebric *et al*, 1999) in which a mobile robot using neural network is able to find the target in an unknown environment.

2.8.2.2 Early 2000: More challenges remain

(Yang and Meng, 2000) discussed a biologically inspired neural network approach to real-time collision-free motion planning of mobile robots or robot manipulators in a non-stationary environment. The real-time robot motion is planned through the dynamic activity landscape of the neural network without a prior knowledge of the dynamic environment, without explicitly searching over the free workspace or the collision paths, and without any learning procedures. The effectiveness and efficiency of the proposed approach were demonstrated through simulation studies. In the paper of (Araujo and De Almeida, 2000) a method for mobile robot navigation in an unknown world using neural network is presented. The learning approach is used to construct a world model, and to learn to navigate from a starting position to a known goal region. Simulation and real-robot results obtained with a Nomad 200 mobile robot were presented to demonstrate the effectiveness of the discussed method. Quantitative results demonstrate the exploration and planning improvements of the proposed navigation approach. (Marichal *et al*, 2001) presented work in which neural network is used for optimisation of fuzzy system for guidance of a mobile robot towards the target. A set of fuzzy rules is optimised according to different criteria. They verified their proposed approach by implementation in two mobile robots. In (Ayrulu and Barshan, 2001) a study was undertaken to investigate the processing of sonar signals using neural networks for robust differentiation of commonly encountered features in indoor robot environments. Their work can find application in areas

where recognition of patterns hidden in sonar signals is required. Some examples are system control based on acoustic signal detection and identification, map building, navigation, obstacle avoidance, and target-tracking applications for mobile robots and other intelligent systems. (Hamze and Clark, 2001) described a view-based mobile robot navigation system relying on self-organizing neural networks. A sequence of view images from a test route was obtained, pre-processed, and used to train a system of self-organizing maps. In (Ma *et al*, 2001) a hybrid intelligent method including fuzzy inference and neural network was presented for real-time self-reaction of a mobile robot in unknown environments. Their method can be used to control a mobile robot based on the present motion situations of the robot in real-time.

(Mbede *et al*, 2002) present an integration of fuzzy local planner and modified Elman neural networks (MENN) approximation-based computed-torque controller for motion control of autonomous robot in dynamic and partially known environments containing moving obstacles. The purpose of the controller, which is designed as a Neuro-fuzzy controller, is to generate the commands for the servo-systems of the robot so it may choose its way to its goal autonomously, while reacting in real-time to unexpected events. The controller demonstrates good performance in adaptation to changes in robot dynamics. The paper of (Zalama *et al*, 2002) describe a neural network model for the reactive behavioural navigation of a mobile robot. From the information received through the sensors the robot can navigate, through a competitive neural network. The robot is able to develop a control strategy depending on sensor information and learning operation. The proposed method did not consider multiple robot navigation. Finally (Berlanga *et al*, 2002) show a new method, in order to learn weights of a neural network controller in autonomous robots called Uniform Co-evolution. The introduction of their method allows the environment to be evolved in the process of learning a general behaviour able to solve problems in different conditions. In particular the proposed method is used to learn reactive robot behaviour for navigation and collision avoidance. Again, the authors have not reported multiple robot navigation.

2.9 Discussion

This chapter details several of the principle study sources that have inspired or paralleled the development of the research work in this thesis. As discussed in section 2.2 familiarities with certain scientific topics facilitates research in which various complex problems can be solve. The traditional way to problem solving was based on the good knowledge and expertise in a specific subject. This has been shown to be insufficient for the development of new tools and techniques. This is the main reason why the filed of artificial life played so important role in the development of recent research works. Scientists endeavour to merge different fields and certainly this is a promising approach to the research development. As this thesis is concerned with the development of modelling, identification and control techniques for the development of architecture for autonomous mobile robots using hybrid approaches some related research works and issues have been reviewed and discussed in this chapter.

In sections 2.3 and 2.4 research work based on co-operative robots and control of multiple robotic systems was discussed. Considering the literature review, it becomes clear that the problem of multi-robot control is challenging and also great motivation for future research. Contributions of related work show that development of control with reliable behaviour with more than one robot co-existing in the same environment still remains a very difficult task. However, it has been almost established that the solution to the problem lies where centralised, decentralised or hybrid control is considered. Recent research works shown that gradually the research community has abandoned the centralised approach as the number of limitations overcome the number of the advantages. In section 2.5 the concept of behaviour-based control was discussed with a number of related contributions. Despite the fact that behaviour-based control is one of the greatest revolutions in robot control, still several issues remain unanswered which generate motivation for further research. Issues like, robot behaviour design, modularity and co-ordination is still of interest. Potential solution to solve these issues can be found within the field of distributed artificial intelligent and distributed systems.

The field of distributed artificial intelligent and distributed system seems to be very promising framework in the research area of development and control of co-operative systems. The literature review shows that the research work of distributed artificial intelligent is still in theoretical stage but it is believed that very soon practical implementations will take place. Already it is shown that the research effort regarding multi-agent systems has well established in the development of more sophisticated software than traditional approaches. In addition, as lower-level processes (perception and actuation) are better understood and implemented, and as computational power increases, the high-level results of distributed artificial intelligence and distributed systems may become increasingly applicable to many practical applications including the control and development of multiple robots incorporating highly intelligent programs.

In section 2.7, background information related to the origin of intelligent control and its use in modelling, identification and control is given. As section 2.7 shows, intelligent control comprises three main approaches (methodologies), which are now widely accepted and established. The use of these methodologies can be successfully implemented to solve several problems in modelling, identification and control of dynamic systems incorporating a specified behaviour. The literature review has shown that fuzzy and neural approaches have been developed for modelling, identification and control of dynamic systems. However, these approaches are still shown signal of suffering with a great number of choice that the designer has to make during the development process. For instance, consider the identification and modelling of a dynamic system using neural network. There are a larger number of learning algorithms available to be used with different methodological approaches to be followed. The question is, which method should be used, and how. Therefore a development time can be dramatically increased or decreased according to a chosen approach.

Section 2.8 presents a literature review of navigation of mobile robot(s) using methodologies from intelligent control. The length of the number of the research works outlined, almost covers the past two decades. It is clear that fuzzy logic is the oldest method used in robot navigation in contrasts to neural networks, which is appears to be relatively new. Using both methodologies the majority of the research works carried out considers a single mobile robot and not navigation of multiple robots. In addition much of the contributions made focusing on robot navigation uses kinematic models and not dynamic models. The literature review also shows that very few works consider solving the problem of robot navigation using behaviours in which certainly is the most promising way. Many research contributions to robot navigation are based on both fuzzy and neural methods with the attempt to solve the problem using one or two models behaviours/models (single fuzzy or neural controller is used with large number of sensory inputs).

2.10 Summary

The work in this thesis shares motivations and goals with a number of related fields. This chapter has presented a literature review based on those fields. At the beginning of the chapter the field of artificial life is introduced. A number of contributions from artificial life based on simulations of multiple autonomous mobile robots relevant to this thesis are given. The literature review of related work based on co-operative robotics and control of multiple mobile robots (multi-agent robotics) is presented, focusing on research contributions in which mobile robots can discriminate each other from the rest of the world based on a local reactive approach. Background information of the birth and origin of the behaviour-based control is given. Advantages and disadvantages of the behaviour-based decomposition control approach in contrast with the traditional functional decomposition are discussed. The role of distributed artificial intelligence and distributed systems in the development of control architectures for multiple autonomous mobile robots is presented. Hence, the more significant sub-fields of distributed artificial intelligence are discussed in more detail as they relate directly to the work

of this thesis. The origin of intelligent control including its approaches is illustrated. Three methodologies in the field of intelligent control are presented two of which are used direct in this thesis. Recent research in modelling, identification and control of dynamic systems using methodologies from intelligent control has been reviewed with reference on related work. Robot navigation using intelligent control methods has been described. In particular the chapter has reviewed robot navigation techniques up to date, focusing on those utilising fuzzy logic and neural networks.

The next chapter presents the main research methodology adopted for the research work carried out in this thesis.

2.11 References

AKKIZIDIS, I. S. and ROBERTS, G. N. 1998. *Fuzzy Modelling and Fuzzy-Neuro Motion Control of an Autonomous Underwater Robot*. The 5th International Workshop on advanced Motion Control. pp. 641-646, Coimbra, Portugal.

ARAUJO, R. and DE ALMEIDA, A. T. 2000. Learning Both a World Model and Navigation Paths in an Unknown Environment. *Intelligent Automation and Soft Computing*, 6 (2), pp. 159-170.

ARKIN, R. C. 1992. Cooperation Without Communication: Multiagent Schema-Based Robot Navigation. *Robotic Systems*, 9 (3), pp. 351-364.

ARKIN, R. C. 1998. *Behaviour-Based Robotics*. USA: MIT Press. 0-262-01165-4.

ARKIN, R. C. and BALCH, T. 1998. Co-operative Multiagent Robotic Systems. In: eds. D. KORTENKAMP, R. P. BONASSO, and R. MURPHY. ed. *Artificial Intelligence and Mobile Robots. Case Studies of Successful Robot Systems*. USA: AAAI Press/The MIT Press, pp. 277-296.

ASAMA, H., HABIB, M. K., ENDO, I., OZAKI, K., MATSUMOTO, A. and ISHIDA, Y. 1991. *Functional Distribution Among Multiple Mobile Robots in an Autonomous and Decentralised Robot System*. Proceedings of IEEE International Conference on Robotics and Automation. pp. 1921-1926, Sacramento, California.

ASAMA, H., MATSUMOTO, Y. and ISHIDA, Y. 1989. *Design of an Autonomous and Distributed Robot System: ACTRESS*. IEEE/RSJ. pp. 283-290,

AYRULU, B. and BARSHAN, B. 2001. Neural Networks for Improved Target Differentiation and Localization With Sonar. *Neural Networks*, 14 (3), pp. 355-373.

AZARM, K. and SCHMIDT, G. 1997. A Decentralised Approach for the Conflict-Free Motion of Multiple Mobile Robots. *Advanced Robotics*, **11** (4), pp. 323-340.

BADRINATH, B. R., ACHRYA, A. and IMIELINSKI, T. 1994. *Structuring Distributed Algorithms for Mobile Hosts*. Proceedings of the 14th International Conference on Distributed Computing Systems. pp. 21-24,

BALCH, T. and ARKIN, R. C. 1998. Behavior-Based Formation Control for Multirobot Teams. *IEEE Transactions on Robotics and Automation*, **14** (6), pp. 926-939.

BARBERA, H. M. and SKARMETA, A. G. 2002. A Framework for Defining and Learning Fuzzy Behaviors for Autonomous Mobile Robots. *International Journal of Intelligent Systems*, **17** (1), pp. 1-20.

BARNES, D. P. 1996. A Behavior Synthesis Architecture for Co-operant Mobile Robots. In: eds. J. O. GRAY and D. G. CALDWELL. ed. *Advanced Robotics & Intelligent Machines*. London: The Institute of Electrical Engineers, pp. 295-314.

BEAUFRERE, B. and ZEGHLOUL, S. 1995a. A Mobile Robot Navigation Method Using a Fuzzy-Logic Approach. *Robotica*, **13**, pp. 437-448.

BEAUFRERE, B. and ZEGHLOUL, S. 1995b. Navigation Method for a Mobile Robot Using a Fuzzy Based Method - Simulation and Experimental Aspects. *International Journal of Robotics & Automation*, **10** (3), pp. 106-113.

BENI, G. 1988. *The Concept of Cellular Robotic System*. IEEE International Symposium on Intelligent Control. pp. 57-62,

BENREGUIEG, M., HOPPENOT, P., MAAREF, H., COLLE, E. and BARRET, C. 1997. Fuzzy Navigation Strategy: Application to Two Distinct Autonomous Mobile Robots. *Robotica*, **15**, pp. 609-615.

BERLANGA, A., SANCHIS, A., ISASI, P. and MOLINA, J. M. 2002. Neural Network Controller Against Environment: A Coevolutionary Approach to Generalize Robot Navigation Behavior. *Journal of Intelligent & Robotic Systems*, **33** (2), pp. 139-166.

BHATTACHARYYA, S. P., CHAPPELLAT, H. and KEEL, L. H. 1995. *Robust Control: The Parametric Approach*. USA: Prentice-Hall Inc. 0-13-781576-X.

BOND, A. H. and GASSER, L. 1988. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers.

BORESTEIN, J., EVERETT, H. R. and FENG, L. 1996. *Navigating Mobile Robots: Systems and Techniques*. USA: A K Peters Ltd. 156881058X.

BROOKS, R. A. 1986. A Robust Layered control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, **RA-2** (1), pp. 14-23.

CALLOUD, P., CHOI, W., LATOMBE, J. C., PAPE, C. L. and YIM, M. 1990. *Indoor Automation With Many Mobile Robots*. IEEE International Workshop on Intelligent Robots and Systems. pp. 67-72.

- CAO, Y. U., FUKUNAGA, A. S. and KAHNG, A. B. 1997. Cooperative Mobile Robots: Antecedents and Directions. *Autonomous Robots*, **4** (1), pp. 7-27.
- CASTELLANO, G., ATTOLICO, G. and DISTANTE, A. 1997. Automatic Generation of Fuzzy Rules for Reactive Robot Controllers. *Robotics and Autonomous Systems*, **22** (2), pp. 133-149.
- CHATILA, R. 1995. Deliberation and Reactivity in Autonomous Mobile Robots. *Robotics and Autonomous Systems*, **16** (2-4), pp. 197-211.
- CHOHRA, A., BENMEHREZ, C. and FARAH, A. 1998. Neural Navigation Approach for Intelligent Autonomous Vehicles (Iav) in Partially Structured Environments. *Applied Intelligence*, **8** (3), pp. 219-233.
- CHOI, D. H. and OH, S. Y. 1997. Real-Time Neural Network Based Camera Localization and Its Extension to Mobile Robot Control. *International Journal of Neural Systems*, **8** (3), pp. 279-293.
- CILIZ, M. K. and ISIK, C. 1989. Fuzzy Rule-Based Motion Controller for an Autonomous Mobile Robot. *Robotica*, **7**, pp. 37-42.
- CZARNEKI, C., JOHN, R. and BENNETT, S. 1995. *The Application of Fuzzy Logic To Real Time Multiple Robot Collision Avoidance*. Proceedings of the ICSC Fuzzy Logic Symposium. pp. C116-C121
- DECKER, K. and LESSER, V. 1993. *A one-shot Dynamics Co-ordination Algorithm for Distributed Sensor Networks*. Proceedings of AAAI. pp. 210-216, Washington, DC.
- DENEUBOURG, J. L., GOSS, S., FRANKS, N., FRANKS, A. S., DETRAIN, C. and CHRETIEN, L. 1990. *The Dynamics of Collective Sorting Robot - Like Ants and Ant - Like Robots*. 1st International Conference on Simulation of Adaptive Behaviour. pp. 356-363,
- DUDEK, G., JENKIN, E., MILIOS, E. and WILKES, D. 1993. *A Taxonomy for Swarm Robots*. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 441-447, Yokohama, Japan.
- DURFEE, E. H., LEE, J. and GMYTRASIEWICZ, P. J. 1993. *Overeager Reciprocal Rationality and Mixed Strategy Equilibria*. Proceedings of AAAI. pp. 225-230,
- EVERETT, H. R. 1995. *Sensors for Mobile Robots: Theory and Applications*. USA: A K Peters, Ltd. 1-56881-048-2.
- FERBER, J. 1999. *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. New York, USA: Addison Wesley Longman Inc. 0-201-36048-9.
- FLOREANO, D. and MONDADA, F. 1998. Evolutionary Neurocontrollers for Autonomous Mobile Robots. *Neural Networks*, **11** (7-8), pp. 1461-1478.
- FU, K. S. 1971. Learning Control Systems and Intelligent Control Systems: An Intersection of Artificial Intelligence and Automatic Control. *IEEE Transactions on Automatic Control*, **16** (1), pp. 70-72.

- FUKUDA, T. and KUBOTA, N. 1999. An Intelligent Robotic System Based on a Fuzzy Approach. *Proceedings of the IEEE*, **87** (9), pp. 1448-1470.
- FUKUDA, T. and NAKAGAWA, S. 1987. *A Dynamically Reconfigurable Robotic System (Concept of a system and Optimal Configurations)*. International Conference on Industrial Electronics, Control, and Instrumentation. pp. 588-595,
- GASOS, J. and MARTIN, A. 1996. A fuzzy Approach to Build Sonar Maps for Mobile Robots. *Computers in Industry*, **32** (2), pp. 151-167.
- GASOS, J. and ROSETTI, A. 1999. Uncertainty Representation for Mobile Robots: Perception, Modeling and Navigation in Unknown Environments. *Fuzzy Sets and Systems*, **107** (1), pp. 1-24.
- GAT, E. 1992. *Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robot*. Proceedings of the National Conference on Artificial Intelligence. pp. 809-815, Menlo Park, CA.
- GENESERETH, M. R., GINSBERG, M. L. and ROSENSCHEIN, J. S. 1986. *Co-operation Without Communication*. Proceedings of AAAI. pp. 51-57,
- GEORGEFF, M. 1983. *Communication & Interaction in Multi-Agent Planning*. Proc. AAAI. pp. 125-129,
- GODJEVAC, J. 1998. Fuzzy Systems and Neural Networks. *Intelligent Automation and Soft Computing*, **4** (1), pp. 27-37.
- GOLDBERG, D. and MATARIC, M. J. 1999. *Coordinating Mobile Robot Group Behavior Using A Model of Interaction Dynamics*. Proceedings of the 3rd International Conference on Autonomous Agents. Seattle, Washington.
- GUTIERRZOSUMA, R., JANET, J. A. and LUO, R. C. 1998. Modelling of Ultrasonic Range Sensors for Localisation of Autonomous Mobile Robots. *IEEE transactions on Industrial Electronics*, **45** (4), pp. 654-662.
- HAMZE, F. and CLARK, J. J. 2001. View-Based Route-Learning With Self-Organizing Neural Networks. *Image and Vision Computing*, **19** (11), pp. 753-761.
- HARRIS, C. J. 1994. *Advances in Intelligent Control*. UK: Taylor & Francis Ltd. 0-7484-0066-4.
- HARRIS, C. J. and BILLINGS, S. A. 1985. *Self-tuning Control: Applications and Methods*. Peter Peregrinus.
- HARTLEY, R. and PIPITONE, F. 1991. *Experiments with the Subsumption Architecture*. IEEE International Conference on Robotics and Automation. pp. 1652-1658, Sacramento, CA .
- HAYKIN, S. 1999. *Neural Networks: A Comprehensive Foundation*. 2nd end. USA: Prentice-Hall, Inc. 0-13-273350-1.
- HEIKKILA, T. and RONING, J. 1992. PEM-Modelling: A Framework for Designing Intelligent Robot Control. *Journal of Robotics and Mechatronics*, **4** (5), pp. 437-444.

HELLENDORRN, H. and DRIANKOV, D. 1997. *Fuzzy Model Identification, Selected Approaches*. New York: Springer-Verlag Berlin Heidelberg. 3-540-62721-9.

HOFFMANN, F. and PFISTER, G. 1997. Evolutionary design of a Fuzzy Knowledge Base for a Mobile Robot. *International Journal of Approximate Reasoning*, **17** (4), pp. 447-469.

HOWARD, A. and KILCHEN, L. 1999. Navigation Using Natural Landmarks. *Robotics and Autonomous Systems*, **26** (2), pp. 99-115.

JAGANNATHAN, S. 1997. Discrete-time Fuzzy Logic Control of a Mobile Robot with an Onboard Manipulator. *International Journal of Systems Science*, **28** (12), pp. 1195-1209.

JANG, J. S. and SUN, C. T. 1995. Neuro-fuzzy Modelling and Control. *Proceedings of the IEEE*, **83** (3), pp. 378-406.

JANG, J. S., SUN, C. T. and MIZUTANI, E. 1997. *Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence*. USA: Prentice-Hall, Inc. 0-13-261066-3.

JEBRIC, B., GROLINGER, K. and VRANJES, B. 1999. Autonomous Agent Based on Reinforcement Learning and Adaptive Shadowed Network. *Artificial Intelligence in Engineering*, **13** (2), pp. 141-157.

JUNG, D. and ZELINSKY, A. 1999. An Architecture for Distributed Cooperative Planning in a Behavior-Based Multi-Robot System. *Robotics and Automation Systems*, **26** (2-3), pp. 149-174.

KAM, M., ZHU, X. X. and KALATA, P. 1997. Sensor Fusion for Mobile Robot Navigation. *Proceedings of the IEEE*, **85** (1), pp. 108-119.

KASSIM, A. A. and KUMAR BVKV 1999. Path Planners Based on the Wave Expansion Neural Network. *Robotics and Autonomous Systems*, **26** (1), pp. 1-22.

KICHERT, W. J. M. and MAMDANI, E. H. 1978. Analysis of a Fuzzy Logic Controller. *Fuzzy Sets and Systems*, **1** (1), pp. 29-44.

KODAIRA, M., OHTOMO, T., TANAKA, A., IWATSUKI, M. and OCHUCHI, T. 1996. Obstacle Avoidance Travel Control of Robot Vehicle Using Neural Network. *systems and Computers in Japan*, **27** (12), pp. 102-112.

KORTENKAMP, D., BONASO, R. P. and MURPHY, R. 1998. *Artificial Intelligence and Mobile Robots*. USA: The MIT Press.

KOSKO, B. 1992. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. USA: Prentice-Hall International, Inc. 0-13-612334-1.

KRAUS, S. 1993. *Agents Contracting Tasks in Non-Collaborative Environment*. Proceedings of AAAI. pp. 243-248,

KUBE, C. R. and BONABEAU, E. 2000. Cooperative Transport by Ants and Robots. *Robotics and Autonomous Systems*, **30** (1-2), pp. 85-101.

KUBE, C. R. and ZHANG, H. 1993. Collective Robotics: From Social Insects to Robots. *Adaptive Behavior*, **2** (2), pp. 189-219.

- LAENGLE, T. and LUETH, T. C. 1994. *Decentralized Control of Distributed Intelligent Robots and Subsystems*. Proceedings of the IFAC Symposium on Artificial Intelligence in Real Time Control (ARTC). Valencia, Spain.
- LANGTON, C. G. 1989. *Artificial Life*. The Proceedings of the Interdisciplinary Workshop in the Synthesis and Simulations of Living Systems. pp. 1-47, Redwood City, CA.
- LEE, E. T. 1995. Applying Fuzzy-Logic to Robot Navigation. *Kybernetes*, **24** (6), pp. 38-43.
- LEE, P. S. and WANG, L. L. 1994. Collision-Avoidance by Fuzzy-Logic Control for Automated Guided Vehicle Navigation. *Journal of Robotic Systems*, **11** (8), pp. 743-760.
- LI, W. 1994a. *Fuzzy Logic Based Perception-Action Behavior Control of a Mobile Robot in Uncertain Environments*. Proc. of IEEE World Congress on Computational Intelligence FUZZ-IEEE. pp. 1626-1631,
- LI, W. 1994b. Fuzzy-Logic-Based Reactive Behavior Control of an Autonomous Mobile System in Unknown Environments. *Engineering Applications of Artificial Intelligence*, **7** (5), pp. 521-531.
- LI, W. and HE, K. Z. 1994. *Sensor-Based Robot Navigation in Uncertain Environments Using Fuzzy Logic*. ASME Inter. Computers in Engineering Conference.
- LIN, C. H. and WANG, L. L. 1997. Intelligent Collision Avoidance by Fuzzy Logic Control. *Robotics and Autonomous Systems*, **20** (1), pp. 61-83.
- LISCANO, R., MANZ, A., STUCK, E. R., FAYEK, R. E. and TIGLI, J. Y. 1995. Using a Blackboard to Integrate Multiple Activities and Achieve Strategic Reasoning for Mobile Robot Navigation. *IEEE Expert*, **10** (1), pp. 24-35.
- LUETH, T. C. and LAENGLE, T. 1994. *Task Description, Decomposition, and Allocation in a Distributed Autonomous Multi-Agent Robot System*. IEEE/RSJ Conference on Intelligent Robots and Systems.
- LYONS, D. M. and HENDRIKS, A. J. 1995. Exploiting Patterns of Interaction to Achieve Reactive Behaviour. *Artificial Intelligence*, **73** (1), pp. 117-148.
- MA, X. W., LI, X. L. and QIAO, H. 2001. Fuzzy Neural Network-Based Real-Time Self-Reaction of Mobile Robot in Unknown Environments. *Mechatronics*, **11**(8), pp. 1039-1052.
- MAAREF, H. and BARRET, C. 1995. *Fuzzy Help in Mobile Robot Navigation*. IEEE Int. Conf. on Robotics & Automation. pp. 865-871,
- MAAREF, H. and BARRET, C. 2002. Sensor-Based Navigation of a Mobile Robot in an Indoor Environment. *Robotics and Autonomous Systems*, **38** (1), pp. 1-18.
- MAES, P. and BROOKS, R. A. 1990. *Learning to Coordinate Behaviors*. AAAI. pp. 796-802, Boston.
- MARICHAL, G. N., ACOSTA, L., MORENO, L., MENDEZ, J. A., RODRIGO, J. J. and SIGUT, M. 2001. Obstacle Avoidance for a Mobile Robot: a Neuro-Fuzzy Approach. *Fuzzy Sets and Systems*, **124** (2), pp. 171-179.

MARTINEZ, A., TUNSTEL, E. and JAMSHIDI, M. 1994. Fuzzy-Logic Based Collision-Avoidance for a Mobile Robot. *Robotica*, **12**, pp. 521-527.

MATARIC, M. J. 1992. Integration of Representation Into Goal-Driven Behavior-Based Robots. *IEEE Transactions on Robotics and Automation*, **8** (3), pp. 304-312.

MBEDE, J. B., ELE, P. and XINHAN, H. 2002. Neuro-Fuzzy Dynamic Obstacle Avoidance for Autonomous Robot Manipulators. *JSME International Journal Series C - Mechanical Systems Machine Elements and Manufacturing*, **45** (1), pp. 286-297.

MICELI, M. and CESTA, A. 1993. *Strategic Social Planning: Looking for Willingness in Multi-agent Domains*. Proceedings of the 15th Annual Conference of the Cognitive Science Society. pp. 741-746,

MIYATA, H., OKHI, M., YOKUCHI, Y. and OHIKA, M. 1996. Control of the Autonomous Mobile Robot Dream-1, for a Parallel Parking. *Mathematics and Computer in Simulation*, **41** (1), pp. 129-139.

MUCIENTES, M., IGLESIAS, R., REGUEIRO, C. V., BUGARIN, A., CARINENA, P. and BARRO, S. 2001. Fuzzy Temporal Rules for Mobile Robot Guidance in Dynamic Environments. *IEEE Transactions on Systems Man and Cybernetics Part C- Applications and Reviews*, **31** (3), pp. 391-398.

MULLER, J. P. 1997. Control Architectures for Autonomous and Interacting Agents: A Survey. *Lecture Notes in Artificial Intelligence*, **1209**, pp. 1-26.

NANAYAKKARA, D. P. T., WATANABE, K., KIGUCHI, K. and IZUMI, K. 2001. Evolutionary Learning of a Fuzzy Behavior Based Controller for a Nonholonomic Mobile Robot in a Class of Dynamic Environments. *Journal of Intelligent & Robotic Systems*, **32** (3), pp. 255-277.

NOREILS, F. R. 1993. Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment. *The Int. Journal of Robotics Research*, **12** (1), pp. 79-98.

NORGAARD, M., RAWN, O., POULSEN, N. K. and HANSEN, L. K. 2000. *Neural Networks for Modelling and Control of Dynamic Systems*. United Kingdom: Springer-Verlag London Ltd. 1-85233-227-1.

ORIOLO, G., ULIVI, G. and VENDITTELI, M. 1997. Fuzzy Maps: A New Tool for Mobile Robot Perception and Planning. *Journal of Robotics Systems*, **14** (3), pp. 179-197.

ORIOLO, G., ULIVI, G. and VENDITTELLI, M. 1998. Real-Time Map Building and Navigation for Autonomous Robots in Unknown Environments. *IEEE Transactions on Systems Man and Cybernetics Part B- Cybernetics*, **28** (3), pp. 316-333.

ORTEGA, J. G. and CAMACHO, E. F. 1994. Neural Network mbpc for Mobile Robot Path Tracking. *Robotics and Computer-Integrated Manufacturing*, **11** (4), pp. 271-278.

OZAKI, K., ASAMA, H. and ENDO, I. 1997. Distributed and Cooperative Object Pushing by Multiple Mobile Robots Based on Communication. *Advanced Robotics*, **11** (5), pp. 501-517.

- PAL, P. K. and KAR, A. 1996. Sonar-Based Mobile Robot Navigation Through Supervised Learning on a Neural-Net. *Autonomous Robots*, **3** (4), pp. 355-374.
- PARKER, L. E. 1993. *Designing control Laws for Cooperative Agent Teams*. Proc. IEEE Int. Conf. on Robotics and Automation. pp. 582-587, Atlanta, GA.
- PARKER, L. E. 1998. *Distributed Control of Multi-Robot Teams: Co-operative Baton-Passing Task*. Proc. IEEE Int. Conf. on Information Systems Analysis & Synthesis. pp. 89-94,
- PARKER, L. E. 1999. Adaptive Heterogeneous Multi-Robot Teams. *Neurocomputing*, **28**, pp. 75-95.
- PICTON, P. 1998. *Neural Networks*. 2nd end. USA: Palgrave. 0-333-80287-X.
- QUOY, M., GAUSSIER, P., LEPRETRE, S. and REVEL, A. 1999. *A Neural Model for the Visual Navigation and Planning of a Mobile Robot*. HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY: Springer-Verlag Berlin.
- RAMIREZ-SERRANO, A. and BOUMEDINE, M. 1996. *Real-Time Navigation in Unknown Environments Using Fuzzy Logic and Ultrasonic Sensing*. Proceedings of the 11th International Symposium on Intelligent Control. pp. 26-31,
- RIDAO, P., BATLLET, J., AMAT, J. and ROBERTS, G. N. 1999. Recent Trends in Control Architectures for Autonomous Underwater Vehicles. *International Journal of Systems science*, **30** (9), pp. 1033-1056 .
- RIVALS, L. 1999. Modeling and Control of an Autonomous Vehicle Using Neural Networks. *Mecanique Industrielle Et Materiaux*, **52** (1), pp. 10-12.
- ROBERTS, G. N. 1999. Intelligent Mechatronics. *Engineering Science and Educational Journal*, **8** (2), pp. 66-72.
- ROSA, R. G. and GARCIAALEGRE, M. C. 1990. Fuzzy-Logic Strategies to Control an Autonomous Mobile Robot. *Cybernetics and Systems*, **21** (2-3), pp. 267-276.
- ROSENSCHEIN, J. S. 1982. *Synchronisation of Multi-agent Plans*. Proceedings of AAAI. pp. 115-119,
- ROSENSCHEIN, J. S. 1993. *Consenting Agents: Negotiation Mechanisms for Multi-agent Systems*. IJCAI. pp. 792-799,
- SARIDIS, G. N. and VALANIDIS, K. P. 1988. Analysis Design of Intelligent Machines. *Automatica*, **24** (1), pp. 123-133.
- SCHOPPERS, M. J. 1987. *Universal Plans for Reactive Robots in Unpredictable Environment*. Proceedings of 10th International Conference on Artificial Intelligence. pp. 1039-1046, Milan, Italy.
- SENEHI, M. K. and KRAMER, T. R. 1998. A Framework for Control Architectures. *International Journal of Computer Integrated Manufacturing*, **11** (4), pp. 347-363.
- SERAJI, H. 2000. Fuzzy Traversability Index: a New Concept for Terrain-Based Navigation. *Journal of Robotic Systems*, **17** (2), pp. 75-91.

SETHI, I. K. and YU, G. N. 1994. Making a Mobile Robot Learn to Determine its Location Using Neural Networks. *Pattern Recognition Letters*, **15** (4), pp. 493-502.

SHIM, H. S., KIM, H. S., JUNG, M. J., CHOI, I. H., KIM, J. H. and KIM, J. O. 1997. Designing Distributed Control Architecture for Cooperative Multi-Agent System and Its Real-Time Application to Soccer Robot. *Robotics and Autonomous Systems*, **21** (2), pp. 149-165.

SONG, K. T. and SHEEN, L. H. 2000. Heuristic Fuzzy-Neuro Network and Its Application to Reactive Navigation of a Mobile Robot. *Fuzzy Sets and Systems*, **110** (3), pp. 331-340.

STEELS, L. 1990. Cooperation Between Distributed Agents Through Self-Organisation. *Decentralized A. I.*, pp. 175-196.

SUGENO, M. and KANG, G. T. 1986. Fuzzy Modeling and Control of a Multilayer Incinerator. *Fuzzy Sets and Systems*, **18**, pp. 329-346.

TAKAGI, T. and SUGENO, M. 1985. Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics*, **15** (1), pp. 116-132.

TAKEUCHI, T., NAGAI, Y. and ENOMOTO, N. 1988. Fuzzy Control of a Mobile Robot for Obstacle Avoidance. *Information Sciences*, **45**, pp. 231-248.

TANI, J. and FUKUMURA, N. 1994. Learning Goal-Directed Sensory-Based Navigation of a Mobile Robot. *Neural Networks*, **7** (3), pp. 553-563.

TSOURVELOUDIS, N. C., VALAVANIS, K. P. and HEBERT, T. 2001. Autonomous Vehicle Navigation Utilizing Electrostatic Potential Fields and Fuzzy Logic. *IEEE Transactions on Robotics and Automation*, **17** (4), pp. 490-497.

TUNSTEL, E., DE OLIVEIRA, M. A. A. and BERMAN, S. 2002. Fuzzy Behavior Hierarchies for Multi-Robot Control. *International Journal of Intelligent Systems*, **17** (5), pp. 449-470.

VAN BREEMEN, A. J. N. 2001. *Agent-Based Multi-Controller Systems: A Design Framework for Complex Control Problems*. Ph.D. Thesis. Twente University Press.

VAN BRUSSEL, H. 1995. *Navigation Issues in Intelligent Autonomous Systems*. International Conference on Intelligent Autonomous Systems. pp. 42-52, Karlsruhe, Germany.

WANG, L. X. 1997. *Course in Fuzzy Systems and Control*. USA: Bernard Goodwin. 0-13-540882-2.

WANG, P. K. C. 1991. Navigation Strategies for Multiple Autonomous Mobile Robots Moving in Formation. *Robotic Systems*, **8** (2), pp. 177-195.

WARWICK, K. 1998. *Recent Developments in Intelligent Control*. Colloquium in Update on Developments in Intelligent Control. pp. 1-4, London, UK.

WATANABE, M., ONOGUCHI, K., KWEON, I. and KUNO, Y. 1992. *Architecture of Behaviour-Based Mobile Robot in Dynamic Environments*. IEEE International Conference on Robotics and Automation. pp. 2711-2718, Nice, France.

- WEISER, M. 1993. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, **36** (7), pp. 74-84.
- WERGER, B. B. 1999. Cooperation Without Deliberation: A Minimal Behavior-Based Approach to Multi-Robot Teams. *Artificial Intelligent*, (110), pp. 293-320.
- WILSON, M. S., KING, C. M. and HUNT, J. E. 1997. Evolving Hierarchical Robot Behaviours. *Robotics and Autonomous Systems*, **22** (3-4), pp. 215-230.
- XU, W. L. and LU, Z. K. 2000. Virtual Target Strategy in Fuzzy Navigation of Reactive Mobile Robots. *Intelligent Automation and Soft Computing*, **6** (4), pp. 303-315.
- XU, W. L. and TSO, S. K. 1996. Real-Time Self-Reaction of a Mobile Robot in Unstructured Environments Using Fuzzy Reasoning. *Engineering Applications of Artificial Intelligence*, **9** (5), pp. 475-485.
- YAGER, R. R. and FILEV, D. P. 1994c. *Essentials of Fuzzy Modelling and Control*. New York: John Wiley & Sons. 0471017612.
- YANG, S. X. and MENG, M. 2000. An Efficient Neural Network Approach to Dynamic Robot Motion Planning. *Neural Networks*, **13** (2), pp. 143-148.
- YOSHIDA, E., ARAI, T. and OTA, J. 1998. Local Communication of Multiple Mobile Robots: Design of Group Behavior for Efficient Communication. *Advanced Robotics*, **11** (8), pp. 759-779.
- YUN, W. S., CHO, D. W. and BEAL, Y. S. 1997. Dynamic Path Planning for Robot Navigation Using Sonar Mapping and Neural Networks. *ASME Transactions, Journal of Dynamic Systems and Control*, **119** (1), pp. 19-26.
- ZADEH, A. G., FAHIM, A. and ELGINDY, M. 1997. Neural Network and Fuzzy Logic Applications to Vehicle Systems: Literature Survey . *International Journal of Vehicle Design*, **18** (2), pp. 132-193.
- ZADEH, L. A. 1973. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-3**, pp. 28-44.
- ZALAMA, E., GOMEZ, J., PAUL, M. and PERAN, J. R. 2002. Adaptive Behavior Navigation of a Mobile Robot. *IEEE Transactions on Systems Man and Cybernetics Part A - Systems and Humans*, **32** (1), pp. 160-169.
- ZHANG, M. L., PENG, S. X. and MENG, Q. H. 1997. Neural Network and Fuzzy Logic Techniques Based Collision Avoidance for a Mobile Robot. *Robotica*, **15**, pp. 627-632 .
- ZHOU, Y., WILKINS, D. and COOK, R. P. 1998. Neural Network Control for a Fire-Fighting Robot. *Software-Concepts and Tools*, **19** (3), pp. 146-152.

3

Research Methodology

3.1 Introduction

The aim of this chapter is to propose, justify and present the main methodology adopted for the research work carried out in this thesis. Although there is no formal method for developing integrated solutions for advanced mobile robotic systems, in most cases, robot systems or different types of controlling robot systems are evaluated using a proof of concept technique, in which the system or the approach is shown to be capable of accomplishing a particular task. More often, two methods are compared. The comparison of those two methods may produce an inaccurate result because a more formal approach needs to be taken to choose which types of methods to compare and for what type of task.

The demand for complex control systems with or without the use of a mathematical model of the plant to be controlled and the effort to model, control, identify and build control systems architectures for autonomous mobile robots has led to recent use of intelligent control methods such as fuzzy logic, neural networks, genetic algorithms, multi-agents systems, etc. However

the effort for several simple, cheap, flexible and more fault-tolerant mobile robots rather than a single powerful robot for each separate task needs an integration of both conventional and intelligent control design techniques.

The proposed methodology for this research work can be broken into the following nine basic steps, described as follows:

1. **Conventional control design.** Automatic control has played an important role in the advance of engineering and science (Gajic and Lelic, 1996) and (Astrom and Wittenmark, 1997). In addition to its extreme significance in the steering of missiles, aircraft-autopilot systems, spacecrafts, underwater vehicles, mobile robots, automatic control has become a crucial and usually integral part of modern manufacturing and industrial processes. Design of Proportional + Integral (PI) speed controller is presented in chapter five for the MIABOT V2 mobile robot. This method was chosen as conventional control design provides good performance when mainly based upon knowledge of the process mathematical model. In addition conventional control is computational simple and fast (this is demonstrated in chapter five where comparison with fuzzy and neural controllers has been made) and in general is much more widely understood and easier to tune (only three terms are tuneable). Appendix C presents the background information of the basic control system structure with more emphasis on PID control.
2. **Optimisation.** The subject of optimisation is an interesting blend of heuristics and rigor, of theory and experiments and may be defined as the science of determining the 'best' solutions to certain mathematically defined problems, which are often models of physical reality. The non-linear control design¹ (NCD) blockset based on constrained optimisation

¹ NCD is registered trademark of the Math Works, Inc. Copyright 1993-1997. All rights reserved.

provided by MATLAB² is used in chapter four and five for tuning/optimisation of physical and control parameters. The physical parameters (moment of inertial, etc.) were identified/optimised for more accurate robot model, whereas controller parameters (PI gains) were tuned to meet desired design requirements. This method is fast, easy to use, suitable for control design of both linear and non-linear systems and gives good performance. Appendix D presents an overview of constrained optimisation and the non-linear control design tool.

3. Robust stability analysis for interval polynomials based on parametric approach.

Almost all dynamic systems depend on varying or uncertain parameters and this is certainly true for small mobile robots. For instance, consider the velocity of a mobile robot (i.e. due to the battery variations), or its mass (i.e. adding or removing components), all these parameters may vary more or less significantly within certain bounds and they influence the system dynamics. Traditional control design approaches consider a fixed operating point in which the controller (compensator) is robust enough to effectively control the plant for different operating conditions. These approaches produce good results if the parameter variations are small or the system dynamics are not too sensitive with respect to these parameters. For significant (large) parameter variations these control design methods reach their performance limits. The parametric robustness analysis approach (Kharitonov's Theorem) is adopted in chapter five as it is an easy to use method and offers fast robust stability testing and analysis based on interval polynomials. Using this method the closed-loop control system of the MIABOT V2 mobile robot is proved to be robustly stable under uncertainty in its dynamics. The interval polynomial problem was first posed by (Faedo, 1953), who attempted to solve it using the Routh-Hurwitz conditions. (Kharitonov, 1978) gave the complete solution with his theorem for real polynomials, which he then extended

² MATLAB is registered trademark of the Math Works, Inc. Copyright 1984-2000. All rights reserved.

to the complex case. Since then many papers have been published based on parametric approach regarding robust stability of uncertain plant (Siljad, 1989) and (Kontogiannis and Munro, 1996). Appendix E presents definitions and theorems (Bhattacharyya *et al*, 1995) related to robust stability testing of the closed-loop control system of the MIABOT V2 mobile robot under uncertainty in its dynamics.

4. **Fuzzy logic systems (FLS).** Fuzzy logic systems are employed in chapter five to model local controller (speed controller) from observation data and in chapter six to model robot behaviours (obstacle avoidance behaviours) for mobile robot navigation. The reason that FLS were adopted is because they employ qualitative linguistic terms that take into account the imprecise nature of real-world processes and systems. Other important advantages of FLS includes: allows the handling of processes that are either modelled inadequately or not representable mathematically; they describes process behaviour based on available empirical or experiential information from sensors systems and/or human operators; they can cope with complex non-linear, multi-variable and time-varying processes without requiring them to be defined in precise mathematical terms.
5. **Artificial neural networks (ANN).** Artificial neural networks are also used in chapter five to model local controller (speed controller) from observation data and to model robot behaviours (obstacle avoidance behaviours) in chapter six for mobile robot navigation. The advantages of using ANN can be summarised as follows: ANN have interesting and attractive features such as learning, self-organisation and the capability to model a large class of non-linear systems. ANN can learn a mapping between an input and an output space and form an associate memory that retrieves the appropriate output when presented with an unseen input. They can also generalise to produce an output when presented with previously unseen inputs. Calculations are in principle carried out in parallel resulting in speed advantages and programming can be done by training rather than defining explicit

instructions. The major advantage of the ANN methodology is that it can produce learning controller for a mobile robot that can operate in an uncertain environment.

6. **Clustering techniques.** Clustering can be an effective technique for dealing with large sets of data. The principal idea is to distil natural groupings of data from a large data set thereby allowing concise representation of the model's behaviour. Subtractive clustering is used in chapter five for the identification of a fuzzy speed controller from observation data. The advantage of using subtractive clustering is that computation is simply proportional to the number of data points and independent of the dimension of the problem under consideration. Having training data from a PI speed controller, construction of a dynamic fuzzy model to control the plant is achieved. Such dynamic modelling is complex and difficult task (large rule base due to many input variables) and is effectively solved using subtractive clustering.
7. **Design of control systems architectures for autonomous mobile robots.** The new control system architecture presented in chapter six is hybrid. In this chapter a description of different techniques that have been used to build control systems for autonomous mobile robots (or agents) and how different architectures can be classified is presented. When comparison between different types of control architectures is undertaken, this comparison should be made based on a number of important properties. Classification among these properties is proposed in this chapter.
8. **Multi-agent systems (MAS).** As mentioned above the new control system architecture in chapter six is hybrid, but also is multi-agent type constructed and orientated. Multi-agent systems theory is relatively new field in control and systems engineering. A special role in the theory and tools for solving complex control problems is attributed to the concept of agent. An agent represents an abstract entity that is able to solve a particular (partial)

problem. Agents have the ability to be combined into a multi-agent system, such that the overall multi-agent system is able to solve a more complex problem. In this chapter brief background information of MAS and classification of the main multi-agent control architectures is presented in order to form the basic of the concept to construct local controllers that consist of several other controllers in chapter six.

9. **Stateflow design tool based on finite state machines (FSM) theory.** Stateflow design tool is based on finite state machines (FSM) theory. In this thesis it is used in chapter six as global state identification mechanism, supervisor-like co-ordination object for several local controllers-agents and also as tool for identification of direction of neighbour robots. The advantage of using stateflow is model visualisation and simulation of complex reactive systems based on the theory of FSM. However, the design can be easily modified, evaluation of results and verification of system's behaviour at any stage of the design can be done successfully. In this chapter the main idea behind of FSM is highlighted followed by description of stateflow's main design steps.

This chapter is organised as follows: The main methodology of fuzzy logic systems and artificial neural network is presented in sections 3.2 and 3.3 respectively. Clustering techniques are discussed in section 3.4. Section 3.5 presents the design methodology of control systems architectures for autonomous mobile robot. Section 3.6 discuss multi-agent systems and their role of how can solve complex control problems. Stateflow design tool based on FSM theory is outlined in section 3.7. A discussion follows in section 3.8 and finally the summary of the chapter is presented in section 3.9.

3.2 Fuzzy logic systems (FLS)

As mentioned in the introduction of this chapter fuzzy logic is used in chapters four and five for modelling and control of local controllers. Therefore this section forms a brief outline of the

theoretical background of fuzzy logic systems highlighting the structure of fuzzy logic and fuzzy control followed by classification of different types of fuzzy models. More details and background information can be found in (Mamdani and Gaines, 1981), (Zimmermann, 1991), (Heske and Heske, 1996), (Berkan and Trubatch, 1997), (Yen and Langari, 1999) and (Nguyen and Walker, 2000).

Every history of logic seems to start with Aristotle (384-322 B.C). Aristotle, the student of Plato, thought of Logic as the science of knowing. Aristotle's most germane, and certainly his most quoted contributions to the foundations of fuzzy logic are two axioms named the Law of Contradiction and the Law of the Excluded Middle. The Law of Contradiction says that a thing cannot belong to a class and not belong to a class at the same time. This is like saying that it cannot be both *raining* and *not raining* at the same time. The Law of the Excluded Middle says that a thing must either belong to a class or not belong to a class. In other words, it must be *raining* or *not raining*. Together these two axioms leave no room for such concepts as *sort-of raining* or *slightly raining*.

The next, and certainly most significant, steps in the development of logic were taken in 1965 in a paper called *Fuzzy Sets* (Zadeh, 1965). Lofti A. Zadeh of the University of California at Berkeley took those steps by coining the name fuzzy logic and defining the mathematical notions of inclusion, union, intersection, complement, etc. for such sets. Zadeh then turned his attention to applying his newly minted fuzzy framework to modelling and automating models of human reasoning. *Approximate Reasoning* is the term used by Zadeh to describe the human ability to process imprecise, incomplete, and possibly unreliable information while reaching concrete conclusions. Since then the development of fuzzy logic theory stimulated alternative means to solve several problems in automatic control.

3.2.1 What is fuzzy logic

Fuzzy logic is a powerful problem-solving methodology with a myriad of applications in embedded control and information processing. Fuzzy provides a remarkably simple way to draw definite conclusions from vague, ambiguous or imprecise information. In a sense, fuzzy logic resembles human decision making with its ability to work from approximate data and find precise solutions. Unlike classical logic, which requires a deep understanding of a system, exact equations, and precise numeric values, fuzzy logic incorporates an alternative way of thinking, which allows modelling complex systems using a higher level of abstraction originating from human knowledge and experience. Fuzzy Logic allows expressing this knowledge with subjective concepts such as very hot, bright red, and a long time, which are mapped into exact numeric ranges.

3.2.2 Structure of fuzzy logic

A logic based on the two truth values *true* and *false* is sometimes inadequate when describing human reasoning. Fuzzy logic uses the whole interval between 0 (*false*) and 1 (*true*) to describe human reasoning.

3.2.2.1 Fuzzy sets

A fuzzy set is a set without crisp or clearly defined boundary. Fuzzy sets can be used to describe vague concepts or linguistic variables. Take for example the set of *young people*. A one year old baby will clearly be a member of the set, and a 100 years old person, will not be a member of this set, but what about people at the age 20, 30 and 40 years? Zadeh proposed a *grade of membership* such that the transition from membership to non-membership is gradual rather than abrupt. The grade of membership for all its members thus describes a fuzzy set. An item's grade of membership is normally a real number between 0 and 1 as shown in Figure 3-1, often denoted by the Greek letter μ . The higher the number the higher the membership.

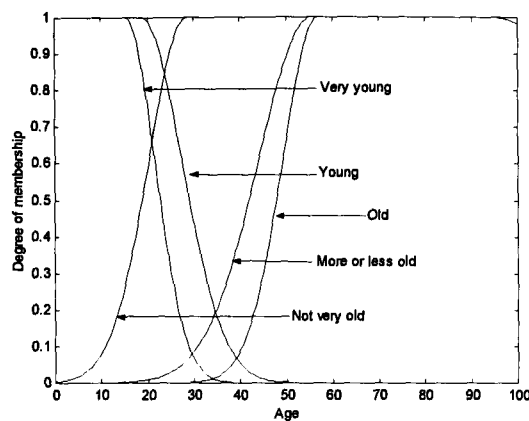


Figure 3-1 The sets *more less old*, *very young*, and *not very young* are derived from *young* and *old*

3.2.2.2 Universe

Elements of a fuzzy set are taken from a *universe of discourse*, or *universe* for short. The universe contains all elements that can come into consideration. Even the universe depends on the context, as for instance the set of young people could have all human beings in the world as its universe. Alternatively it could be the numbers between 0 and 100. These would then represent age as shown in Figure 3-1.

3.2.2.3 Membership functions

Every element in the universe of discourse is a member of the fuzzy set to some grade, maybe even zero. The function that ties a number to each element x of the universe is called the *membership function* $\mu(x)$. A fuzzy set can be described by a membership function whose membership values are strictly monotonically increasing, monotonically decreasing or monotonically increasing then monotonically decreasing for elements in the universe of discourse. Several types of basic functions can be used for membership functions, examples includes singleton, triangular, trapezoidal, s-shaped (called an s-curve), bell shaped (called π -curve), and a reverse s-shaped (called z-curve). Figure 3-2 shows an example of both triangular and trapezoid-shaped build in membership functions.

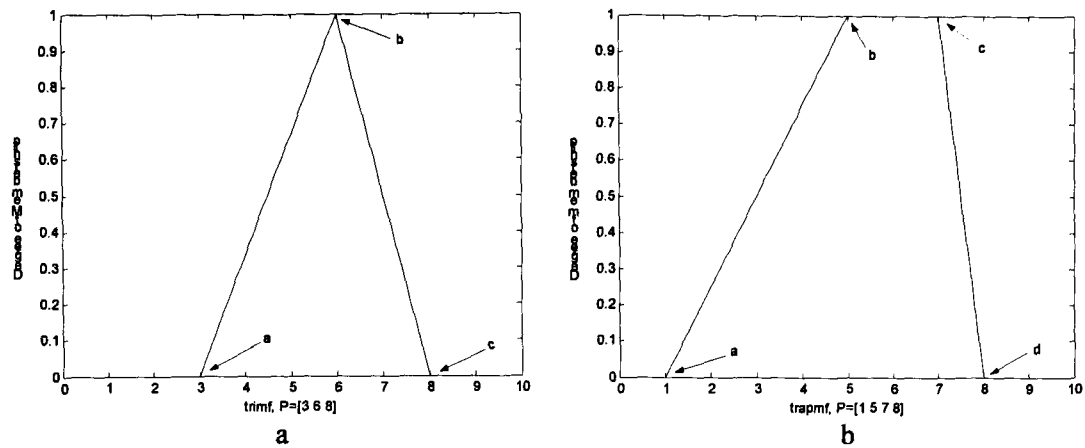


Figure 3-2 Examples of membership functions: (a) triangular-shaped (b) trapezoidal-shaped

The mathematical expression of the most commonly used membership functions is given in the following. The mathematical expression of all currently used membership functions can be found in several sources such as (Klir and Yuan, 1995) and (Passino and Yurkovich, 1998).

Triangular. Equation (3.1) represents triangular membership function in which b is a modal value, and a and c denote the lower and upper bounds, respectively, for non-zero of $f(x; a, b, c)$.

$$f(x; a, b, c) = \begin{cases} 0, & \text{if } x \leq a \\ \frac{x-a}{b-a}, & \text{if } a \leq x \leq b \\ \frac{c-x}{c-b}, & \text{if } b \leq x \leq c \\ 0, & \text{if } c \leq x \end{cases} \quad (3.1)$$

Sometimes it is more convenient to use the notation explicitly highlighting the membership function's parameters. In this case the result is:

$$f(x; a, m, b) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \quad (3.2)$$

Trapezoidal. A trapezoidal membership function (3.3) is specified by four parameters (a, b, c, d) and is defined by:

$$f(x; a, b, c, d) = \begin{cases} 0, & \text{if } x \leq a \\ \frac{x-a}{b-a}, & \text{if } a \leq x \leq b \\ 1, & \text{if } b \leq x \leq c \\ \frac{d-x}{d-c}, & \text{if } c \leq x \leq d \\ 0, & \text{if } d \leq x \end{cases} \quad (3.3)$$

Singleton. A singleton membership function is defined by one parameter only. This model can be used to translate the precise crisp input for fuzzification or to represent the inference solution.

When variable x is a , its membership $\mu(x)$ takes value 1 as defined in (3.4).

$$f(x; a) = \begin{cases} 1, & \text{if } x = a \\ 0, & \text{if } x \neq a \end{cases} \quad (3.4)$$

3.2.2.4 Fuzzy sets operations

The membership function is obviously a crucial component of a fuzzy set. It is therefore natural to define operations on fuzzy sets by means of their membership functions. In fact a fuzzy set operation creates a new set from one or several given sets. In more general terms, the most commonly used fuzzy sets operations are defined as Intersection (AND), Union (OR) and Complement (NOT). For example, consider two fuzzy sets A and B on the universe X , for a given element x of the universe, the intersection is defined as:

$$\mu_{A \cap B}(x) = \mu_A(x) \hat{t} \mu_B(x) \quad (3.5)$$

where \hat{t} is the notation of the triangular norm or t -norm, which is a fuzzy conjunction operator (Yen and Langari, 1999). The Minimum and Algebraic Product are most commonly used to calculate the t -norm for a fuzzy intersection (Pedrycz and Gomide, 1998) as shown in (3.6) and (3.7):

$$\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \min[\mu_A(x), \mu_B(x)] \quad (3.6)$$

$$\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \mu_A(x) \mu_B(x) \quad (3.7)$$

The union is defined as:

$$\mu_{A \cup B}(x) = \mu_A(x) \hat{s} \mu_B(x) \quad (3.8)$$

where \hat{s} is the notation of the triangular s -norm, which is a fuzzy disjunction operator (Yen and Langari, 1999). The Maximum is suggested and most commonly used to calculate the s -norm for a fuzzy union (Zadeh, 1965) as shown in (3.9):

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \quad (3.9)$$

Finally the complement is defined by:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \quad (3.10)$$

3.2.3 Design of fuzzy logic controller

Design of a fuzzy controller requires more design decisions than usual, for example regarding rule base, inference engine, defuzzification, and data pre and post processing. There are specific components characteristic of a fuzzy controller to support a design procedure. In the block diagram in Figure 3-3, the fuzzy controller is between pre-processing block and a post-processing block.

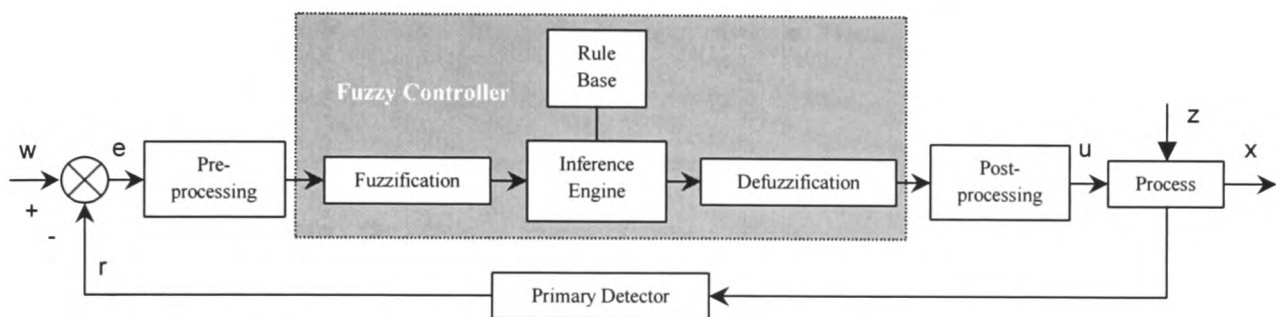


Figure 3-3 Structure of fuzzy controller

Where w is the setpoint, x is the process output, u is the control action, r is the measured value, e is the error and z is the process noise. The following briefly explains the Figure 3-3 block by block. More details can be found in (Cox, 1999).

3.2.3.1 Pre-processing

The inputs are most often hard or *crisp* measurements from some measuring equipment, rather than linguistic. A pre-processor, conditions the measurements before they enter the controller. Examples of pre-processing are: normalisation or scaling onto a particular, standard range; filtering in order to remove noise. When the input to the controller is *error*, the control strategy is a static mapping between input and control signal. A dynamic controller would have additional inputs, for example derivatives, integrals, or previous values of measurements

backwards in time. These are created in the pre-processor thus making the controller multi-dimensional, which requires many rules and makes it more difficult to design. The pre-processor then passes the data on to the controller.

3.2.3.2 Fuzzification

The first block inside the controller is *fuzzification*, which converts each piece of input data to degrees of membership by a lookup in one or several membership functions. The fuzzification block matches the input data with the conditions of the rules to determine how well the condition of each rule matches that particular input instance. There is a degree of membership for each linguistic term that applies to that input variable. In other words fuzzification interface involves the following functions: measures the value of input variables; performs a scale mapping that transfers the range of values of input variables into corresponding universes of discourse; performs the function that converts input data into suitable linguistic values, which may be viewed as labels of fuzzy sets. At this design stage it is important to set up parameters such as type and number of membership functions.

3.2.3.3 Rule base

The rules may use several variables both in the condition and the conclusion of the rules. The controllers can therefore be applied to both multi-input multi-output (MIMO) problems and single-input single-output (SISO) problems. The controller may actually need both the *error* and the *change of error* as inputs. To simplify, this section assumes that the control objective is to regulate some process output around a prescribed setpoint or reference. Therefore the rule base will contain rules in the *if-then* format. Usually different formats are used to represent the rule base. One example can be the rule format in the following:

IF error is *Neg* and change in error is *Pos* THEN output is *Zero*

The names *Neg*, *Pos* and *Zero* are labels of fuzzy sets. The same set of rules could be presented in a *relational* format or the *tabular linguistic* format for shorter representation.

3.2.3.4 Inference engine

Inference is the act of drawing a conclusion based on a premise. In the case of fuzzy logic rule based systems, premises are spelled out as a combination of antecedents on the IF-side of rules. The consequents of a fuzzy rule can be interpreted as the conclusions drawn if the premises of a fuzzy rule are satisfied. The fuzzy inference procedure specifies how the IF-side truth value applies to the consequents specified in the rule. The strength of the conclusion simply indicates a degree of belief in the actual value or action specified in the rule consequent.

Characteristically, the degree to which the premise of a fuzzy rule is satisfied lies on the continuum of values $[0,1]$. What is expected for any inference procedure is that the strength of the conclusion should track with the strength of its associated premises. A premise truth of zero should lead to a conclusion with zero strength, and a premise truth of one should lead to a conclusion with maximal strength. In between the extremes of premise truth, the strength of a conclusion should increase as the strength of its premise increases. Implication and aggregation both are operating within the inference engine mechanism. In general there are three methods of inference engine: correlation minimum, correlation product and min-max (Heske and Heske, 1996).

3.2.3.5 Defuzzification

Defuzzification is needed to translate the fuzzy output of a fuzzy controller to a numerical representation (crisp). When a fuzzy controller is considered from the theoretical point of view, the fuzzy output can be a multi-dimensional fuzzy set (fuzzy relation). This assumes that the controller can have multiple outputs (SISO or MIMO system) which causes the fuzzy output of the controller to be a multi-dimensional fuzzy set. Some of the most commonly defuzzification

methods include: centre of gravity, centre of largest area, mean of maxima, first of maxima, middle of maxima, height, left most, right most and weighted average defuzzification. The choice of defuzzification method depends on the context of the control problem (Ross, 1995) and (Reznik *et al*, 2000). In the following only the centre of gravity defuzzification method is described due to its popularity whereas the other methods can be found in several sources such as (Klir and Yuan, 1995) and (Yen and Langari, 1999).

$$u = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)} \quad (3.11)$$

The crisp output value u is the abscissa under the centre of gravity of a fuzzy set. Here x_i is a running point in a discrete universe, and $\mu(x_i)$ is its membership value in the membership function. For the continuous case, replace the summations by integrals. It is much used method although its computational complexity is relatively high.

3.2.3.6 Post-processing

Output scaling is also very important operation in the design of fuzzy logic controllers. In case the output is defined on a standard universe, this must be scaled to engineering units, for instance, volts, meters, etc.

3.2.4 Types of fuzzy models

Given a model (heuristic or analytical) of the physical system to be controlled and specifications for its desired behaviour, the design objective in fuzzy control is to design a feedback control law in the form of a set of fuzzy rules such as that the closed loop system exhibit the desired behaviour. To achieve this design goal different types of fuzzy model can be employed. (Driankov and Palm, 1998) has classified these types of fuzzy models as follows: Mamdani

fuzzy model, Takagi-Sugeno (TS) fuzzy model, relational fuzzy model, differential equations non-linear model and modified TS fuzzy model approximating a given differential equations non-linear model. In this thesis both Mamdani and Takagi-Sugeno fuzzy models are used. Each fuzzy model has to offer advantages over the other. In the following an overview is given for both Mamdani and Takagi-Sugeno fuzzy models.

3.2.4.1 Mamdani model

The Mamdani fuzzy model was first introduced by (Mamdani and Assilean, 1975). Using this model, Mamdani, developed the first fuzzy logic controller. Most fuzzy control systems developed in the 80's use the Mamdani model. The main idea of the Mamdani controller is to describe process states by means of linguistic variables and to use these variables as inputs to control rules. The Mamdani method is more intuitive and well-suited to human input and also more widespread method. However, this model is computational expensive and cannot be used for optimisation. In the following the Mamdani model is described.

Consider the following rule base, where X , Y and Z are linguistic variables:

$$R_i : \text{IF } X \text{ is } A_i \text{ and } Y \text{ is } B_i \text{ THEN } Z \text{ is } C_i \quad (i=1,2,\dots,n) \quad (3.12)$$

Given the input fact (x_0, y_0) , the goal is to determine the output “ Z is C ”. The first step to make is to fuzzify the given input. The fuzzifier maps the input data $x_0 \in U_x$ into the fuzzy set A and $y_0 \in U_y$ into fuzzy set B . The next step is to evaluate the truth-value for the premise for each rule, and then apply the result to the conclusion part of each rule using the fuzzy implication. The Membership functions defined on the input variables are applied to their actual values to determine the degree of truth for each rule premise. The degree of truth for a rule's premise is computed for the rule base in equation (3.12) as follows:

$$a_i = \mu_{A_i \text{ and } B_i}(x_0, y_0) = \min(\mu_{A_i}(x_0), \mu_{B_i}(y_0)) \quad (3.13)$$

If a rule's premise has nonzero degree of truth the rule is activated. The next step is to find the output, C'_i , of each of the rules:

$$\mu_{C'_i}(w) = \mu_{(A_i \text{ and } B_i) \rightarrow C_i}(x_0, y_0, w), \quad \forall w \in W \quad (3.14)$$

In *MIN* inferencing (or Mamdani implication rule) the implication is interpreted as a fuzzy *AND* operator:

$$\mu_{C'_i}(w) = \mu_{A_i \text{ and } B_i}(x_0, y_0) \text{ and } \mu_{C_i}(w) = \min(\mu_{A_i \text{ and } B_i}(x_0, y_0), \mu_{C_i}(w)) \quad (3.15)$$

In the rule aggregation step, all fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable. The purpose is to aggregate all individual rule outputs to obtain the overall system output. In the *MAX* composition, the combined output fuzzy subset C^* is constructed by taking the maximum over all the fuzzy subsets assigned to the output variable by the inference rule:

$$\mu_{C^*}(w) = \max(\mu_{C'_1}(w), \mu_{C'_2}(w), \dots, \mu_{C'_n}(w)) \quad (3.16)$$

Normally, the defuzzification step is executed as the last step and the most commonly used method in the centre of gravity described in section 3.2.3.5.

3.2.4.2 Takagi-Sugeno model

The Takagi-Sugeno (TS) model was introduced by (Takagi and Sugeno, 1985) about one decade after the Mamdani model. The main motivation for developing this model is to reduce the number of rules required by the Mamdani model, especially for complex and high-dimensional problems. The Takagi-Sugeno model is computationally more efficient and well suited to optimisation and adaptive techniques and to mathematical analysis. It has also guaranteed continuity of the output space and it works well with linear techniques such as PID control. The fuzzification of the inputs and the application of the fuzzy operator are similar to Mamdani model. However the Takagi-Sugeno model differs from Mamdani model by introducing crisp functions as the consequences of the rules. This structure offers a systematic approach to generate fuzzy rules from a given input-output data set. A Takagi-Sugeno rule set is of the following form:

$$R_i : \text{IF } X \text{ is } A_i \text{ and } Y \text{ is } B_i \text{ THEN } z_i = f_i(x_0, y_0), \quad (i = 1, 2, \dots, n) \quad (3.17)$$

Where, (x_0, y_0) is the input. The antecedent of each rule is a set of fuzzy propositions connected with the *AND* operator. The consequent of each rule is a crisp function of the input vector $[x_0, y_0]$. By means of the fuzzy sets to the antecedent propositions the input domain is softly partitioned in smaller regions where the mapping is locally approximated by the crisp functions f_i . Combining the rules and their affects differs from the Mamdani method considerably. One variation of the TS inference system uses the weighted sum criterion to combine all the local representations in a global approximator, by:

$$z = \frac{\sum_{i=1}^r \mu_i z_i}{\sum_{i=1}^r \mu_i} \quad (3.18)$$

Where, μ_i is the degree of fulfilment of the i th rule and r is the number of the rules in the rule base.

3.3 Artificial neural networks (ANN)

Artificial neural networks are employed in chapters five and six for modelling and control of local controllers. This section forms a brief introduction of the theoretical background of ANN. More details and background information can be found in literature in the contributions of (Kosko, 1992), (Taylor and Lisboa, 1993), (Harris, 1994), (Welstead, 1994), (Jang *et al*, 1997), (Ng, 1997), (Picton, 1998), (Haykin, 1999) and (Li *et al*, 2001).

The aim of ANN is to model networks of biological neurons in the brain. The human brain and nervous system have amassing properties. Although brain cells operates about seven orders of magnitude slower than switching elements of modern computers, the brain is capable of performing tasks which are impossible for computers. The main reasons behind this statement are massive parallelism and asynchronous operation of the human brain. The ANN structure is parallel composed of many computational elements connected by links with variable weights.

3.3.1 Biological neuron

The human brain consists of about hundred billions (10^{11}) different types of neurons. Each of them has about 1,000 – 10,000 connections to other neurons. The nervous system has several kinds of nerve cells (neurons), but they all share common features. Figure 3-4 shows the structure of a typical biological neuron. The neuron is formed of a *soma* (i.e., cell body), *dendrites*, and an *axon*. The dendrites receive signals from the other neurons, and the axon passes a signal to the other neurons. A junction between the axon and the dentate is called a

synapse. One axon makes typically a few thousand of *excitatory* or *inhibitory* synapses with other neurons.

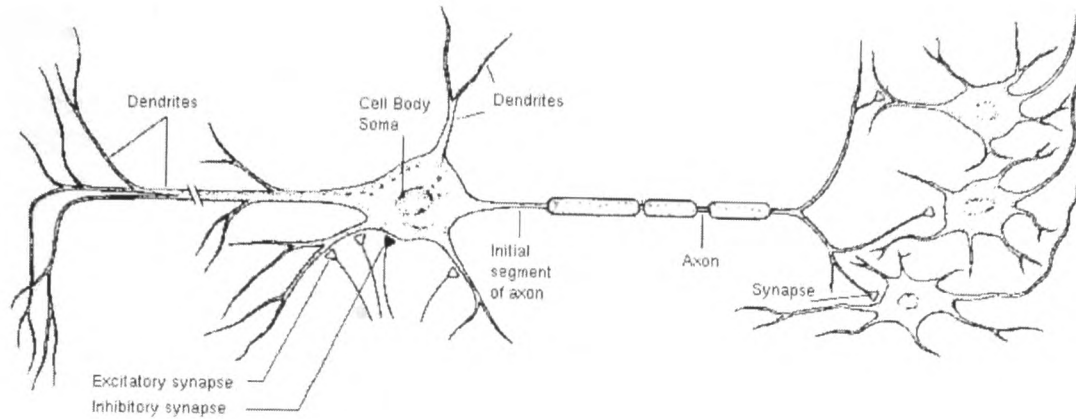


Figure 3-4 Biological neuron

The process of transmitting a signal from one neuron to another is chemically complex and is beyond the scope of this thesis. Briefly, the receiving cell's *electrical potential* raises or lowers depending on the incoming signal. When this potential reaches a *threshold*, an *action potential* of fixed strength and duration is sent down the axon. The neuron “fires” and the action potential is transmitted through the axon to the synapses with other cells. After a *refractory period* the cell can fire again (Ganong, 1987) and (Hertz *et al*, 1991).

3.3.2 Model of artificial neuron

The first model of an artificial neuron was proposed in 1943 by (McCulloch and Pitts, 1943). The standard artificial neuron is a processing element whose output is calculated by multiplying its inputs by a weight vector, summing the results and applying an activation function to the sum. Figure 3-5 shows the model of a neuron, which forms the basics for designing the ANN. The main basic elements of the neuron are described as follows (Haykin, 1999):

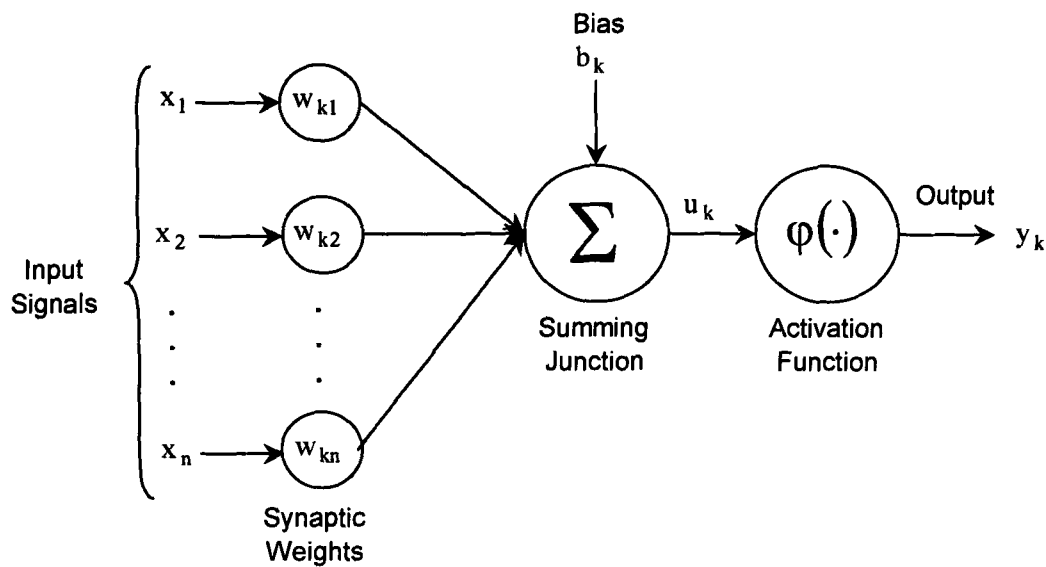


Figure 3-5 Model of a neuron

Synaptic weights. This is a set of synapses or connecting links, each of which is characterised by a weight or strength on its own. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj} .

Summing junction. This is an adder for summing the input signals, after they have been weighted by the respective synapses of the neuron.

Activation function. This transforms the result of the adder and limits the amplitude of the output of a neuron. Generally, the normalised amplitude range of the output of the neuron is given as the closed unit interval $[0,1]$, or alternative $[-1,1]$.

Bias. This has the effect of increasing or lowering the net input of the activation function, depending on whether it is positive or negative, respectively.

In mathematical terms the neuron's functionality can be described with equations (3.19) and (3.20) as follows:

$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (3.19)$$

$$y_k = \varphi(u_k + b_k) \quad (3.20)$$

Where x_1, x_2, \dots, x_m are the input signals, $w_{k1}, w_{k2}, \dots, w_{km}$ are the synaptic weights of the neuron k , u_k is the output of the summing junction, b_k is the bias, $\varphi(\cdot)$ is the activation function and y_k is the output signal of the neuron.

3.3.2.1 Types of activation function

The activation function, $\varphi(\cdot)$, defines the output of a neuron in terms of the activity level at its input. The activation function can have various shapes depending on the application (Jang *et al*, 1997). The most commonly used types of activation functions are shown in Figure 3-6.

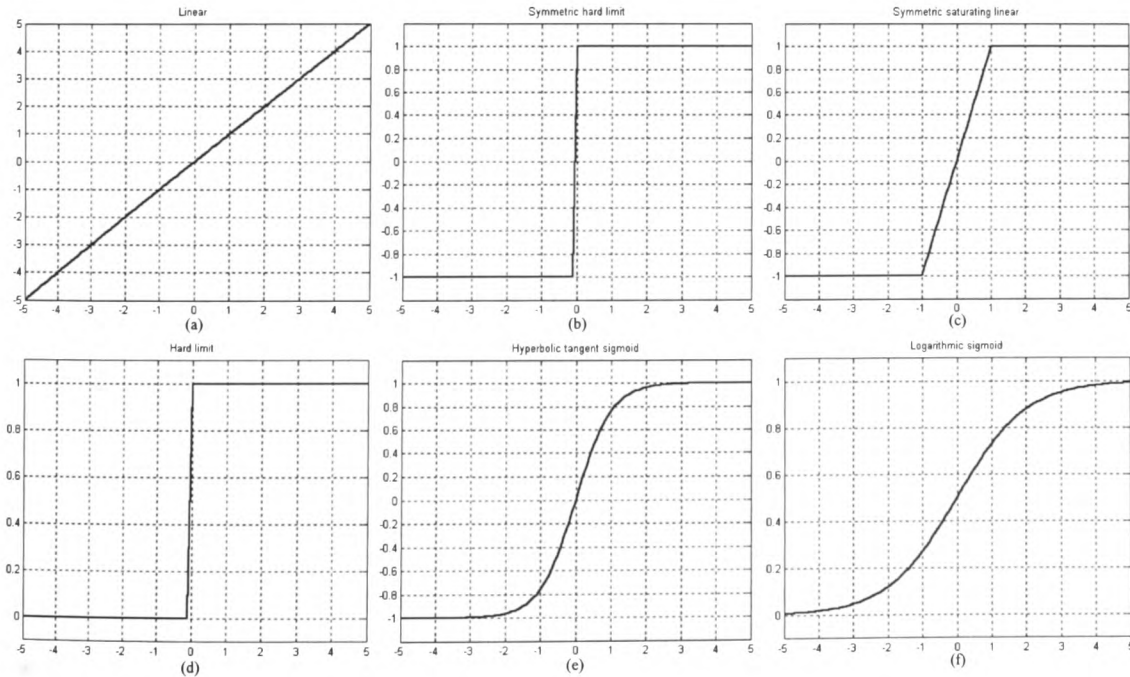


Figure 3-6 Common activation functions

In mathematical terms the activation functions shown in Figure 3-6 can be expressed as follows:

Linear. A linear (Figure 3-6(a)) activation function's output is simply equal to its input as shown in equation (3.21).

$$f(x) = x \quad (3.21)$$

Symmetric hard limit. A symmetric hard limit (Figure 3-6(b)) activation function's output is defined by:

$$f(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \quad (3.22)$$

Symmetric saturating linear. A symmetric saturating linear (Figure 3-6(c)) activation function's output is defined in equation (3.23) where k is constant and greater than zero.

$$f(x) = \begin{cases} 1 & x \geq (1/k) \\ kx & (-1/k) \leq x < (1/k) \\ -1 & x < (-1/k) \end{cases} \quad (3.23)$$

Hard limit (or step). A hard limit (Figure 3-6(d)) activation function's output is defined by:

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (3.24)$$

Hyperbolic tangent sigmoid. A hyperbolic tangent sigmoid (Figure 3-6(e)) activation function's output is defined by:

$$f(x) = (e^x - e^{-x}) / (e^x + e^{-x}) \quad (3.25)$$

Logarithmic sigmoid. A logarithmic sigmoid (Figure 3-6(f)) activation function's output is defined by:

$$f(x) = 1 / (1 + e^{-x}) \quad (3.26)$$

The construction of a neural network deepens upon many parameters. These parameters include: type of activation function, network architecture and learning method. In the following section the main types of ANN architectures are presented.

3.3.3 ANN architectures

ANN architectures can be divided into two main categories depending on the kind of learning that is incorporated in the network (Pfeifer and Scheier, 1999). In the first category are neural networks that require no teacher and are said to be *unsupervised*. Unsupervised ANN usually divided into two types; Hebbian learning and Kohonen maps. Hebbian learning comes in many variations but usually takes place when the nodes on both sides of the connection are simultaneously active (sometimes within a given interval). The advantages of Hebbian learning includes: it is simple and requires little computation; it is purely local, meaning that for learning, only the neuron itself and its neighbours need to be considered; it is biologically plausible (Churchland and Sejnowski, 1992). Kohonen maps (Kohonen, 1982) are widely used in many applications such motor control, navigation, etc. The basic architecture of Kohonen map is described as follows: In the map layer, lateral (connections are called lateral if they link nodes within a layer, rather than between layers) connections are excitatory for close neighbours, inhibitory for those further away and neutral for the ones still further out. Patterns are presented

to the model at the input layer, and depending on the particular architecture and choice of parameters, the system will eventually learn a particular categorisation of the input space. The Kohonen maps can be used if the classification data are unknown, can be used to map high-dimensional spaces into low-dimensional ones and as does Hebbian learning, Kohonen maps have certain neurobiological plausibility (Kohonen, 1989). The second category comprises neural networks that require a teacher and are said to be *supervised*. Supervised ANN usually divided into two network topologies; feedforward neural networks (FNN) and recurrent (or feedback) neural networks (RNN). Supervised FNN and RNN are discussed in more detail in the following section. However more emphasis will be given to FNN as used both in chapters five and six.

3.3.3.1 Feedforward neural networks (FNN)

FNN are constructed of one (perceptron neural network) or more (multilayer perceptron neural network) hidden layers between the input and output layers as shown in Figure 3-7. As can be observed from Figure 3-7 the network is divided into input layer, output layer and hidden layer(s). The neurons are forward connected between adjacent layers, signals propagate only in the direction from the input layer, through intermediate hidden layers neurons, to the output layer. FNN are classified as fully connected, if every neuron in the layer of the network is connected to every neuron in the adjacent forward layer. However, if some of the communication links (synaptic connections) are missing from the network, it is said that the network is partially connected. It has been proved that any non-linear function can be approximated using FNN with one hidden layer having non-linear activation functions (Cybenko, 1989), (Hornik *et al*, 1989) and (Funahashi, 1989). The same result or even better can be achieved if more than one non-linear hidden layers form the network. (De Villiers and Barnard, 1993) have demonstrated that FNN with two hidden layers are more prone to fall into local minima, which might give better approximation for some specific problems.

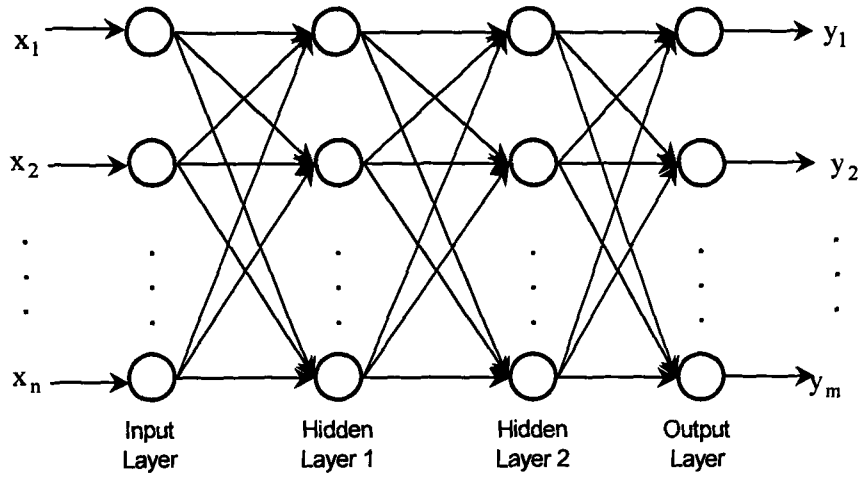


Figure 3-7 Feedforward neural network

The general relationship in a FNN between the input x and the output y is represented by the following equation (3.27).

$$y(\xi, x) = f \left(\sum_n \xi_n f \left(\sum_j \xi_j f \left(\dots f \left(\sum_i \xi_i x_i \right) \right) \right) \right) \quad (3.27)$$

In the particular case in Figure 3-7 with two hidden layers, equation (3.27) can be expressed as follows:

$$y(\xi, x) = f_4 (\xi_4 f_3 (\xi_3 f_2 (\xi_2 f_1 (\xi_1 x)))) \quad (3.28)$$

Where $\xi_i = [w_i, b_i]$ with $i = 1, 2, 3, 4$ is the vector parameter of each layer, w are the weights, b are the biases, x is the input vector and f are the activation functions for each layer of neurons. In order to allow the output of the network to be any real number within a certain bound, the activation function of the output layer is usually chosen to be a linear function. In order to guarantee the approximation properties of the network, the hidden layers activation

functions are chosen to be a non-linear function with known derivatives (this requirement is due to the backpropagation algorithm). Two of the commonly used non-linear functions in the hidden layer are: the hyperbolic tangent sigmoid function shown in equation (3.25) with values in $[-1,1]$ and logarithmic sigmoid function shown in equation (3.26) with values in $[0,1]$.

3.3.3.2 Recurrent neural networks (RNN)

RNN are different from FNN in that their structure incorporates at least one feedback loop as shown in Figure 3-8. In general, the output of every neuron is fed back with varying gains (weights) to the input of all neurons. It is claimed that the presence of feedback loops has a profound impact on the learning capability of the network and on its performance (Ku and Lee, 1995). The feedback loops commonly involve unit delays if dealing with discrete-time systems, or integrators in the continuous-time case. The RNN may be preferred to FNN when the measured plant outputs are highly corrupted by noise and the dynamics of the non-linear process are complex and unknown

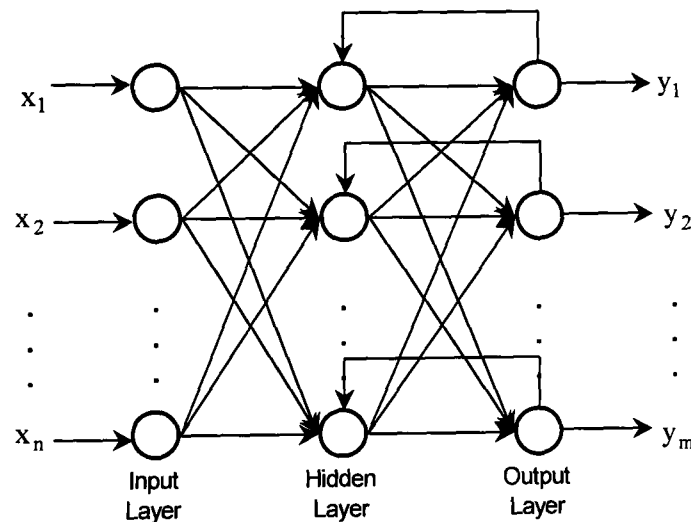


Figure 3-8 Recurrent neural network

3.3.4 Training of feedforward multilayer perceptron NN

The multilayer perceptron (MLP) neural networks were not used in the past because of lack of an effective training technique. (Werbos, 1974) developed an algorithm called back-propagation in 1974, but his achievement remained almost unknown for almost a decade. (Rumelhart *et al*, 1986) re-discovered the technique and later a closely related approach was proposed in (Cichocki and Unbehauen, 1994) and (Faussett, 1994). Since the work of (Rumelhart *et al*, 1986) back-propagation algorithm has been successfully used for training of MLP networks. In addition, more recently, the back-propagation algorithm has also been adopted for training of RNN. A feedforward MLP network should perform a specific non-linear mapping or association task which can be expressed in terms of a given input/output pattern (pairs). These input/output relations are called a set of training examples. Training of the MLP network consists of the adaptation of all synaptic or connection weights in such a way that the discrepancy between the actual output signals and the desired signals, averaged over all training input examples, is as small as possible. In other words, the back-propagation algorithm can be considered as an unconstrained optimisation training problem of a suitably constructed error or cost function. A full description of the back-propagation algorithm is given in Appendix F. There are several different back-propagation training algorithms available in the literature. They have a variety of different computation and storage requirements and no one algorithm is best suited to all situations. The MATLAB library offers a wide range of different back-propagation training algorithms some of which include: basic gradient descent algorithm, gradient descent with momentum algorithm, Resilient back-propagation algorithm, Fletcher-Reeves conjugate gradient algorithm, Polak-Ribiere conjugate gradient algorithm, Powell-Beale conjugate gradient algorithm, Scaled conjugate gradient algorithm, BFGS quasi-Newton method algorithm, one step secant method algorithm, adaptive learning rate algorithm, Levenberg-Marquardt algorithm and Bayesian regularisation algorithm. It is very difficult to know which of the above algorithms will be the faster for a given problem as this depends upon many

factors. Some of these factors include complexity of the problem, number of data points in the training set, number of weights and biases in the network and the error goal. The training algorithm used for the feedforward MLP networks in chapters five and six is Levenberg-Marquardt (Levenberg, 1944) and (Marquardt, 1963). This algorithm was chosen because it was found to have the fastest convergence over the others with accurate training. More details of the Levenberg-Marquardt algorithm can be found in Appendix F.

3.4 Clustering

Clustering is used in chapter five for the identification of a dynamic model (fuzzy model from PI speed controller). The problem of automatic generation of fuzzy IF-THEN rules is one of the most important issues in the development of fuzzy systems models. Clustering of numerical data, which is one of the most fundamental issues in pattern recognition and system modelling algorithms, can be successfully used to solve the aforementioned problem. The main purpose of clustering is to distill natural groupings of data from a large data set, producing a concise representation of the system's behaviour. To demonstrate the importance of clustering techniques consider the clustering problem in Figure 3-9. It is obvious that there are four clusters in Figure 3-9a when the objective function is based on distance between the elements. In this case reasoning easily can create clusters, such as *IF X and Y are small THEN it is cluster 1; IF X and Y are large THEN it is cluster 2*, and so forth. On the other hand the clustering problem in Figure 3-9b is complex since every data point appears to be equidistant from each other. The important question here is whether a mathematical technique can do better job than the human brain for cases depicted in Figure 3-9b. In the following sections some definitions and notation in general cluster analysis is given. The main three offline clustering techniques are also discussed with more emphasis on subtractive clustering as this technique is implemented in chapter five for identification of dynamic fuzzy controller.

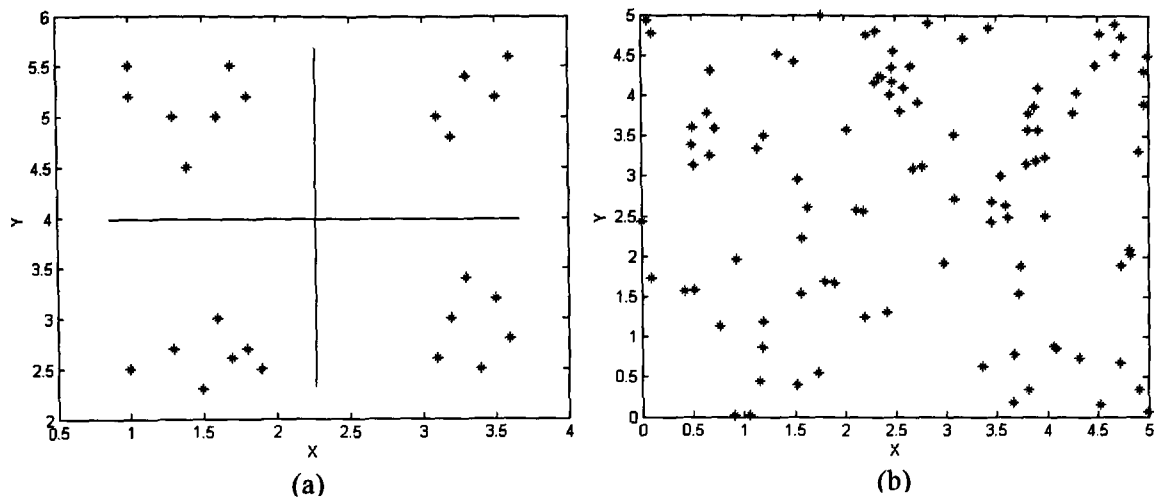


Figure 3-9 Clustering based on distance: (a) Interpretable by human, (b) ambiguous to human.

3.4.1 Definitions and notations in cluster analysis

Prior the description of the off-line clustering techniques that can be used in identification, modelling and control strategies, some definitions and notations used in the cluster analysis are given. Note that the following sections only indicate the main definitions and notations of the cluster analysis whereas more details can be found in the literature (Bezdek, 1981), (Klir and Yuan, 1995), (Yen and Langari, 1999) and (Bezdek *et al*, 1999).

3.4.1.1 The data used in cluster analysis

One of the most important advantages of clustering techniques is that they can be applied to data that is quantitative (numerical), qualitative (categorical) or mixture of both. In chapter five where cluster analysis takes place for the identification of dynamic model the data considered are quantitative. In general, the data are typically observations and/or records of some physical process of a real system and/or control action. Each observation and/or record consists of n measured variables, grouped into an n -dimensional Euclidean space \mathcal{R}^n column vector

$\Psi_k = [\psi_{1k}, \psi_{2k}, \dots, \psi_{nk}]^T$, $\psi_k \in \mathbb{R}^n$. A set of N observations and/or records is denoted by $\Psi = \{\Psi_k | k = 1, 2, \dots, N\}$ and is represented as an $n \times N$ matrix as follows:

$$\Psi = \begin{bmatrix} \psi_{11} & \psi_{12} & \dots & \psi_{1N} \\ \psi_{21} & \psi_{22} & \dots & \psi_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{n1} & \psi_{n2} & \dots & \psi_{nN} \end{bmatrix} \quad (3.29)$$

In pattern recognition terminology, the columns of the matrix are called patterns or objects, the rows are called the features or attributes, and Ψ is called the pattern or data matrix. The meaning of the columns and rows of Ψ depends on the context of the classification problem.

3.4.1.2 Definition of clusters

Various definitions of a cluster can be formulated, depending on the objective of clustering. (Bezdek, 1981) defined a cluster as a group of homogeneous classes or objects that are more similar to one another than to members of other clusters. The term “similarity measure” has an important effect on the clustering analysis results since it indicates which mathematical properties of the data set, i.e. distance, connectivity, intensity should be used and in what way in order to identify the clusters. Distance can be measured among the data vectors themselves, or as a distance from a data vector to some prototypical object of the cluster. The prototypes (which are usually the centre of the clusters) or centroids are usually not known in advance, and are sought by the clustering algorithms simultaneously with the partitioning of the data. The centre of clusters may be vectors of the same dimension as the data objects, but they can also be defined as high-level geometrical objects, such as linear or non-linear subspaces or functions.

3.4.2 Clustering algorithms (off-line)

This section presents three of the most representative off-line clustering techniques frequently used for fuzzy modelling (fuzzy c-means, mountain and subtractive). According to (Jang *et al*, 1997) clustering techniques are validated on the basis of the following two assumptions.

1. Similar inputs to the target system to be modelled should produce similar outputs. In other words the target system to be modelled is a smooth input-output mapping.
2. These similar input-output pairs are bundled into clusters in the training data set. This means that the data set required conforming to some specific type of distribution. (Jang *et al*, 1997) claims that this is not always true. Therefore clustering techniques used for fuzzy modelling are highly heuristic, and finding a data set to which clustering techniques cannot be applied satisfactory is not uncommon.

3.4.2.1 Fuzzy C-means clustering

The Fuzzy C-means³ (FCM) clustering method was proposed by (Bezdek, 1974), as an improvement over previous clustering method called C-means (Dunn, 1974). FCM clustering method clusters the data by minimising the total “distance” of each data point to the cluster centre. In particular, FCM algorithm is an iterative optimisation algorithm that minimises the following cost function.

$$J = \sum_{k=1}^n \sum_{i=1}^c \mu_{ik}^m \|x_k - v_i\|^2 \quad (3.30)$$

³ C-means clustering method very often called K-means or hard C-means

Where, n is the number of data points, c is the number of clusters, x_k is the k th data point, v_i is the i th cluster centre, μ_{ik} is the degree of membership (between 0 and 1) of the k th data in the i th cluster and m is a weighting exponent greater than 1 (typically $m=2$, $m \in [1, \infty)$). The degree of membership μ_{ik} is defined by equation (3.31).

$$\mu_{ik} = \frac{1}{\sum_{j=1}^c \left(\frac{\|x_k - v_i\|}{\|x_k - v_j\|} \right)^{2/(m-1)}} \quad (3.31)$$

Starting with a desired number of clusters c and an initial guess for each cluster centre v_i , $i=1,2,\dots,c$, FCM clustering method will converge to a solution for v_i that represents either a local minimum or a saddle point of the cost function (Bezdek *et al*, 1999). The quality of the FCM solution depends strongly on the choice of initial values such as the number of clusters, clusters centres, etc. This can be concluded as disadvantage of FCM, as the critical problem is how to determine the optimal number of clusters. To overcome this problem another fast algorithm is needed to determine the initial cluster centres or the FCM algorithm must run several times with a different sets of initial clusters. For detailed treatment of FCM clustering algorithm including its variants and convergence properties, the reader is referred to (Bezdek, 1981).

3.4.2.2 Mountain clustering method

The mountain clustering method, proposed by (Yager and Filev, 1994a), (Yager and Filev, 1994b), (Yager and Filev, 1994c), is a relatively simple and effective approach for estimating the number and initial locations of cluster centres on the basis of a density measure called the mountain function. This method makes a grid of the data space and computes a potential value of each grid point based on the distance to the actual data points (a grid point with many data

points nearby will have a high potential value). The first cluster centre is chosen the grid point with the highest potential value. The main philosophy behind this method is that once the first cluster centre is chosen, the potential of all grid point are reduced according to their distance from the cluster centre (a greatly reduced potential will have grid point near the first cluster centre). The next cluster centre is placed at the grid point with the highest remaining potential value. This procedure (acquiring new cluster centre and reducing the potential of surrounding grid point) is repeated until the potential of all grid points falls below a threshold. The general form of the mountain method can be divided into three steps as follows:

1. Step one involves forming a grid on the data space, where the intersections of the grid lines constitute the candidates for cluster centres, denoted as a set Y .
2. Step two involves the construction of the mountain function representing a data density measure. The height of the mountain function at an a point $v \in Y$ is given as follows:

$$h^m(v) = \sum_{i=1}^N \exp\left(-\frac{\|v - \psi_i\|^2}{2\sigma^2}\right) \quad (3.32)$$

Where ψ_i is the i th data point and σ is an application-specific constant, which determines the height as well the smoothness of the resultant mountain function. Equation (3.32) shows that each data point ψ_i contributes to the height of the mountain function $h^m(v)$ at v . This contribution is inversely proportional to the distance between ψ_i and v . The higher the mountain function value at a point the higher is its potential to be a cluster centre.

3. Step three involves the use of mountain function to define the cluster centres (this is achieved by sequentially destruction of the mountain function). The first cluster centre

c_1 is the point in the candidate centres Y that has the greatest value for the mountain function i.e., $c = \max(h^m(v))$. Obtaining the next cluster centre requires eliminating the effect of the just-identified centre, which is typically surrounded by a number of points that also have high-density scores. This can be done by revising the mountain function as follows:

$$h_{\varepsilon}^m(v) = h_{\varepsilon-1}^m(v) - h^m(c_{\varepsilon}) \exp\left(-\frac{\|v - c_{\varepsilon}\|^2}{2\varsigma^2}\right) \quad (3.33)$$

Where ε is the number of the cluster and ς is a positive constant similar to the parameter σ . The subtracted exponential part in equation (3.33) is a Gaussian function inversely proportional to the distance between v and c_{ε} , as well as being proportional to the height $h^m(c_{\varepsilon})$ at the centre. Note that after subtraction, the new mountain function reduces to zero at $v = c_1$. The ε cluster centre is again selected as the point in Y that has the largest value for the new mountain function. This process of revising the mountain function and finding the next cluster continues until a sufficient number of cluster centres is attained.

The main advantage of the mountain clustering method is that it does not require a predefined number of clusters and also is less sensitive to noise than other clustering methods such as FCM (Pal and Chakraborty, 2000). The main disadvantage of the mountain clustering method is that it is computationally expensive and the amount of computation grows rapidly with the increase in the dimensionality of the data.

3.4.2.3 Subtractive clustering

Subtractive clustering is used in chapter five for the identification/construction of local models (fuzzy speed controller). In sections 3.4.2.1 and 3.4.2.2 two well-established clustering algorithms for fuzzy model identification were presented. It was shown that both algorithms had disadvantages such as determination of optimal number of clusters (FCM) and increase in computation as the dimensionality of the problem increases (mountain clustering). To overcome these problems subtractive clustering is an alternative approach to solve complex and high-dimensional problem. In the following, a subtractive clustering algorithm is presented in more details than the previously presented algorithms as this has been used extensively in this thesis.

Subtractive clustering proposed by (Chiu, 1994) is an extension of the mountain clustering method. Using subtractive clustering data points (not grid points) are considered as the candidates for cluster centres. The computation is simply proportional to the number of data points and independent of the dimension of the problem under consideration. Consider a collection of n data points $\{x_1, \dots, x_n\}$ in an M -dimensional space. Without the loss of generality, the data points are assumed to have been normalised within a hypercube. Since each data point is a candidate for cluster centres, a density measure at data x_i is defined as:

$$D_i = \sum_{j=1}^n e^{-a\|x_i - x_j\|^2}, \quad a = \frac{4}{r_a^2}, \quad r_a > 0 \quad (3.34)$$

Hence, a data point will have a high-density value if it has many neighbouring data points. The radius r_a defines a neighbourhood. Data points outside this radius contribute only slightly to the density measure. After computing the density measure for each point, the one with the higher density is selected as the first cluster centre. Let x_{c_1} be the centre of the first group and D_{c_1} its density. Then, the density measure for each data point x_i is revised by the formula:

$$D_i = D_i - D_{c_i} e^{-\beta \|x_i - x_{c_i}\|^2}, \beta = \frac{4}{r_b^2}, r_b > 0 \quad (3.35)$$

The radius r_b represents the radius of the neighbourhood for which significant density measure reduction will occur. The radius for reduction of density should be to some extent higher than the neighbourhood radius to avoid closely spaced clusters. The value is typically, $r_b = 1.5r_a$. Since the points closer to the cluster centre will have their density measure strongly reduced, the probability for those points to be chosen as the next cluster is lower. This procedure is carried out iteratively, until the stopping criteria are reached. The algorithm is presented as follows.

Subtractive clustering algorithm

if $D_k > \varepsilon^{up} D_{c_i}$ (D_k is the density of location of the k th cluster centre x_{Ck})

Accept x_{Ck} as the next cluster centre and continue

else if $D_k < \varepsilon^{down} D_{c_i}$

Reject x_{Ck} and end the clustering process

else

Let d_{min} be the shortest distance between x_k and all previously found cluster centres

if $\frac{d_{min}}{r_a} + \frac{D_k}{D_{c_i}} \geq 1$

Accept x_k as the next cluster centre and continue

else

Reject x_{Ck} and set the density at x_{Ck} to 0

Select the data point with the next highest density as the new x_{Ck} and re-test

end if

end if

Here, ε^{up} specifies a threshold above which the point is selected as a centre, and ε^{down} specifies the threshold below which the point is definitely rejected. Typically, $\varepsilon^{up} = 0.5$ and $\varepsilon^{down} = 0.15$. If the density measure fails in the gray region, then checking of data points is required to identify where they provide a good trade-off between having a significant density measure and being sufficiently far from existing clusters. At the end of clustering procedure, a set of fuzzy rules will have been obtained. Each cluster will represent a rule. However, since the clustering procedure is conducted in a multidimensional space, fuzzy sets must be obtained. As each axis of the multidimensional space refers to a variable, the centres of the membership functions for that variable are obtained by projecting the centre of each cluster in the corresponding axis. As for the widths, they are obtained on the basis of the neighbourhood radius r_a , defined while performing subtractive clustering. Since Gaussian membership functions are used, their standard deviations are computed as follows:

$$\sigma_{ij} = r_a \frac{\max(x_{kj}) - \min(x_{kj})}{\sqrt{8}}, k = 1, \dots, N \quad (3.36)$$

3.5 Design of control systems architectures for autonomous mobile robots

The control architecture for autonomous mobile robots presented in chapter six is hybrid. The scope of this section is to classify and discuss the main design methodology of control systems architectures for autonomous mobile robots. In section 3.5.1 some definitions related to design of control system architectures are given. Section 3.5.2 presents a short classification of the currently used control architectures. The main key issues when designing such control system architectures for complex control systems are discussed in section 3.5.3. In section 3.5.4 the

most important properties, which can characterise a control system architecture for autonomous mobile robots, is discussed.

3.5.1 Definitions used in design of control system architectures for autonomous mobile robots

In most books and articles terms such as robot, autonomy, etc, are used without any explicit definition. Sometimes this is confusing as definitions regarding the same thing are variance to each other. Some definitions of the main terms used in this thesis are given in the following, just for consistency purposes. Note, that the following definitions cannot be treated or considered to be unique or the best.

Robot: According to the Robotics Industry Association (RIA) “*a robot is a programmable, multi-functional, manipulator designed to move material, parts, tools, or specialised devices through variable programmed motions for the performance of a variety of tasks*”. This definition is quite restrictive, excluding mobile robots, among other things. In this thesis the robot’s definition used is the one by (Arkin, 1998) in which “*robot is a machine able to extract information from its environment and use knowledge about its world to move safely in a meaningful and prospective manner*”.

Autonomy: For the scope of this thesis autonomy will be defined “*as the ability of a system to perform complex tasks without human guidance while coping with an unknown and changing environment*”.

Autonomous mobile robot: (Russel and Norvig, 1995) has given the following definition for an autonomous mobile robot. “*An autonomous mobile robot can make decisions on its own, guided by the feedback it gets from its physical sensors*”. This definition fully defines the term of autonomous robot used in this thesis.

Behaviour: *What an autonomous robot is observed doing. It can be seen also as the result of an interaction of a robot with its environment (Pfeifer and Scheier, 1999).*

Behaviour control: *Set of mechanisms that determine the behaviour in which the robot will engage (Pfeifer and Scheier, 1999).*

Behaviour-based robotics: *When the overall robot design is decomposed, not into functional components (learning, planning, etc.) but into a number of behaviours such as avoid obstacle, reach target, etc (Brooks, 1986).*

Control architecture: Two definitions have been chosen to define control architecture. The first is by (Pfeifer and Scheier, 1999) which states: “*structure that determines the robot-environment coupling, that is how the sensory and motor signals are processed to produce behaviour*”. The second definition is also given by (Pfeifer and Scheier, 1999) and states: “*the control architecture of a robot defines how the job of generating actions from percepts is organised*”

3.5.2 Classification of control architectures

According to (Ridao *et al*, 1999) the control architectures can be classified in three main categories: deliberative, reactive and hybrid. Each category is discussed in brief in the following sections. More details can be found in (Muller, 1997), (Ridao *et al*, 1999) and (Senehi and Kramer, 1998).

3.5.2.1 Deliberative architectures

These architectures are strongly based on traditional artificial intelligence techniques based on planning and also on a world model. Deliberative architectures are usually very difficult to

adapt in a dynamic environment where changes are taking place very fast. In general, they present a predictable behaviour with some degree of reactivity through re-planning. Two major architectural principles that are often incorporated in a deliberative architecture are the hierarchical and centralised structure. Hierarchical architectures use a functional and hierarchical decomposition where the tasks are decomposed into subtasks and the control is organised in progressive levels with different levels of data abstraction. Centralised architectures (see section 3.5.3.1) are organised as a set of modules communicated through a central control module.

3.5.2.2 Reactive architectures

The main philosophy behind the reactive architectures (Brooks, 1986), also known as behavioural architectures, is that the modules in which the system is built up of are behaviour producing instead of functional as in deliberative architectures. Normally, the missions are described as a sequence of phases with a set of active behaviours. The behaviours continuously react to the situation sensed by the perception system. Each of these behaviours pursues its own goal and can be defined using rules or any kind of link between sensor and actuator module. The main robot's global behaviour emerges from the combination of the elemental active behaviours. These types of architectures are more suited for dynamic environments where changes are taking place very fast.

3.5.2.3 Hybrid architectures

Hybrid architectures are integrating deliberative and reactive activities, where deliberative elements are used for planning and problem solving and reactive elements are used for obtaining a quick response action to situations that the system is not able to predict (real-time control). According to (Ridao *et al*, 1999) hybrid architectures are normally structured into three layers: planning layer, control execution layer and functional reactive layer. At the upper layer a planner transforms the mission into a set of tasks to be executed by the control system

(symbolic reasoning takes places at this layer). The second layer may be seen as a task refinement layer plus its execution control. In this case the execution control drives the system into a sequence of phases, where each phase is characterised by a set of active behaviours. Finally the last layer is a function reactive layer, normally executed asynchronously to the rest of the system.

3.5.3 Main key issues of the design of control systems architectures for multiple autonomous mobile robots

As mentioned in the introduction of this thesis the use of multiple robots offers a wide range of advantages over the use of a single robot. However, the architecture design requirements for a multiple mobile co-operative robot system are complex and have several differences compared to single robot architecture. In the following sections the six key issues, regarding the architecture design of multiple autonomous mobile robots is discussed in brief.

3.5.3.1 Centralised, decentralised or hybrid control

The first decision that has to be made is whether the control architecture will be centralised, decentralised or one form of hybrid. There is no specific law or any particular restriction of which control form is the best. This is because the control architecture that is suitable for a given task may not be flexible and suitable for another. In centralised control architecture, decisions are made in a central mechanism or in a single control unit (agent), and afterwards transmitted to the executive components (robots). Due to the complexity of the hierarchical planning system the development of these architectures is difficult because it is not easy to determine the suitability of this concept in advance. With the decentralised control architecture, each robot makes its own decisions and performs only these decisions without having any connection with a central mechanism or single control unit. This is very important for a multiple mobile co-operative robot system, because problems such as fault-tolerance, the difficulty of adding new features into the system, and re-implementation of the system are automatically

avoided. The research literature has been dominated with works on decentralised control architecture, avoiding the centralised approach. Hybrid control architecture - often designers decide to adopt a hybrid solution of the decentralised and centralised approach. If the problem lies where the decentralised system needs an internal central unit a hybrid solution has to be adopted.

3.5.3.2 Heterogeneous or homogeneous robots

A second step in the design of the architecture for multiple mobile co-operative robots is the selection between heterogeneous and homogeneous robot characteristics. The robots should be either homogeneous or heterogeneous, depending on the task being undertaken. A homogeneous robot team consists of a number of robots, which have the same skills and capabilities. Most of the research projects involve homogeneous robot teams. This choice makes the design process much easier, because it minimises the complexity of task allocation. Co-operation may involve a team of robots provided with different skills (the mechanical design may be different also), referred to as heterogeneous robots. The complexity of the design is increased dramatically compared with the design process with homogeneous robots.

3.5.3.3 Co-operation with or without communication

Communication between multiple robots may make the team able to perform some co-operative tasks more efficiently. However, achieving co-operation within the robot team with communication produces several advantage i.e. achieving very complex tasks and also disadvantages i.e. waiting time, and transmission error. The above benefits and restrictions of communication led the research community to distinguish between explicit (with) or implicit (without) communication. If the robots (agents) communicate with each other directly, or if there is broadcast communication between them, then, this form of communication is called explicit communication. If the robots do not communicate with each other, or there is no

broadcast communication between them, but there is communication through the world environment, then this form of communication is called implicit communication.

3.5.3.4 Making robots that work as a team

The key issue in team working is how well the modelling between robots and environment has been developed. This key issue has a strong relationship with the communication key issue (modelling of purely communicating robots, or not).

3.5.3.5 Multiple robots path planning

Multiple robot path planning differs from single robot path planning in several ways. A mobile robot has to avoid obstacles and also other robots. To address this particular problem is very complicated problem, which should be taking into account when designing control architecture of autonomous mobile robots.

3.5.3.6 Learning

The final key issue in architecture design is learning in multiple robot systems. Changes within the robot's environment can result in poor robot performance and decreases the ability to achieve a given task. One solution to this problem is to introduce a learning method. The chosen method will increase the robot's performance and its ability to respond correctly to environment changes. This desirable result will be achieved by learning, so the robot system will be able to optimise and set its own control features. The main problem is that compared to single robot learning, co-operative learning adds the challenge of a much larger search space, awareness of other team members, and also the synthesis of the individual behaviours with respect to the task given to the group.

3.5.4 Properties of control architectures

When comparison between different types of control architectures is undertaken, this comparison should be made based on a number of important properties. (Pettersson, 1997) and (Yavuz and Bradshaw, 2002) have proposed a number of properties for comparison between different types of control architectures. Classification among these properties is proposed and discussed in brief as follows:

Modularity. In order to facilitate adding of new components or functionality into a control architecture then modularity is important. A modular architecture can produce a flexible robot(s) that can be easily adapted to different applications and environments. Modularity can be achieved either by focusing on independent modules or independent behaviours.

Robustness. The control architecture should be able to continue to function during unexpected situations. Reactive and hybrid architectures are generally more robust than deliberative ones.

Fault tolerance. In order for a robot to be able to achieve its goal in an environment that is dangerous or unsuitable for humans despite component failures it must be able to function without any possibility of repair. In that case the robot must be able to detect and isolate possible faults.

Distribution. Related to the fault tolerance of architecture is its distribution. As mentioned in section 3.5.3.1 the problem of centralised architecture is that the central parts become a bottleneck that slows down the control system. The problem, which may occur utilising distributed architecture, is communication and co-ordination that can be very difficult. An example of a fully distributed architecture is ALLIANCE developed by (Parker, 1999) which is tolerant to faults in the distributed modules.

Reactivity. This property of control architecture provides the robot(s) with the ability to act in short and predictable time on any sensory input. Reactivity is very important if the mobile robot has to operate in unknown and unstructured environment. Behavioural architectures are more suited to this property whereas this can often be a problem with deliberative ones as some type of modelling before acting on the sensory input is required.

Adaptability. Adaptability is considerable property, since control architectures will need to be adapted and extended during the lifetime of the robot. An important aspect of adaptability is the ability to adapt to changes in the environment by dynamic learning or dynamic switching. In some cases in order to achieve adaptability, integration of problem solving and learning in the control system is considered.

Planning. This property allows the robot to simulate itself and its environment in real time (unsuitable for reactive architectures).

Co-operation. One-way to increase modularity, distribution and robustness is to use co-operative agents. Co-operative agents having different skills and capabilities for specific subtasks, have the ability to solve problems more efficiently and effectively than single agent. An important aspect of co-operation is communication between the agents (communication of goals, states, etc.).

Uncertainty. Uncertainty has been one of the main aspects of study in mobile robotics because plays an important role in many real-time applications (Min *et al*, 1997). As *prior* knowledge of the environment may be incomplete and time variant, therefore uncertainty is one of the main design constraints in control systems architecture. The control architecture should be able to

cope with the dynamic changes in the environment that occur simultaneously with the operation of the robot.

Learning. On-line learning also is an important issue to be considered as it has promising features for design and implementation of more flexible and adaptive control systems architectures.

Goal oriented. Navigation of the mobile robot implies the meaningful progress towards the achievement of the goal. Therefore control architectures must combine deliberative with reactive planning in order to be successful in navigational tasks.

Efficiency. Control architecture designed for complex- real-time systems must provide the means by which the system may accomplish its objectives efficiently. The control architecture must be able to satisfy real-time constraints, safety and promote fault-tolerance.

Easy of application. Crucial consideration in the design of control architectures for mobile robots is the ease with which a system may be developed, tested, debugged and understood.

Optimal control (operation). Control architectures must be able to provide control design methods to choose the best behaviour or controller in an optimal manner. Therefore control architectures must incorporate deliberative reasoning and hierarchical structures.

3.6 Multi-agent systems (MAS) in control engineering

As mentioned previously the new control system architecture in chapter six is hybrid but also is multi-agent type constructed and orientated. During the past few years, the need for large scale and complex systems has become obvious. The main problem is often the design of the intelligent control structure for such complex systems, and also for system components if the

overall system consists of several separate systems (i.e., controllers). Multi-agent systems (MAS) theory is a relatively new field in control and systems engineering and can be used successfully to solve the aforementioned problem. A special role in the theory and tools for solving complex control problems is attributed to the concept of agent⁴. An agent represents an abstract entity that is able to solve a particular (partial) problem. Agents have the ability to be combined into a multi-agent system, such that the overall multi-agent system is able to solve a more complex problem. In this section the background information of MAS, classification of the main agent control architectures and the concept to construct local controllers that consist of several other controllers using MAS is presented.

3.6.1 Autonomous agents

In traditional artificial intelligence (AI) and cognitive science, computer models have been the predominant tools. Synthetic methodology, however, can be extended to include not only simulations, but also physical systems, artificial creatures, behaving in the real world (Pfeifer and Scheier, 1999). As mentioned in section 3.5.1 “autonomous” designates independence from human control. Typically, autonomous agents have the form of mobile robots and can be used as models of biological systems, human or animals. To date, autonomous agents behave in the real world without the intervention of a human. They have sensors to perceive the environment, and they perform actions that change the environment. In other words autonomous agents are systems in their own right as shown in Figure 3-10. This is the main reason why autonomous agents are well suited to explore issues in the study of control engineering and artificial intelligence in general.

⁴ Introducing the concept of an agent in a precise and technical manner is a difficult thing to do, as there is no generally accepted definition of it (Van Breemen and De Vries, 2001). In this thesis the word agent is used to represent both a physical entity (i.e., robot) and virtual entity (software component). Good source of reference regarding variety of agent definitions is given in (Ferber, 1999).

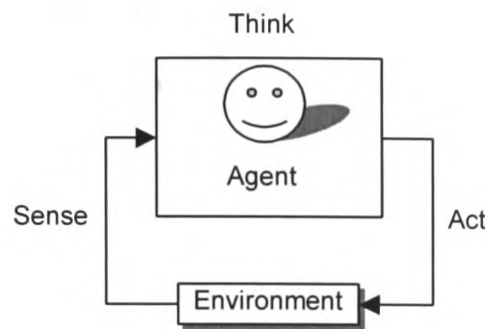


Figure 3-10 An agent as entity (physical or virtual) that senses, thinks and acts in some environment in order to achieve its goal.

3.6.2 Multi-agent systems

Despite the fact that agents are often presented as entities that solve problems in order to realise their goal, sometimes many problems are far too complex to be handled by an individual agent. The solution to such complex problems can be often obtained using a group of agents. From the terminology point of view a group or society of agents is called multi-agent system (MAS). In recent years the study of such systems (MAS) has become a new field of research (Stone and Velose, 2000) and (Van Breemen and De Vries, 2001). (Lesser, 1998) has defined MAS as a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver. According to (Jennings *et al*, 1998), a general design methodology of MAS does not exist. (Stone and Velose, 2000) have identified several good reasons justifying why a MAS approach should be adopted. Some of these reasons include: *distributed problem* (a MAS is suitable for problems which are distributed in nature), *robustness* (a MAS that has redundant agents might tolerate failures in one or several of the agents, and is thus more robust than a centralised system), *scalability* (because agents are modular, it should be easy to add and remove agents from the MAS), *simpler implementation* (again because agents are modular, implementing a MAS should be

easier than implementing one overall centralised system), and *parallelism* (to speed up the computation time needed for solving a problem, some parts could be executed in parallel). Detail background information about design aspects, co-ordination, structure and some of the key issues in agent technology of MAS, can be in found in the work of (Lesser, 1998), (Ferber, 1999) and (Van Breemen and De Vries, 2000).

3.6.3 Agent control architectures

Agents are not common in control engineering. Possible reasons behind this could be the following. Firstly, the field of multi-agent systems is relatively new. This means that merging MAS and control engineering has not get happened. Secondly, merging MAS and control engineering is very challenging and difficult task, because control theory has a strong mathematical foundations, whereas the field of MAS is mainly focused on abstract descriptions of the system.

At the present time, it is a non-trivial problem to design the architecture of an agent, given a specification of its behaviour. In the literature it has been documented that there are several different reasoning models which an agent can possess (Pfeifer and Scheier, 1999). However each particular model can be implemented by several different control architectures. (Ferber, 1999) presents and discuss a list of the main agent control architectures based on type of approach, subordination structure, pairing and constitution as shown in Table 1, along with their associated parameters. The new hybrid control architecture presented in chapter six draws its design from the most of the main agent control architectures shown in Table 1 but most from competitive tasks architecture, production rules architecture, connectionist, dynamic system and multi-agent architecture.

Type of architecture	Approach	Type of component	Subordination structure	Coupling structure	Constitution
Horizontal modular	Horizontal functional	Module	Hierarchical	Fixed (but progressive)	Predefined
Blackboard	Functional	Task	Hierarchical (Meta)	Variable	Predefined
Subsumption	Vertical functional	Primitive task	Hierarchical	Fixed	Predefined
Competitive tasks	Vertical functional	Task + Primitive task	Hierarchical (Competition)	Variable	Predefined
Production rules	Functional	Rule	Hierarchical (Meta)	Variable	Predefined
Classifiers	Vertical functional	Rule	Hierarchical	Evolutionary	Predefined
Connectionist	Vertical functional	Formal neuron	Egalitarian	Fixed (by weight)	Predefined
Dynamic system	Vertical functional	Stimuli-command relationship	Egalitarian	Fixed (but progressive)	Emergent
Multi-agent	Object/functional	Agent	Egalitarian	Variable	Emergent

Table 1 Main agent control architectures (after (Ferber, 1999))

3.7 Stateflow design tool based on finite state machines theory

The methodology chosen in chapter six for the design and modelling of global state identification mechanism, supervisor-like co-ordination object responsible for several local controller-agents and also the tool for identification of direction of neighbour robots is stateflow, which is based on the theory of finite state machines (FSM).

3.7.1 Finite state machines (FSM)

Finite state machines are widely used in the modelling and control of system in various areas (Moor and Raisch, 1999), (Singh and Nowick, 2000), (Carter, 2001), (Sato and Gohara, 2001), (Giua, 2001) and (Roze and Cordier, 2002). Descriptions using FSM are useful to represent the flow of complex control problems and are amenable to formal analysis such as model and control algorithm checking. An FSM consists of a finite number of states and conditional transitions between them. The FSM reads input symbols from the finite alphabet and produces output symbols (actions) taken from another (or possible the same) finite alphabet, while

jumping between the different states. More details in the operation of FSM can be found in (Gill, 1962), (Harel, 1987), (Hatley and Imtiaz, 1988), (Villa *et al*, 1997) and (Kam *et al*, 1997). In the following, the basic definition of a FSM either as non-deterministic or deterministic is given.

Definition 3-14 (non-deterministic finite state machine or simple FSM): A NDFSM is defined as a 5-tuple $M = \langle S, I, O, T, R \rangle$ where S represents the finite state space, I represents the finite input space and O represents the finite output space. T is the transition relation defined as a characteristic function $T: I \times S \times S \times O \rightarrow B$. On an input i , the FSM at present state p can transit to a next state n and output o if and only if $T(i, p, n, o) = 1$ (i.e., (i, p, n, o) is a transition). There exist one or more transitions for each combination of present state p and input i . $R \subseteq S$ represents the set of reset states.

Definition 3-15 (deterministic finite state machine or completely specified FSM): A DFSM is defined as a 6-tuple $M = \langle S, I, O, \delta, \lambda, r \rangle$ where S represents the finite state space, I represents the finite input space and O represents the finite output space. δ is the next state function defined as $\delta: I \times S \rightarrow S$ where $n \in S$ is the next state of present state $p \in S$ on input $i \in I$ if and only if $n = \delta(i, p)$. λ is the output function defined as $\lambda: I \times S \rightarrow O$ where $o \in O$ is the output of the present state $p \in S$ on input $i \in I$ if and only if $o = \lambda(i, p)$. $r \in S$ represents the unique reset state.

3.7.2 Stateflow

The stateflow design tool provided by (The MathWorks, 1997) is a powerful graphical design and development tool for complex control and supervisory logic problems. The advantages of using stateflow can be summarised as follows: model visualisation, simulation of complex

reactive systems based on FSM, design and development of deterministic supervisory control systems, easily design modification, evaluation of results, verification of system's performance at any stage of the design and integration with Simulink platform for system analysis and modelling. Another, important property of stateflow is that enables the representation of hierarchy, parallelism and history. Hierarchy is useful for designing very complex control systems, parallelism is useful as two or more orthogonal states can be active at the same time and history provides the means to specify the destination state of a transition based on historical information. An example of stateflow diagram is shown in Figure 3-11. The main components of the diagram are briefly discussed in the following.

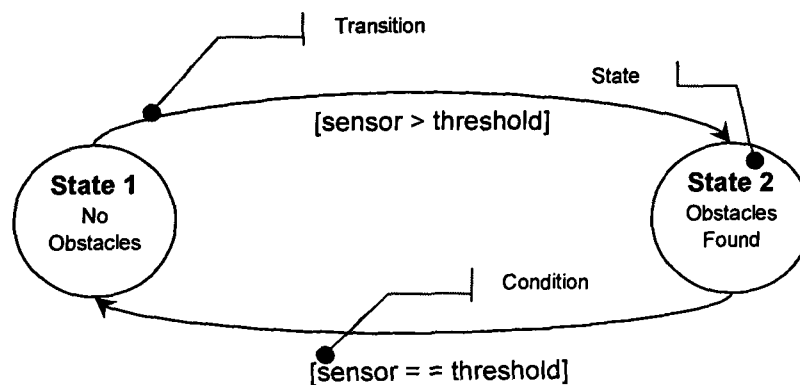


Figure 3-11 An example of 2-state stateflow diagram based on FSM theory

As can be observed from Figure 3-11, state⁵, transition and condition are the main components of the stateflow diagram (more details of all components comprising a stateflow diagram can be found in (The MathWorks, 1997)). State describes a mode of an event-driven system. The activity or inactivity of the state dynamically changes based on events and conditions (events are non-graphical objects and thus not represented directly in the Figure 3-11). A transition is a graphical object that links, in most cases, one object to another. One end of the transition is attached to a source object and the other end to a destination object. The source is where the

⁵ The idea of "state" as a concept in the representation of systems was first introduced in 1936 by Turing, A. M. (TURING, A. M. 1936. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. London Math. Soc.*, **42** (2), pp. 230-265)

transition begins and the destination is where the transition ends. A transition describes the circumstances under which the system moves from one state to another. A condition is a Boolean expression specifying that a transition occur, given that the specific expression is true. Figure 3-11 shows that if a sensor is equal to threshold then no obstacle has been found, thus the system it will stay in state 1. If the sensor is greater than the threshold then the system will jump to state 2. The system will return to state 1 if the sensor reading is equal to a threshold.

3.8 Discussion

As mentioned in the introduction the main scope of this chapter is to propose, justify and present the main research methodology adopted for the research work carried out in this thesis. The main methodology was broken down into nine basic steps. Steps one, two and three present the design methodology for the low-level control of the proposed control architecture, which is modelling and control of the MIABOT V2 mobile robot. A new concept within the first three design steps is the constrained optimisation using the non-linear design (NCD) tool. Although constrained optimisation is well documented in the literature NCD is a relatively new tool providing both tuning of control parameters (controller gains) and optimisation/identification of either physical parameters (optimisation of physical parameters, such as moment of inertia can be achieved). The next three steps propose the design methodology of fuzzy, neural and clustering techniques. One of the main advantages of fuzzy systems is that there is no need to have a mathematical model of the system, it is possible to control non-linear plants and using comprehensive linguistic rules and it is possible to implement expert human knowledge and experience. However, it is inappropriate to note that fuzzy systems cannot solve all control problems. Like any other design methodology the use of fuzzy logic has some drawbacks, such as no global or systematic method for the transformation of the human experience into the rule base of the fuzzy system. In addition, it is not possible to demonstrate stability of the controlled system since the model is not known and it is not guaranteed that rules are coherent (the possibility of mismatch between the rules exists). The advantages of neural networks have

already discussed in the introduction of this chapter. However, it is important to discuss and identify drawbacks of ANN, if any, and possible similarities with fuzzy systems. A considerable drawback of ANN is that knowledge extraction and knowledge representation are difficult. This results in some kind of integration between fuzzy and neural systems. For instance, automatic design and fine-tuning of the membership functions used in fuzzy control through learning by neural networks. Clustering techniques described in section 3.4 can be considered as a helping tool for developing both fuzzy and neural systems. The final three steps of the proposed design methodology are very closely related. In chapter six, the final three steps of the proposed methodology are merged for the total integration of the control system. In section 3.5 it was shown the background information of the design of control systems architecture for autonomous mobile robot followed by the main design methodology of multi-agent systems in section 3.6 and stateflow design in section 3.7. It was shown that there are still open areas for research on merging of different fields, such as multi-agent systems and control engineering.

3.9 Summary

As mentioned in the introduction of this chapter there is no formal method for developing integrated solutions for advanced mobile robotic systems. This chapter proposes, justifies and presents the main methodology adopted for the research work carried out in this thesis. The main methodology has been broken down into nine basic steps. Each step is presented individually focusing on design/modelling issues following a discussion of either advantages or disadvantages when the particular method is adopted. The first step concerns conventional control design, which is used in chapter five to model speed controller for the mobile robot. The second step refers to constrained optimisation using non-linear control design tool for tuning/optimisation of physical and control parameters in chapters four and five. The design methodology for fast robust stability testing and analysis based on interval polynomials is discussed in the third step, where, using the parametric robustness analysis approach (Kharitonov's Theorem) the closed-loop control system (controller and plant) is proved to be

robustly stable under uncertainty in robot dynamics. Steps four, five and six discuss the main methodology for modelling and identification of local controllers using fuzzy logic systems, artificial neural networks and clustering techniques. Steps seven and eight define the methodology regarding the design of control systems architectures of mobile robots and multi-agent systems as additional tool in development of control architectures. The stateflow design tool based on finite state machines theory is the final step of the proposed design methodology. Using this tool, model visualisation and construction of complex reactive systems can be achieved. In particular this design tool is used in chapter six as a global state identification mechanism, supervisor-like co-ordination object for several local controllers-agents and also as tool for identification of direction of neighbour robots.

The contributions of this chapter are: A proposed design methodology for developing integrated solutions for autonomous mobile robotic systems and classification of the main design methodology (properties) of control systems architectures for autonomous mobile robots. Fuzzy systems, neural networks and clustering techniques are described under a unifying theory. Discussion based on merging multi-agent systems and control engineering.

In the next chapter the design methodology of steps one and two is used for the derivation and establishment of a dynamic model of the MIABOT V2 mobile robot.

3.10 References

- ARKIN, R. C. 1998. *Behaviour-Based Robotics*. USA: MIT Press. 0-262-01165-4.
- ASTROM, K. J. and WITTENMARK, B. 1997. *Computer-Controlled Systems*. 3rd end. USA: Prentice Hall. 0133148998.
- BERKAN, R. C. and TRUBATCH, S. L. 1997. *Fuzzy systems Design Principles: Building Fuzzy IF-THEN Rule Bases*. USA: IEEE Press. 0-7803-1151-5.
- BEZDEK, J. 1974. Cluster Validity with Fuzzy Sets. *Journal of Cybernetics*, **3** (3), pp. 58-71.

- BEZDEK, J. C. 1981. *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press. 0-306-40671-3.
- BEZDEK, J. C., KELLER, J., KRISNAPURAM, R. and PAL, N. R. 1999. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. USA: Kluwer Academic Publishers.
- BHATTACHARYYA, S. P., CHAPPELLAT, H. and KEEL, L. H. 1995. *Robust Control: The Parametric Approach*. USA: Prentice-Hall Inc. 0-13-781576-X.
- BROOKS, R. A. 1986. A Robust Layered control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, **RA-2** (1), pp. 14-23.
- CARTER, J. 2001. Finite State Machines. *Power Engineering Journal*, **15** (1), pp. 15.
- CHIU, S. L. 1994. Fuzzy Model Identification Based on Cluster Estimation. *Journal of Intelligent and Fuzzy Systems*, **3** (2), pp. 267-278.
- CHURCHLAND, P. S. and SEJNOWSKI, T. J. 1992. *The Computational Brain*. USA: MIT Press. 0262531208.
- CICHOCKI, A. and UNBEHAUEN, R. 1994. *Neural Networks for Optimisation and Signal Processing*. USA: John Wiley & Sons. 0471930105.
- COX, E. 1999. *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems*. 2nd end. USA: Academic Press. 0-12-194456-5.
- CYBENKO, G. 1989. Approximation by Superposition of Sigmoidal Functions. *Mathematics of Control, Signals and Systems*, **2**, pp. 183-192.
- DE VILLIERS, J. and BARNARD, E. 1993. Backpropagation Neural Nets with One and Two Hidden Layers. *IEEE Transactions on Neural Networks*, **4** (1), pp. 136-141.
- DRIANKOV, D. and PALM, R. 1998. *Advances in Fuzzy Control*. USA: Physica-Verlag. 3-7908-1090-8.
- DUNN, J. 1974. A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact, Well Separated Cluster. *Journal of Cybernetics*, **3** (3), pp. 32-57.
- FAEDO, S. 1953. A New Stability Problem for Polynomials with Real Coefficients. *Ann. Scuola Norm. Pisa Sci. Fis. Mat. Ser.*, **3** (7), pp. 53-63.
- FAUSSETT, L. V. 1994. *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*. USA: Prentice-Hall. 0133341860.
- FERBER, J. 1999. *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. New York, USA: Addison Wesley Longman Inc. 0-201-36048-9.
- FUNAHASHI, K. 1989. On the Approximate Realisation of Continuous Mappings by Neural Networks. *Neural Networks*, **2** (3), pp. 183-192.
- GAJIC, Z. and LELIC, M. 1996. *Modern Control Systems Engineering*. UK: Prentice Hall Europe. 0-13-134116-2.
- GANONG, W. F. 1987. *Review of Medical Psychology*. USA: Prentice-Hall, Inc. 0838582826.

- GILL, A. 1962. *Introduction to the Theory of Finite-State Machines*. New York: McGraw-Hill Book Company, Inc.
- GIUA, A. 2001. Model Reduction of Finite-State Machines by Contraction. *IEEE Transactions on Automatic Control*, **46** (5), pp. 797-801.
- HAREL, D. 1987. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, **8** (3), pp. 231-274.
- HARRIS, C. J. 1994. *Advances in Intelligent Control*. UK: Taylor & Francis Ltd. 0-7484-0066-4.
- HATLEY, D. J. and IMTIAZ, A. P. 1988. *Strategies for Real-Time System Specification*. New York: Dorset House Publishing Company, Inc.
- HAYKIN, S. 1999. *Neural Networks: A Comprehensive Foundation*. 2nd end. USA: Prentice-Hall, Inc. 0-13-273350-1.
- HERTZ, J., KROGH, A. and PALMER, R. G. 1991. *Introduction to the Theory of Neural Computation*. USA: Addison-Wesley. 0201515601.
- HESKE, T. and HESKE, J. N. 1996. *Fuzzy Logic for Real World Design*. USA: Annabooks. 0-929392-24-8.
- HORNIK, K., STINCHCOMBE, M. and WHITE, H. 1989. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, **2** (5), pp. 359-366.
- JANG, J. S., SUN, C. T. and MIZUTANI, E. 1997. *Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence*. USA: Prentice-Hall, Inc. 0-13-261066-3.
- JENNINGS, N. R., SYCARA, K. and WOOLDRIDGE 1998. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, **1**, pp. 275-306.
- KAM, T., VILLA, T., BRAYTON, R. K. and SANGIOVANNI-VINCENTELLI, A. 1997. *Synthesis of Finite State Machines: Functional Optimisation*. USA: Kluwer Academic Publishers. 0-7923-9892-4.
- KHARITONOV, V. L. 1978. Asymptotic Stability of an Equilibrium Position of a Family of Systems of Linear Differential Equations. *Differential Uraunen*, **14**, pp. 2086-2088.
- KLIR, G. J. and YUAN, B. 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. USA: Prentice-Hall Inc. 0-13-101171-5.
- KOHONEN, T. 1982. Self-organised Formation of Topological Correct Feature Maps. *Biological Cybernetics*, **43** (1), pp. 59-69.
- KOHONEN, T. 1989. *Self-organisation and Associative Memory*. 3rd end. USA: Springer Verlag. 0387513876.
- KONTOGIANNIS, E. and MUNRO, N. 1996. *The Fundamental Dominance Condition for MIMO Systems with Parametric Uncertainty*. Proceedings of the IEE/IFAC Control '96. pp. 1202-1207, UK.

- KOSKO, B. 1992. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. USA: Prentice-Hall International, Inc. 0-13-612334-1.
- KU, C. C. and LEE, K. Y. 1995. Diagonal Recurrent Neural Network for Dynamical Systems Control. *IEEE Transactions on Neural Networks*, **6** (1), pp. 144-155.
- LESSER, V. R. 1998. Reflections on the Nature of Multi-agent Coordination and its Implications for an Agent Architecture. *Autonomous Agents and Multi-agent Systems*, **1**, pp. 89-111.
- LEVENBERG, K. 1944. A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly Journal of Applied Mathematics and Physics*, **25** (2), pp. 164-168.
- LI, H., CHEN, G. L. P. and HUANG, H. P. 2001. *Fuzzy Neural Intelligent Systems: Mathematical Foundation and the Applications in Engineering*. USA: CRC Press. 0-8493-2360-6.
- MAMDANI, E. H. and ASSILEAN, S. 1975. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Machine Studies*, **7** (1), pp. 1-13.
- MAMDANI, E. H. and GAINES, B. R. 1981. *Fuzzy Reasoning and its Applications*. USA: Academic Press. 0-12-467750-9.
- MARQUARDT, D. W. 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *J. Soc. Indust. Appl. Math*, **11** (2), pp. 431-441.
- MCCULLOCH, W. S. and PITTS, W. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, **5**, pp. 115-133.
- MIN, B. K., CHO, D. W. and LEE, S. J. 1997. Sonar Mapping of a Mobile Robot Considering Position Uncertainty. *Robotics Computer Integrating Manufacturing*, **13** (1), pp. 41-49.
- MOOR, T. and RAISCH, J. 1999. Supervisory Control of Hybrid Systems Within a Behavioural Framework. *Systems & Control Letters*, **38** (3), pp. 157-166.
- MULLER, J. P. 1997. Control Architectures for Autonomous and Interacting Agents: A Survey. *Lecture Notes in Artificial Intelligence*, **1209**, pp. 1-26.
- NG, G. W. 1997. *Application of Neural Networks to Adaptive Control of Intelligent Systems*. UK: Research Studies Press Ltd. 0-86380-214-1.
- NGUYEN, H. T. and WALKER, E. A. 2000. *A First Course in Fuzzy Logic*. 2nd end. USA: Chapman & Hall/CRC. 0-8493-1659-6.
- PAL, N. R. and CHAKRABORTY, D. 2000. Mountain and Subtractive Clustering Method: Improvements and Generalizations. *International Journal of Intelligent Systems*, **15** (4), pp. 329-341.
- PARKER, L. E. 1999. Adaptive Heterogeneous Multi-Robot Teams. *Neurocomputing*, **28**, pp. 75-95.
- PASSINO, K. M. and YURKOVICH, S. 1998. *Fuzzy Control*. USA: Addison Wesley Longman, Inc. 0-201-18074-X.

PEDRYCZ, W. and GOMIDE, F. 1998. *An Introduction to Fuzzy Sets. Analysis and Design*. USA: MIT. 0-262-16171-0.

PETTERSSON, L. 1997. *Control Systems Architectures for Autonomous Agents*. A Survey Study . pp. 1-16, Royal Institute of Technology.

PFEIFER, R. and SCHEIER, C. 1999. *Understanding Intelligence*. USA: The MIT Press. 0-262-16181-8.

PICTON, P. 1998. *Neural Networks*. 2nd end. USA: Palgrave. 0-333-80287-X.

REZNIK, L., GHANAYEM, O. and BOURMISTROV, A. 2000. PID plus Fuzzy Controller Structures as the Design Base for Industrial Applications. *Engineering Applications of Artificial Intelligence*, **13** (4), pp. 419-430.

RIDAO, P., BATLLET, J., AMAT, J. and ROBERTS, G. N. 1999. Recent Trends in Control Architectures for Autonomous Underwater Vehicles. *International Journal of Systems science*, **30** (9), pp. 1033-1056 .

ROSS, T. J. 1995. *Fuzzy Logic with Engineering Applications*. USA: McGraw-Hill Inc. 0070539170.

ROZE, L. and CORDIER, M. O. 2002. Diagnosing Discrete-Event Systems: Extending the "Diagnoser Approach" to Deal With Telecommunication Networks. *Discrete Event Dynamic Systems-Theory and Applications*, **12** (1), pp. 43-81.

RUMELHART, D. E., HILTON, G. E. and WILLIAMS, R. J. 1986. Learning Representations of Back-propagation Errors. *Nature*, **323**, pp. 533-536.

RUSSEL, S. and NORVIG, P. 1995. *Artificial Intelligence - A Modern Approach*. Englewood Cliffs, New Jersey: Prentice-Hall International, Inc.

SATO, S. and GOHARA, K. 2001. Fractal Transition in Continuous Recurrent Neural Networks. *International Journal of Bifurcation and Chaos*, **11** (2), pp. 421-434.

SENEHI, M. K. and KRAMER, T. R. 1998. A Framework for Control Architectures. *International Journal of Computer Integrated Manufacturing*, **11** (4), pp. 347-363.

SILJAD, D. D. 1989. Parameter Space Methods for Robust Control Design: A Guided Tour. *IEEE Transactions on Automatic Control*, **34** (7), pp. 674-688.

SINGH, M. and NOWICK, S. M. 2000. Synthesis for Logical Initializability of Synchronous Finite- State Machines. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **8** (5), pp. 542-557.

STONE, P. and VELOSE, M. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robotics*, **8** (3), pp. 345-383.

TAKAGI, T. and SUGENO, M. 1985. Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics*, **15** (1), pp. 116-132.

- TAYLOR, M. and LISBOA, P. 1993. *Techniques and Applications of Neural Networks*. UK: Ellis Harwood Ltd. 0-13-062183-8.
- THE MATHWORKS, I. 1997. *Stateflow User's Guide*.
- VAN BREEMEN, A. J. N. and DE VRIES, T. J. A. 2000. *An Agent-based Framework for Designing Multi-controller Systems*. Proceedings of the Fifth International Conference on the Practical Applications of Intelligent Agents and Multi-agent Technology. pp. 219-235, Manchester, UK.
- VAN BREEMEN, A. J. N. and DE VRIES, T. J. A. 2001. Design and Implementation of a Room Thermostat Using an Agent-based Approach. *Control Engineering Practice*, 9 (3), pp. 233-248.
- VILLA, T., KAM, T., BRAYTON, R. K. and SANGIOVANNI-VINCENTELLI, A. 1997. *Synthesis of Finite State Machines: Logic Optimisation*. USA: Kluwer Academic Publishers. 0-7923-9892-0.
- WELSTEAD, S. T. 1994. *Neural Network and Fuzzy Logic Applications in C++*. USA: John Willey & Sons, Inc. 0-471-30974-5.
- WERBOS, P. J. 1974. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences*. PhD Thesis. Harvard University.
- YAGER, R. R. and FILEV, D. P. 1994a. Approximate Clustering via the Mountain Method. *IEEE Transactions of Systems, Man and Cybernetics*, 24 (8), pp. 1279-1284.
- YAGER, R. R. and FILEV, D. P. 1994b. Generation of Fuzzy Rules by Mountain Clustering. *Journal of Intelligent and Fuzzy Systems*, 2 (3), pp. 209-219.
- YAGER, R. R. and FILEV, D. P. 1994c. *Essentials of Fuzzy Modelling and Control*. New York: John Wiley & Sons. 0471017612.
- YAVUZ, H. and BRADSHAW, A. 2002. A New Conceptual Approach to the Design of Hybrid Control Architecture for Autonomous Mobile Robots. *Journal of Intelligent & Robotic Systems*, 34 (1), pp. 1-26.
- YEN, J. and LANGARI, R. 1999. *Fuzzy Logic: Intelligence, Control and Information*. USA: Prentice-Hall, Inc. 0-13-525817-0.
- ZADEH, L. A. 1965. Fuzzy Sets. *Information and Control*, 8, pp. 338-353.
- ZIMMERMANN, H. J. 1991. *Fuzzy Set Theory - and Its Applications*. 2nd ed. USA: Kluwer Academic Publishers. 0-7923-9075-X.

4

Modelling of MIABOT V2 Mobile Robot

4.1 Introduction

As mentioned in the introduction of this thesis the design and testing of the new hybrid multi-agent control architecture presented in chapter six and seven is highly dependent on the accuracy of the mathematical model describing the system to be controlled. Therefore the main purpose of this chapter is to derive and establish a dynamic model of the MIABOT V2 mobile robot.

In the research on autonomous mobile robots, experiments are very important and many experimental approaches towards mobile robot research have been done. However, the heavy cost of a large number of experiments is a serious problem in the development of control algorithms. To avoid the cost of experiments, simulations using mathematical models of the plant have been a popular method for research on mobile robotics. For instance, to show results

of planning algorithms (Borenstein, 1995), (Will and Zak, 1997), (Louste and Liegeois, 2000), (Pruski and Atassi, 2000), (Tuneski *et al*, 2001) and (Egerstedt and Hu, 2002) to analyse and design motion control systems (Menezes de Oliveira *et al*, 2000), (Pajaziti *et al*, 2000), (Herrmann, 2001) and (Kodagoda *et al*, 2002) and to investigate sensor characteristics (Everett, 1995), (Bemporad *et al*, 2000) and (Han *et al*, 2001). Therefore for the design of control systems the importance of a mathematical model, describing the system to be controlled, is well recognised.

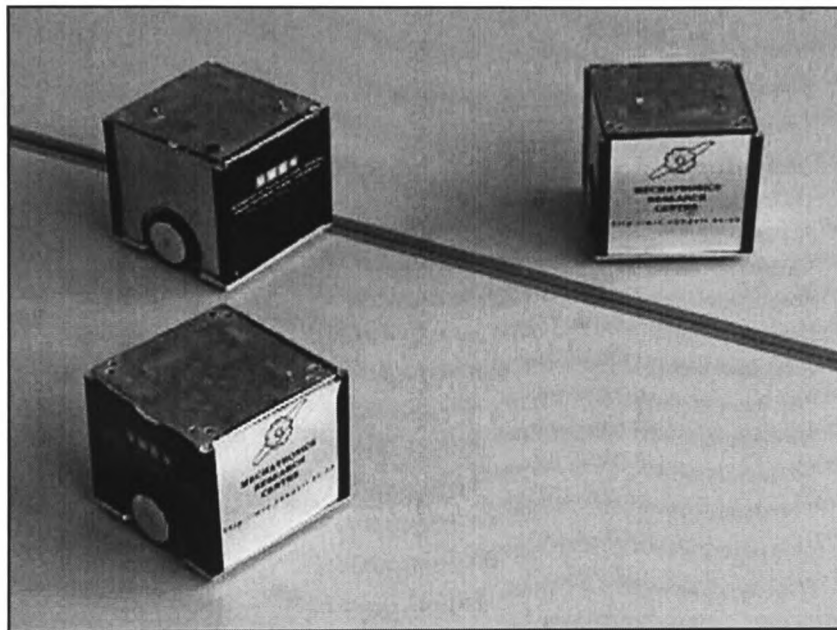


Figure 4-1 MIABOT V2 mobile robots

Modelling the behaviour of dynamic systems is quite a complex task. Generally the model should describe, acceptably well, the relevant dynamics of the plant. The more complex the model is, the more difficult the controller design will become. For this reason, it is common practice to distinguish between models used for control design purposes from those used for simulation purposes. In the former, only the relevant dynamics are represented, often in a way

that are compatible with the selected control design technique, while the latter it is important that the discrepancies between the real plant response and the model response is reduced.

In a general case, any mobile robot can be modelled with both a kinematic and a dynamic model. The kinematic model of a mobile robot is required to give its global description and to understand its manoeuvrability properties. However, there are cases in which the kinematic model is necessary in order to analyse the behaviour of the robot within the framework of the theory of nonholonomic systems. In this case the controllability, the reducibility, and the stabilisability of the kinematic model are also analysed. The dynamic model will give a complete description of the dynamics of the system including the generalised forces produced by the actuators. Similar to the kinematic model, the dynamic model can be used also to analyse controllability, reducibility, and the stabilisability.

However, the position accuracy of the robot is mainly affected by mechanical disturbances in which the most severe is wheel slippage (due to accelerations and fast turning). Most of the disturbances have been identified and are discussed below.

In order to minimise slip, the contact surface between wheels and floor should have a high friction coefficient, so rubber tyres are used. However, it is difficult to obtain rubber tyres with exactly the same diameter. In addition, distributed loads will slightly compress one tyre more than the other, thus changing its rolling radius. Wheels with different diameters cause the robot to travel along an arc, rather along a straight line, even if the motors are running at equal speeds. There is a contact area, rather than a contact point, between the wheel and the floor. This causes an uncertainty about the effective distance between the drive wheels, creating inaccuracies when turning. Another major mechanical disturbance is caused by misalignment of the drive wheels. This affect will produce a lateral drag force resulting in a curved path even when both wheels are exactly of the same diameter and are rotating at the same velocity. The condition of the

batteries affects the robot performance a great deal. If the batteries are ‘down’ there is a greater drop in voltage when they are loaded and vice versa. Finally it was found that at slow speeds the motors are not very reliable. This is more obvious particularly at starting when the input is low (this is basically due to friction). Taking into consideration all these disturbances, it is worthwhile to note that the modelling process is quite a complex and challenging task.

This chapter is organised as follows: Section 4.2 introduce the main features of the MIABOT V2 mobile robot. Analytically, from section 4.2.1 to 4.2.3 the main components of the mobile robot, the drive train and the DC motors are presented. In section 4.3 the kinematic model is derived with some discussion based on nonholonomic systems. Section 4.4 gives the full non-linear dynamic model of the robot, followed by, in section 4.5, the illustration of the linearised model. To verify the accuracy of the model both experiments and simulation studies were conducted. The main results of this chapter are presented in Section 0. A discussion follows in section 4.7 and finally the summary of the chapter is presented in section 4.8.

4.2 Main features of MIABOT V2 mobile robot

Merlin Systems Corporation Ltd, manufacture the MIABOT V2 mobile robot. The robot is currently used as a robot footballer in the Mechatronics Research Centre and in other Universities throughout the UK (Plymouth University, Salford University, Open University, Essex University, etc.). Therefore a précised dynamic model is required for construction and testing of control algorithms. Such a model is not available from the manufacturer. Prior to the derivation of both the kinematic and dynamic equations a brief description of the mobile robot is given (only the main components will be presented here, more details can be found in Appendix A).

4.2.1 Main components

Figure 4-2 shows an exploded view of the MIABOT V2 mobile robot. The main components of the mobile robot, indicated in Figure 4-2, are as follows: Modular style metal case (main body) 8cm^3 [1], main board (including CPU, H-Bridge, etc.) [2], two rechargeable Nickel Metal Hydride (NIMH) battery packs [3], drive train [4] (including motors [5], encoders [6]) and wheels [7] equipped with O-rings [8].

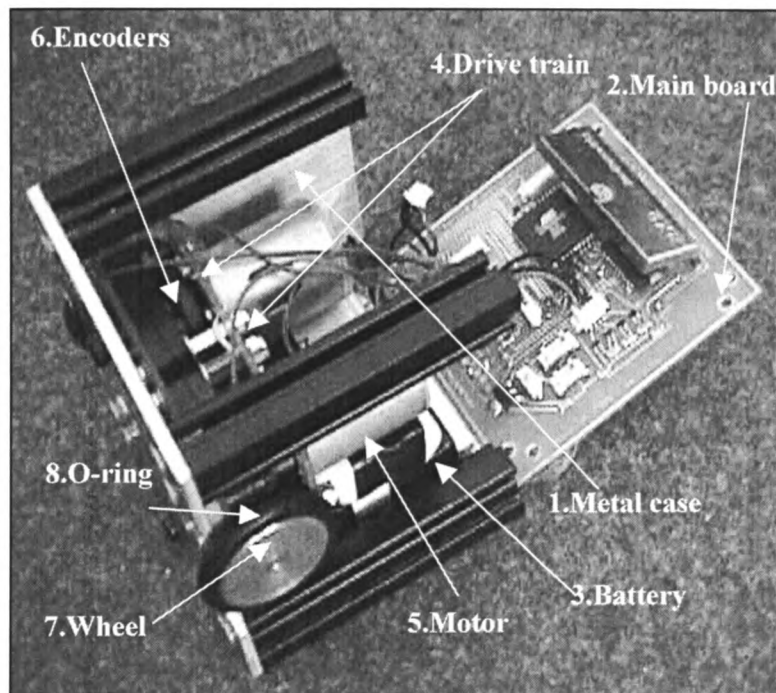


Figure 4-2 An exploded view of the mobile robot

4.2.2 Drive train

The drive train of the mobile robot consists of two 4.5-12V DC motors, with worm gear drives (one 10x6mm and one 6x6mm) to each wheel. The wheels are 32mm in diameter and fitted with O-rings to reduce slippage. The overall width of the drive train is approximately 80mm. Each motor shaft is monitored via a combination of phototransistors and infrared LED's, which

provide feedback for the ATMEL (AT90s8515) microprocessor. All the information is gathered by the microprocessor and each motor is driven independently.

4.2.3 DC motors

The DC motors used by the drive train are produced by Mabuchi RC-280SA-20120 giving 29g.cm torque at 6400rpm taking 0.57A from a 6V supply at maximum power. The stall torque from a 6V supply is 175g.cm at a current of 2.85A. The armature resistance of the motor is 2.11ohm. Back electromagnetic force EMF is 2.16V at 3000rpm. The robot is able to achieve speeds of up to 1.2m/s.

4.3 Establishing the kinematic model

As mentioned in the introduction of this chapter the kinematic model of a mobile robot is required to give its global description and to understand its manoeuvrability properties. Most conventional mobile robots use either a *tricycle* design where one wheel is steered and driven or a *differential drive* design (i.e. two drive wheels, each with its own motor; note that this is the type of design of the MIABOT V2 mobile robot). In (Campion *et al*, 1996) structural properties and classification of general kinematic models of wheeled mobile robots (WMR) can be found. Several other examples of derivation of kinematic models of WMR are available in the literature. For example, (Muir and Neuman, 1987), (Killough and Pin, 1992), (Dudek *et al*, 1993) and (Borestein *et al*, 1996).

Before the derivations of the kinematic equation it is important to state the difference between *holonomic* and *nonholonomic* robot (this is an important property due to the difficulty of nonholonomic systems to be stabilised in a posture by a smooth time-invariant state feedback). It is necessary to draw the distinction between what the actuators do, namely, turning or steering the wheels, and what these motions do in the environment. In this case, the side effect of the

wheel motion is to move the robot to any point in a three-dimensional space. If the number of controllable degrees of freedom is only two, and the total degrees of freedom is three this is *nonholonomic* robot. In general, a nonholonomic robot has fewer controllable degrees of freedom than total degrees of freedom. As a rule, the greater the difference between controllable and total degrees of freedom, the harder it is to control the robot. A robot with a trailer has four degrees of freedom but only two are controllable, and takes considerable skill to drive in reverse. If the number of total and controllable degrees of freedom of the system is the same, the robot is *holonomic*. MIABOT V2 is a nonholonomic robot as the number of controllable degrees of freedom is two (v, ω or u_l, u_r), which is less than the total degrees of freedom (x, y, θ). Therefore the mathematical expression of this kind of system can be represented as follows:

Mobile robots whose motion is subjected to a set of p nonintegrable constraints involving time derivatives of the configuration vector q are classified as nonholonomic systems (Neimark and Fufaen, 1972). The constraints usually take the form:

$$G(q)\dot{q} = 0 \quad (4.1)$$

with the $(n-p)$ independent columns of the $p \times n$ matrix $G(q)$ forming the base for the nonholonomic constraint condition:

$$\dot{q} = K(q)u \quad (4.2)$$

Note that the number of control inputs is less than the dimension of the system, i.e. under actuation with $u \in \mathbb{R}^{n-p}$ follows from Equation (4.1). Consider a set of wheels with independent wheel motors as the MIABOT V2 mobile robot with nonholonomic kinematics, as

shown in Figure 4-3. Assuming no slippage the motion of each wheel is restricted to its longitudinal direction with velocities u_l and u_r , respectively, by a single nonholonomic constraint ($p=1$). In other words, no motion can occur along the lateral robot co-ordinate axis y_r . Also shown in Figure 4-3 is the robot configuration $\mathbf{q} = (x, y, \vartheta) \in \mathcal{R}^3$ in the global co-ordinate frame (x_g, y_g) . Control inputs are the two wheel velocities u_l and u_r , which may be translated into the translational and rotational velocity variables $\mathbf{u} = (v, \omega) \in \mathcal{R}^2$ for convenience. The motion of the wheel set in the global co-ordinate frame is given in the following.

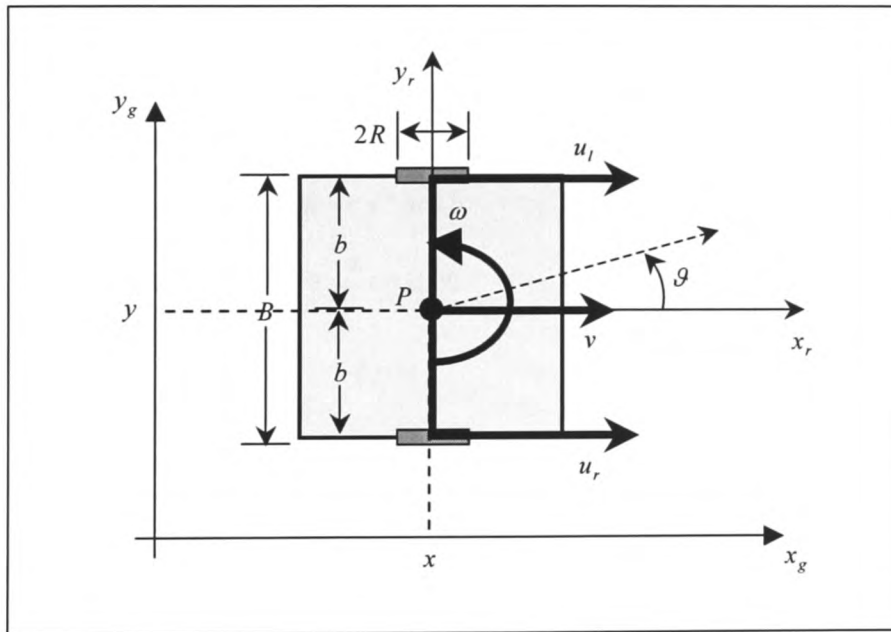


Figure 4-3 Schematic layout of the mobile robot

Let point P denote the centre of the mobile robot in Figure 4-3. Let \dot{x} be the time derivative of the global x position of point P and \dot{y} the time derivative of the global y position of the point

P. Let $\dot{\vartheta}$ be the time derivative of the global orientation angle (angle from the x_g axis in Figure 4-3). Also, let $B = 2b$, where b is defined as the half distance between the two wheels. Then the first order kinematic equations of the mobile robot are defined as follows:

$$\dot{x} = \frac{1}{2} * \cos \vartheta * u_l + \frac{1}{2} * \cos \vartheta * u_r \quad (4.3)$$

$$\dot{y} = \frac{1}{2} * \sin \vartheta * u_l + \frac{1}{2} * \sin \vartheta * u_r \quad (4.4)$$

$$\dot{\vartheta} = -\frac{1}{B} * u_l + \frac{1}{B} * u_r \quad (4.5)$$

The general form of Equations (4.3), (4.4) and (4.5) can be represented in the following if control inputs are v (linear velocity) and ω (angular velocity).

$$\dot{x} = v * \cos \vartheta \quad (4.6)$$

$$\dot{y} = v * \sin \vartheta \quad (4.7)$$

$$\dot{\vartheta} = \omega \quad (4.8)$$

4.4 Establishing the dynamic model

(Yamamoto and Yun, 1994) considered dynamic modelling of a wheeled mobile robot taking into consideration nonholonomic constraints. Their work is mainly based on control of a mobile platform with a manipulator. Lagrange multipliers were used in order to implement the Lagrange equations of motion of the mobile platform. (Bloch *et al*, 1992) have also discussed more complicated dynamic modelling of mobile robots. The derivation of the equations of motion of MIABOT V2 is similar to (Kimoto. K. and Yuta, 1995) and (Feddema *et al*, 1997) except that wheel slippage is added due to linear accelerations and fast turning. Also the

parameters of the right and left sides are assumed equal. The following notations will be used in the derivation of the constrained and dynamic equations.

- B : Wheel base
- D : Friction constant of the wheel
- J : Rotational moment of inertia
- J_{act} : Active moment of inertia
- J_m : Moment of inertia about the wheel
- J_w : Moment of inertia about the motor axis
- K_b : Motor's back EMF
- K_t : Torque constant
- M : Mass
- M_{act} : Active mass
- R : Radius of each driving wheel
- V_l : Left motor voltage
- V_r : Right motor voltage
- P : Center of mass of the robot
- Ω : Motor armature resistance
- b : Distance between the driving wheel and the axis of symmetry
- f_r : Force generated by the right wheel
- f_l : Force generated by the left wheel
- γ : Ratio of the motor gearbox
- θ : Robot's orientation (heading angle)
- τ_l : Left motor torque value

- τ_r : Right motor torque value
- v : Linear velocity
- u_l : Forward velocity of the left wheel
- u_r : Forward velocity of the right wheel
- ω : Angular velocity
- ω_l : Left wheel angular velocity
- ω_r : Right wheel angular velocity
- x : Direction on x axis in the global co-ordinate frame (x_g, y_g)
- σ : Slippage factor
- σ_v : Slippage factor due to linear accelerations
- $\sigma_{\dot{\theta}}$: Slippage factor due to angular accelerations
- σ_l : Slippage factor for left wheel
- σ_r : Slippage factor for right wheel
- y : Direction on y-axis in the global co-ordinate frame (x_g, y_g)
- x_r : x-axis of robot co-ordinate frame
- y_r : y-axis of robot co-ordinate frame
- x_g : X-axis of global co-ordinate frame
- y_g : Y-axis of global co-ordinate frame

Referring to Figure 4-3 the robot's equations of motion are:

$$M \frac{dv}{dt} = f_r + f_l \quad (4.9)$$

$$J \frac{dw}{dt} = \frac{B}{2} (f_r - f_l) \quad (4.10)$$

Assuming no slippage, the linear and angular velocities of the robot from the odometry are:

$$v = \frac{1}{2} (R\omega_r + R\omega_l) \quad (4.11)$$

$$\omega = \frac{1}{B} (R\omega_r - R\omega_l) \quad (4.12)$$

In reality this assumption is not true. In this case the odometry is based on simple equations that are easily implemented and that utilise data from inexpensive incremental wheel encoders. However, odometry is also based on the assumption that wheel revolutions can be translated into linear displacement relative to the ground. This assumption is only of limited validity (Borenstein and Koren, 1995). One extreme example is wheel slippage: if one wheel was to slip then the associated encoder would register wheel revolutions even though these revolutions would not correspond to a linear displacement of the wheel. In (Borestein *et al*, 1996) two categories are listed in which inaccuracies in the translation of wheel encoder readings into a linear motion can occur. In this work the wheel slippage is of interest, as the mass of the robot is very small in accordance with very fast DC motors. In general, the main reasons for wheel slippage is slippery floors, robot accelerations, fast turning, poor contact with the floor and finally due to both external and internal forces (interaction with other objects and castor wheels)

Therefore the model of the MIABOT V2 mobile robot contains two different types of slippage. The first slippage has been modelled due to the linear accelerations and the second due to the angular accelerations (fast turning). The general form of the slippage factor σ is given within the interval $0 < \sigma < 1$.

The wheel slippage was calculated conducting experiments with different initial robot accelerations. Comparison was made on the feedback information from the wheel encoders (wheel revolutions were counted) against the robot's actual linear displacement on the ground. The value of slippage factor shown in Figure 4-4 and Figure 4-5 was based on averaged results. The slippage factors of the MIABOT V2 mobile robot due to linear (Figure 4-4) and angular (Figure 4-5) accelerations were estimated to be:

$$\sigma_{\dot{v}} = f_1(\dot{v}) = a_5 \dot{v}^5 + a_4 \dot{v}^4 + a_3 \dot{v}^3 + a_2 \dot{v}^2 + a_1 \dot{v} + a_0 \quad (4.13)$$

$$\sigma_{\ddot{\theta}} = f_2(\ddot{\theta}) = b_5 \ddot{\theta}^5 + b_4 \ddot{\theta}^4 + b_3 \ddot{\theta}^3 + b_2 \ddot{\theta}^2 + b_1 \ddot{\theta} + b_0 \quad (4.14)$$

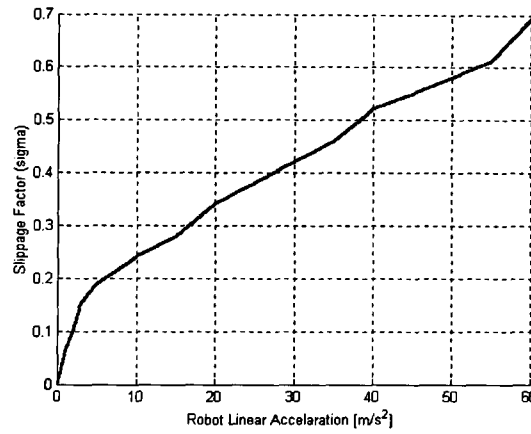


Figure 4-4 Approximation of slippage factor due to robot linear acceleration

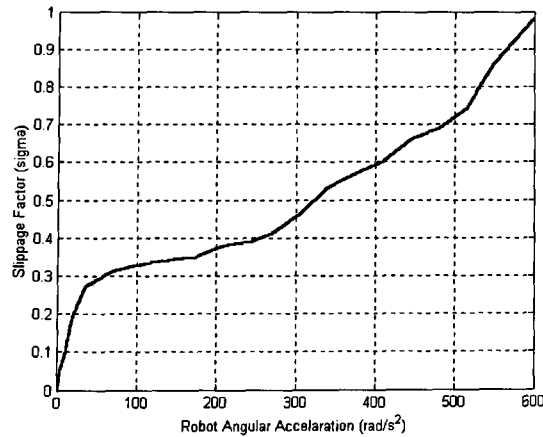


Figure 4-5 Approximation of slippage factor due to robot angular acceleration

Where \dot{v} and $\ddot{\theta}$ denotes the linear and angular acceleration of the robot respectively. Then the slippage factors for the left and right wheel are:

$$\sigma_l = \max(\sigma_{\dot{v}}, \sigma_{\ddot{\theta}}^l) \quad (4.15)$$

$$\sigma_r = \max(\sigma_{\dot{v}}, \sigma_{\ddot{\theta}}^r) \quad (4.16)$$

Equations (4.15), (4.16) show that the dominant slippage factor will be used to derive the final linear and angular velocities of the wheels. Referring to Equations (4.11), (4.12) the linear and angular velocities of the mobile robot are obtained from the following:

$$v = \frac{1}{2} (R(1 - \sigma_r)\omega_r + R(1 - \sigma_l)\omega_l) \quad (4.17)$$

$$\omega = \frac{1}{B} (R(1 - \sigma_r)\omega_r - R(1 - \sigma_l)\omega_l) \quad (4.18)$$

The force generated by each wheel is related to the motor torque, which in turn is related to the applied voltage of the motor by (note γ is the gear ratio):

$$\tau_r = \gamma J_m \dot{\omega}_r + \frac{1}{\gamma} (J_w \dot{\omega}_r + D\omega_r + Rf_r) \quad (4.19)$$

$$\tau_l = \gamma J_m \dot{\omega}_l + \frac{1}{\gamma} (J_w \dot{\omega}_l + D\omega_l + Rf_l) \quad (4.20)$$

$$\tau_r = \frac{K_I}{\Omega} V_r - \frac{\gamma K_b K_I}{\Omega} \omega_r \quad (4.21)$$

$$\tau_l = \frac{K_I}{\Omega} V_l - \frac{\gamma K_b K_I}{\Omega} \omega_l \quad (4.22)$$

The velocity and acceleration of the robot in the x and y directions are:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = v \begin{bmatrix} \cos \vartheta \\ \sin \vartheta \end{bmatrix} \quad (4.23)$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \dot{v} \begin{bmatrix} \cos \vartheta \\ \sin \vartheta \end{bmatrix} + v \dot{\vartheta} \begin{bmatrix} -\sin \vartheta \\ \cos \vartheta \end{bmatrix} \quad (4.24)$$

Note that $\omega = \dot{\vartheta}$ and $v = \sqrt{\dot{x}^2 + \dot{y}^2}$. Combining Equations (4.23) and (4.24), the resulting equations of motion are:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \cos \vartheta \\ \sin \vartheta \end{bmatrix} \left\{ \frac{\gamma K_I}{R \Omega M_{act}} (V_r + V_l) - \frac{2}{R^2 M_{act}} \left(D + \frac{\gamma^2 K_b K_I}{\Omega} \right) \sqrt{\dot{x}^2 + \dot{y}^2} \right\} + \begin{bmatrix} -\sin \vartheta \\ \cos \vartheta \end{bmatrix} \dot{\vartheta} \sqrt{\dot{x}^2 + \dot{y}^2} \quad (4.25)$$

$$\ddot{\vartheta} = \frac{\gamma B K_I}{2 R \Omega J_{act}} (V_r - V_l) - \frac{B^2}{2 R^2 J_{act}} \left(D + \frac{\gamma^2 K_b K_I}{\Omega} \right) \dot{\vartheta} \quad (4.26)$$

Where the active mass and moment of inertia are:

$$M_{act} = M + \frac{2}{R^2} (\gamma^2 J_m + J_w) \quad (4.27)$$

$$J_{act} = J + \frac{B^2}{2 R^2} (\gamma^2 J_m + J_w) \quad (4.28)$$

The dynamic model of the mobile robot has been modelled in Simulink and for simplicity reasons the following four factors were used:

$$P_1 = \frac{\gamma K_I}{R \Omega M_{act}} \quad Q_1 = \frac{2}{R^2 M_{act}} \left(D + \frac{\gamma^2 K_b K_I}{\Omega} \right)$$

$$P_2 = \frac{\gamma B K_I}{2R\Omega J_{act}}$$

$$Q_2 = \frac{B^2}{2R^2 J_{act}} \left(D + \frac{\gamma^2 K_b K_I}{\Omega} \right)$$

To increase the model's accuracy factors P_1, P_2, Q_1, Q_2 were estimated using the constrained optimisation method described in Appendix D. Experiments using the mobile robot took place to verify the model. Comparison between real and simulated trajectories was made based on different input commands. Figure 4-6 shows a picture of the experimental trials.

The Simulink block of robot's input/output dynamic model is shown in Figure 4-7. The control inputs are left and right voltage for the DC motors. The initial values include initial position of the robot and initial speed either as linear or angular. In order for the robot's model to be *user-friendly* and for more convenience regarding manipulation of initial values and values of physical parameters, a mask block has been developed. In addition, Figure 4-8 shows the Simulink submodel of the mobile robot in more detail including all the interconnections

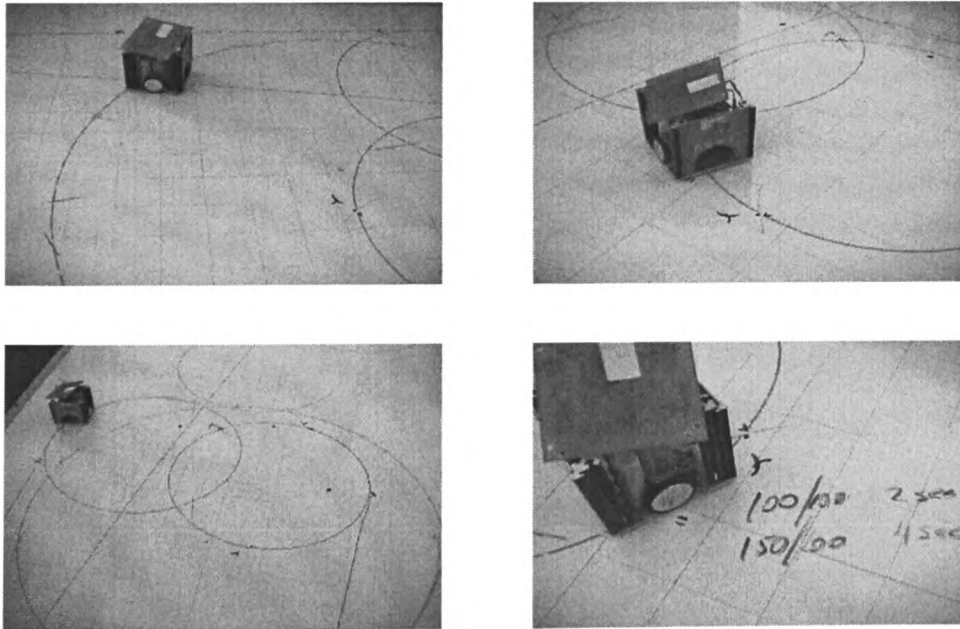


Figure 4-6 Real-time experiments to obtain and verify the accuracy of the open-loop model.

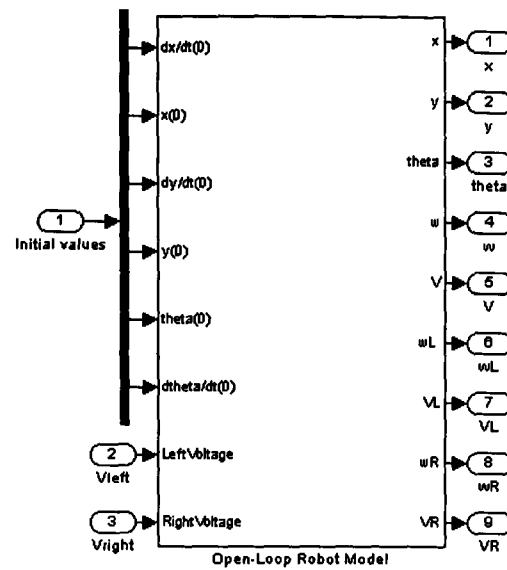


Figure 4-7 Robot's input/output open-loop dynamic model in Simulink

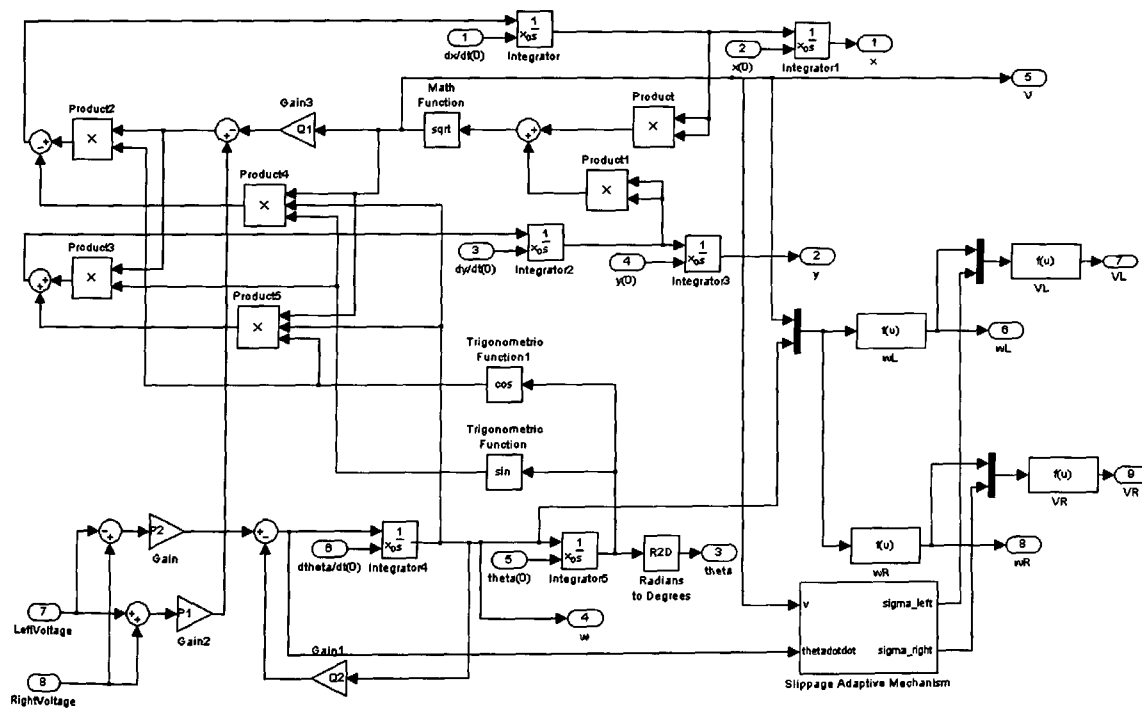


Figure 4-8 Submodel of robot's input/output open-loop dynamic model

4.5 Linearised model

Most physical systems have non-linear elements, but in some circumstances it may be possible to treat them as linear. Then the edifice of linear mathematics, which is very highly developed, can be employed to yield solutions. Sometimes the system operates over a small range of input values, and then the non-linearities can often be effectively approximated by linear functions. This is referred to as operation about some reference point or nominal trajectory. However if the non-linear equations are known, then the linearised form of these equations are often called perturbation equations. When designing and considering a linear controller or stability analysis takes place for a vehicle (mobile robot), it is often necessary to establish linearised models around representative operating points. In this case the linearised model is extracted from the full non-linear model of the vehicle. For the MIABOT V2 mobile robot, the non-linear model is composed of the non-linear equations of motion described in Equations (4.25) and (4.26). Although in chapter five the control design is based on the non-linear model of the robot the linear model is required for the robust stability testing, which takes place also in chapter five. The non-linear model can be written in the state-space form as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (4.29)$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}) \quad (4.30)$$

Where, \mathbf{x} , \mathbf{u} and \mathbf{y} denote states, control inputs and outputs respectively.

By linearising the non-linear model Equations (4.29), (4.30) around a steady operating point $\mathbf{x}_0, \mathbf{u}_0$, a linearised state-space model can be obtained in the following form:

$$\Delta \dot{\mathbf{x}} = \mathbf{A} \Delta \mathbf{x} + \mathbf{B} \Delta \mathbf{u} \quad (4.31)$$

$$\Delta y = C\Delta x + D\Delta u \quad (4.32)$$

Where,

$$\Delta x = x - x_0, \quad \Delta u = u - u_0, \quad \Delta y = y - y_0 = h(x, u) - h(x_0, u_0)$$

And,

$$\mathbf{A} = \frac{\partial f}{\partial x} \bigg|_{(x_0, u_0)}, \quad \mathbf{B} = \frac{\partial f}{\partial u} \bigg|_{(x_0, u_0)}, \quad \mathbf{C} = \frac{\partial h}{\partial x} \bigg|_{(x_0, u_0)}, \quad \mathbf{D} = \frac{\partial h}{\partial u} \bigg|_{(x_0, u_0)}$$

Once a linearised state-space has been established, the transfer function can be obtained directly from Equations (4.31) and (4.32), i.e.

$$\Delta Y(s) = G(s) \cdot \Delta U(s) \quad (4.33)$$

Where,

$$G(s) = C[sI - A]^{-1}B + D \quad (4.34)$$

Around a representative operating point of (x_0, u_0) the linearised state-space model is established using the *linmodv5* MATLAB function. The matrices **A**, **B**, **C** and **D** are:

$$\mathbf{A} = \begin{bmatrix} -63 & -63 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -45 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (4.35)$$

$$\mathbf{B} = \begin{bmatrix} 6.45 & 6.45 \\ 0 & 0 \\ -105 & 105 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (4.36)$$

$$\mathbf{C} = \begin{bmatrix} 1.03 & 1.03 & -0.04 & 0 & 0 & 0 \\ 1.03 & 1.03 & 0.04 & 0 & 0 & 0 \end{bmatrix} \quad (4.37)$$

$$\mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (4.38)$$

The state vector of the state-space model is:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \quad (4.39)$$

From the linearised state-space model and using the Equation (4.34), the transfer function matrix can be obtained in the following:

$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \end{bmatrix} \quad (4.40)$$

Where, after pole-zero cancellation becomes:

$$G(s) = \begin{bmatrix} \frac{10.99s + 572.4}{s^2 + 108s + 2835} & \frac{2.317s + 26.24}{s^2 + 108s + 2835} \\ \frac{2.317s + 26.24}{s^2 + 108s + 2835} & \frac{10.99s + 572.4}{s^2 + 108s + 2835} \end{bmatrix} \quad (4.41)$$

4.6 Results

To verify the robot model, experiments took place in order to compare real and simulated robot trajectory. Figure 4-9 shows a comparison of both trajectories when the control input was 3.5V for both motors at $t = 0$ and 4.5V on right and 3.5V on left motor at $t = 2s$ (note that the input command was transmitted to it from PC host through the serial port). It can be observed that the non-linear robot's dynamic model is very close to the real dynamics of the plant. The root mean square error (RMSE) was found to be 1.51. Measuring errors were not considered for the real trajectory shown in Figure 4-9 as they were obtained using a pen on the robot moving on a fixed grid.

Following the non-linear model linearisation it is necessary to verify the linearised model established. The direct way of verification is to compare the responses of the linearised model and the non-linear model from which the linearised model is extracted under the same perturbations of the control inputs. The comparison of the responses under perturbations of control inputs is shown in Figure 4-10 and Figure 4-11. In Figure 4-10 the left motor is perturbed to achieve left wheel linear velocity 0.1m/s at $t = 0$ and 1m/s at $t = 1s$. In Figure 4-11 the right motor is perturbed to achieve a right wheel linear velocity 0.8m/s at $t = 0$ and

concluded that the linearised model is acceptable. The RMSE in Figure 4-10 is 0.05 and in Figure 4-11 is 0.03. The peaks in Figure 4-10 at $t = 2s$ and in Figure 4-11 at time $t \approx 1s$ are due to the influence of one wheel to another (system is coupled). As mentioned earlier the non-linear model will be used in chapter five for control system design and the linearised model for the robust stability analysis of the uncertain closed-loop dynamic system.

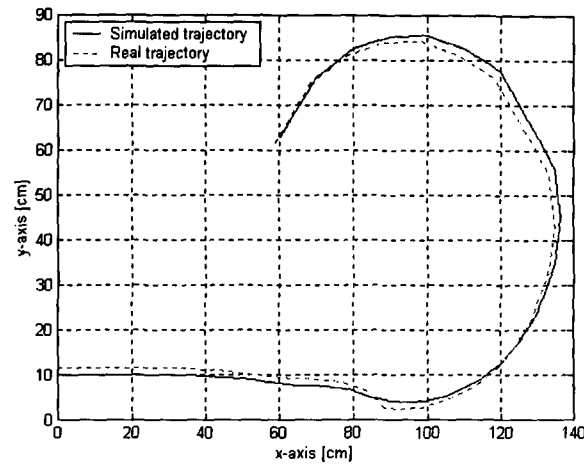


Figure 4-9 Comparison of real and simulated robot trajectory (RMSE=1.51)

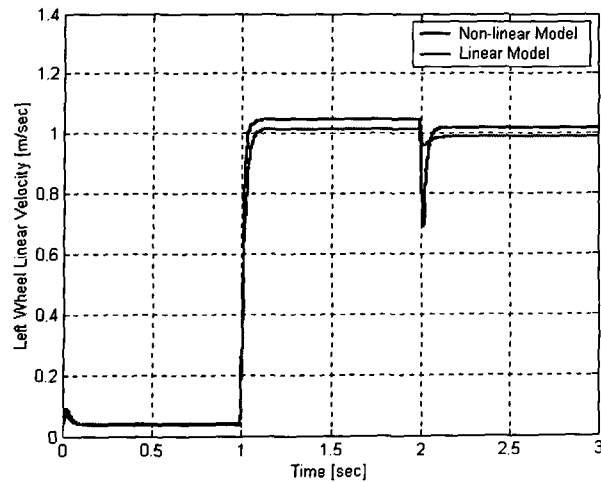


Figure 4-10 Comparison of non-linear and linear response for the left wheel linear velocity under perturbations of control inputs (RMSE=0.05).

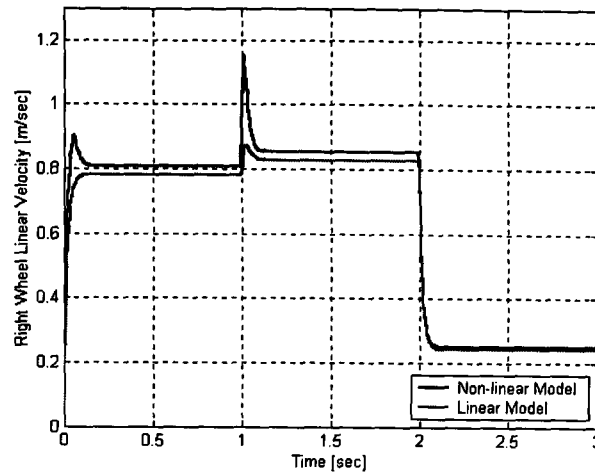


Figure 4-11 Comparison of non-linear and linear response for the right wheel linear velocity under perturbations of control inputs (RMSE=0.03)

4.7 Discussion

As mentioned in the introduction of this chapter, the main objective is to derive both kinematic and dynamic model of the differential drive MIABOT V2 mobile robot. A discussion of some of the modelling issues as well as the results is presented in this section.

At the introduction of this chapter was discussed the difficulty to model mobile robots due to mechanical disturbances in which affects their position accuracy. For this reason several factors were identified showing that obtaining a good model of a mobile robot is very difficult and challenging task. Conducting experiments and optimising physical parameters the robot dynamic model was established. Figure 4-9 demonstrates verification on a dynamic model of the mobile robot compared to the real system's dynamics, in which the resultant position errors are significant small. The RMSE of the comparison between real and simulated trajectory found to be 1.51. This very good result taking into consideration that the most mechanical disturbances that affect the mobile robot were overcome.

Finally having representative dynamic model results in representative linearised model as shown in Figure 4-10 and Figure 4-11. This is desirable and useful result that provides the opportunity for reliable robust stability testing for uncertainty in robot dynamics.

4.8 Summary

This chapter presents the modelling of MIABOT V2 mobile robot. The robot is small size measuring 8cm^3 and is steered and driven by differential drive design utilising two DC motors enabling robot's speed up to 1.2m/s . The first order kinematic model of the robot was derived in order to understand its manoeuvrability properties and to produce information about its global description. However, as the robot behaviour is related to the framework and theory of holonomic and nonholonomic systems some discussions were made to show in which category MIABOT V2 falls (holonomic or nonholonomic). Then the full non-linear dynamic model of the robot was established for complete description of its system's dynamics. To improve the model accuracy real experiments took place. The non-linear control design blockset based on constrained optimisation method was used for identification of several robot physical parameters. However, to further improve the model, wheel slippage was introduced and modelled. Contacting real experiments, two different types of slippage were considered, one due to linear accelerations and another due to angular accelerations (fast turning). The robot model found to be very close to the real dynamics of the plant considered. The linearised model of the robot was extracted and some comparisons were made to show the validity of the linearised model against the non-linear. Results were presented, and show that the linearised model is acceptable, comparing non-linear and linear response under perturbations of the same control inputs.

The contributions of this chapter are: Modelling of MIABOT V2 mobile robot with both a kinematic and a dynamic model. Optimisation and identification of parameters of physical

components (i.e moments of inertia) conducting experiments and using the non-linear control design tool. Modelling of wheel slippage based on experiments for more accurate feedback control design.

In the next chapter the full non-linear model is used for the design of a speed controller and the linearised model is used for verification of the robust stability of the closed-loop system under uncertainty in robot dynamics.

4.9 References

BEMPORAD, A., DI MARCO, M. and TESI, A. 2000. Sonar-Based Wall-Following Control of Mobile Robots. *Journal of Dynamic Systems Measurement and Control-Transactions of the ASME*, **122** (1), pp. 226-230.

BLOCH, A. M., REYHANOGLU, M. and MCCLAMROCH, N. H. 1992. Control and Stabilisation of Non-holonomic Dynamic Systems. *IEEE Transactions on Automatic Control*, **37** (11), pp. 1746-1757.

BORENSTEIN, J. 1995. Control and Kinematic Design of Multi-Degree-of-Freedom Mobile Robots with Compliant Linkage. *IEEE Transactions on Robotics and Automation*, **11** (1), pp. 21-35.

BORENSTEIN, J. and KOREN, Y. 1995. Motion Control Analysis of a Mobile Robot. *Transactions of ASME, Journal of Dynamics, Measurement and Control*, **109** (2), pp. 73-79.

BORESTEIN, J., EVERETT, H. R. and FENG, L. 1996. *Navigating Mobile Robots: Systems and Techniques*. USA: A K Peters Ltd. 156881058X.

CAMPION, G., BASTIN, G. and DANDREANOVEL, B. 1996. Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots. *IEEE Transactions on Robotics and Automation*, **12** (1), pp. 47-62.

DUDEK, G., JENKIN, E., MILIOS, E. and WILKES, D. 1993. *A Taxonomy for Swarm Robots*. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 441-447, Yokohama, Japan.

EGERSTEDT, M. and HU, X. M. 2002. Hybrid Control Approach to Action Coordination for Mobile Robots. *Automatica*, **38** (1), pp. 125-130.

EVERETT, H. R. 1995. *Sensors for Mobile Robots: Theory and Applications*. USA: A K Peters, Ltd. 1-56881-048-2.

- FEDDEMA, J. T., ROBINET, R. D. and DRIESSEN, B. J. 1997. *Explaining Finite State machine Characteristics Using Variable Structure Control*. Proceedings of SPIE (The International Society of Optical Engineering). Sensor Fusion and Decentralised Control in Autonomous Robotics Systems. pp. 46-54, Pittsburgh, Pennsylvania.
- HAN, K. B., KIM, H. Y. and BAEK, Y. S. 2001. Corridor Navigation of the Mobile Robot Using Image Based Control. *KSME International Journal*, **15** (8), pp. 1097-1107.
- HERRMANN, J. M. 2001. Dynamical Systems for Predictive Control of Autonomous Robots. *Theory in Biosciences*, **120** (3-4), pp. 241-252.
- KILLOUGH, S. M. and PIN, F. C. 1992. *Design of Ommidirectional and Holonomic Wheeled Platform Design*. Proceedings of IEEE Conference on Robotics and Automation. pp. 84-90, Nice, France.
- KIMOTO, K. and YUTA, S. 1995. Autonomous Mobile Robot Simulator - A Programming Tool for Sensor - Based Behavior. *Autonomous Robots*, **1**, pp. 131-148.
- KODAGODA, K. R. S., WIJESOMA, W. S. and TEOH, E. K. 2002. Fuzzy Speed and Steering Control of an AGV. *IEEE Transactions on Control Systems Technology*, **10** (1), pp. 112-120.
- LOUSTE, C. and LIEGEOIS, A. 2000. Near Optimal Robust Path Planning for Mobile Robots: the Viscous Fluid Method With Friction . *Journal of Intelligent & Robotic Systems*, **27** (1-2), pp. 99-112.
- MENEZES DE OLIVEIRA, V., ROBERTO DE PIERI, E. and LAGES, W. F. 2000. *Neural Networks in the Control of a Mobile Platform*. 6th IFAC Symposium on Robot Control. pp. 181-186, Vienna, Austria.
- MUIR, P. F. and NEUMAN, C. P. 1987. Kinematic Modelling of Wheeled Mobile Robots. *Journal of Robotic Systems*, **4** (2), pp. 281-329.
- NEIMARK, J. I. and FUFAEN, N. A. 1972. *Dynamics of Nonholonomic Systems*. USA: American Mathematical Society. 0821815830 .
- PAJAZITI, A., SHALA, A., PAJAZITI, A. and LIKAJ, R. 2000. *Velocity and Torque Feedback Control of a Nonholonomic Mobile Robot Using Fuzzy-Neural Networks*. 6th IFAC Symposium on Robot Control. pp. 43-48, Vienna, Austria.
- PRUSKI, A. and ATASSI, A. 2000. Sensor Information Space for Robust Mobile Robot Path Planning. *Robotica*, **18**, pp. 415-421.
- TUNESKI, A. I., VUKOBRATOVIC, M. K. and DIMIROVSKI, G. M. 2001. Adaptive Control of Multiple Robots Manipulation on a Dynamical Environment. *Proceedings of the Institution of Mechanical Engineers Part I- Journal of Systems and Control Engineering*, **215** (14), pp. 385-404.
- WILL, A. B. and ZAK, S. H. 1997. Modelling and Control of an Automated Vehicle. *Vehicle System Dynamics*, **27** (3), pp. 131-155.
- YAMAMOTO, Y. and YUN, X. 1994. Coordinating Locomotion and manipulation of a Mobile Robot. *IEEE Transactions on Automatic Control*, **39** (6), pp. 1326-1332.

5

Control, Robust Stability Analysis and Discovery of Fuzzy/Neural Local Models

5.1 Introduction

The aims of this chapter is to present the control, the robust stability analysis of the MIABOT V2 mobile robot and the discovery of fuzzy-neural local models from observation data. Three speed controllers have been developed and comparison is made based on performance criteria and length of execution time. Reliable speed controller for the MIABOT V2 mobile robot is vital in this thesis for credible control strategy development and testing in chapters six and seven.

In a general control scheme, the plant is affected by input signals, some of which (the control inputs) are accessible to the controller, and some of which (the disturbance inputs) are not. Some of the plant signals (the tracking outputs) are to be controlled, and some of the plant

signals (the measurement outputs) are available to the controller. A controller generates control inputs to the plant with the objective of having the tracking outputs closely approximate to the reference inputs. With open-loop control, there is no feedback from the motors, informing the robot's program how fast the wheels are turning or how far the robot has travelled. Depending on terrain, slippage in wheel contacts, etc. the commanded voltages (to the drive motors) do not necessarily imply particular speeds. To overcome this problem closed-loop control is used using a speed controller. The main traditional idea is to take the desired velocity command, send that command to the motors, measure how fast the motors actually turn, and then measure the speed and compare this to the commanded speed. In this case and depending on the control design chosen, several control parameters have to be tuned or optimised. For instance if the control design involves Proportional Integral and Derivate (PID) action then three gains or parameters have to be tuned/optimised. In many cases this is a very difficult task and also a non-linear control problem. In this chapter the tuning/optimisation process is solved using the Non-linear Control Design (NCD) blockset provided by MATLAB. As described in Appendix D the NCD blockset converts constraint bound data and tuneable variable information into a constrained optimisation problem. Then the tuneable variables are adjusted in an attempt to better achieve the constraints on system signals defined by the NCD interface. The constrained optimisation problem is solved using Sequential Quadratic Programming (SQP) algorithm and Quasi-Newton gradient search techniques.

As mentioned in chapter three section 3.5.4, one of the most important properties of control architectures is modularity. Therefore three types of speed controllers have been developed based on conventional control theory, fuzzy logic and neural networks. The performance and the length of execution time of each controller are compared against the other controllers. The design of the intelligent controllers (fuzzy and neural) is based on clustering techniques and learning with a teacher, which is also referred as supervised learning. In addition this study also

addresses the problem of identifying, modelling and controlling systems with dynamical behaviour.

In many applications, the structure, linearity and order of the system are considered fixed and the parameters of the system are allowed to change to a certain extent. Changes in parameters of parts of the system determine changes in the coefficients of the closed-loop system dynamics. Only very simple cases are described by coefficient variations free of independence. Complex processes are characterised by rather complex relations between changes in the parameters and changes in the coefficients of the closed-loop system. Nevertheless, even in those cases, sufficient conditions are required for allowable parameter changes without causing inadmissible change in the dynamic system behaviour. Usually two approaches are available when considering uncertainty models and robustness; parametric (Interval polynomial, The Edge Theorem etc) and nonparametric robustness analysis (The basic perturbation model, Structured singular value, etc.). MIABOT V2 mobile robot is still under further development in order to increase its capabilities and functionality (adding new sensors and deployment of other hardware). The problem, which arises in this case, is whether the control law implemented is sufficient to incorporate all these changes, which rapidly affect the robot dynamics. As mentioned in chapter three the parametric robustness analysis approach (Kharitonov's Theorem) is adopted in this thesis as it offers fast robust stability testing and analysis based on interval polynomials.

The remainder of this chapter is organised as follows: Section 5.2 discusses design requirements and the selection criteria of speed controller for the mobile robot. Section 5.3 presents the development and tuning of the PI controller. The modelling and identification of fuzzy controller based on subtractive clustering is given in section 5.4. Section 5.5 presents the development of a neural controller based on supervised learning. Comparison of all three types of controllers based on performance criteria and execution time is covered in section 5.6. Robust

stability analysis of the closed-loop control system of the mobile robot under uncertainty in robot dynamics is presented in section 5.7. A discussion follows in section 5.8 and finally the summary of the chapter is presented in section 0.

5.2 Controlling the mobile robot

Already mentioned in the introduction that one of the aims of this chapter is the development of speed controller for the mobile robot for credible control strategy development and testing in chapters six and seven. In the following sections the selection of the speed controller is presented, followed by design requirements derived for the mobile robot.

5.2.1 Selection of speed controller

The control problem is to design speed controller for the MIABOT V2 mobile robot based on the multivariable non-linear model derived in chapter four. In general, the selection of the design method depends upon many parameters such as performance criteria, process knowledge, linearities, non-linearities of the system etc. Several controller design methods can be found in the literature (Gajic and Lelic, 1996), (Skogestad and Postlethwaite, 1996) and (Burnham *et al*, 1999). However, these methods may differ in complexity, flexibility and in the amount of process knowledge used. For instance, Ziegler-Nichols method is based on knowledge of the process transfer function at the operating point. Pole placement is also based on knowledge of the process transfer function. Using this method a controller that gives desired closed-loop poles can be found. A drawback of this method is that complex models lead to complex controllers. Other methods include, dominant pole design (only few poles are assigned), loop shaping, H_∞ control design (it assumes system interconnection structure matrix is known), Cohen-Coon method (main design criterion is rejection of load disturbances), Haalman method, Chien-Hrones-Reswick method (modification of Ziegler-Nichols method), etc. More details for controller design methods can be found in (Astrom and Hagglund, 1995).

The design of MIABOT's V2 speed control law is based on PI MIMO controller (described in section 5.3) and the NCD optimisation method described in Appendix D. It has previously been mentioned that this method is simple, fast, easy-to-use, suitable for control design of non-linear models and gives good performance. The main objective is the optimisation and tuning of the speed controller parameters such as the closed-loop system in Figure 5-1 meets the design requirements defined in the following section 5.2.2. However, it is important the closed-loop response to be robust to the uncertainty in the robot dynamics. This aspect will be discussed in more detail in section 5.7

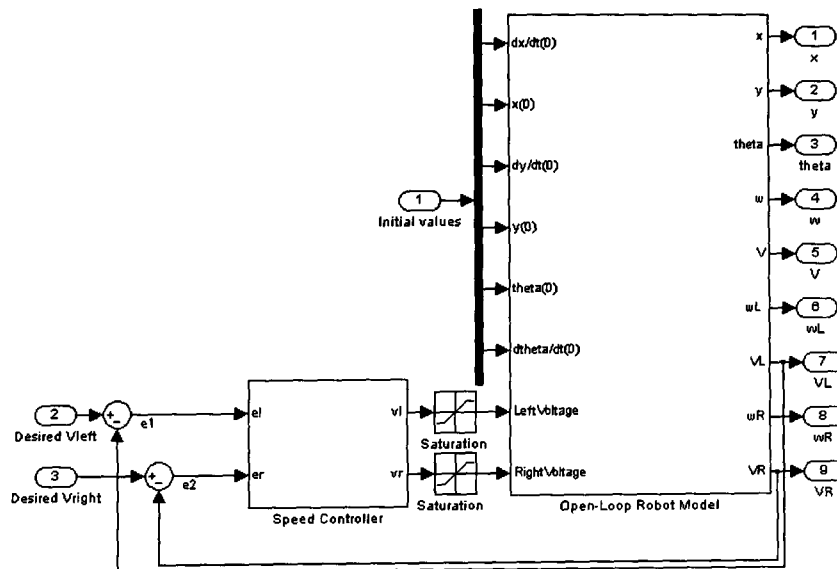


Figure 5-1 Closed-loop robot model with speed controller

5.2.2 Design requirements

The next step in controlling the mobile robot is to specify performance criteria. When operating from the 6V supply the mobile robot will reach speeds up to 1.2m/s. Ideally the robot should be able to accelerate up to this speed in less than 0.15s (i.e. to match open-loop performance, see chapter four, Figure 4-11). Since the most basic requirement of the MIABOT V2 mobile robot is that it should operate at the desired velocity, the steady-state error of the wheel speed should be zero. The other performance requirement is that the robot must accelerate to its

steady-state speed as soon as it is commanded. In this case a settling time of 0.25s is desirable, with an overshoot of less than 5% .

5.3 PI controller

Chapter four presented the dynamic model of the plant to be controlled with two inputs and two outputs. The inputs are left and right voltages of the left and right motor respectively. The outputs are the actual left and right velocities of the left and right wheel respectively. The control action selected is PI instead of PD or PID for the following reasons: PI control action decreases steady-state error in a system's output response to a step input and also removes steady-state offsets, including those caused by disturbance input signals. Derivative action is omitted, as it has no affect on constant offsets in either the reference tracking or disturbance rejection responses. This is because the derivative of a constant error is zero and so the controller does not respond to the presence of the constant error. This is quite the opposite of the use of integral control as was previously mentioned. In order to introduce more freedom into the control design the PI controller is modelled as a state-space system as shown in Equations (5.1) and (5.2) with a null A matrix and B defined as the identity matrix. Matrices C and D are the tuneable variables K_i and K_p respectively for a total of eight tuneable variables (note that the size of both matrices is 2×2). The final form of the PI controller is shown in Equation (5.3), (5.4) and in Figure 5-2 as expanded Simulink model. The controller can be represented in the general state space form as follows:

$$\dot{x} = Ax + Bu \quad (5.1)$$

$$y = Cx + Du \quad (5.2)$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \quad (5.3)$$

$$\begin{bmatrix} v_l \\ v_r \end{bmatrix} = \begin{bmatrix} K_{i11} & K_{i12} \\ K_{i21} & K_{i22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} K_{p11} & K_{p12} \\ K_{p21} & K_{p22} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \quad (5.4)$$

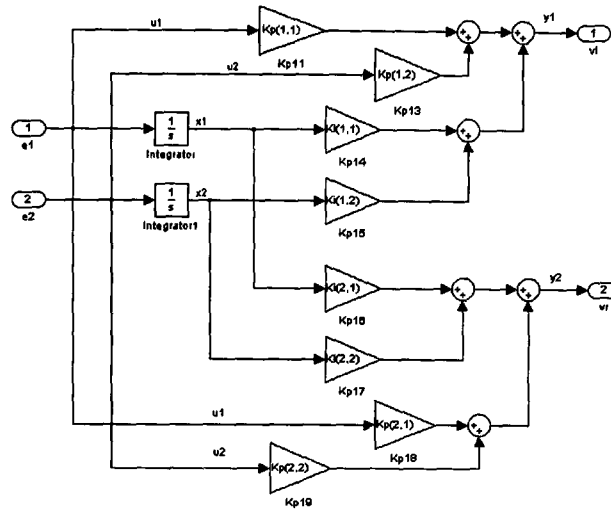
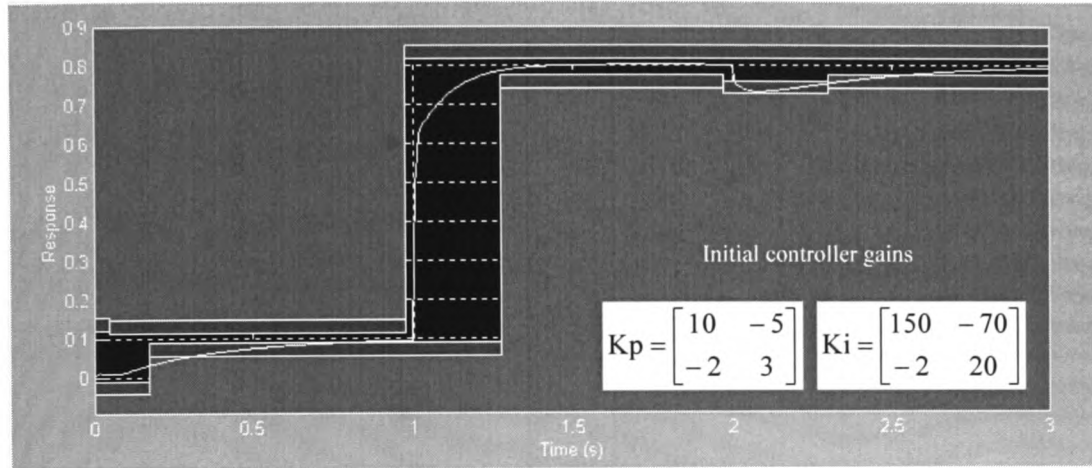


Figure 5-2 Proportional Integral MIMO speed controller with eight tuneable variables

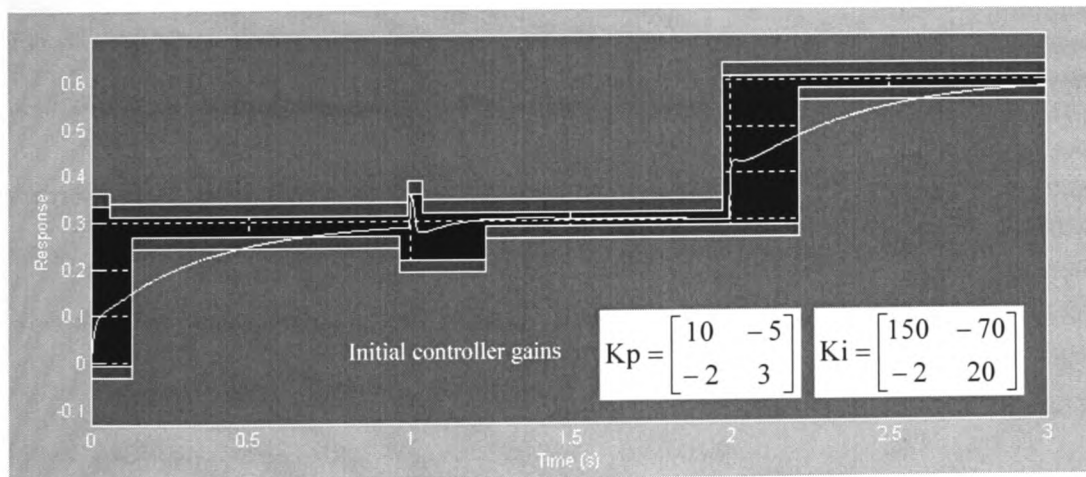
5.3.1 Controller tuning

The non-linear control design (NCD) blockset provided by MATLAB is used for tuning/optimisation of the controller parameters. It uses time domain constraint bounds to represent lower and upper bounds on response signals. When the optimisation procedure starts NCD converts the constraint bound data and tuneable variable information (K_i and K_p) into a constrained optimisation problem. As described in Appendix D the constrained optimisation problem is solved by a Sequential Quadratic Programming (SQP) algorithm and Quasi-Newton gradient search techniques. The main idea behind the controller's parameter optimisation is the minimisation of the maximum constraint violation (in other words NCD minimises the maximum (weighted) constrained error). The set-up for upper and lower bounds is shown in Figure 5-3 and is defined as follows: For upper bounds constraints, the difference between the constraint boundary and the simulated error defines the constraint error. The constraint error for lower bound constraints is defined as the difference between the simulated error and the constraint boundary. The subroutine of the SQP method solves a Quadratic Programming (QP) problem at each iteration. At each iteration, the upgrade and estimate of the Hessian of the

Lagrangian is achieved. At this stage a line search method is required in which a merit function is used to solve the problem. Finally the implementation of the SQP subproblem attempts to satisfy the Kuhn-Tucker equations, which are necessary conditions for optimality of the constrained optimisation problem. Both (Fletcher, 1987) and (Gill *et al*, 1991) provide good references and more details on the above algorithms.



(a)



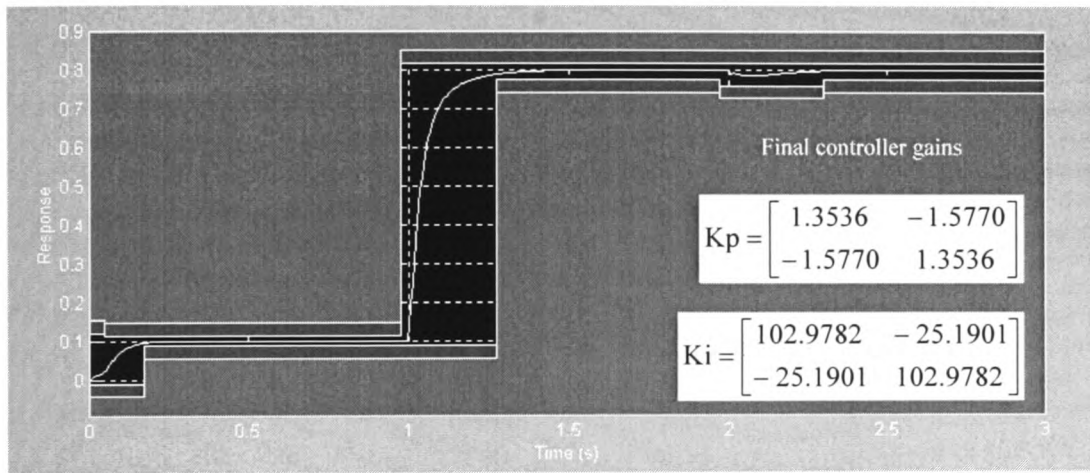
(b)

Figure 5-3 Constraint windows with initial controller gains before optimisation for left (a) and right (b) response

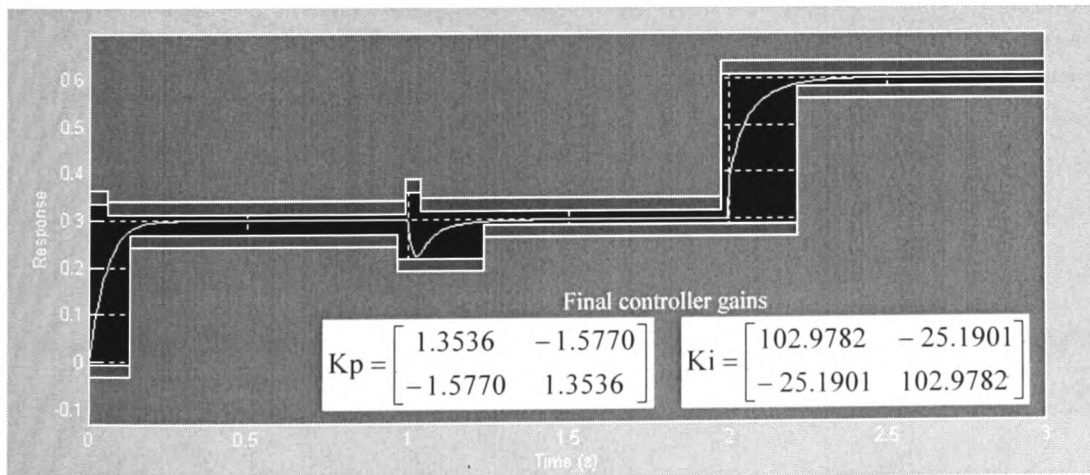
5.3.2 Results of tuning

The required parameters for the PI controller were tuned and optimised successfully as shown in Figure 5-4. Equation (5.5) shows the output equation where it can be observed that the controller is symmetrical. This is an expected result as the mobile robot was modelled as being symmetrical with equal parameters on both left and right sides.

$$\begin{bmatrix} v_l \\ v_r \end{bmatrix} = \begin{bmatrix} 102.9782 & -25.1901 \\ -25.1901 & 102.9782 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1.3536 & -1.5770 \\ -1.5770 & 1.3536 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \quad (5.5)$$



(a)



(b)

Figure 5-4 Constraint windows with final controller gains after optimisation for left (a) and right (b) response

To test and verify the control algorithm the mobile robot was simulated with different step input command. The test was carried out for 3s using step input on both wheels as desired input velocity command. The comparison of both uncompensated and compensated responses under perturbations of control inputs is shown in Figure 5-5(a) and Figure 5-5(b). In Figure 5-5(a) the desired input linear velocity for the right wheel is 0.8m/s at $t=0$ and 0.2m/s at $t=1$ s. In Figure 5-5(b) the desired input linear velocity for the left wheel is 0.2m/s at $t=0$ and 1m/s at $t=2$ s. Comparing the two responses (uncompensated and compensated) with the reference response, it can be concluded that the control algorithm is successful, in that the robot produces the desired performance as defined in section 5.2.2 i.e. there is less than 5% , overshoot on the response, the rise time is less than 0.15s and the settling time is less than 0.25s . The peaks in Figure 5-5(a) at $t=1$ s and in Figure 5-5(b) at time $t=2$ s are due to the influence of one wheel to another (system is coupled). However, it is also important to observe the control effort as the robot has supply of 6V , both Figure 5-6(a) and Figure 5-6(b) show that the motor input voltages do not exceed 6V and are therefore acceptable.

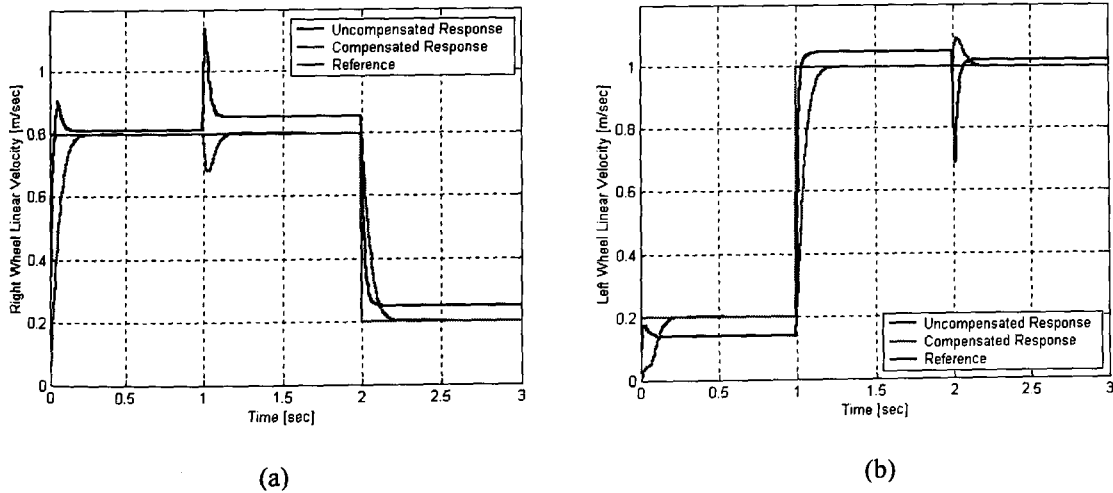


Figure 5-5 Comparison of uncompensated compensated and reference response (PI control). (a) Step input for the right wheel (b) Step input for the left wheel

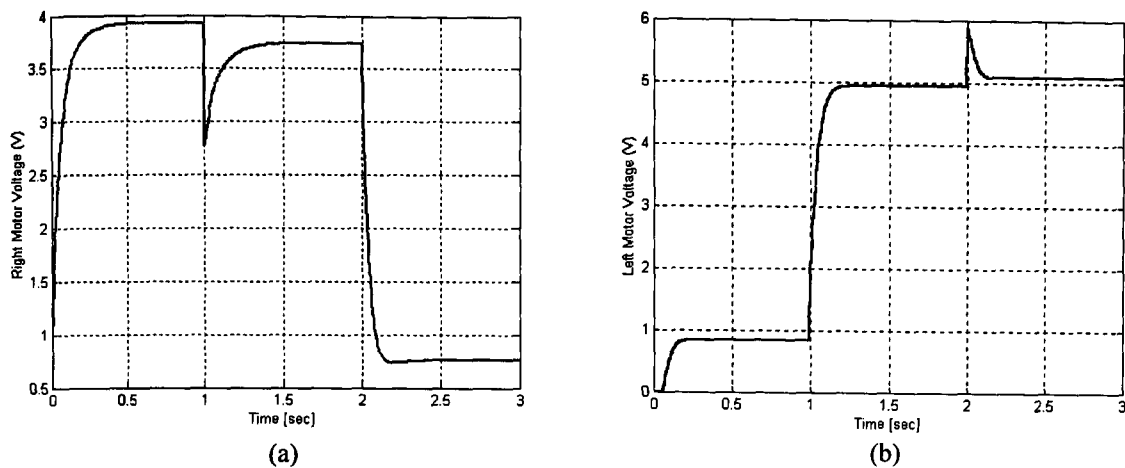


Figure 5-6 Control effort for the right (a) and left (b) motor (PI control).

5.4 Fuzzy controller using PI controller as a teacher

This section presents the development of a dynamic fuzzy controller, where the fuzzy controller replaces the PI controller. Chapter three presented the main design methodology of fuzzy logic systems and clustering techniques for model identification. However, the establishment of an input-output model for a process is still a very important problem in dynamic systems analysis. The replacement of the PI controller with fuzzy is difficult and challenging task. The reason behind this statement is that the plant to be controlled is MIMO with dynamical behaviour. In addition, the relationship of one output to another is coupled. Therefore in this case the design of fuzzy controller is non-trivial task. Fuzzy controller outputs are static functions of the controlled input. Dynamical behaviour of a controller, like differential or integral action, is emulated by extending the controller function to more inputs. Those inputs are delayed values of inputs and outputs. One problem with this is the large number of inputs, which results a large number of rules in the rule base of the fuzzy controller. To design such a controller manually is a very complicated and difficult task. Also, it is known that a fuzzy controller with more rules is not necessarily better than one with fewer rules. Thus, it is important to develop efficient rules for such controllers. As the scope of fuzzy controller development is to compare its execution

time and its performance against the PI controller, subtractive clustering is used. Despite the existence of many clustering techniques, subtractive clustering was chosen for the reasons already mentioned in chapter three section 3.4.2.3. This method generates Sugeno type fuzzy controller (see chapter three, section 3.2.4.2) with the fewer number of rules (this will produce faster controller). The steps of the algorithmic methodology for the design procedure of the fuzzy controller with dynamical behaviour based on subtractive clustering are defined as follows:

5.4.1 Selection of input-output data

The first step of the proposed algorithmic methodology is to select and collect input-output data using the PI controller from section 5.3 as a teacher. Figure 5-7 shows a Simulink model, where the inputs and outputs of the PI controller are recorded for the construction of the fuzzy controller. Equations (5.6) and (5.7) shows in symbolic form the input-output recorded data to be used to define the multi-dimensional input-output space.

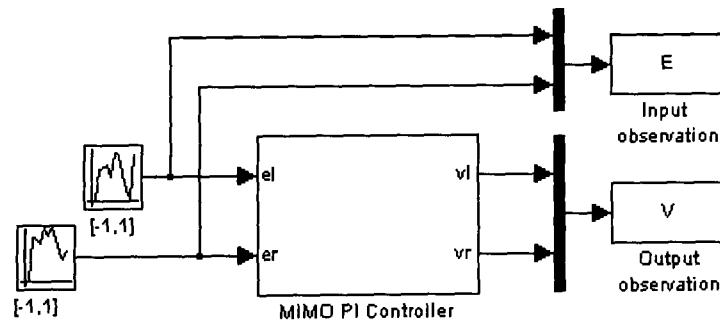


Figure 5-7 Input-output observations for data collection

$$E_{FL} = \begin{bmatrix} el_1 & er_1 \\ el_2 & er_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ el_n & er_n \end{bmatrix} \quad (5.6)$$

$$V_{FL} = \begin{bmatrix} vl_1 & vr_1 \\ vl_2 & vr_2 \\ \vdots & \vdots \\ vl_m & vr_m \end{bmatrix} \quad (5.7)$$

5.4.2 Subtractive clustering to define the number of rules

As mentioned in section 5.4 dynamical behaviour of a controller, such as differential or integral action, is emulated by extending the controller function to more inputs. Those inputs are delayed values of inputs and outputs. Thus, as the PI controller (see Figure 5-1) has dynamical behaviour due to the appearance of integrator the fuzzy controller should have as additional inputs delayed values of inputs and outputs. Figure 5-8 shows the schematic layout of inputs and output of the fuzzy controller with sampling time and delay of 10ms.

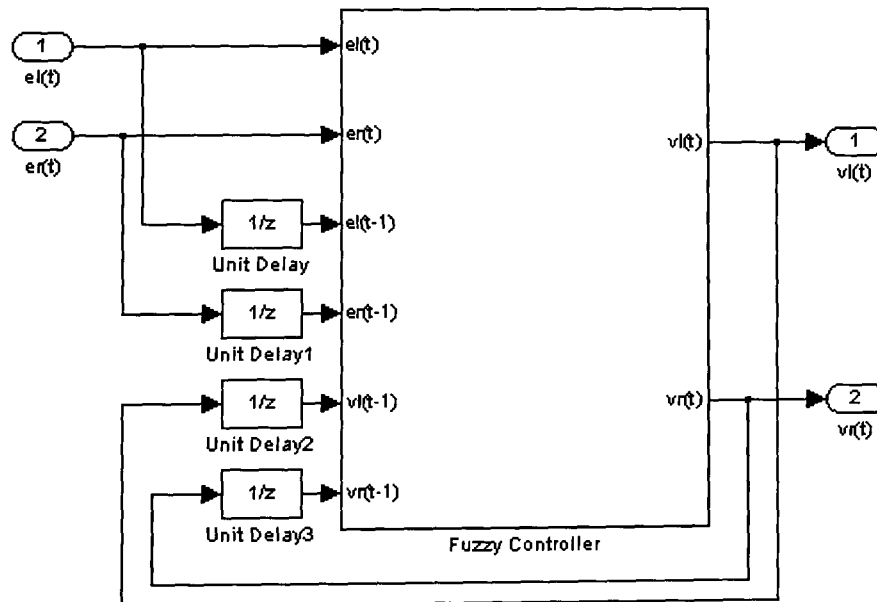


Figure 5-8 Schematic layout of fuzzy controller with delayed inputs and outputs

The next step is to construct the multi-dimensional input-output space and to apply subtractive clustering for automatic rule generation of the fuzzy inference system. The multi-dimensional input-output space is shown in Equation (5.8).

$$\psi_{FL} = \begin{bmatrix} vl_2 & vr_2 & el_2 & er_2 & el_1 & er_1 & vl_1 & vr_1 \\ vl_3 & vr_3 & el_3 & er_3 & el_2 & er_2 & vl_2 & vr_2 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ vl_n & vr_n & el_n & er_n & el_{n-1} & er_{n-1} & vl_{n-1} & vr_{n-1} \end{bmatrix} \quad (5.8)$$

The subtractive clustering algorithm presented in chapter three generates a model from data given in Equation (5.8), when specific cluster radius is defined (see chapter three section 3.4.2.3). The cluster radius indicates the range of influence of a cluster when a data space is considered as a unit hypercube. Specifying a small cluster radius will usually yield many small clusters in the data and thus many rules in the rule base of the fuzzy controller and vice versa. The radius cluster is selected to be 0.5 (it is recommended as initial value, if the resulting model is inaccurate then the cluster radius needs to be changed). Applying subtractive clustering a fuzzy inference system is returned. In this case the controller type for the fuzzy inference system is a first order Sugeno model with 64 rules in the rule base. The number of rules is small considering the large number of input variables (six) to the fuzzy controller. The next step is to verify the fuzzy controller performance.

5.4.3 Fuzzy controller testing and verification

To test and verify the fuzzy controller the mobile robot was simulated with step input commands. The experiment was carried out for 3s using step input on both wheels as desired input velocity command. The comparison of both reference and actual responses under perturbations of control inputs is shown in Figure 5-9(a) and Figure 5-9(b). In Figure 5-9(a) the desired input linear velocity for the right wheel is 0.5m/s at $t=0$ and 0.3m/s at $t=2s$. In

Figure 5-9(b) the desired input linear velocity for the left wheel is 0.4 m/s at $t = 0$ and 0.8 m/s at $t = 1\text{ s}$.

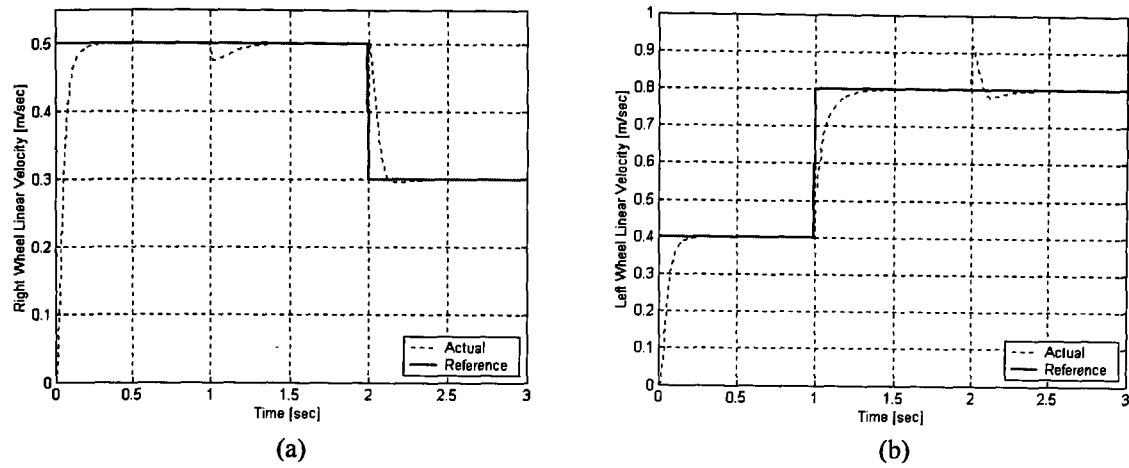


Figure 5-9 Comparison between the reference and actual response (fuzzy control). (a) Step input for the right wheel (b) Step input for the left wheel

Comparing the two responses (reference and actual), it can be concluded that fuzzy control is successful, in that the robot produces the desired performance as defined in section 5.2.2. The control effort, illustrated in Figure 5-10(a) and Figure 5-10(b) show that the motor input voltages do not exceed 6 V .

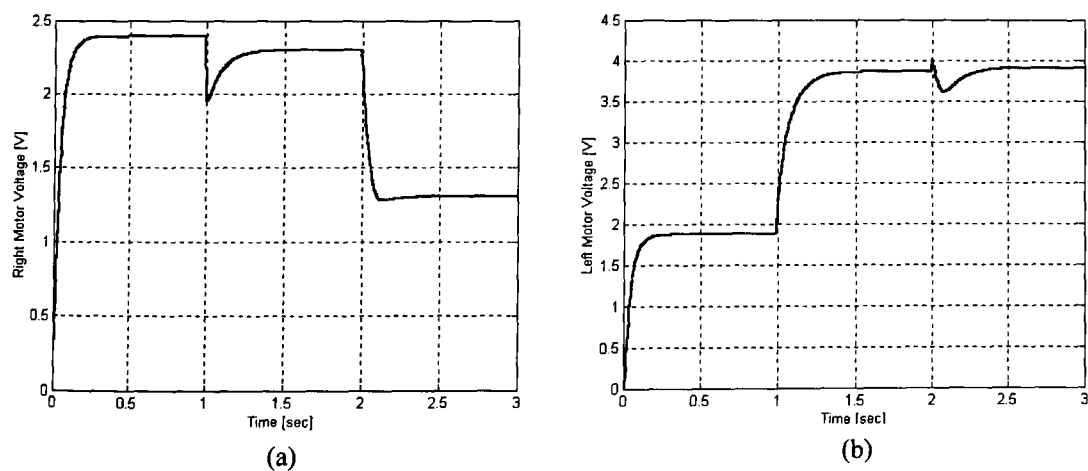


Figure 5-10 Control effort for the right (a) and left (b) motor (fuzzy control).

5.5 Neural controller using PI controller as a teacher

This section presents the development of a dynamic neural controller, where the neural network controller replaces the PI controller. In chapter three the main design methodology of neural networks and their training techniques was presented. In this section feedforward neural network is trained in order to be able to control the speed of the mobile robot. Already mentioned in section 5.4 that fuzzy systems are usually static mapping of input-output. Similarly, neural network is static mapping indicating that theoretically it is not feasible to control or identify a dynamic system. To extend this essentially steady state mapping to the dynamic domain is to adopt an approach similar to linear theory of ARMA (Auto Regressive Moving Average) modelling. As in section 5.4 the ANN is fed with past outputs and inputs values as shown in Figure 5-11.

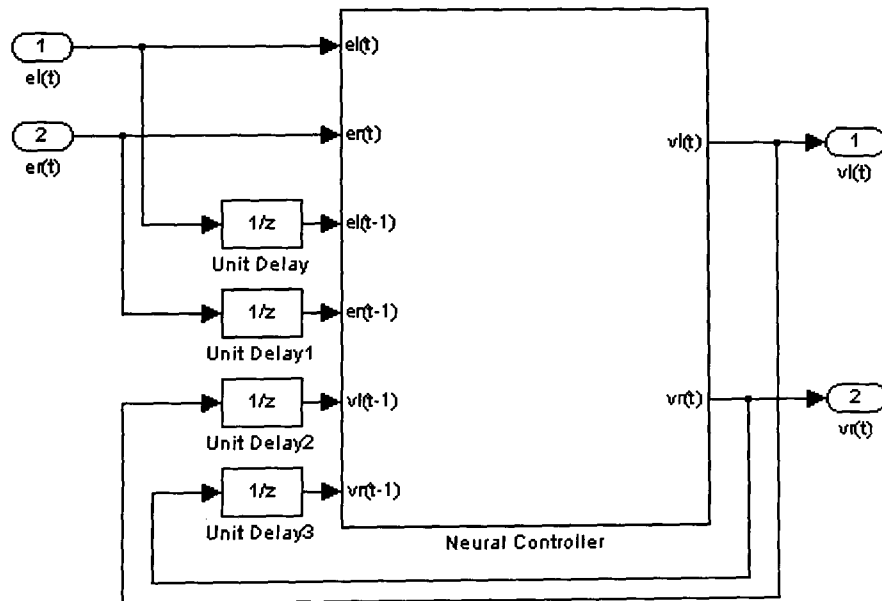


Figure 5-11 Schematic layout of neural controller with delayed inputs and outputs

5.5.1 Training the neural controller

The main design methodology when training a neural network is shown in Figure 5-12. At the beginning of the training, data are collected or generated for both training and testing of the neural network.

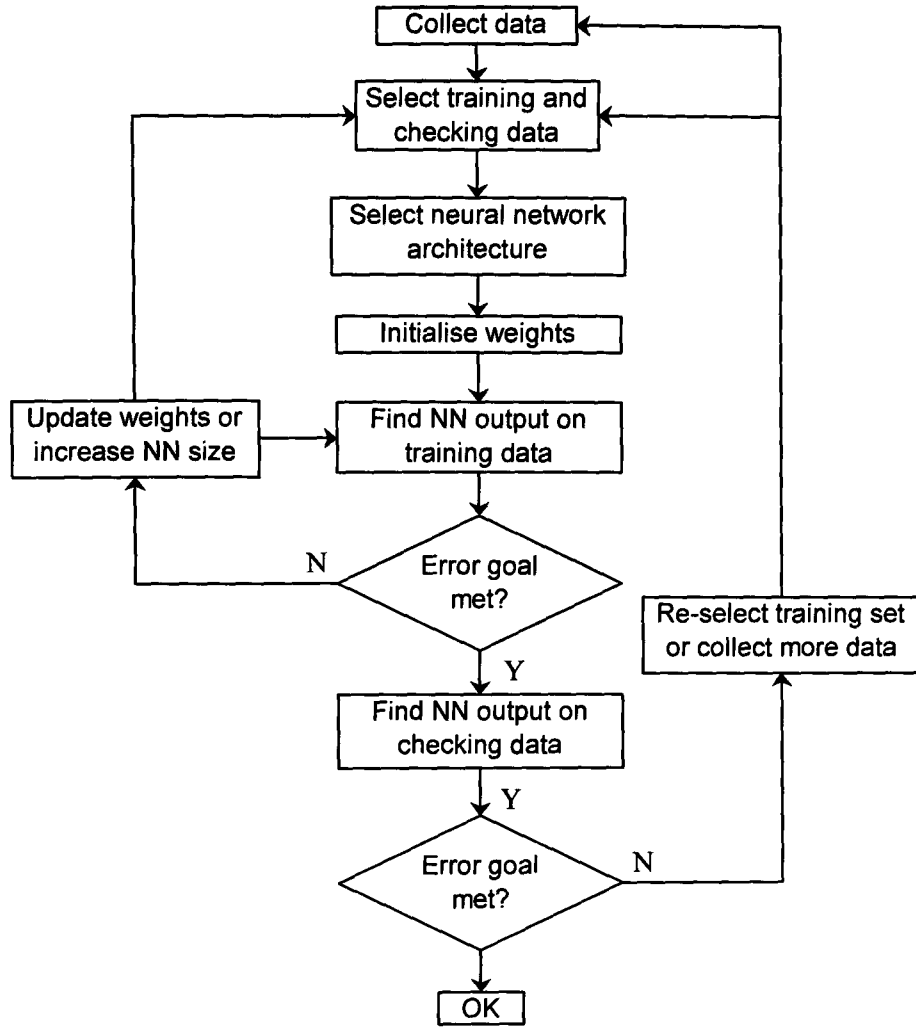


Figure 5-12 Neural network training flow chart

Equations (5.9) and (5.10) shows in symbolic form the input-output collected data to be used to define input and target data of the neural network.

$$E_{NN} = \begin{bmatrix} el_1 & el_2 & \dots & el_n \\ er_1 & er_2 & \dots & er_n \end{bmatrix} \quad (5.9)$$

$$V_{NN} = \begin{bmatrix} vl_1 & vl_2 & \dots & vl_m \\ vr_1 & vr_2 & \dots & vr_m \end{bmatrix} \quad (5.10)$$

Using Equations (5.9) and (5.10) the input training data are shown in Equation (5.11) and the target data are shown in Equation (5.12) as follows:

$$E_{NN}^{\text{tr}} = \begin{bmatrix} e_{l_2} & e_{l_3} & . & . & . & e_{l_n} \\ e_{r_2} & e_{l_3} & . & . & . & e_{r_n} \\ e_{l_1} & e_{l_2} & . & . & . & e_{l_{n-1}} \\ e_{r_1} & e_{r_2} & . & . & . & e_{r_{n-1}} \\ v_{l_1} & v_{l_2} & . & . & . & v_{l_{n-1}} \\ v_{r_1} & v_{r_2} & . & . & . & v_{r_{n-1}} \end{bmatrix} \quad (5.11)$$

$$V_{NN}^{\text{tr}} = \begin{bmatrix} v_{l_2} & v_{l_3} & . & . & . & v_{l_n} \\ v_{r_2} & v_{r_3} & . & . & . & v_{r_n} \end{bmatrix} \quad (5.12)$$

The next step is to define the neural network architecture. According to (Hines, 1997) there are two options available when an ANN architecture has to be defined. The first option is to start with a fairly large network that is sure to have enough degrees of freedom (large number of neurons in the hidden layer(s)) to train to the desired error goal. Then, once the network is trained, the network size is reduced until the smallest size of network that trains remains. The second option available is to start with a small network and gradually to increase its size until the network trains and its error goal is met. In this work the second option has been selected which involves initially a fairly small network architecture. After the network is chosen, the weights and biases are initialised and the network is ready for training.

Using the input and target data from Equations (5.11) and (5.12) the training process starts. During training the weights and biases of the network are iteratively adjusted to minimise the network performance function (error goal). Usually the performance function for feedforward networks is the mean square error (MSE), which is the average squared error between the network outputs and the target outputs. In chapter three section 3.3.4 the background theory behind training of feedforward multilayer perception NN was given, included the different

training algorithms available in the literature when network training is considered. All of these algorithms use the gradient performance function to determine how to adjust the weights to minimise the error goal. The gradient is determined using a technique called backpropagation (see chapter three section 3.3.4 and Appendix F), which involves performing computations backwards through the network. In this thesis the algorithm of Levenberg-Marquardt is used in order to increase the speed of convergence. For more details of the algorithm see Appendix F. The final structure of the dynamic NN controller is shown in Figure 5-13.

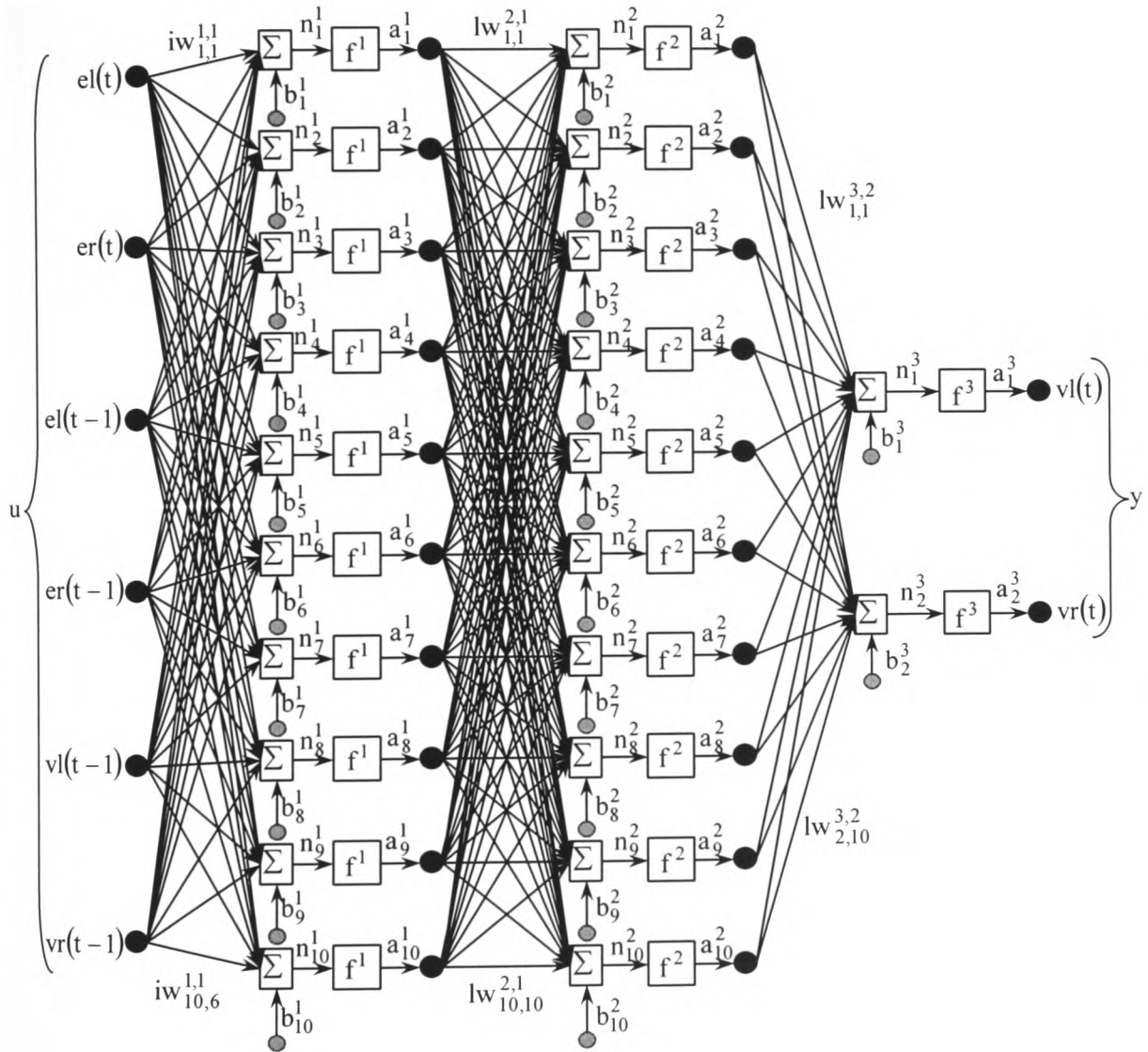


Figure 5-13 Structure of dynamic NN using PI controller as a teacher

The network shown in Figure 5-13 has two hidden layers (ten neurons on each layer) and an output layer consisting of two neurons. The input and output to the network are column vectors as shown in Equations (5.13) and (5.14).

$$u = \begin{bmatrix} el(t) \\ er(t) \\ el(t-1) \\ er(t-1) \\ vl(t-1) \\ vr(t-1) \end{bmatrix} \quad (5.13)$$

$$y = \begin{bmatrix} vl(t) \\ vr(t) \end{bmatrix} \quad (5.14)$$

Each layer has a weight matrix \mathbf{W} , a bias vector \mathbf{b} , and an output vector \mathbf{a} . The bias vector of the first, second and third layer is defined by the following equation.

$$\mathbf{b}^1 = \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \\ b_4^1 \\ b_5^1 \\ b_6^1 \\ b_7^1 \\ b_8^1 \\ b_9^1 \\ b_{10}^1 \end{bmatrix}, \quad \mathbf{b}^2 = \begin{bmatrix} b_1^2 \\ b_2^2 \\ b_3^2 \\ b_4^2 \\ b_5^2 \\ b_6^2 \\ b_7^2 \\ b_8^2 \\ b_9^2 \\ b_{10}^2 \end{bmatrix}, \quad \mathbf{b}^3 = \begin{bmatrix} b_1^3 \\ b_2^3 \end{bmatrix} \quad (5.15)$$

The weight matrix of the first, second and third layer is given by Equations (5.16), (5.17) and (5.18) as follows:

$$IW^{1,1} = \begin{bmatrix} iw_{1,1}^{1,1} & iw_{1,2}^{1,1} & iw_{1,3}^{1,1} & iw_{1,4}^{1,1} & iw_{1,5}^{1,1} & iw_{1,6}^{1,1} \\ iw_{2,1}^{1,1} & iw_{2,2}^{1,1} & iw_{2,3}^{1,1} & iw_{2,4}^{1,1} & iw_{2,5}^{1,1} & iw_{2,6}^{1,1} \\ iw_{3,1}^{1,1} & iw_{3,2}^{1,1} & iw_{3,3}^{1,1} & iw_{3,4}^{1,1} & iw_{3,5}^{1,1} & iw_{3,6}^{1,1} \\ iw_{4,1}^{1,1} & iw_{4,2}^{1,1} & iw_{4,3}^{1,1} & iw_{4,4}^{1,1} & iw_{4,5}^{1,1} & iw_{4,6}^{1,1} \\ iw_{5,1}^{1,1} & iw_{5,2}^{1,1} & iw_{5,3}^{1,1} & iw_{5,4}^{1,1} & iw_{5,5}^{1,1} & iw_{5,6}^{1,1} \\ iw_{6,1}^{1,1} & iw_{6,2}^{1,1} & iw_{6,3}^{1,1} & iw_{6,4}^{1,1} & iw_{6,5}^{1,1} & iw_{6,6}^{1,1} \\ iw_{7,1}^{1,1} & iw_{7,2}^{1,1} & iw_{7,3}^{1,1} & iw_{7,4}^{1,1} & iw_{7,5}^{1,1} & iw_{7,6}^{1,1} \\ iw_{8,1}^{1,1} & iw_{8,2}^{1,1} & iw_{8,3}^{1,1} & iw_{8,4}^{1,1} & iw_{8,5}^{1,1} & iw_{8,6}^{1,1} \\ iw_{9,1}^{1,1} & iw_{9,2}^{1,1} & iw_{9,3}^{1,1} & iw_{9,4}^{1,1} & iw_{9,5}^{1,1} & iw_{9,6}^{1,1} \\ iw_{10,1}^{1,1} & iw_{10,2}^{1,1} & iw_{10,3}^{1,1} & iw_{10,4}^{1,1} & iw_{10,5}^{1,1} & iw_{10,6}^{1,1} \end{bmatrix} \quad (5.16)$$

$$LW^{2,1} = \begin{bmatrix} lw_{1,1}^{2,1} & lw_{1,2}^{2,1} & lw_{1,3}^{2,1} & lw_{1,4}^{2,1} & lw_{1,5}^{2,1} & lw_{1,6}^{2,1} & lw_{1,7}^{2,1} & lw_{1,8}^{2,1} & lw_{1,9}^{2,1} & lw_{1,10}^{2,1} \\ lw_{2,1}^{2,1} & lw_{2,2}^{2,1} & lw_{2,3}^{2,1} & lw_{2,4}^{2,1} & lw_{2,5}^{2,1} & lw_{2,6}^{2,1} & lw_{2,7}^{2,1} & lw_{2,8}^{2,1} & lw_{2,9}^{2,1} & lw_{2,10}^{2,1} \\ lw_{3,1}^{2,1} & lw_{3,2}^{2,1} & lw_{3,3}^{2,1} & lw_{3,4}^{2,1} & lw_{3,5}^{2,1} & lw_{3,6}^{2,1} & lw_{3,7}^{2,1} & lw_{3,8}^{2,1} & lw_{3,9}^{2,1} & lw_{3,10}^{2,1} \\ lw_{4,1}^{2,1} & lw_{4,2}^{2,1} & lw_{4,3}^{2,1} & lw_{4,4}^{2,1} & lw_{4,5}^{2,1} & lw_{4,6}^{2,1} & lw_{4,7}^{2,1} & lw_{4,8}^{2,1} & lw_{4,9}^{2,1} & lw_{4,10}^{2,1} \\ lw_{5,1}^{2,1} & lw_{5,2}^{2,1} & lw_{5,3}^{2,1} & lw_{5,4}^{2,1} & lw_{5,5}^{2,1} & lw_{5,6}^{2,1} & lw_{5,7}^{2,1} & lw_{5,8}^{2,1} & lw_{5,9}^{2,1} & lw_{5,10}^{2,1} \\ lw_{6,1}^{2,1} & lw_{6,2}^{2,1} & lw_{6,3}^{2,1} & lw_{6,4}^{2,1} & lw_{6,5}^{2,1} & lw_{6,6}^{2,1} & lw_{6,7}^{2,1} & lw_{6,8}^{2,1} & lw_{6,9}^{2,1} & lw_{6,10}^{2,1} \\ lw_{7,1}^{2,1} & lw_{7,2}^{2,1} & lw_{7,3}^{2,1} & lw_{7,4}^{2,1} & lw_{7,5}^{2,1} & lw_{7,6}^{2,1} & lw_{7,7}^{2,1} & lw_{7,8}^{2,1} & lw_{7,9}^{2,1} & lw_{7,10}^{2,1} \\ lw_{8,1}^{2,1} & lw_{8,2}^{2,1} & lw_{8,3}^{2,1} & lw_{8,4}^{2,1} & lw_{8,5}^{2,1} & lw_{8,6}^{2,1} & lw_{8,7}^{2,1} & lw_{8,8}^{2,1} & lw_{8,9}^{2,1} & lw_{8,10}^{2,1} \\ lw_{9,1}^{2,1} & lw_{9,2}^{2,1} & lw_{9,3}^{2,1} & lw_{9,4}^{2,1} & lw_{9,5}^{2,1} & lw_{9,6}^{2,1} & lw_{9,7}^{2,1} & lw_{9,8}^{2,1} & lw_{9,9}^{2,1} & lw_{9,10}^{2,1} \\ lw_{10,1}^{2,1} & lw_{10,2}^{2,1} & lw_{10,3}^{2,1} & lw_{10,4}^{2,1} & lw_{10,5}^{2,1} & lw_{10,6}^{2,1} & lw_{10,7}^{2,1} & lw_{10,8}^{2,1} & lw_{10,9}^{2,1} & lw_{10,10}^{2,1} \end{bmatrix} \quad (5.17)$$

$$LW^{3,2} = \begin{bmatrix} lw_{1,1}^{3,2} & lw_{1,2}^{3,2} & lw_{1,3}^{3,2} & lw_{1,4}^{3,2} & lw_{1,5}^{3,2} & lw_{1,6}^{3,2} & lw_{1,7}^{3,2} & lw_{1,8}^{3,2} & lw_{1,9}^{3,2} & lw_{1,10}^{3,2} \\ lw_{2,1}^{3,2} & lw_{2,2}^{3,2} & lw_{2,3}^{3,2} & lw_{2,4}^{3,2} & lw_{2,5}^{3,2} & lw_{2,6}^{3,2} & lw_{2,7}^{3,2} & lw_{2,8}^{3,2} & lw_{2,9}^{3,2} & lw_{2,10}^{3,2} \end{bmatrix} \quad (5.18)$$

The activation function used for the first two layers is hyperbolic tangent sigmoid (tansig) whereas linear (purelin) activation function is used for the last layer. The mathematical expressions of the most commonly used activation functions including hyperbolic tangent sigmoid and linear is given in chapter three section 3.3.2.1. Note that the chain rule that is used in deriving the back-propagation algorithm necessitates the computation of the derivative of the activation functions. The derivative of hyperbolic tangent sigmoid and linear activation function is given by:

control effort, illustrated in Figure 5-15(a) and Figure 5-15(b) show that the motor input voltages do not exceed 6V .

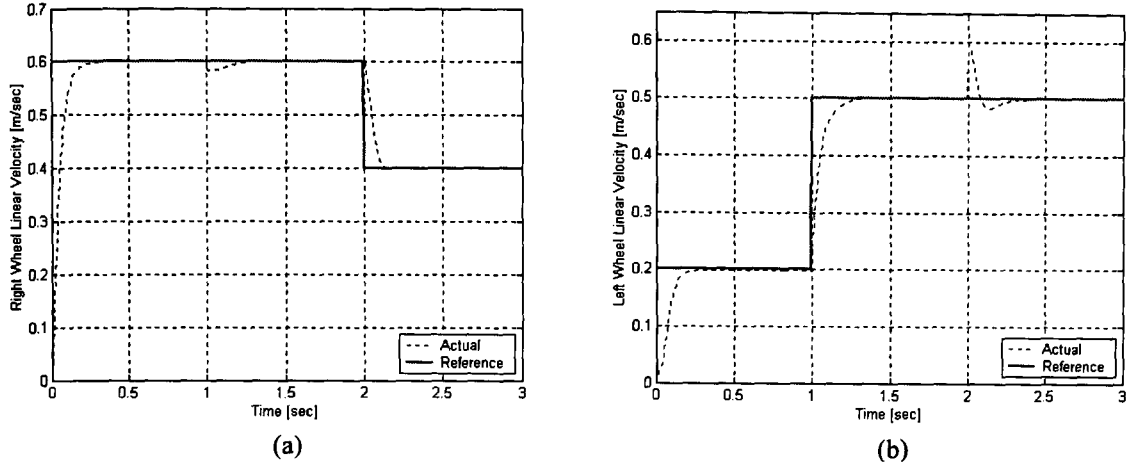


Figure 5-14 Comparison between the reference and actual response (neural control). (a) Step input for the right wheel (b) Step input for the left wheel.

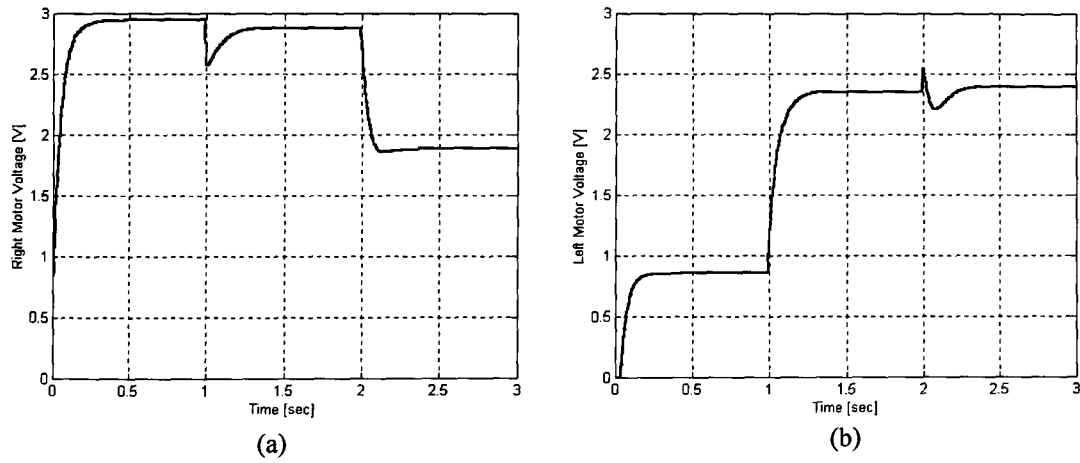


Figure 5-15 Control effort of the left and right motor respectively (neural control)

5.6 Comparison between the different controllers

In this section comparison of all three types of controllers is made based on their execution time and performance. As mentioned in the introduction of this chapter the overall control architecture should be modular. To address this modularly even from the low-level control unit,

all three types of controllers have been created as a subsystem as shown in Figure 5-16. Each controller can be selected from the mask block shown in Figure 5-17.

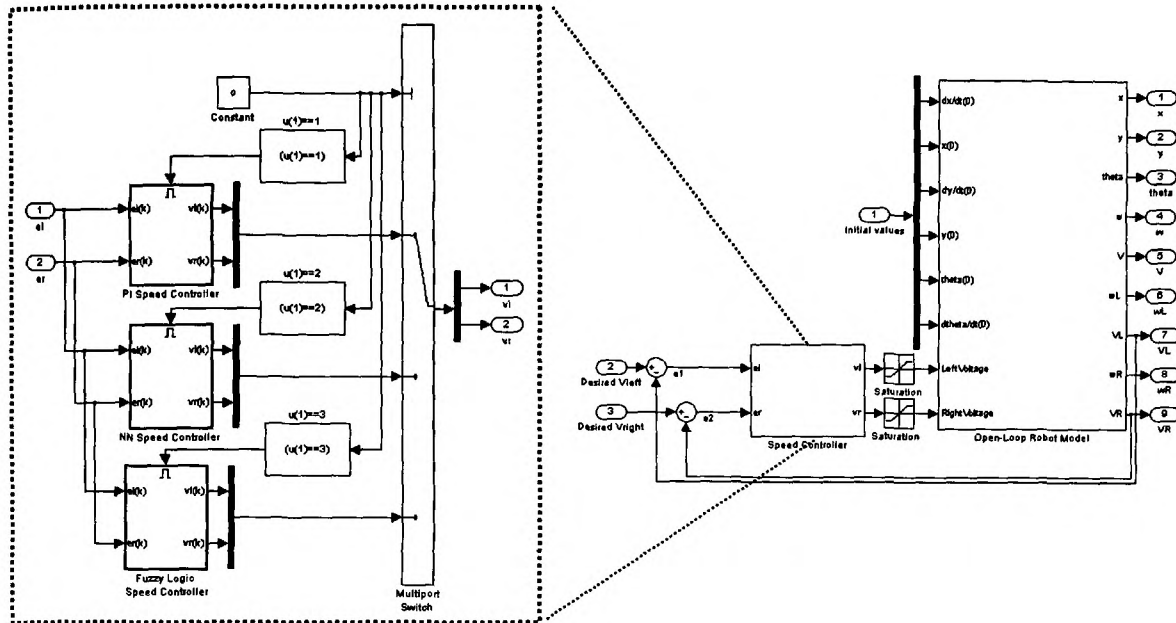


Figure 5-16 Subsystem of speed controller (PI, FL and NN controller)

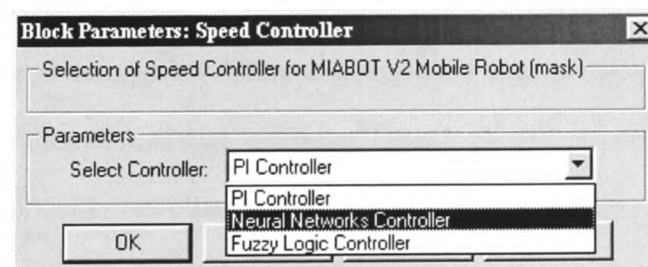


Figure 5-17 Mask block for selection of speed controller

To compare the execution time and performance of all three types of controllers the mobile robot was commanded under the same perturbations of the control inputs. The experiments were carried out for 3s using step input on both wheels as desired input velocity command. The

desired input linear velocity for the right wheel is 0.3m/s at $t = 0$ and the desired input linear velocity for the left wheel is 0.1m/s at $t = 0$ and 0.6m/s at $t = 1$ s. The comparison of both reference and actual responses under perturbations of the same control inputs of all three types of controllers is shown in Figure 5-18, Figure 5-19 and Figure 5-20. Table 1 illustrates the performance of each controller in terms of integral square error (ISE), integral absolute error (IAE) and integral time absolute error (ITAE) for each response (left and right), and also the elapsed time taken for 3s simulation time.

Examining the time responses obtained from Figure 5-18 to Figure 5-20 the following conclusions can be drawn. From Table 1 it can be seen that PI and fuzzy speed controllers have a marginal advantage against the neural controller (red shaded numbers show the best performance). In particular the PI controller has the smallest ISE error of all controllers, whereas the fuzzy controller has the smallest IAE and ITAE error. This simulation study shows that both fuzzy systems and neural networks can successfully model and control a dynamic system when their training is based upon a teacher with good input-output training data. However it is shown that a fuzzy or neural controller is not necessary better than classical PI controller and vice versa. In chapter seven the performance of this controller is validated when the mobile robot has to accomplish different navigational tasks. Another important remark in this study is the execution time of each controller. Table 1 shows that neural controller is almost three times slower than PI controller whereas a fuzzy controller is almost twice slower than neural controller. The explanation behind the large execution time of fuzzy controller lies in terms that fuzzy controller has a large number of parameters. This indicates the degree of complexity when fuzzy controller performs operations such as fuzzification, rule base evaluation and in defuzzification.

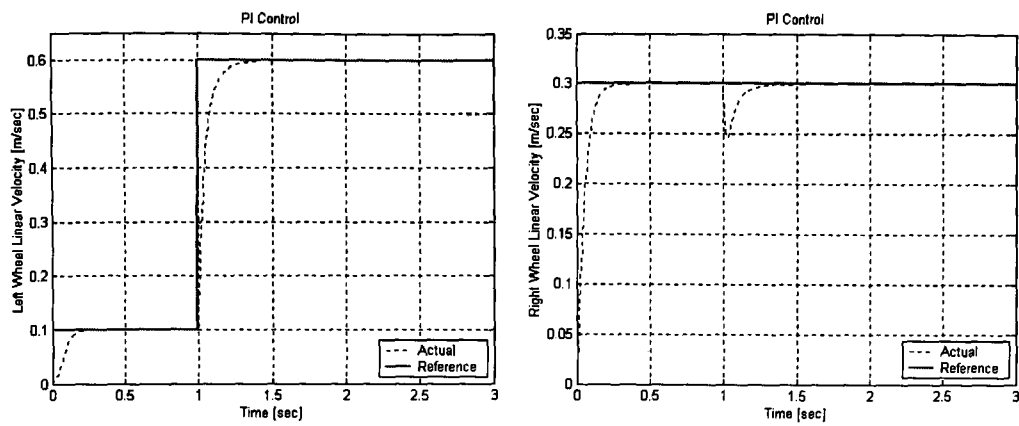


Figure 5-18 Performance of PI control

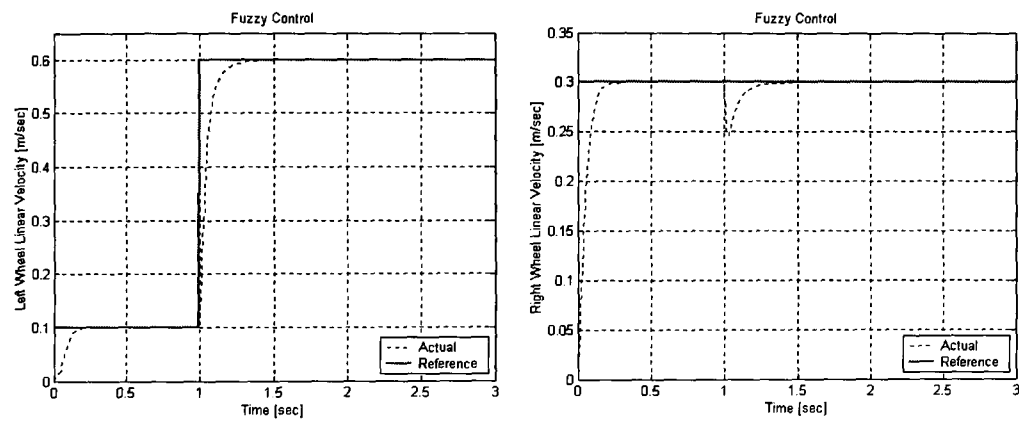


Figure 5-19 Performance of fuzzy control

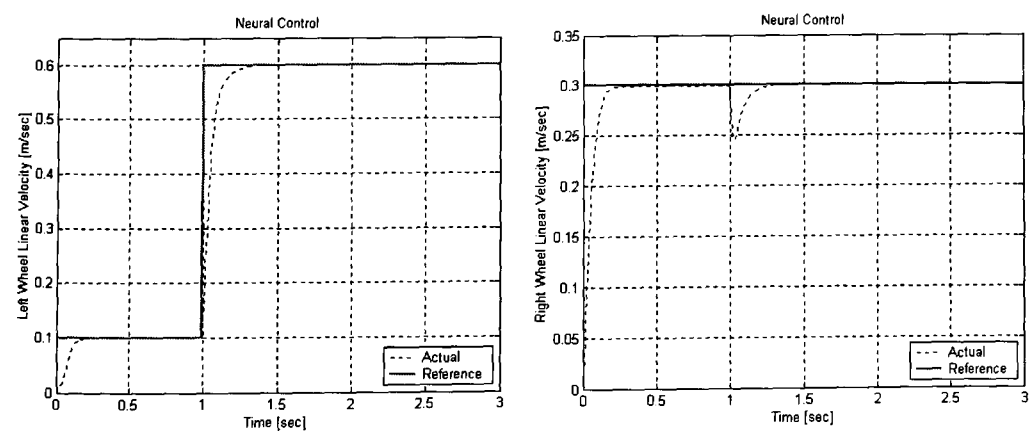


Figure 5-20 Performance of neural control

Performance criteria and execution time of PI, fuzzy and neural speed controller							
Type of controller	ISE		IAE		ITAE		Elapsed time (sec)
	Left	Right	Left	Right	Left	Right	
PI	0.008047	0.003207	0.03458	0.02218	0.02778	0.006594	1.2600
Fuzzy	0.008324	0.003269	0.03362	0.02169	0.02681	0.006483	7.0300
Neural	0.008333	0.003286	0.03604	0.025	0.032	0.01318	3.9600

Table 1 Comparison of performance and execution time of PI, fuzzy and neural controller

5.7 Robust stability testing of MIABOT V2 closed-loop

In this section the robust stability of the closed-loop control system of the MIABOT V2 mobile robot under uncertainty in robot dynamics is tested and proved. In Appendix E, definitions and theorems from (Bhattacharyya *et al*, 1995) related to robust stability analysis for interval polynomials is given. Description of uncertainty structure (Appendix E, section 2) is given through several definitions following by both definitions and theorems regarding Value Sets and Zero Exclusion condition (Appendix E, section 3). Then the Kharitonov's theorem (Barmish, 1994) is described in brief following theorems on robust stability testing via graphics (Appendix E, section 4).

The following sections are organised as follows: The application (robust stability testing of mobile robot closed-loop control system under uncertainty in robot dynamics) is presented in section 5.7.1. The robust stability analysis of the closed-loop control system is verified using graphical techniques in section 5.7.2 (see Appendix E, section 5 for background information). Section 5.8 presents a discussion based on the importance of the selection of the interval polynomial family.

5.7.1 The application

Figure 5-1 shows the MIABOT's actual closed-loop control system consisting of MIMO speed controller and an uncertain plant (mobile robot). The open-loop transfer function matrix (TFM) for the mobile robot was derived in chapter four as shown in Equation (5.21). To test and prove

the robust stability of the closed-loop system under uncertainty in robot dynamics the transfer function of the PI speed controller is used, which is easy to understand and analyse (note that three types of speed controllers exist).

$$G(s) = \begin{bmatrix} \frac{10.99s + 572.4}{s^2 + 108s + 2835} & \frac{2.317s + 26.24}{s^2 + 108s + 2835} \\ \frac{2.317s + 26.24}{s^2 + 108s + 2835} & \frac{10.99s + 572.4}{s^2 + 108s + 2835} \end{bmatrix} \quad (5.21)$$

The MIMO PI speed controller $G_c(s)$ of the mobile robot can be described in the following TFM form:

$$G_c(s) = \begin{bmatrix} \frac{1.354s + 103}{s} & \frac{-1.577s - 25.19}{s} \\ \frac{-1.577s - 25.19}{s} & \frac{1.354s + 103}{s} \end{bmatrix} \quad (5.22)$$

Thus Equation (5.23) describes the overall closed-loop TFM, $G'(s)$.

$$G'(s) = \begin{bmatrix} \frac{10.87s^3 + 1679s^2 + 7.806e004s + 1.079e006}{s^4 + 129.7s^3 + 6265s^2 + 1.345e005s + 1.079e006} & \frac{-13.75s^3 - 877.2s^2 - 1.135e004s - 0.01564}{s^4 + 129.7s^3 + 6265s^2 + 1.345e005s + 1.079e006} \\ \frac{-13.75s^3 - 877.2s^2 - 1.135e004s - 0.01564}{s^4 + 129.7s^3 + 6265s^2 + 1.345e005s + 1.079e006} & \frac{10.87s^3 + 1679s^2 + 7.806e004s + 1.079e006}{s^4 + 129.7s^3 + 6265s^2 + 1.345e005s + 1.079e006} \end{bmatrix} \quad (5.23)$$

According to the definitions and theorems from Appendix E sections 2, 3 and 4 the robust stability of the closed-loop system under uncertainty in robot dynamics of Equation (5.23) can be tested and proved. The characteristic equation of the closed-loop system can be written as interval polynomial in the following form:

$$p(s, q) = q_4 s^4 + q_3 s^3 + q_2 s^2 + q_1 s + q_0 \quad (5.24)$$

Where,

$$\mathbf{q} = [q_4, q_3, q_2, q_1, q_0] \quad (5.25)$$

is the vector of uncertain parameters, and assume that,, $q_3 \in [123.2, 136.2]$, $q_2 \in [5951, 6579]$, $q_1 \in [127780, 141230]$ and $q_0 \in [1025100, 1133000]$ (note that no closed interval is taken for q_4 as it is always 1) then the uncertainty bounding set (Appendix E, Definition E-1) is:

$$Q = \left\{ \mathbf{q} \left| \begin{array}{l} q_0 \in [1025100, 1133000], q_1 \in [127780, 141230] \\ q_2 \in [5951, 6579], q_3 \in [123.2, 136.2] \end{array} \right. \right\} \quad (5.26)$$

The above interval polynomial family is denoted by writing an interval polynomial family of the form:

$$p(s, \mathbf{q}) = s^4 + [123.2, 136.2]s^3 + [5951, 6579]s^2 + [127780, 141230]s + [1025100, 1133000] \quad (5.27)$$

Where $0 \notin [q_4^-, q_4^+] = 1$. Thus interval polynomial family $P(s, Q)$ has invariant degree. From Definition E-10 given in Appendix E the four fixed Kharitonov polynomials are derived as follows:

$$K_1(s) = s^4 + 136.2s^3 + 6579s^2 + 127780s + 1025100 \quad (5.28)$$

$$K_2(s) = s^4 + 123.2s^3 + 5951s^2 + 141230s + 1133000 \quad (5.29)$$

$$K_3(s) = s^4 + 136.2s^3 + 5951s^2 + 127780s + 1133000 \quad (5.30)$$

$$K_4(s) = s^4 + 123.2s^3 + 6579s^2 + 141230s + 1025100 \quad (5.31)$$

Using Routh criterion it is easy to verify that all four Kharitonov polynomials are stable. Hence it can be concluded that the closed-loop control system is robustly stable under uncertainty in robot dynamics.

5.7.2 Verification of the closed-loop robust stability

To verify that the closed-loop control system is robustly stable further testing using graphics is performed using the Theorem E-3 and E-4 given in Appendix E, section 5. The characteristic equation of the closed-loop system is given from Equation (5.23). Equations (5.24) and (5.25) provide the uncertainty vector \mathbf{q} and the uncertainty bounding set Q . It was shown earlier that the interval polynomial family $P(s, Q)$ has invariant degree, therefore in accordance with Theorem E-3, the first step in the graphical test for robust stability requires that at least one polynomial in $P(s, Q)$ that is stable. Using the midpoint of each interval from Equation (5.26) \mathbf{q}^* is obtained as follows:

$$\mathbf{q}^* = (1, 129.7, 6265, 134505, 1079050) \quad (5.32)$$

Then,

$$p(s, \mathbf{q}^*) = s^4 + 129.7s^3 + 6265s^2 + 134505s + 1079050 \quad (5.33)$$

Using the Routh criterion it is easy to verify that $p(s, \mathbf{q}^*)$ is stable. The cut-off frequency can be calculated from Equation (E-11) as follows:

$$\omega_c = 1 + \frac{\max\{1133000, 141230, 6579, 136.2\}}{1} = 1133001 \text{ rad/s} \quad (5.34)$$

Therefore the Kharitonov rectangles can be plotted to verify the stability of the closed-loop system under uncertainty in robot dynamics for frequency range $\omega \in [0, 1133001] \text{ rad/s}$. For more convenient, frequency range $\omega \in [0, 100] \text{ rad/s}$ plot is shown in order the zero point in the graph to be visible. Figure 5-21 shows the Kharitonov rectangles for the closed-loop system. Figure 5-22 shows the plot of the frequency sweeping function $H(\omega)$. Since the origin is excluded from the Kharitonov rectangles (Figure 5-21) it is concluded that the closed-loop control system under uncertainty in robot dynamics is robustly stable. The same conclusion can be obtained from Figure 5-22 because it can be observed that the frequency sweeping function $H(\omega)$ is positive for all $\omega \in [0, 100] \text{ rad/s}$.

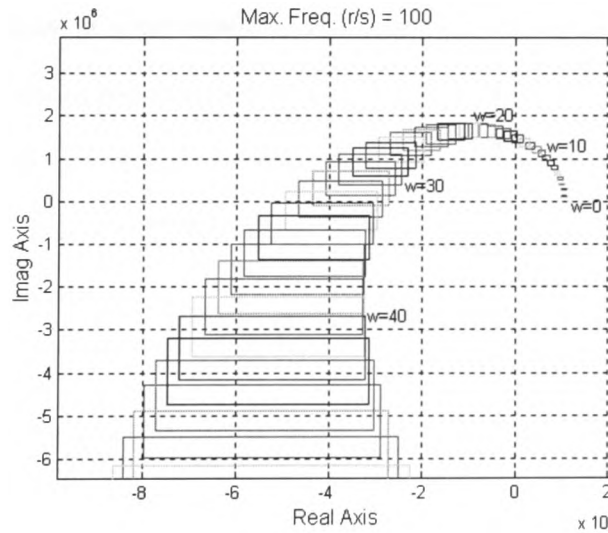


Figure 5-21 Kharitonov rectangles for the controlled closed-loop control system

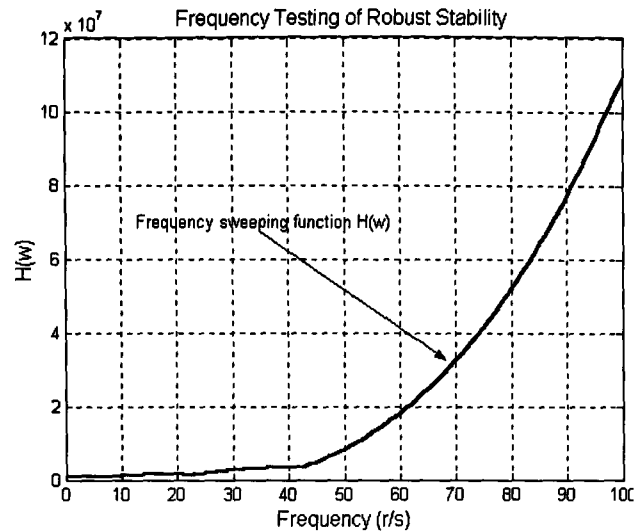


Figure 5-22 A plot of $H(\omega)$ versus ω .

5.8 Discussion

As mentioned in the introduction of this chapter, the objective was to develop successful speed control law for the mobile robot, to perform analysis and testing of the closed-loop robust stability under uncertainty in robot dynamics and to discover fuzzy-neural local models from observation data. In order to add modularity into the overall control architecture three types of controllers (PI, fuzzy and neural) were developed based on classical and intelligent methods. A discussion of some of the control, identification and robust stability algorithmic methodology issues is presented in this section.

In sections 5.2 and 5.3 was discussed how difficult is to successfully control the mobile robot as it is multivariable with coupled interconnection structure. Although there exists many controller designs methods in the literature the control solution in which can produce satisfactory result is still a difficult decision that has to be made. A successfully control law for the MIABOT V2 found to be PI MIMO controller with eight tuneable parameters. This control structure has to offer an advantage over the traditional PI controller as it gives more freedom for tuning. In other

words it is much better to control a process by tuning eight parameters rather than two in terms of controller freedom. The disadvantages of this approach are that the tuning process becomes very complex and highly non-linear problem. To overcome this problem the non-linear control design tool was used for the tuning/optimisation of the control parameters. The use of NCD in this chapter has demonstrated to be capable to solve high non-linear constrained optimisation and also successful tuning of PI controller based on non-linear model of the robot (note that most PID tuning methods are based on the linear model of the plant).

The next algorithmic methodology covered in this chapter is the problem of identifying, modelling and controlling dynamical systems. Sections 5.4 and 5.5 give two different approaches when dynamical system has to be developed based on clustering and supervised training using a teacher. In particular, section 5.4 demonstrates how fuzzy control can be developed to replace the PI controller. This algorithmic methodology is based on clustering of observed input-output data, which can successfully produce concise representation of the system's behaviour. Subtractive clustering was chosen, as it is capable to solve complex and high dimensional nature problems. However, disadvantages of other clustering techniques such as determination of optimal number of clusters and increase in computation as the dimensionality of the problem increases are overcome using this method. Using subtractive clustering a fuzzy inference system is generated with the minimum number of rules. This is desirable result especially when the fuzzy system has a high number of inputs. Results show that fuzzy logic can successfully model and control system with dynamical behaviour.

In section 5.5 an algorithmic methodology is proposed in which neural controller can model and control the plant based on training using the PI controller as a teacher. Results show that if appropriate network size has been defined and the training process (see Figure 5-12) is based on good input-output training data then the control law obtained is good and satisfactory.

In section 5.6 comparisons between the different controllers has been made. Using the same input command the controllers were tested for 3s simulation time. Examining the time responses obtained the results show that PI and fuzzy speed controllers have a marginal advantage against the neural controller. In particular the PI controller has the smallest ISE error of all controllers, whereas the fuzzy controller has the smallest IAE and ITAE error. It is interesting how fuzzy and neural controller controls the plant having dynamical behaviour characteristics. As mentioned above this is a result of good input-output observation data and appropriate size of neural network architecture. The main point highlighted already in section 5.6 is the execution time of each controller. The simulation study shows the huge advantage of PI control compared to any other control methods in terms of computational complexity. PI appears to be simple and effective. However both fuzzy and neural controllers despite their computational complexity equally control the plant. A question then arises of which control method is the best. Although a neural controller does not provide better performance in this simulation study this is not true when the system to be controlled is unknown and highly non-linear. These conclusions lead to the path of further study of possible combinations and development of hybrid system consisting of classical and intelligent techniques.

In section 5.7 the robust stability of the closed-loop was tested and proved under uncertainty. The uncertainty of the closed-loop system was modelled by replacing the coefficients of the closed-loop characteristic equation of the MIMO system with closed interval polynomials. Although the robust stability was proven, a question remains of how to map a closed-loop characteristic equation of system to the system's physical parameters (especially for MIMO systems). For example, the mobile robot for which the robust analysis took place weighs 0.5 kg. If there was a need for 10% increase of its mass (i.e. adding more sensing elements) how the coefficients of the closed-loop characteristic equation will change is of interest. To map the change of the robot's mass to the change in the coefficients of the closed-loop characteristic equation is very difficult. In order to demonstrate this, consider the modified open-loop robot

transfer function matrix in Equation (5.35), and the closed-loop transfer function of the system in Equation (5.36) resulting from the 10% increase in mass.

$$G_M(s) = \begin{bmatrix} \frac{10.06s + 504.2}{s^2 + 102.3s + 2577} & \frac{1.659s + 23.11}{s^2 + 102.3s + 2577} \\ \frac{1.659s + 23.11}{s^2 + 102.3s + 2577} & \frac{10.06s + 504.2}{s^2 + 102.3s + 2577} \end{bmatrix} \quad (5.35)$$

$$H_M(s) = \begin{bmatrix} \frac{11s^3 + 1576s^2 + 7.096e004s + 9.814e005}{s^4 + 124.3s^3 + 5793s^2 + 1.223e005s + 9.814e005} & \frac{-13.62s^3 - 846.4s^2 - 1.032e004s}{s^4 + 124.3s^3 + 5793s^2 + 1.223e005s + 9.814e005} \\ \frac{-13.62s^3 - 846.4s^2 - 1.032e004s}{s^4 + 124.3s^3 + 5793s^2 + 1.223e005s + 9.814e005} & \frac{11s^3 + 1576s^2 + 7.096e004s + 9.814e005}{s^4 + 124.3s^3 + 5793s^2 + 1.223e005s + 9.814e005} \end{bmatrix} \quad (5.36)$$

It can be observed that the intervals used for Equation (5.25) do not include all the variations in coefficients resulting from a 10% increase in mass. Care must therefore be taken in selecting the most suitable interval in order to accommodate the range of the expected parameter variations. The closed-loop control system described in Equation (5.36) was tested again for robust stability based on new uncertainty bounding set given in equation (5.37) and was found to be robustly stable.

$$Q = \left\{ \mathbf{q} \begin{cases} q_0 \in [981400, 1133000], q_1 \in [122300, 141230] \\ q_2 \in [5793, 6579], q_3 \in [123.2, 136.2] \end{cases} \right\} \quad (5.37)$$

5.9 Summary

This chapter presented the control, the robust stability analysis of the MIABOT V2 mobile robot and the discovery of fuzzy-neural local models from observation data. At the first instance PI control law was developed based on full non-linear model and design requirements derived in section 5.2.2 for the mobile robot. PI control action was selected, as it is computational simple and fast to design. However, as already mentioned there exists a clear relationship between PI (or PID) and system response parameters. The tuning/optimisation of the PI control parameters

was achieved using the non-linear control design tool (NCD). This design methodology was fast, easy to use and suitable for the control design, which was based on non-linear model. In addition the method was found to produce good performance and robust control under plant uncertainty. The results of tuning show that PI control action successfully controls the plant, providing performance within the design requirements. Then a design methodology to replace the PI controller by fuzzy logic controller was proposed. Based on subtractive clustering a fuzzy logic controller was identified using the PI controller as a teacher. The fuzzy controller was modelled with dynamical behaviour by emulating the controller function to more inputs (delays of inputs and outputs). To extend this design methodology a multilayer feedforward neural network with dynamical behaviour was developed based on supervised learning. Comparison was made of all three types of controllers based on their performance and execution time. It was shown that all controllers control the plant within the design requirements where the PI controller has the smallest ISE error of all controllers, whereas the fuzzy controller has the smallest IAE and ITAE error. However, the execution time found to vary dramatically between the controllers. The PI control action found to be as the fastest controller whereas the neural controller found to be faster than the fuzzy controller. Finally the closed-loop control system was tested for robustness under uncertainty in robot dynamics. Using the parametric robustness analysis approach (Kharitonov's Theorem) the closed-loop control system of the MIABOT V2 mobile robot was tested and proved to be robustly stable under uncertainty in robot dynamics. This method proved to be easy and fast to use. To reinforce the result of the approach taken the robust stability analysis of the closed-loop control system was verified using graphical techniques. The work covered in this chapter is used in chapter six as an algorithmic methodology for the design of local behaviours and also in chapter seven to form the low-level control unit of the proposed hybrid control architecture where its evaluation is taking place.

The contributions of this chapter are: Development of speed control action (MIMO speed controller) for the mobile robot based on non-linear model and non-linear control design tool.

Identification, modelling and control of dynamic system (controller) using fuzzy control based on subtractive clustering and neural control based on supervised learning (learning with a teacher). Comparison of time response performance and length of execution time between PI, fuzzy and neural controller. Testing of the closed-loop control system based on parametric robustness analysis approach under uncertainty in robot dynamics.

In the next chapter using the non-linear model of the mobile robot derived in chapter four and the speed control law from this chapter including the algorithmic methodologies for identification, modelling and control of dynamic system, a hybrid multi-agent type control architecture is developed for navigation of multiple autonomous mobile robots in unknown environment based on implicit communication.

5.10 References

- ASTROM, K. J. and HAGGLUND, T. 1995. *PID Controllers: Theory, Design and Tuning*. 2nd end. USA: Instrument Society of America. 1-55617-516-7.
- BARMISH, B. R. 1994. *New Tools for Robustness of Linear Systems*. New York: Macmillan Publishing Company. 0-02-306055-7.
- BHATTACHARYYA, S. P., CHAPPELLAT, H. and KEEL, L. H. 1995. *Robust Control: The Parametric Approach*. USA: Prentice-Hall Inc. 0-13-781576-X.
- BURNHAM, K. J., DUNOYER, A. and MARCROFT, S. 1999. Bilinear Controller with PID Structure. *IEE Journal of Computing and Control Engineering*, **10** (2), pp. 63-69.
- FLETCHER, R. 1987. *Practical Methods for Optimisation*. 2nd end. UK: John Wiley & Sons Ltd. 0-471-91547-5.
- GAJIC, Z. and LELIC, M. 1996. *Modern Control Systems Engineering*. UK: Prentice Hall Europe. 0-13-134116-2.
- GILL, P. E., MURRAY, W. and WRIGHT, M. H. 1991. *Numerical Linear Algebra and Optimisation*. Addison Wesley.
- HINES, J. W. 1997. *Fuzzy and Neural Approaches in Engineering*. USA: John Wiley & Sons, Inc. 0-471-19247-3.
- SKOGESTAD, S. and POSTLETHWAITE, I. 1996. *Multivariable Feedback Control. Analysis and Design*. United Kingdom: John Wiley & Sons Ltd. 0-471-94277-4.

6

Hybrid Control Architecture for Navigation of Multiple Autonomous Mobile Robots

6.1 Introduction

This chapter presents the development of a novel hybrid multi-agent based control architecture called CAROS (Co-operative Autonomous RObotic Systems) for navigation of multiple autonomous mobile robots in unknown static and/or dynamic environment. The proposed architecture takes the advantages of various control structure types thereby integrating them in a way that results in an overall increase in synergy.

The remainder of this chapter is organised as follows: Section 6.2 discusses the need for hybrid control architectures for autonomous mobile robots in order to form a more robust, flexible and modular control system. An overview of the proposed hybrid multi-agent type control architecture for navigation of multiple autonomous mobile robots is presented in section

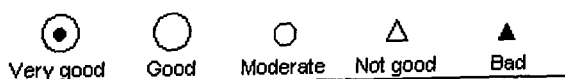
6.3. Section 6.4 presents the control strategy chosen for the navigation of mobile robots in an unknown environment. Sections 6.4.1 and 6.4.2 illustrates the control strategy decomposition into global and local strategy. In section 6.5 the design and implementation methodology of the control architecture is given. The low-level control unit of the control architecture is presented in section 6.6. Sensor modelling and sensor sensitivity is described in section 6.7. In particular in this section three types of sensor have been modelled based on their sensitivity characteristics. Robot localisation during the navigation tasks is discussed in section 6.8. Section 6.9 presents the analysis of the speed and orientation adaptive mechanism (SOAM) used in CAROS. In particular section 6.10 describes how the SOAM is organised using local controllers by means of agents. The co-ordination method for these controllers-agents is presented in section 6.10.1. Section 6.11 presents an algorithmic methodology for modelling the robot's behaviours. In this section is also shown that the implementation of complex control systems can be overcome by decomposing the global task into simpler well-defined behaviours. Sections 6.12, 6.13, 6.14 and 6.15 describe the implementation of four well-defined behaviours for the navigation of multiple robots in an unknown environment. In particular static and dynamic obstacle avoidance is considered. The proposed algorithmic methodology uses both fuzzy logic and neural networks. Hybrid solutions are also proposed for a novel approach of identification of moving direction of neighbour robots. A proposed method for co-ordination and parallel switching of the modelled behaviours is described in section 6.16. A discussion follows in section 6.17, and the summary including the main contributions of this chapter is presented in section 6.18.

6.2 Why hybrid architecture?

In chapter two, section 2.5 it was shown that the behaviour-based approach in robotic control could effectively produce robust performance in complex and dynamic domains. However, strong assumptions that purely reactive systems make can serve as a disadvantage at times. These assumptions according to (Arkin, 1998) may include environmental lack of temporary

consistency and stability, insufficient symbolic representation of world knowledge and finally difficult to localise the robot in respect to a world model. However, in some environments these assumptions may not be completely valid. For instance, if an accurate world model exists then perhaps a deliberative control architecture (method) may be preferred. Therefore it can be concluded that hybrid systems capable for incorporating both deliberative, reactive and may be other systems have the potential to deliver a satisfactory solution for flexible and robust control systems architecture. In chapter three section 3.5.2 a classification and some discussion of control architectures was presented. In addition section 3.5.4 (chapter three) described the main requirements/properties of control architecture. However the identification of what are the “best” characteristics of each control approach in terms of requirements/properties is still a matter of interest. In Figure 6-1 an analysis is presented based on the Quality Functional Deployment tool (QFD)¹ (Bossert, 1991) for the identification of the relationship of the requirements/properties of control architecture versus the control architecture specifications.

<div>Requirements/ Properties</div> <div>Control architecture specifications</div>	Modularity	Robustness	Fault tolerance	Distribution	Reactivity	Adaptability	Planning	Co-operation	Easy of application	Uncertainty	Optimal control	Goal oriented	Learning	Efficiency
Deliberative systems	▲	△	△	△	▲	○	●	▲	▲	▲	●	●	●	△
Reactive systems	●	●	○	○	●	○	△	●	●	●	▲	○	○	●
Hierarchical structure	△	△	△	▲	△	○	●	△	△	△	●	●	●	△
Behaviour-based structure	●	●	○	●	●	○	△	●	●	●	△	△	△	●
Centralised control	○	△	▲	▲	△	△	○	△	△	△	●	○	○	△
Distributed control/Multi-agent	○	●	●	●	●	○	△	●	●	●	△	△	△	●
Sensor fusion	△	▲	▲	△	△	▲	○	▲	△	△	○	○	▲	▲
Behaviour selection	●	●	○	●	●	○	▲	●	○	●	○	▲	▲	●



Very good Good Moderate Not good Bad

Satisfaction levels

Figure 6-1 QFD analysis of control architecture specifications versus requirements/properties

¹ Figure 6-1 sometimes called L Shaped Matrix Diagram. In the L Shape, two interrelated groups of items are presented in line and row format. The choice of satisfaction levels is subjective (Bossert, 1991).

The results from Figure 6-1 clearly indicate that the most promising solution in control systems architecture design is the hybrid approach. Therefore a deliberative upper module is needed to make high-level navigational planning and decision-making, whereas a low-level reactive module is needed to take care of the real-time issues related to the interactions with the environment. A schematic layout of this approach (hybrid) is shown in Figure 6-2. The middle module interacts between the deliberative and reactive module supervising the accomplishment of the tasks.

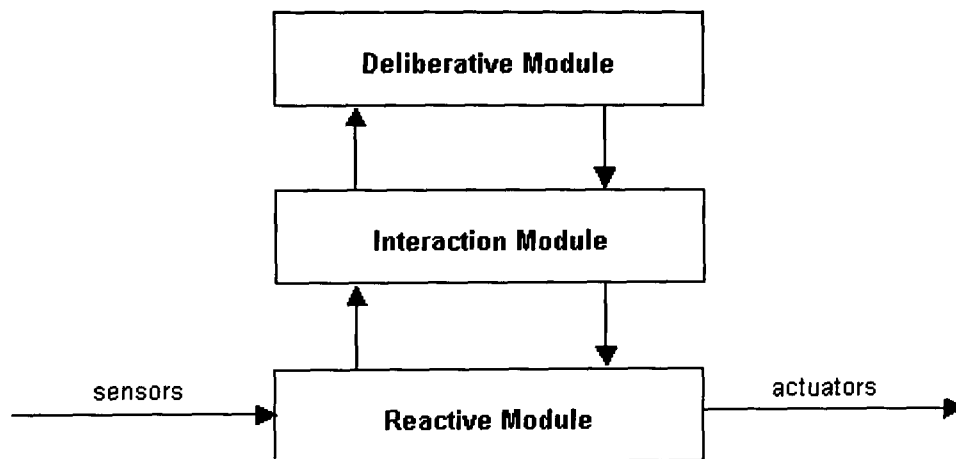


Figure 6-2 Hybrid approach

6.3 Overview of CAROS

In this section a general overview of the CAROS hybrid control architecture for navigation of autonomous mobile robots is presented. The hybrid control architecture draws its design from competitive tasks architecture, production rules architecture, connectionist architecture, dynamic system architecture, multi-agent architecture and subsumption architecture. Figure 6-3 shows the overview of the control architecture where it can be seen that the control architecture consists of the following parts: Sensory data, action co-ordinator mechanism, a set of local controllers modelling the robots behaviours, speed and orientation adaptive mechanism comprising of a set of local controller-agents, the robot model and its control mechanism and finally the robot working environment. The main characteristics of the proposed control architecture are summarised in Table 6-1.

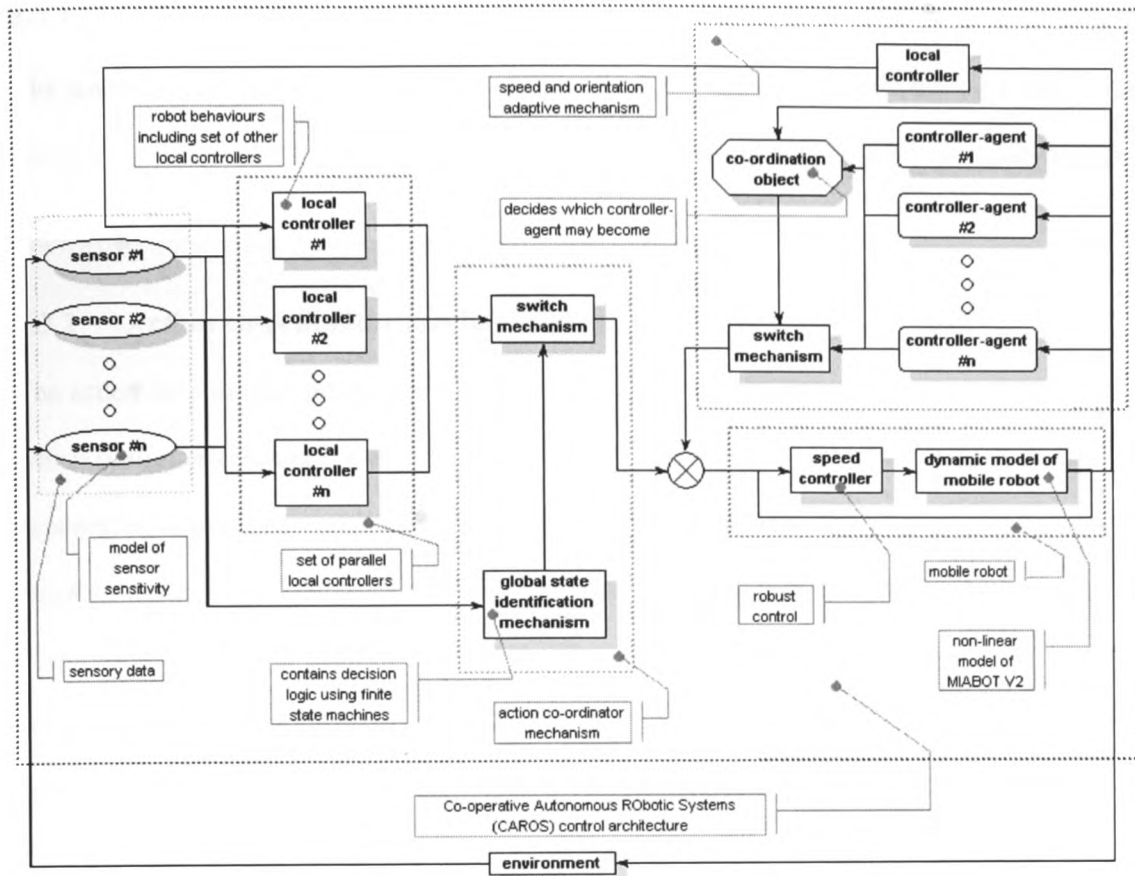


Figure 6-3 Structure of CAROS

Property	Value
Name	Co-operative Autonomous Robotic Systems (CAROS)
Developer	Alexandros Mouzakitis
Type	Hybrid Multi-Agent Control Architecture
Design Method	Experimental Simulation
Behavioural Encoding	Discrete. Fuzzy Logic, Neural Networks and Stateflow. Compatible with MATLAB Toolboxes
Approach	Object/Vertical Functional
Type of Component	Agent/Stimuli-Command Relationship/Tasks and Primitive Actions /Rules/Formal Neuron/FSM
Co-ordination Method	Egalitarian/Hybrid Hierarchical Competition
Coupling Structure	Variable
Constitution	Emergent/Predefined
Programming Method	MATLAB/Simulink and C
Robots Fielded	MIABOT V2

Table 6-1 Main characteristics of CAROS

6.3.1 Sensory data

The sensory data block is used to model sensitivity of three types of sensors for the robot's local navigation. The three types of sensors used were ultrasonic, infrared and no-shape sensors, which are all presented in detail in section 6.7.

6.3.2 Action co-ordinator mechanism (ACM)

The action co-ordinator mechanism is used to identify the robot/system states and then decide which controller/behaviour should be activated. The number of required states is not unique but depends on a particular application and designer choice. The design tool used to model the robot's states is stateflow based on theory of finite state machines (FSM). In chapter three an overview of the design methodology of stateflow and FSM was presented. The output of the control logic of the FSM decides which behaviour has to be activated according to the current state of the system. Note that action co-ordinator mechanism and robot behaviours share the same type and number of inputs coming from the sensory data block. A non-linear switch mechanism then is in charge to release the control output commanded by the global state identification mechanism (GSIM). This kind of action co-ordination draws its advantages from the competitive tasks architecture in which once a functional module is selected the module is activated and the previous one deactivated. However the deactivated module is not totally switched off, it continues to receive sensory data in parallel with FSM. Thus, the module it exercises a monitoring activity, which consists in receiving data from sensors and calculating the values of the selection parameters, which depend on it. The ACM is presented in more detail in section 6.16

6.3.3 Decision-making mechanism (DMM)

The decision-making mechanism is the largest part of the proposed control architecture. Only four independent local controllers were employed in this thesis to model the DMM (note that the number of local controllers is dependent upon the particular application). Each local controller (behaviour) runs completely independently, is responsible for a predefined task and

provides the action co-ordinator mechanism with control commands according to its own rate and time constraints. The main design philosophy behind the decision-making mechanism is based on behaviour-based approach. The classical artificial intelligent approach to robot control is to divide the problem in a task-based style. As mentioned previously in chapter two, the major problem with this approach was that all subsystems must work well in order for the robot to function at all. The behaviour-based approach came to solve this problem. With the new intelligent control approach several behaviours are used to control the robot. For this work, each of the behaviour (functional module-local controllers) is designed to perform a particular task such as to orientate and approach the goal, to avoid neighbour robots and also collisions with static obstacles. For instance, one of the major differences between this work and the subsumption architecture is that the coupling structure is not fixed. The variable coupling structure of the CAROS architecture allows more flexibility and modularity into the control system.

The design of the functional modules is not unique. The most popular behaviour design is the one by (Brooks, 1986), although other researchers have developed behaviour on more or less the same principle (Arkin, 1989) and (Mataric, 1992). For this work the principle of the production systems architecture, FSM and connectionists architecture is proposed for the design of the major functional modules. Fuzzy logic is one form of production systems architecture. This form of control architecture was shown in chapter three that consists of the perception, the database with the inference engine and the rule base, and finally the execution. In terms of fuzzy logic the functional module consists of the fuzzification mechanism, the inference engine and the defuzzification mechanism, which produces the crisp output as the control command. Neural networks, which are one form of connectionist architecture, also used to model standalone or hybrid behaviours in accordance with FSM (stateflow-neural approach). As mentioned in chapter three the use of fuzzy logic, neural networks and FSM provide good tools for translating human knowledge into mathematical terms, easy to identify, model and apply learning

techniques for non-linear systems and finally provides model visualisation and control of complex reactive control systems. The DMM is presented in more detail in section 6.11

6.3.4 Speed and orientation adaptive mechanism (SOAM)

The speed and orientation adaptive mechanism, which forms another part of the control architecture, is in charge of two tasks. The first task is to define the local operating speed of the mobile robot (local operating speed depends on control strategy) whereas the latter task is to provide feedback for one of the local controllers in the decision-making mechanism responsible to correct the mobile robot orientation towards the target. This adaptive mechanism consists of a set of local controller-agents, one local controller and a co-ordination object. According to (Van Breemen and De Vries, 2000) a controller-agent “is a largely autonomous locally operating controller that consists of a control algorithm (in the form of an update and a calculate function), an operating regime characterisation, initialisation and finalisation functions and an interface to co-ordinate its behaviour in order to handle dependencies among controller-agents”. The Co-ordination object solves dependencies between controller-agents based on their interactions. As can be observed from Figure 6-3 the co-ordination object has as inputs the inputs and outputs of the controller-agents. The SOAM is presented in more detail in section 6.9

6.3.5 Mobile robot and environment

The last two parts of the control architecture comprise the mobile robot and the robot's environment. The mobile robot block consists of the full non-linear model of the MIABOT V2 mobile robot described in chapter four. Three types of controllers developed in chapter five, for reliable speed control can be selected to control the robot's velocity. Realistic model and robust speed control of the mobile robot are required for the implementation of the hybrid control architecture (note that most control architecture assume kinematic model of the mobile robot in which the robot's dynamics and speed remains unconstrained). The environment simulation block is used in order to simulate static obstacles and sensor readings (see sections 6.7 and 6.13).

6.4 Control strategy

In this section an introduction based on control strategy chosen for the navigation of the mobile robots is presented. The control strategy has been decomposed into two sub-control strategies (global and local control strategies) as shown in Figure 6-4 and furthermore presented in sections 6.4.1 and 6.4.2. Although the CAROS control architecture is based on the control strategy presented here, it still remains a generic architecture capable of solving other complex control problems. For instance, the control architecture may be used for navigation of other types of vehicles such as a different type of wheeled mobile robot, for ship motion control, for aircraft control etc., The control architecture can also be applicable in other applications which may involve problems of control and decision making for a given process.

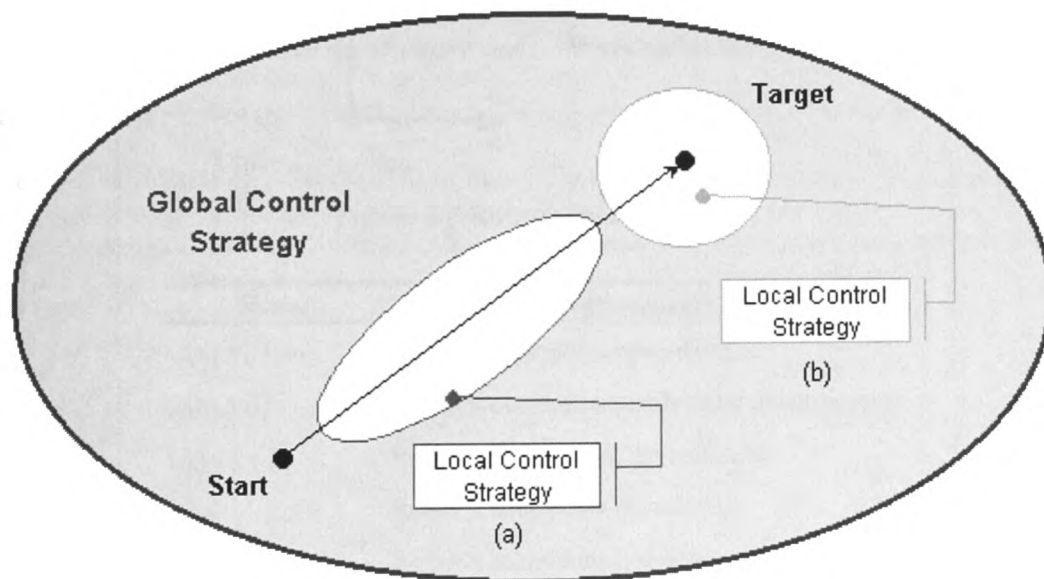


Figure 6-4 Control strategy for robot(s) navigation

6.4.1 Global control strategy

The global control strategy of the mobile robot(s) during the navigation cycle is defined as shown in Figure 6-5. The co-ordinates and the orientation of the robot at time t are x_S , y_S and θ_S (note that subscript S stands for “sensed”). The start and target points of the robot are

(x_0, y_0) and (x_T, y_T) respectively. The orientation at start and target points of the robot is defined as ϑ_0 and ϑ_T respectively in which the orientation angle is measured anticlockwise from the x axis (see chapter four, section 4.3). The robot's velocity at start and target points is defined as V_0 and V_T respectively. The robot desired velocity during the mission is defined by V_D . If there were no obstacles (either static or dynamic) during the navigation, the robot would instantly turn towards its target from the start. Table 6-2 summarises the variables used for the robot's co-ordinates, orientation and velocity during navigation.

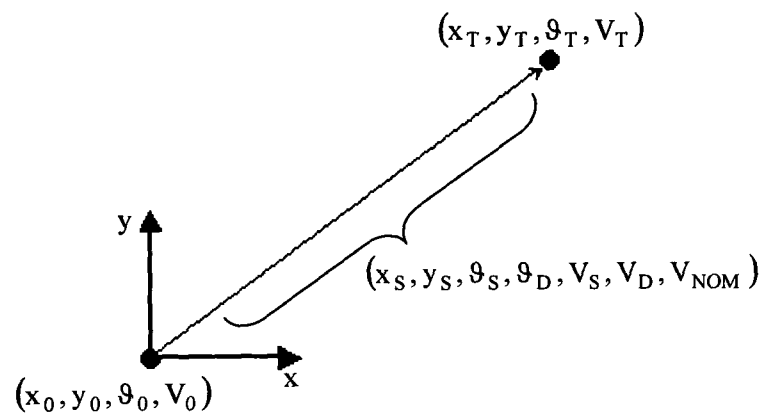


Figure 6-5 Robot(s) global strategy

Name	Representation
(x_0, y_0)	Robot's start point co-ordinates
(x_S, y_S)	Robot's current (sensed) point co-ordinates
(x_T, y_T)	Robot's target point co-ordinates
ϑ_0	Robot's start point orientation
ϑ_S	Robot's current orientation
ϑ_D	Robot's desired orientation
ϑ_T	Robot's target point orientation
V_0	Robot's start point linear velocity
V_S	Robot's current linear velocity
V_D	Robot's desired linear velocity
V_{NOM}	Robot's nominal linear velocity
V_T	Robot's target point linear velocity

Table 6-2 Variables used for robot navigation

6.4.2 Local control strategy

The robot(s) local control strategy is divided into two phases described as follows:

Phase A (local control strategy (a) as shown in Figure 6-4) is the case in which the mobile robot detects and moves towards the target, but obstacles either as static or dynamic obstructing its path i.e. the main goal during Phase A is to approach close to the target, avoiding possible obstacles on the way. Therefore the robot will move around the obstacle if it is static or it will use traffic rules if it is dynamic. If at the same time both static and dynamic obstacles have been found to obstruct the robot's path, then priority is given to the closest obstacle. Section 6.11 presents in detail the analysis of the proposed method of both static and dynamic obstacle avoidance.

Phase B begins when robot enters a predefined region (circle) around the target. At the end of the mission the robot has to reach the target with predefined angle and speed. To achieve this, the local control strategy from Phase A is switched off. The main idea is to generate a virtual trajectory from the entry point to the target, which satisfies boundary conditions specified with the approaching angle and speed. Virtual trajectory is specified by two virtual points. Positions of the virtual points are calculated dynamically, depending on the entry point co-ordinates and approach requirements. Figure 6-6 shows a schematic layout of Phase B.

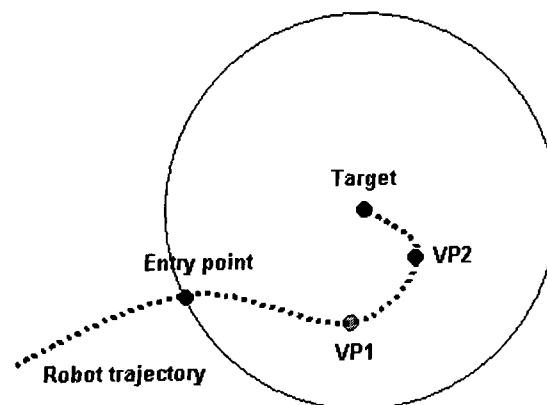


Figure 6-6 Phase B of local control strategy

6.5 Design and implementation methodology

Once the main principles and structure of the control architecture is defined, the next step is the implementation of the control architecture. The implementation of CAROS took place on a MATLAB/Simulink based simulator. The control architecture was tested throughout a large number of experiments until the simulation reached the desired results. The main design methodology was based on iterative cycle of simulations, analysis of the results, leading to further improvement of the control architecture towards its final configuration. In the following sections the design and implementation of CAROS control architecture is presented in more detail.

6.6 Low-level control

The simulation of the control architecture relies on the ability to realistically simulate a mobile robot (MIABOT V2, or any other robot) within its working environment. Hence, the modelling and simulation of the vehicle was considered the first problem to face. The dynamics, which are certainly important for a mobile robot including its local control laws, have been taken into account. The dynamic model of the mobile robot derived in chapter four and its speed control laws (PI, fuzzy and neural controller) derived in chapter five comprise the low-level control unit in CAROS control architecture. Parameters of the low-level control unit such as initial values and/or initial vector for the mobile robot and/or selection of speed controller are set using the dialog boxes shown in Figure 6-7.

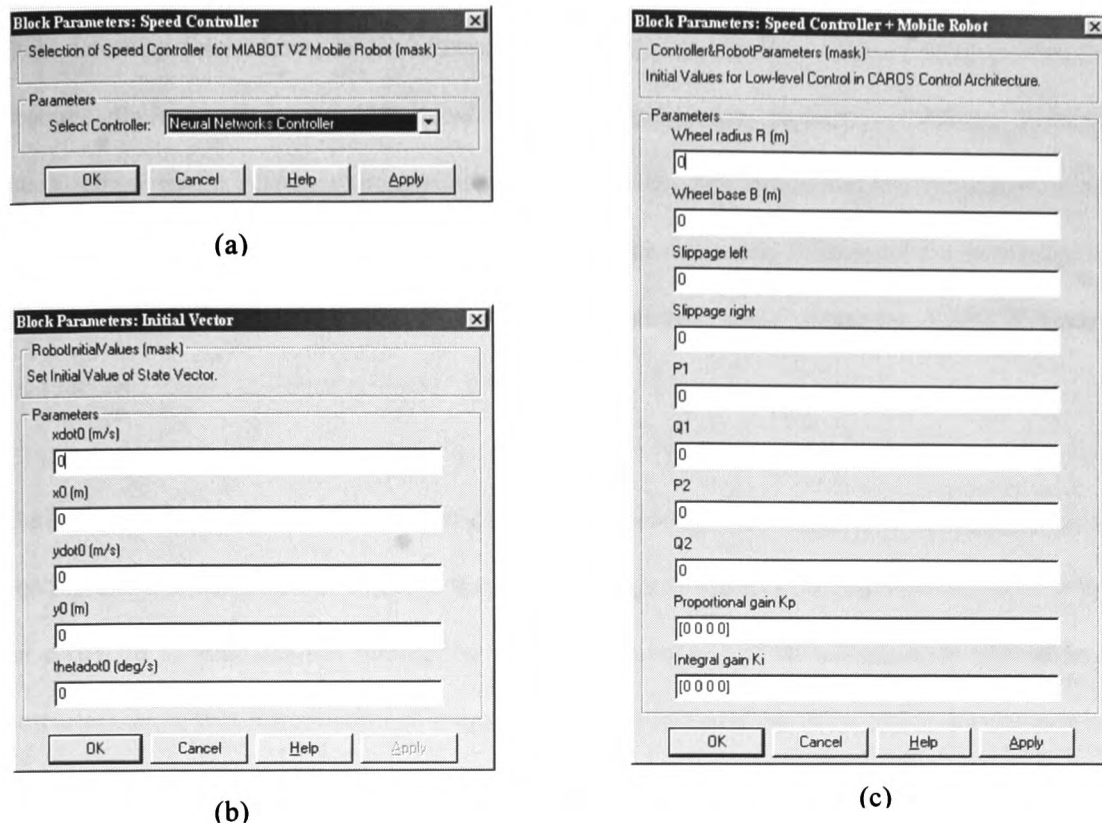


Figure 6-7 Dialog boxes used in low-level control unit. (a) Selection of speed controller (PI, fuzzy or neural) (b) Initial value of state vector for the robot(s) (c) Initial value for robot dynamics and speed controller gains

6.7 Modelling sensors and sensor sensitivity

Truly autonomous control implies the ability of a free-roaming robot to travel anywhere so desired, subject to nominal considerations of terrain traversability. Furthermore at the robot, task and environment are tightly linked, the overall behaviour of the robot is the result of the interaction of these three components.

Sensors are devices that can sense and measure physical properties of the environment, such as distance, temperature, size, weight, etc. They deliver low-level information about the environment the robot is working in. According to (Everett, 1995) sensors are divided into two categories, the navigational referencing sensors and the collision avoidance sensors. Collision avoidance sensors usually operate over short ranges incorporating low resolution. The field of view should provide sufficient coverage for a turning robot, and allow enough time to stop or

change course. (Nehmzow, 1999) has characterised all sensors by a number of properties that describe their capabilities. The most important properties include sensitivity, linearity, measurement range, response time, accuracy, repeatability, resolution and finally type of output. In this section the modelling of collision avoidance type sensors is considered for the navigation of the MIABOT V2 mobile robot(s) in the simulation trials using the CAROS control architecture.

During the simulation trials three type of sensors were used: infrared, ultrasonic (or modular) and no-shape sensors. It was decided that the use of three sensors on each mobile robot would be sufficient to simulate the interaction between the robot and its environment (detection of obstacles). Note that introduction of more sensors may increase the robot ability to navigate but the computational complexity of the simulation increases dramatically (too slow to run). All sensors are set in the front face of the robot. A support was created in order to give two rotation axes for each sensor and so there is the possibility to choose different horizontal and vertical positions. The following schema and picture in Figure 6-8 shows the arrangement of the sensors on the mobile robot.

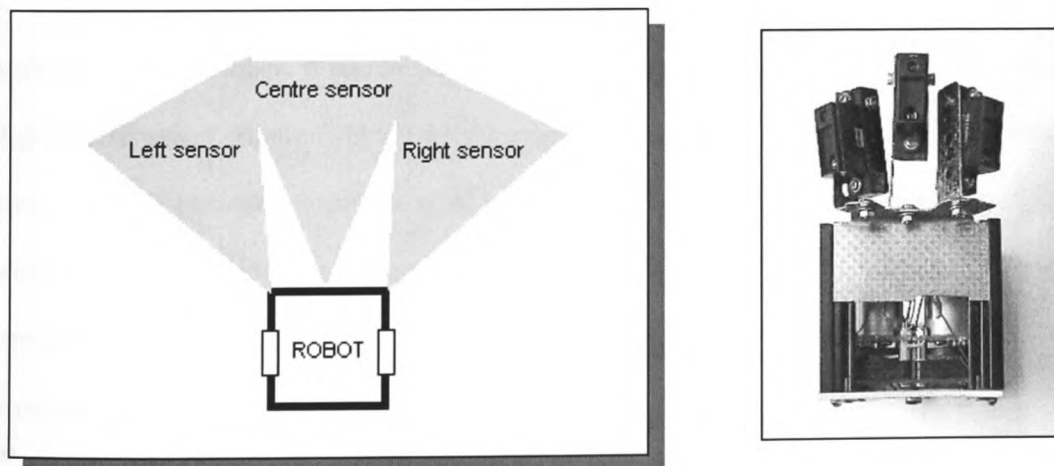


Figure 6-8 Schematic layout of sensor arrangement.

6.7.1 Infrared sensors

The infrared sensor (Sharp GP2D02) was the first type of sensor to be modelled as they have already been used with the MIABOT V2 mobile robot. Infrared sensors are probably the simplest type of non-contact sensors and they are widely used in mobile robots to detect obstacles. They are safe, inexpensive and fairly easy to use. Provided all objects in the robot's environment have a uniform colour and surface structure, infrared sensors can be calibrated to measure distance from the objects. However, in realistic scenarios surfaces of objects have different colours, which results a larger or smaller amount of light. To overcome this problem in mobile robot navigation it is usually best to start avoiding the object as soon as it is detected. Figure 6-10 shows the modelling of infrared sensor sensitivity based on real experiments with the mobile robot. It can be seen that the sensor sensitivity it varies between 0 and 1 according to the distance of the obstacle detected up to the range of 0.8m .

6.7.2 Ultrasonic sensors

Ultrasonic sensors are the second type of sensor sensitivity modelled. Ultrasonic sensors are also safe, low-cost and easy to use. The main design philosophy behind ultrasonic sensors is the same that is used by bats². The sensitivity of this type of sensor is not uniform, but consists of a main lobe and side lobes. It can be used to obtain distances up to 10m through direct time-of-flight measurement. Figure 6-11 shows the modelling of ultrasonic sensor sensitivity based on given 0.7 value of slope. Slope is a value between 0 and 1, which can be selected to change the sensor sensitivity in terms of sharpness (note that zero value of slope will give an no-shape sensor presented in the next paragraph). It can be seen that the sensor sensitivity it varies between 0 and 1 according to the distance of the obstacle detected up to the range of 0.8m .

² A chirp, that is a short (e.g. 1.2 ms), powerful pulse of a range of frequencies, is emitted, and its reflection of objects ahead of the sensor is detected by a receiver (Nehmzow, 1999).

6.7.3 No-shape sensors (infrared)

This is a simple sensor unable to distinguish distance. With this type of sensor if an object is detected the sensor sensitivity is 1 and 0 otherwise. Therefore this sensor does not consider distance to the object but only information if it is present. Figure 6-10 shows the modelling of no-shape sensor sensitivity. It can be seen that the sensor sensitivity is either 0 or 1 according to the appearance of the obstacle detected up to the range of 0.8m .

As can be seen from Figure 6-9 all sensors can be monitored during the robot(s) navigation. In addition, sensor modelling is generic and modular. This means that at any time several sensor parameters can be adjusted such as sensor position, range, sensitivity and type of sensor to be used.

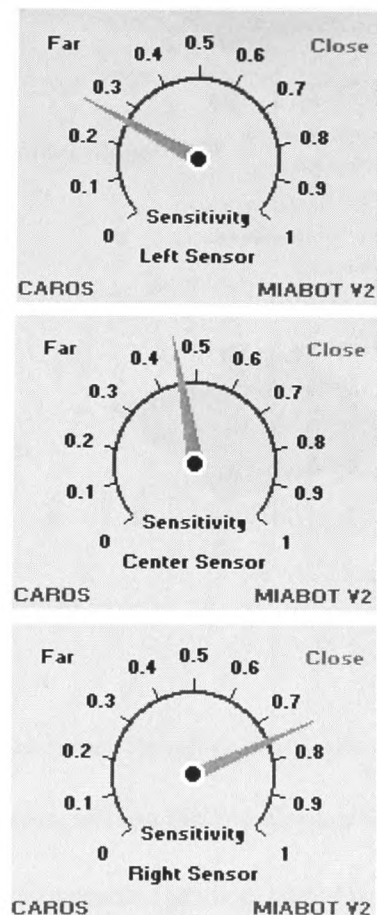
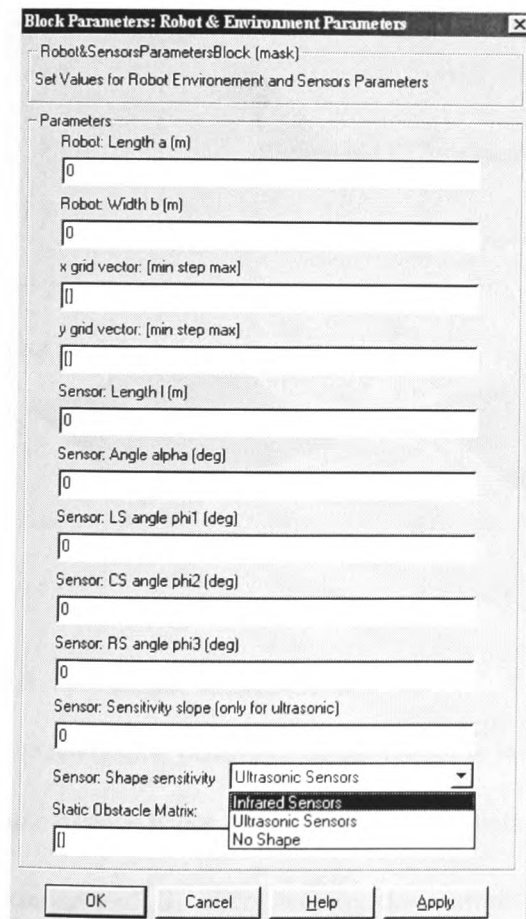


Figure 6-9 Dialog box for environment/sensor parameters and gauges used to read sensor sensitivity

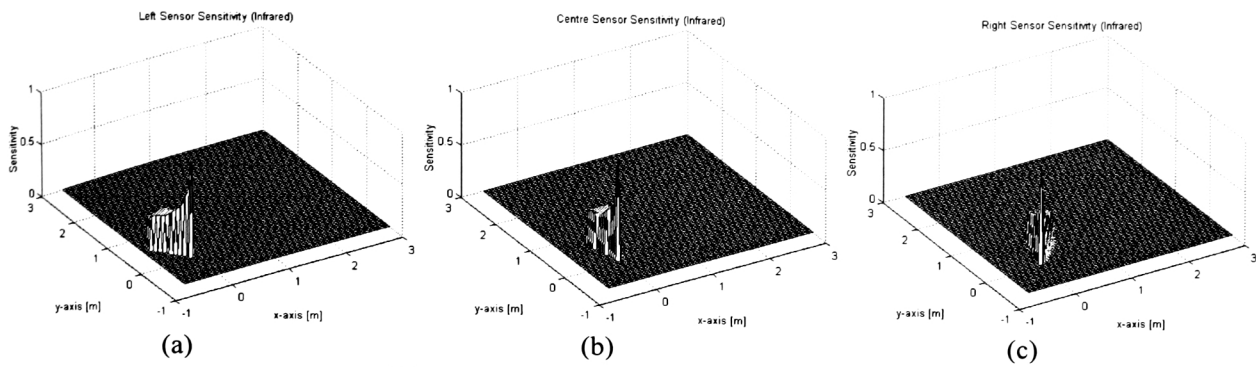


Figure 6-10 Infrared sensor sensitivity

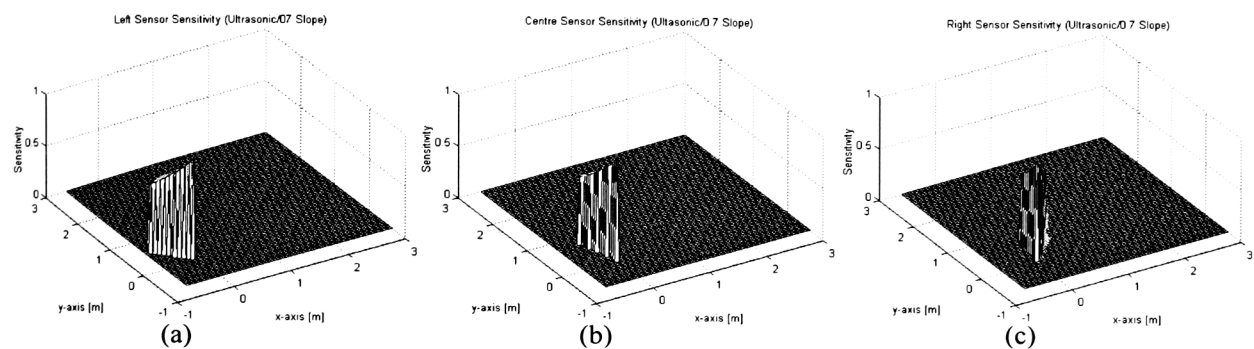


Figure 6-11 Ultrasonic sensor sensitivity (0.7 Slope)

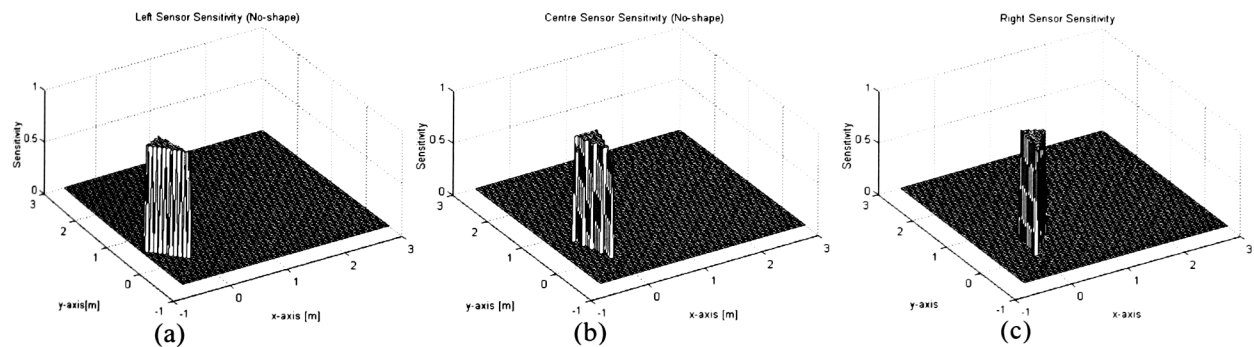


Figure 6-12 No-shape sensor sensitivity

6.8 Robot localisation

The navigation behaviour of the robots is in a two-dimensional plane. Therefore, the positional state of each robot is given by a co-ordinate point (x_i, y_i) characterising the two-dimensional position and θ_i characterising the current direction of the movement (heading direction or orientation). The process of accurately determining the position and orientation of a moving

robot with respect to a reference frame is a fundamental requirement for autonomous navigation. Different kinds of techniques have been developed to tackle the problem of robot localisation and these techniques according to (Roumeliotis and Bekey, 1997) can be sorted into two main categories: *Relative (local) localisation*, which consists of evaluating the position and the orientation through integration of information provided by diverse sensors. The integration is started from the initial position and its continuously updated in time. *Absolute (global) localisation*, which is the technique, that permits the robot to find its way directly in the domain of evaluation of the system. These methods usually rely on navigation beacons, active or passive landmarks, map matching or satellite-based signals i.e. global positioning system (GPS). (Canudas De Witt and Sordalen, 1992), (Hou and Muller, 1992), (Eren and Fung, 1997), (Astolfi, 1999), (Jetto *et al*, 1999), (Mutambara and Durrant-Whyte, 2000), (Sasiadek and Hartana, 2000) and (Conticelli *et al*, 2000) have demonstrated methods of how to estimate both position and orientation of a mobile robot with navigational behaviour in two-dimensional space. Some of their methods include the use of fuzzy logic adapted Kalman filter, extended Kalman filter and design of non-linear observers. The problem of robot localisation is not the scope of this thesis, therefore the full estimate of robot's position and orientation ($\hat{x}_i, \hat{y}_i, \hat{\theta}_i$) it is assumed.

6.9 Analysis of SOAM

In this section, an analysis of speed and orientation adaptive mechanism (SOAM) based on organisation of local controllers by means of agents is described. In addition also described the method of co-ordinating these controllers-agents using a co-ordination object based on FSM.

6.10 Organising SOAM using local controllers by means of agents

The main objective of the SOAM is to provide the mobile robot with the desired speed and orientation according to current local control strategy (i.e. Phase A or B). The local controllers used in the SOAM are organised by means of agents. From the control point of view, systems

are analysed by focusing on models of the dynamic behaviour of variables of the plant. One of the major concepts of control theory is to analyse robustness, stability and other performance properties. As mentioned in chapter three, the field of multi-agent systems is concerned with analysing and designing systems in terms of agents. The uses of multi-agent systems provides an answer to the question of how to deal with distributed systems with focus on goals and organisations, an area which is almost unexplored in the control community. However, it is well known that a controller is usually designed to operate contentiously the whole operating time of the control system. A controller that only operates in some restricted part of the operating regime of the controller system comes close to the concept of agent. SOAM is multi-agent constructed and orientated as the majority of its operation belongs to a restricted part of the operating regime of the control architecture (only operates when the robot enters Phase B). The capability of the mobile robot to reach the target with predefined speed and orientation when it enters Phase B is achieved through the use of local controllers organised by means of agents as summarised and shown in Table 6-3.

Name	Controller-agent Function Representation	Agent Goal
find_VP1	$([g_T], [x, y]_T, [X, Y]_{entry}, [\epsilon], [f], [g]) \rightarrow [x, y]_{VP1}$	Calculate co-ordinates of first virtual point
find_VP2	$([g_T], [RVP2], [x, y]_T) \rightarrow [x, y]_{VP2}$	Calculate co-ordinates of second virtual point
approach_target	$([x, y]_T, [x, y]_S) \rightarrow [g_D^1]$	Orientate towards the target
approach_VP1	$([x, y]_T, [x, y]_S, [x, y]_{VP1}) \rightarrow [g_D^2]$	Orientate towards the first virtual point
approach_VP2	$([x, y]_T, [x, y]_S, [x, y]_{VP2}) \rightarrow [g_D^3]$	Orientate towards the second virtual point
get_target	$[g_T] \rightarrow [g_D^4]$	Reach the goal with predefined angle
adjust_speed	$([x, y]_T, [x, y]_S, [V_T], [V_{NOM}]) \rightarrow [V_D]$	Reach the goal with predefined speed

Table 6-3 Controller-agents used in CAROS and SOAM for robot navigation

At the beginning of the navigation the robot(s) target location, predefined target approach angle, nominal robot speed and other initial values can be set using the dialog boxes shown in Figure 6-13.

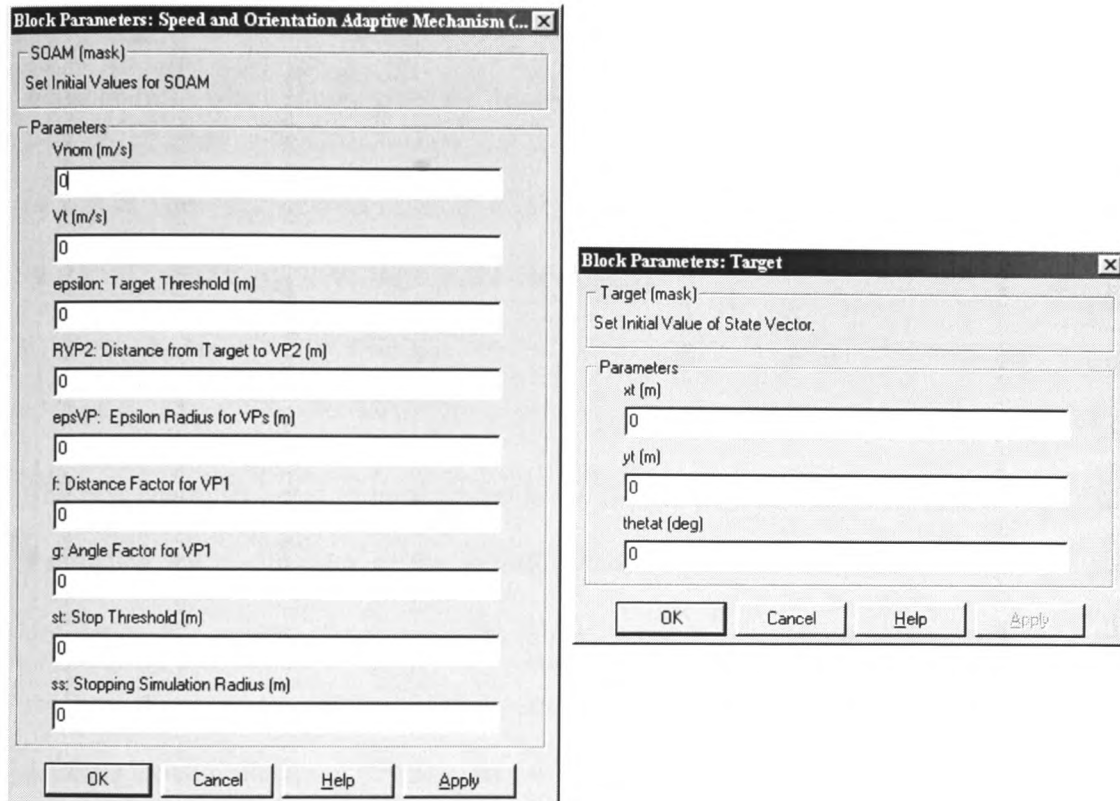


Figure 6-13 Dialog boxes used to set initial values in robot(s) navigation

In order to explain the co-ordinating work undertaken by the controller-agents when the robot enters Phase B the following example is considered. The mobile robot is initially located at point (x_0, y_0) and its goal is to reach the target (x_T, y_T) with predefined angle ϑ_T and predefined speed V_T . As shown in Figure 6-14 the robot enters at point (x_e, y_e) the predefined region (circle) denoted by the radius ϵ taken from the target point. At this stage the SOAM is activated and therefore the scope of the set of local controllers-agents is to find two virtual

points and also to provide the robot with its desired speed and orientation towards the virtual points and the target. The task of the *adjust_speed* controller-agent is to continuously calculate the desired robot's speed until the robot reaches the target. The inputs to the *adjust_speed* controller-agent are nominal robot speed (V_{NOM}), robot's linear velocity at the target point (V_T), distance from the robot's current position to the target (d), target threshold (ϵ) and the threshold to stop at the target (st). The output of *adjust_speed* controller-agent is given in Equation (6.1), which simply gives the desired robot speed denoted by V_D .

$$V_D = \begin{cases} V_{NOM} & \text{if } d > \epsilon \\ \frac{(V_{NOM} - V_T) \cdot d + V_T \cdot \epsilon - st \cdot V_{NOM}}{\epsilon - V_T} & \text{if } d > st \text{ AND } d < \epsilon \\ V_T & \text{if } d \leq st \end{cases} \quad (6.1)$$

The next controller-agent to be activated is the *find_VP2*. This controller-agent is responsible for finding the co-ordinates of the second virtual point. The inputs are robot's target point orientation ϑ_T , target point co-ordinates (x_T, y_T) and the predefined distance of virtual point two from the target denoted by RVP2 (see dialog box in Figure 6-13). Equations (6.2) and (6.3) calculate the co-ordinates of virtual point two as shown in Figure 6-15.

$$x_{VP2} = x_T - (RVP2 \cdot \cos(\vartheta_T \cdot \pi/180)) \quad (6.2)$$

$$y_{VP2} = y_T - (RVP2 \cdot \sin(\vartheta_T \cdot \pi/180)) \quad (6.3)$$

When virtual point two has been found the *find_VP1* controller-agent is activated. This controller-agent is responsible for finding the co-ordinates of the first virtual point. The inputs are robot's target point orientation ϑ_T , target point co-ordinates (x_T, y_T) , entry point co-ordinates (x_e, y_e) , target threshold (ϵ), distance factor for virtual point one (f) and angle

factor for virtual point one (g). Note that the distance and angle factor for virtual point one have been introduced in order to provide more flexibility and modularity into the proposed approach (for other applications distance and angle factors can be modified using the dialog box in Figure 6-13). At the beginning angle ψ can be calculated using the four-quadrant inverse tangent and the co-ordinates of entry and target points as shown in Equation (6.4).

$$\psi = \left(\tan^{-1} \frac{Y_T - Y_E}{X_T - X_E} \right) \cdot 180 / \pi \quad (6.4)$$

Then angle ξ is calculated based on angles ψ , θ_T and the distance and angle factors f and g as follows:

$$\xi = g \cdot \psi + (1 - g) \cdot \theta_T \quad (6.5)$$

Using the RVP2 distance from target for virtual point two, the distance from target to virtual point one RVP1 can be calculated as follows:

$$RVP1 = f \cdot RVP2 + (1 - f) \cdot \epsilon \quad (6.6)$$

Having found ξ and RVP1 the co-ordinates of virtual point one are given by Equations (6.7) and (6.8). Figure 6-16 shows the schematic layout of virtual point one calculation.

$$x_{VP1} = x_T - (RVP1 \cdot \cos(\xi \cdot \pi / 180)) \quad (6.7)$$

$$y_{VP1} = y_T - (RVP1 \cdot \sin(\xi \cdot \pi / 180)) \quad (6.8)$$

At this stage the virtual points VP1 and VP2 have been found and therefore the robot can follow the virtual trajectory generated. The problem, which occurs at Phase B, is that robot's desired orientation is changing dynamically according to which virtual point it is pass in. To solve the problem, four controller-agents are used. The first controller-agent, called *approach_target*, is used to generate the desired robot orientation from the start point (x_0, y_0) until the robot enters the predefined region ε . The output of *approach_target* controller-agent is given in Equation (6.9).

$$\theta_D^1 = \tan^{-1} \frac{y_T - y_S}{x_T - x_S} \cdot 180 / \pi \quad (6.9)$$

When the robot enters the predefined region ε another controller-agent, *approach_VP1*, is used to navigate the robot towards the VP1 virtual point. Equation (6.10) shows the output of the *approach_VP1* controller-agent.

$$\theta_D^2 = \tan^{-1} \frac{y_T - y_S - (RVP1 \cdot \sin(\xi \cdot \pi / 180))}{x_T - x_S - (RVP1 \cdot \cos(\xi \cdot \pi / 180))} \cdot 180 / \pi \quad (6.10)$$

When the robot reaches virtual point one, *approach_VP2* controller-agent is used to navigate the robot towards the virtual point two (VP2). The desired robot orientation towards the virtual point two is calculated as follows:

$$\theta_D^3 = \tan^{-1} \frac{y_T - y_S - (RVP2 \cdot \sin(\theta_T \cdot \pi / 180))}{x_T - x_S - (RVP2 \cdot \cos(\theta_T \cdot \pi / 180))} \cdot 180 / \pi \quad (6.11)$$

Finally when the robot reaches virtual point two, *get_target* controller-agent is used to navigate the robot towards the target point denoted by (x_T, y_T) . The output of this controller-agent is the same as the given predefined target angle as follows:

$$\theta_D^4 = \theta_T \quad (6.12)$$

In some cases the interface of a controller-agent is achieved by activation request and acknowledge signals. The activation request and acknowledge signals are used to co-ordinate the behaviour of the controller-agent with the other controller-agents of the overall controller. For instance in (Van Breemen, 2001) a controller-agent behaves just like a local controller either as being *active* or *inactive*. In this thesis the controller-agents are switched *on* and *off* using the enabled subsystem block provided by MATLAB. Using the enable subsystem block a subsystem (controller-agent) can be either enabled or disabled according to a specific control input (request). Although, a controller-agent does not execute while it is disabled, its output signal is still available to other controllers. While a controller-agent is disabled its output can be selected to hold its previous value or can be reset to its initial condition. The blocks of controller-agents, which have been modelled using the MATLAB/Simulink can be found in Appendix B.

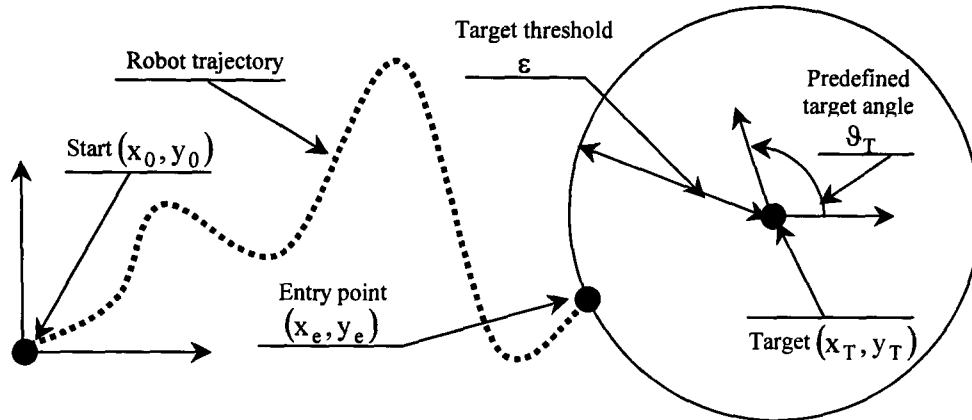


Figure 6-14 Schematic view of mobile robot navigation when enters Phase B

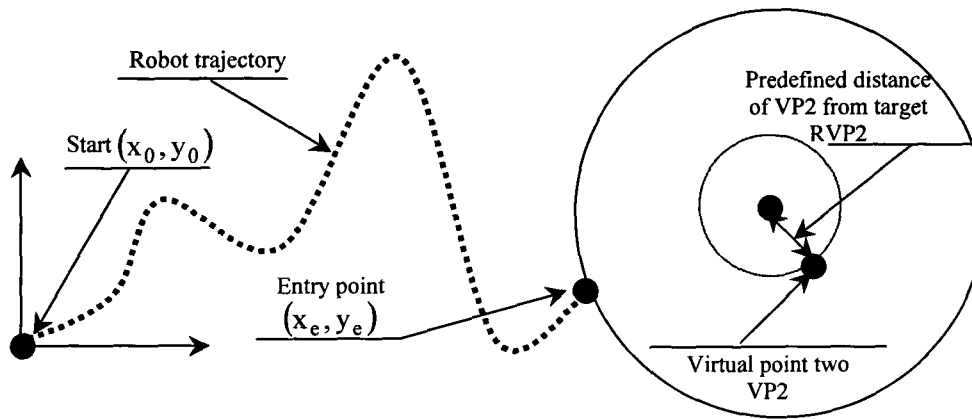


Figure 6-15 Schematic view of virtual point two calculation

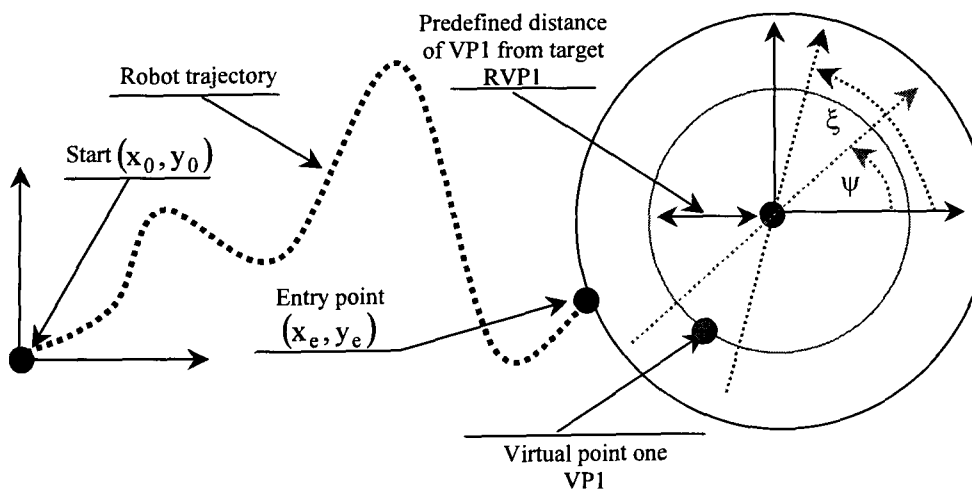


Figure 6-16 Schematic view of virtual point one calculation

6.10.1 Co-ordination of local controller-agents using stateflow

In section 6.10 the use of a controller-agent was proposed to implement a locally operating controllers. To co-ordinate several controller-agents, a mechanism is needed that activates/de-activates them. So, a co-ordination object that retains the decision taking part of the supervisor and hence still must solve problems such as conflicts (several local controllers might generate valid control signals at the same time), deadlock (at the same time none of the controllers might be relevant), bumpless transfer (there should be smooth transition between the outputs of the local controller) and shattering (ideally should be no situation of fast switching between the local controllers) is needed. In the field of multi-agent systems, the design of a co-ordination mechanism is handled as a separate issue. In the literature, a large number of different design methods for co-ordination objects can be found as well as several different types of co-ordination mechanisms (Malone and Crowston, 1994) and (Van Breemen, 2001). Among all the different types of co-ordination mechanisms such as parallel, master-slave, competitive and co-operative, supervisor-like co-ordination mechanism was selected as it combines several useful characteristics such as use of switch, timing diagram, FSM, flowchart and scheduler/planner (scheduler/planner determines the order in which the controller-agents should be active during operating time of the overall controller). The co-ordination object used here is the co-ordination mechanism of supervisor-type modelled with the use of FSM. The use of FSM was chosen as have been proven to work/operate superiorly in the field of hybrid systems modelling and analysis of multi-controller systems (Boel *et al*, 1999) and (Van Den Bosch and Heemels, 1999). In the following a description of the supervisor-like co-ordination object shown in Figure 6-17 is presented.

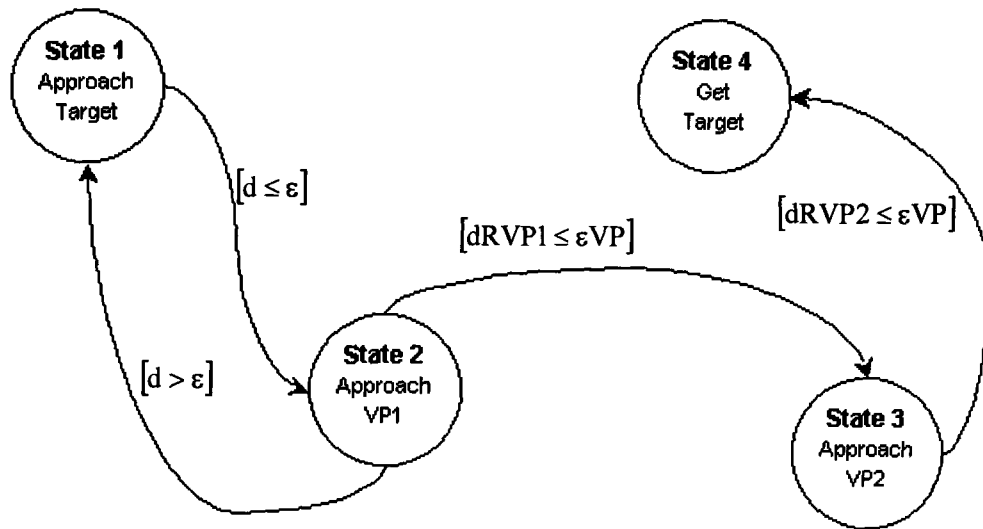


Figure 6-17 Supervisor-like co-ordination object.

Some definitions of FSM and brief background theory regarding stateflow were given in chapter three section 3.7.1. The supervisor-like co-ordination object shown in Figure 6-17 was modelled using a Moore finite state machine and it functions as follows: The task, when the robot enters the predefined region denoted by ϵ , and until it reaches the target, can be described as an ordered sequence of steps. A stateflow diagram (FSM) is then, designed to accomplish each step incorporating transitions between steps³. As can be observed from Figure 6-17 the task has been decomposed into four states. Each state is responsible for activating one of the controller-agents described in section 6.10. At the beginning of the co-ordination object execution the system will stay in state one if the robot has not reached the predefined region denoted by ϵ and therefore the *approach_target* controller-agent is activated (\mathcal{G}_D^1). The co-ordination object it will jump into state two at the point where the robot enters the predefined region ϵ . This means that the condition of the first transition connecting first and second states is satisfied ($d \leq \epsilon$) and hence controller-agent *approach_VP1* is activated (\mathcal{G}_D^2). However, at this state not only does the

³ Each step may be composed of sub-steps or subtasks also performed sequentially.

system activate controller-agent *approach_VP1* but also records the robot's entry point coordinates (x_e, y_e) (Note that (x_e, y_e) are required for the calculation of virtual point one). At this stage it is possible that the mobile robot will go outside the predefined region. Therefore a transition is used from state two to state one which can only be true if $d > \epsilon$. The system will jump into state three if the distance from the current robot position to the virtual point one (d_{RVP1}) is less than or equal to a radius of predefined circle around both the virtual points denoted by ϵ_{VP} (see the dialog box in Figure 6-13). Automatically when the co-ordination object jumps into state three the controller-agent *approach_VP2* is activated (g_D^3). State four will be active if the distance from the current robot position to the virtual point two (d_{RVP2}) is less than or equal to a radius of predefined circle around both the virtual points denoted by ϵ_{VP} . This is the final state of the co-ordination object at which the controller-agent *get_target* is activated (g_D^4). Note that there are not any transitions from state four to state three, and from state three to state two as the distance from the current robot location to the target is very small and therefore no obstacles are assumed to be present.

6.11 Modelling the robot(s) behaviours

Building behaviour-based control systems for robots has become a major alternative to the traditional robot design methodology. In particular to build robot behaviours is a tough job because the designer has to predict the interactions between the robot and the environment as well as to deal with them. However, according to (Brooks, 1990) the extension of the behaviour-based approach to design complete autonomous robots for more complex task still present some challenges. The first concern is how to code a single behaviour module always capable of dealing with the information in the uncompleted known world to achieve a specific task. The second one is actually the action selection problem, it concerns how to decide, for a system including multiple behaviours to handle a variety of situations, which behaviour or behaviours should be active at any particular time to achieve various tasks. All these problems

will become even more challenging when the number of behaviour modules involved increases to follow the increase in task complexity. The capability of each robot to navigate safe and successfully throughout its mission is achieved through the breaking down of its control system into a set of four behaviours as shown in Table 6-4 co-ordinated by the action co-ordinator mechanism (note that the total number of behaviours is dependent on the particular application). In the following sections each of the behaviours is described.

Name	Function Representation	Goal
go_to	$([\vartheta]_D, [\vartheta]_S) \rightarrow [\delta V_{LD}^1, \delta V_{RD}^1]$	Go towards the target
avoid_static	$([sS_L, sS_C, sS_R]) \rightarrow [\delta V_{LD}^2, \delta V_{RD}^2]$	Ignore target and turn away from static obstacles
avoid_dynamic	$([sD_L, sD_C, sD_R, T]) \rightarrow [\delta V_{LD}^3, \delta V_{RD}^3]$	Ignore target and avoid dynamic obstacles
avoid_trap	$([sD_L, sD_C, sD_R]) \rightarrow [\delta V_{LD}^4, \delta V_{RD}^4]$	Avoid trap situation

Table 6-4 Behaviours used in CAROS for robot navigation

6.12 Analysis of *go_to* robot behaviour

The *go_to* behaviour is implemented to steer and move the robot towards the location of the target. Although this is very simple behaviour, when combined with the other behaviours it produces a very robust and reliable navigation system. The input to *go_to* behaviour is ϑ_{error} which is the difference between ϑ_D and ϑ_S angles as shown in Figure 6-18. The behaviour's output is δV_{LD}^1 and δV_{RD}^1 representing the difference in left and right wheel desired velocity respectively. This difference is subtracted from the desired robot velocity (V_D) as shown in Figure 6-18.

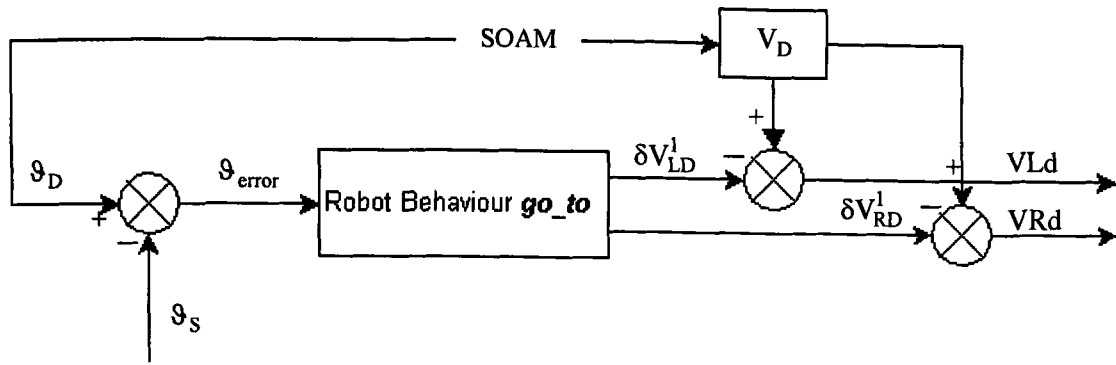


Figure 6-18 Schematic layout of *go_to* robot behaviour

Figure 6-19 shows a schematic layout of robot's possible orientation during its mission. More specifically, Figure 6-19(a) shows that robot orientation is towards the target and therefore current robot orientation ϑ_S is equal to desired orientation ϑ_D and therefore $\vartheta_{error} = 0$. Figure 6-19(b) illustrates a case in which the robot needs to move in a direction that is ϑ_{error} (positive) away from the direction that the robot is currently pointing. For instance, if $\vartheta_{error} = 90$ degrees, then the robot's direction of motion should be "left". Similarly, Figure 6-19(c) illustrates an example in which ϑ_{error} is negative and therefore the robot's direction of motion should be "right".

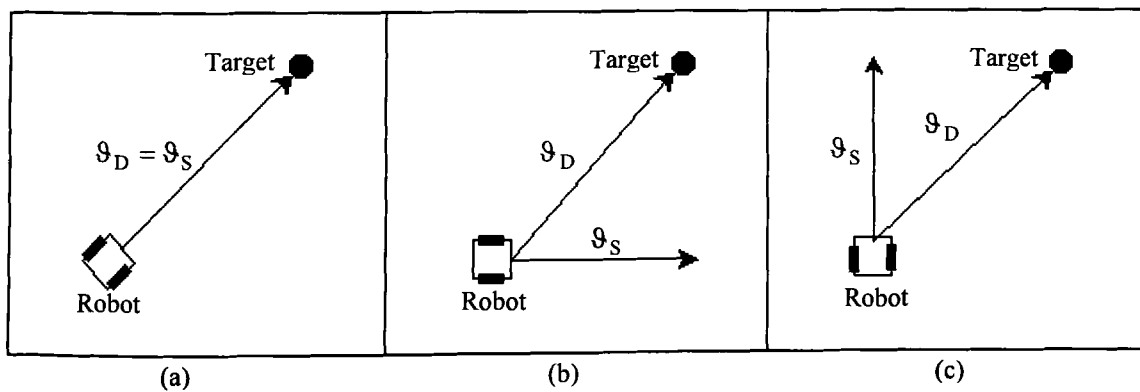


Figure 6-19 Schematic layout of possible robot orientation

From the analysis of Figure 6-19 it can be concluded that the robot's *go_to* behaviour can be modelled with a continuous odd function $\delta V_{LD(RD)}^1 = \delta V_{LD(RD)}^1(\vartheta_{error})$, defined in the first and third quadrant and which satisfies condition $\delta V_{LD}^1(0) = 0$ or $\delta V_{RD}^1(0) = 0$ as shown in Figure 6-20.

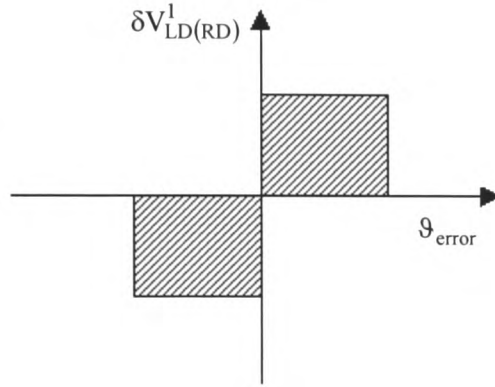


Figure 6-20 Function defined in the first and third quadrant (shaded regions)

The unknown odd function $\delta V_{LD(RD)}^1 = \delta V_{LD(RD)}^1(\vartheta_{error})$ can be selected from a family of functions Φ shown in Equation (6.13), which satisfies all required conditions:

$$\delta V_{LD(RD)}^1 = K \cdot \tanh(S \cdot \vartheta_{error}) \quad (6.13)$$

Where $K > 0$ and $S > 0$ are scaling factors. The scaling factors used in *go_to* robot behaviour are $K = 0.2$ and $S = 0.02$. To increase modularity a dialog box shown in Figure 6-21 has been created for automatic generation of odd function incorporating different values of scaling factors.

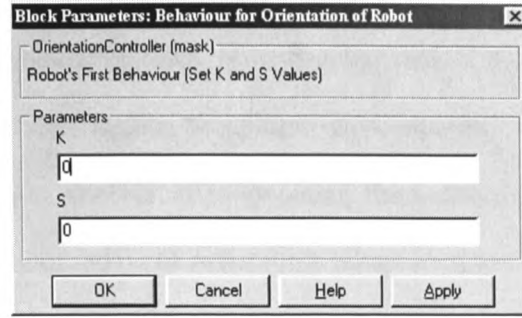


Figure 6-21 Dialog box for scaling factor selection used in *go_to* robot behaviour

Equations (6.14) and (6.15) describe the internal structure of *go_to* robot behaviour shown in Figure 6-18.

$$\delta V_{LD}^1 = 0.2 \cdot \tanh(0.02 \cdot \vartheta_{\text{error}}) \quad (6.14)$$

$$\delta V_{RD}^1 = -0.2 \cdot \tanh(0.02 \cdot \vartheta_{\text{error}}) \quad (6.15)$$

The graphical representation of equations (6.14) and (6.15) describing the *go_to* behaviour is shown in Figure 6-22. It is clear that the behaviour's outputs δV_{LD}^1 and δV_{RD}^1 always have a opposite sign due to the systems symmetry except when $\vartheta_{\text{error}} = 0$ where both outputs are equal to zero.

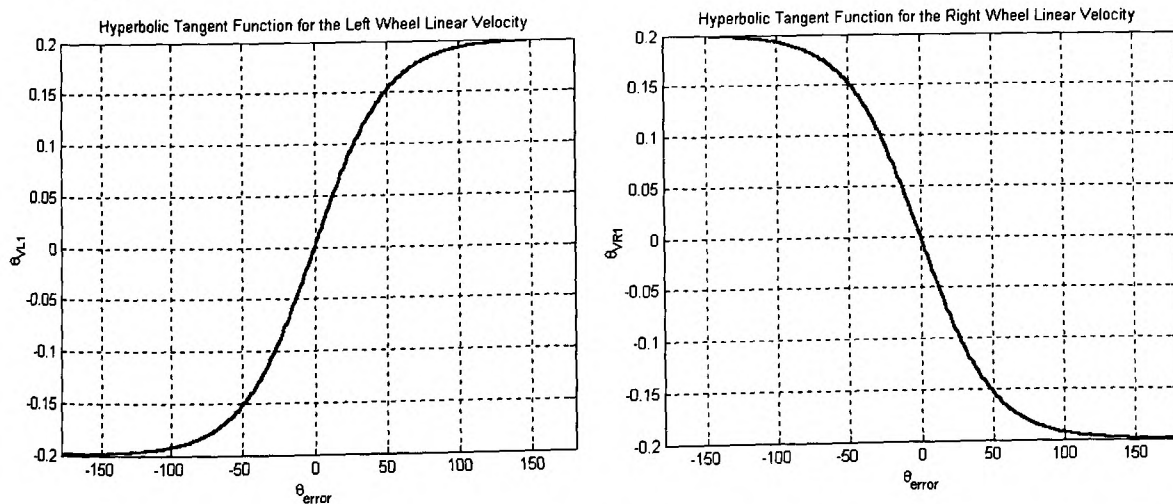


Figure 6-22 Shape of hyperbolic tangent functions describing the output of *go_to* behaviour with $K = 0.2$ and $S = 0.02$ scaling factors.

6.13 Analysis of *avoid_static* robot behaviour

The aim of intelligent robotics research is to develop mobile robots capable of navigating autonomously in unstructured and/or unexplored environments. This development requires intelligent control strategies capable of overcoming the uncertainties presented by the real world. To ensure the safety of the robot system it is necessary that it is able to navigate without colliding with obstructions in its environment. Therefore the most important behaviour in the context of robot safety is the obstacle avoidance behaviour. In chapter two several methodologies were discussed for achieving collisions avoidance. In this thesis two types of obstacle avoidance behaviours have been developed, the *avoid_static* and the *avoid_dynamic* robot behaviour. In this section the first behaviour is described whereas the latter is discussed in section 6.14.

The *avoid_static* robot behaviour is responsible for monitoring the sensory information regarding static obstacles in the robot's environment and to move the robot away from an obstacle before a collision occurs. This information is received from the sensors described in section 6.7. For instance, if an obstruction is detected to the left of the robot, a command is issued to turn the robot to the right. If an obstacle is detected to the right, the robot will turn to the left. Similarly, if an obstruction is detected at the front of the robot, a turn to either the left or the right will follow. As stated in Table 6-1 the behavioural encoding is achieved using fuzzy logic, neural networks and stateflow.

The *avoid_static* robot behaviour was implemented using both fuzzy and neural techniques. Fuzzy logic was chosen since encoding behaviour as a set of fuzzy rules is considered a natural way for humans to describe the expected functioning of the behaviour. On the other hand encoding behaviours using neural networks, several interesting features such as learning and modelling a wide class of non-linear systems is achieved. Calculations are in principle carried out in parallel resulting in speed advantages (as was shown in chapter five) and programming

can be done by training rather than defining explicit instructions. Therefore encoding behaviours using neural networks can produce learning a controller for a mobile robot that can operate in an uncertain environment. The static obstacle avoidance using fuzzy logic is described in section 6.13.1 whereas the static obstacle avoidance using neural networks is described in section 6.13.2. The results presented in chapter seven show that both behavioural encoding methods have to produce some positives and negative aspects. Throughout the thesis has mentioned that the overall control architecture is modular. To address this modularity at this stage both fuzzy and neural behavioural encoding methods implementing *avoid_static* behaviour were created as a subsystem as shown in Figure 6-23. Each controller can be selected from the dialog box shown in Figure 6-24. In addition, the fuzzy inference system (FIS) and the neural network unit have been encoded with a format compatible with the MATLAB fuzzy and neural toolboxes. Therefore any files developed in MATLAB implementing other behaviours can be directly migrated to the CAROS architecture and used without any modifications.

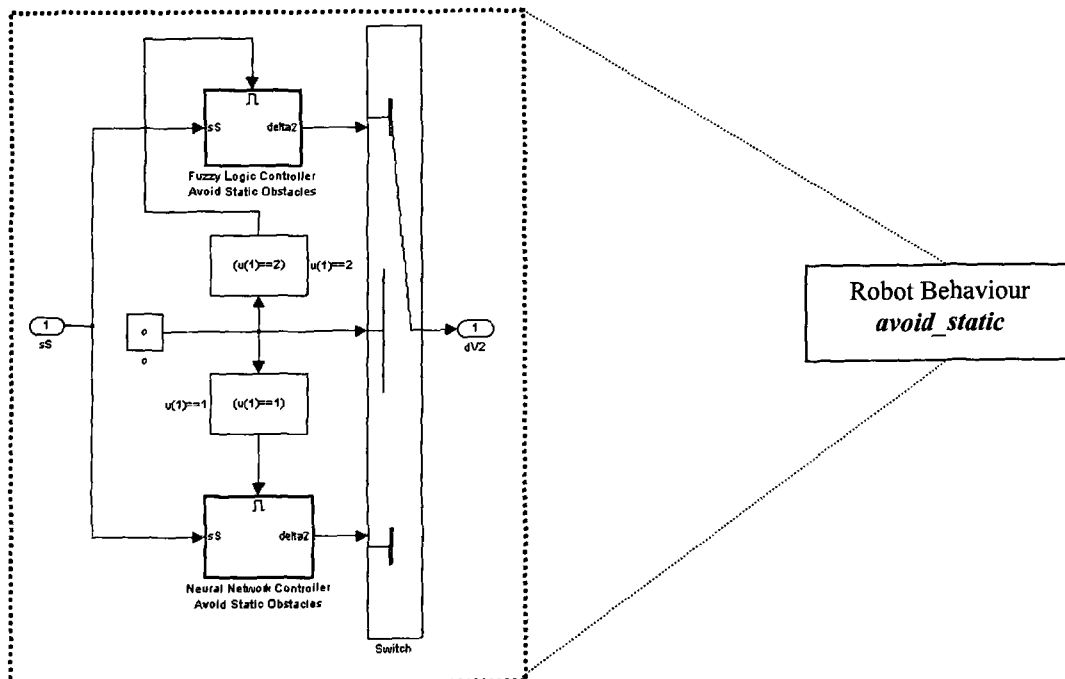


Figure 6-23 Subsystem of *avoid_static* robot behaviour

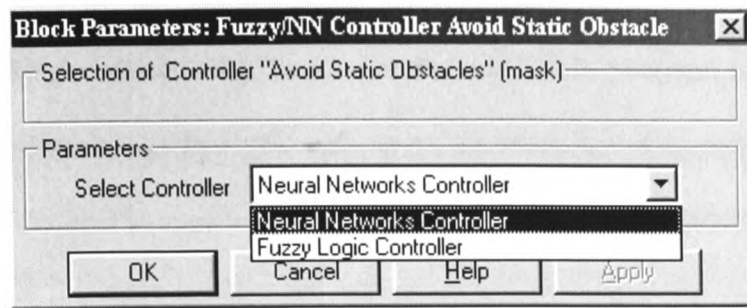


Figure 6-24 Dialog box used in *avoid_static* robot behaviour selection

6.13.1 Static obstacle avoidance using fuzzy controller

The mobile robot's reactive control problem consists of a direct map between input space (i.e. sensor data) into the output space (i.e. control commands). Fuzzy logic is used to implement this matching taking into account the information provided by the sensors and qualitative relations between input and outputs. The *avoid_static* robot behaviour with fuzzy encoding has the same number of inputs as signals provided by the sensors described in section 6.7. These inputs are signals of left sensor (sSL), centre sensor (sSC) and right sensor (sSR) taking measurements from static obstacles. The *avoid_static* behaviour's output is δV_{LD}^2 and δV_{RD}^2 representing the difference in left and right wheel desired velocity. This difference is subtracted from the desired robot velocity (V_D). Schematic layout of *avoid_static* behaviour using fuzzy logic is shown in Figure 6-25

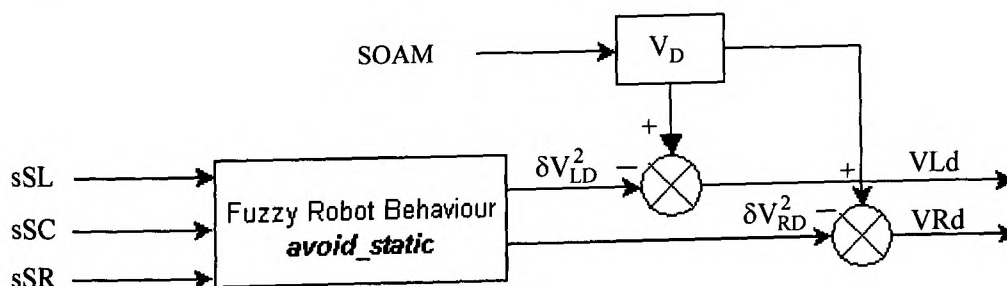


Figure 6-25 Schematic layout of *avoid_static* robot behaviour with fuzzy encoding

Linguistic terms such as “*far*” (F) and “*near*” (N) are defined for the left sensor (sSL), centre sensor (sSC) and right sensor (sSR). Terms such as “small” (S), “medium” (M) and “big” (B) are defined for the δV_{LD}^2 and δV_{RD}^2 variables in the robot’s linear left and right velocities. The type of fuzzy model used to implement *avoid_static* robot behaviour is Mamdani (Mamdani and Assilean, 1975) as it is more intuitive and well-suited to human input (the design methodology behind Mamdani fuzzy model was presented in chapter three, section 3.2.4.1). The MIN fuzzy set operation is used defined by the intersection operator AND. Implication was achieved using the operator MIN, whereas for aggregation the MAX operator was used. The centre of gravity defuzzification method is used because it usually yields superior results (Braae and Rutherford, 1978). The membership functions used for the input and output variable are generalised bell curve membership functions (using this kind of function any shape of membership function can be easily achieved and therefore offers flexibility in the design process). The mathematical expression of the generalised bell function is given in Equation (6.16). The shape and position of this function can be defined using three parameters a , b and c . Parameter b is usually positive whereas parameter c locates the centre of the curve.

$$f(x; a, b, c) = \frac{1}{1 + \left| \frac{x - c}{a} \right|^{2b}} \quad (6.16)$$

Where x is a vector, which defines the universe of discourse.

The membership functions described above are shown in Figure 6-26 and Figure 6-27. Table 6-5 and Table 6-6 lists the parameters defining the membership functions for input and output variables.

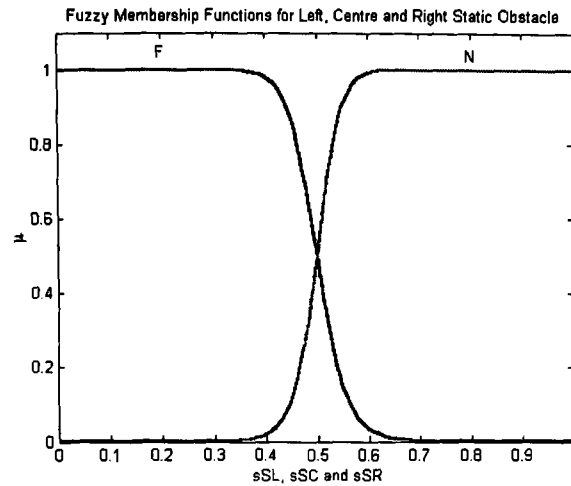


Figure 6-26 Fuzzy membership functions used for the input variables of *avoid_static* robot behaviour

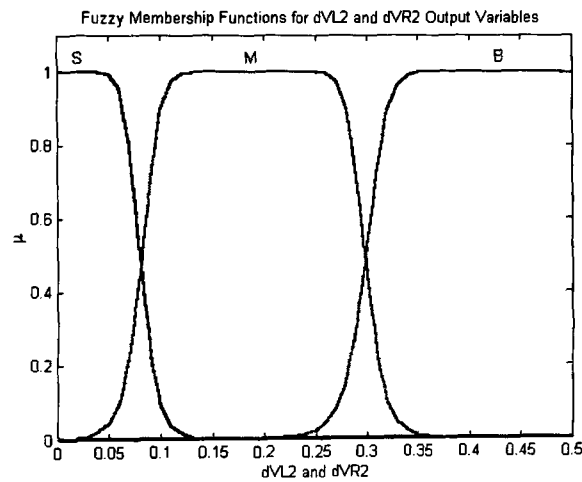


Figure 6-27 Fuzzy membership functions used for the output variables of *avoid_static* robot behaviour

Variables	Parameters	Far (F)	Near (N)
SSL, sSC and sSR	a	0.5	0.5
	b	9.1	10.7
	c	0.0025	1

Table 6-5 Parameters of fuzzy membership functions for *avoid_static* input variables

Variables	Parameters	Small (S)	Medium (M)	Big (B)
δV_{LD}^2 and δV_{RD}^2	a	0.08	0.108	0.125
	b	5.2	5.94	5.74
	c	0	0.19	0.424

Table 6-6 Parameters of fuzzy membership functions for *avoid_static* output variables

When the robot is very close to an obstacle, it must change its speed and steering angle in order to avoid the obstacle. The fuzzy rules used for obstacle avoidance by robots are listed in Table 6-7. All the rules were obtained heuristically using human reasoning. For instance, in rule 7, the left static obstacle is “near”, the right static obstacle is “far” and the front static obstacle is “near”, then the robot should turn to the right as soon as possible (sharp turn) to avoid the obstacles in front and on its left. For the above situation the left velocity should decrease slowly and right velocity should decrease fast as shown in Figure 6-28.

Rule No.	sSL	sSC	sSR	δV_{LD}^2	δV_{RD}^2	Action
1	Far	Far	Far	Small	Small	Slow down little
2	Far	Far	Near	Medium	Small	Slow down and turn left smooth
3	Far	Near	Far	Medium	Small	Slow down and turn left smooth
4	Far	Near	Near	Big	Small	Slow down and turn left sharp
5	Near	Far	Far	Small	Medium	Slow down and turn right smooth
6	Near	Far	Near	Medium	Medium	Slow down medium
7	Near	Near	Far	Small	Big	Slow down and turn right sharp
8	Near	Near	Near	Big	Big	Slow down fast

Table 6-7 List of fuzzy rules incorporated in fuzzy *avoid_static* robot behaviour

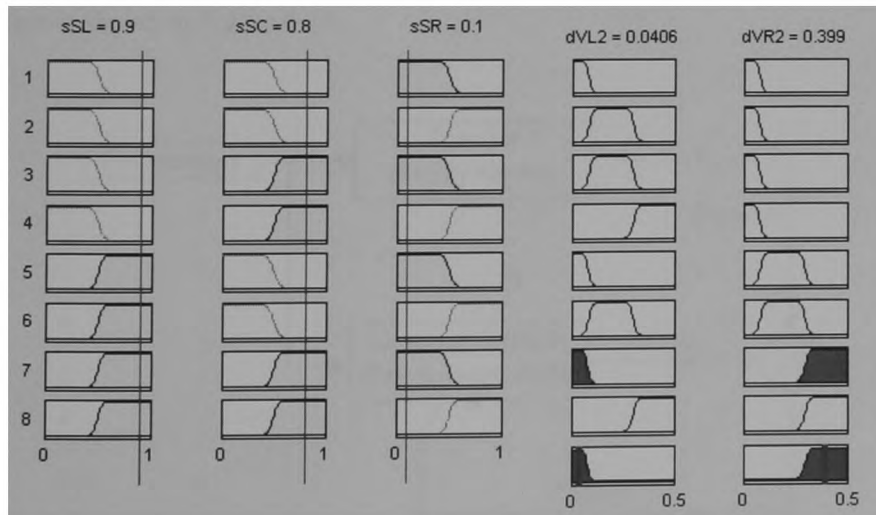


Figure 6-28 Values of δV_{LD}^2 and δV_{RD}^2 when rule 7 is activated

In the following section the *avoid_static* robot behaviour is replaced by neural network. Both behaviours (fuzzy and neural) are compared in chapter seven where the testing of the CAROS control architecture is presented.

6.13.2 Static obstacle avoidance using neural controller

This section describes the development of the *avoid_static* robot behaviour using neural networks instead of fuzzy logic. The schematic layout of the *avoid_static* robot behaviour incorporating neural encoding is the same as in Figure 6-25. The difference between the fuzzy and neural encoding is mainly based on the internal control structure/architecture used. In chapter three the main design methodology of neural networks and their training techniques were presented. In chapter five an algorithmic methodology was proposed for identification, modelling and control of dynamic system using neural networks based on supervised learning (learning with a teacher). In particular recorded data from PI controller were used to train the neural network. In this section a similar algorithmic methodology is used to train static nature neural network using the fuzzy model derived in section 6.13.1 as a teacher. The robot using the neural network controller should be able to avoid static obstacles in the robot(s) working environment at least as well as the fuzzy model. The form of neural network learning proposed in this section is shown in Figure 6-29.

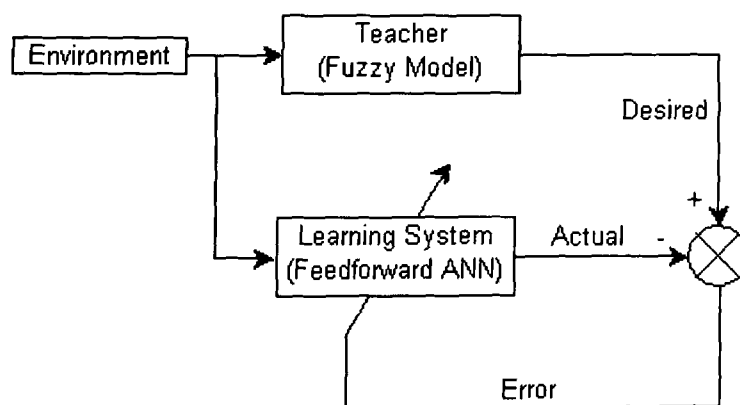


Figure 6-29 ANN learning using fuzzy model as a teacher.

As can be seen from Figure 6-29 the teacher (fuzzy model) having knowledge of the environment (signals from sensors) and also having knowledge of the control output should be (control values of δV_{LD}^2 and δV_{RD}^2). Therefore it can be assumed that the fuzzy model knowledge can be used as one kind of representation of a set of input-output data. The environment is, however unknown to the neural network. At the beginning of the learning process the fuzzy model (teacher) and the neural network are both exposed to the training vector drawn from the environment (signals sSL, sSC and sSR). At this stage the fuzzy model is able to provide the neural network with a desired response for that particular training vector. The desired response represents the action to be performed by the neural network. The network parameters are adjusted using both the training vector and the error signal. The network parameters are adjusted until the neural network matches the fuzzy model response up to a predefined error. When the network is fully trained the fuzzy model is removed.

The neural network architecture for the *avoid_static* robot behaviour is defined using the design principles discussed in chapter five. At the beginning a small size network was selected and gradually increases until the error goal is met. The network architecture chosen is feedforward multilayer perceptron network. The performance function is defined by the mean square error (MSE), which is the average squared error between the network output response and the fuzzy model output response. As in chapter five the backpropagation technique is used and also the algorithm of Levenberg-Marquardt to increase the convergence speed. The final structure of the static neural network incorporated in the *avoid_static* robot behaviour is shown in Figure 630.

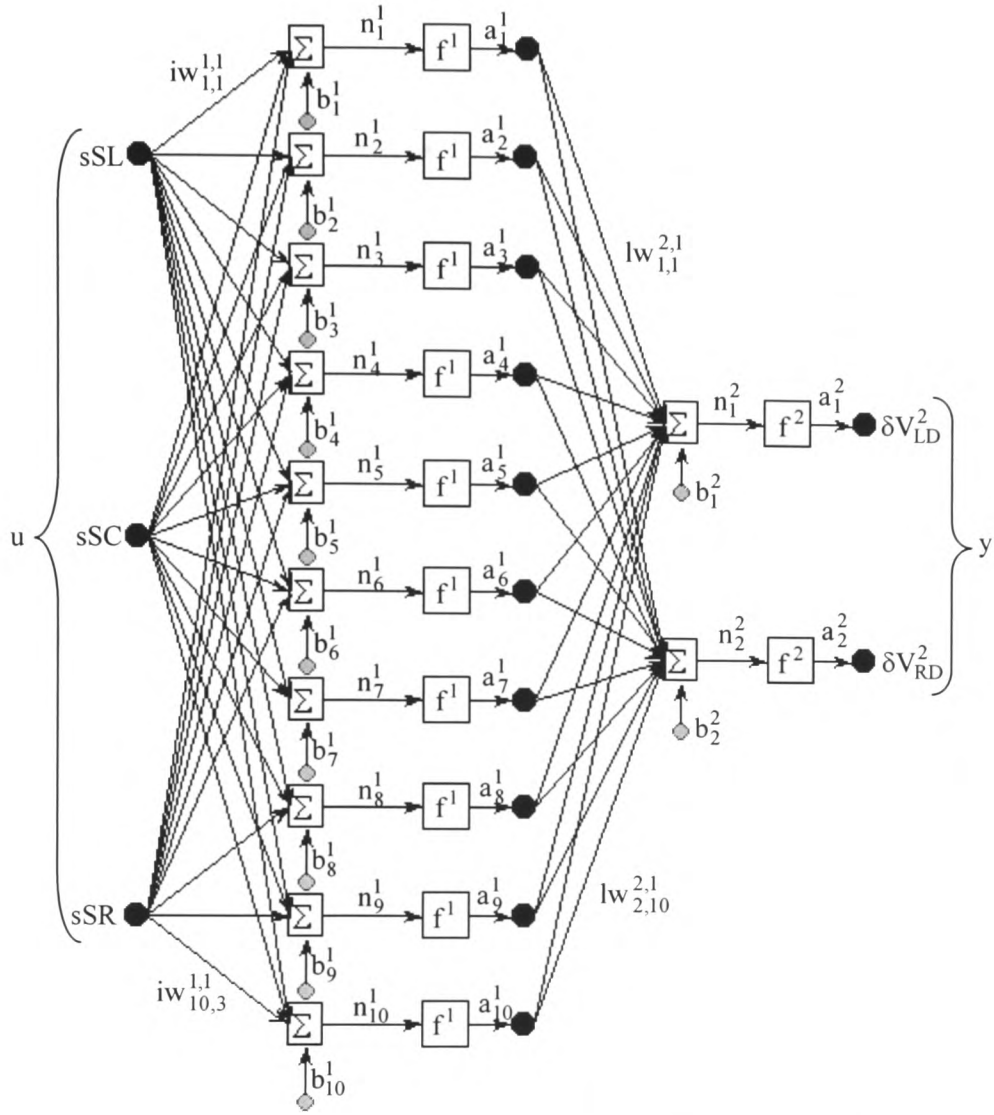


Figure 6-30 Structure of static NN used in *avoid_static* robot behaviour

The network shown in Figure 6-30 has one hidden layer with ten neurons and an output layer consisting of two neurons. The input and output to the network are column vectors as shown in Equations (6.17) and (6.18).

$$u = \begin{bmatrix} sSL \\ sSC \\ sSR \end{bmatrix} \quad (6.17)$$

$$y = \begin{bmatrix} \delta V_{LD}^2 \\ \delta V_{RD}^2 \end{bmatrix} \quad (6.18)$$

Each layer has a weight matrix W , a bias vector b , and an output vector a . The bias vector of the first and second layer is defined by the following equation.

$$b^1 = \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \\ b_4^1 \\ b_5^1 \\ b_6^1 \\ b_7^1 \\ b_8^1 \\ b_9^1 \\ b_{10}^1 \end{bmatrix} \quad b^2 = \begin{bmatrix} b_1^2 \\ b_2^2 \end{bmatrix} \quad (6.19)$$

The weight matrix of the first and second layer is given by Equations (6.20) and (6.21).

$$IW^{1,1} = \begin{bmatrix} iw_{1,1}^{1,1} & iw_{1,2}^{1,1} & iw_{1,3}^{1,1} \\ iw_{2,1}^{1,1} & iw_{2,2}^{1,1} & iw_{2,3}^{1,1} \\ iw_{3,1}^{1,1} & iw_{3,2}^{1,1} & iw_{3,3}^{1,1} \\ iw_{4,1}^{1,1} & iw_{4,2}^{1,1} & iw_{4,3}^{1,1} \\ iw_{5,1}^{1,1} & iw_{5,2}^{1,1} & iw_{5,3}^{1,1} \\ iw_{6,1}^{1,1} & iw_{6,2}^{1,1} & iw_{6,3}^{1,1} \\ iw_{7,1}^{1,1} & iw_{7,2}^{1,1} & iw_{7,3}^{1,1} \\ iw_{8,1}^{1,1} & iw_{8,2}^{1,1} & iw_{8,3}^{1,1} \\ iw_{9,1}^{1,1} & iw_{9,2}^{1,1} & iw_{9,3}^{1,1} \\ iw_{10,1}^{1,1} & iw_{10,2}^{1,1} & iw_{10,3}^{1,1} \end{bmatrix} \quad (6.20)$$

$$LW^{2,1} = \begin{bmatrix} lw_{1,1}^{2,1} & lw_{1,2}^{2,1} & lw_{1,3}^{2,1} & lw_{1,4}^{2,1} & lw_{1,5}^{2,1} & lw_{1,6}^{2,1} & lw_{1,7}^{2,1} & lw_{1,8}^{2,1} & lw_{1,9}^{2,1} & lw_{1,10}^{2,1} \\ lw_{2,1}^{2,1} & lw_{2,2}^{2,1} & lw_{2,3}^{2,1} & lw_{2,4}^{2,1} & lw_{2,5}^{2,1} & lw_{2,6}^{2,1} & lw_{2,7}^{2,1} & lw_{2,8}^{2,1} & lw_{2,9}^{2,1} & lw_{2,10}^{2,1} \end{bmatrix} \quad (6.21)$$

The final output y of the *avoid_static* robot behaviour using neural network encoding illustrated in Figure 6-30 is calculated as follows

$$y = \text{purelin} \left(\underbrace{\underbrace{\underbrace{\mathbf{LW}^{2,1} \text{tansig} \left(\underbrace{\underbrace{\mathbf{IW}^{1,1} \mathbf{u} + \mathbf{b}^1}_{n^1}}_{a^1} \right) + \mathbf{b}^2}_{n^2}}_{a^2}} \right) \quad (6.22)$$

6.14 Analysis of *avoid_dynamic* robot behaviour

It was mentioned in section 6.13 that the most important behaviour in the context of robot safety is the obstacle avoidance behaviour. The *avoid_dynamic* is the third robot behaviour developed in this thesis and it is presented in this section. According to the global control strategy the task of each robot is to reach its target position avoiding collisions with static and dynamic obstacles. As a robot plans its motion based on the local information from its sensors (see section 6.7), effectively each robot is a moving obstacle for the other robots (note there is no direct communication between robots). Therefore a robot has no knowledge about other objects (static obstacles or robots) within its working environment until it senses them. The task of the *avoid_dynamic* robot behaviour is to monitor the sensory information regarding dynamic obstacles (other robots) in the robot's environment, to identify the direction of the moving obstacle and finally to provide the robot with a set of traffic rules so that a possible collision with other robots is avoided. An assumption made in this section is that the sensory data block can distinguish between a moving object (robot) and a stationery object. The sensory information from left, centre and right of the robot are denoted by dSL , dSC and dSR respectively. As can be seen from Table 6-1 the behavioural encoding of all robot behaviours is achieved using fuzzy, neural and stateflow. The *avoid_dynamic* robot behaviour is hybrid incorporating two types of behaviours: a hybrid stateflow-fuzzy discussed in section 6.14.2 and

a hybrid stateflow-neural discussed in section 6.14.3. A schematic layout of the *avoid_dynamic* robot behaviour is shown in Figure 6-31. The internal structure of the *avoid_dynamic* robot behaviour is shown in Figure 6-33. A desired hybrid controller in this behaviour is selected using the dialog box in Figure 6-32. In addition, the design methodology followed with the *avoid_static* robot behaviour is adopted here, which means that any files developed in MATLAB implementing other types of behaviours can be directly migrated to CAROS architecture and used without any modifications.

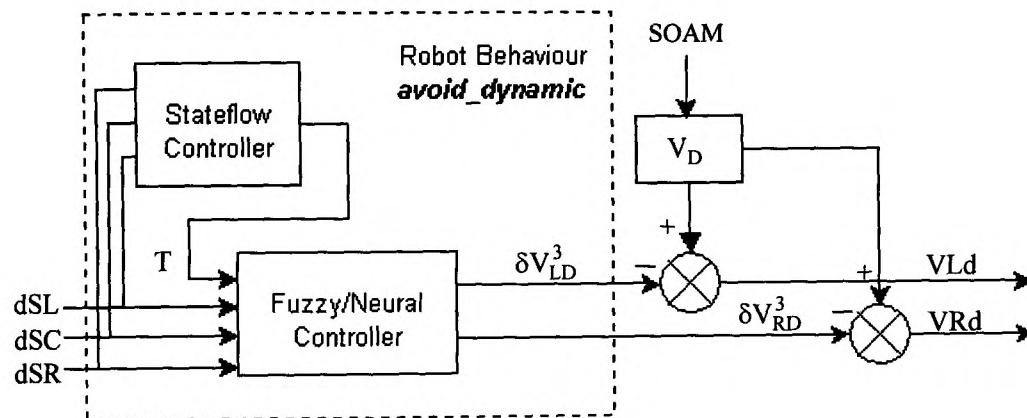


Figure 6-31 Schematic layout of a hybrid *avoid_dynamic* robot behaviour

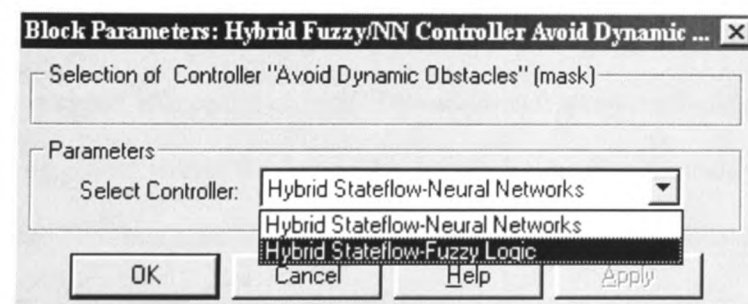


Figure 6-32 Dialog box used in *avoid_dynamic* robot behaviour selection

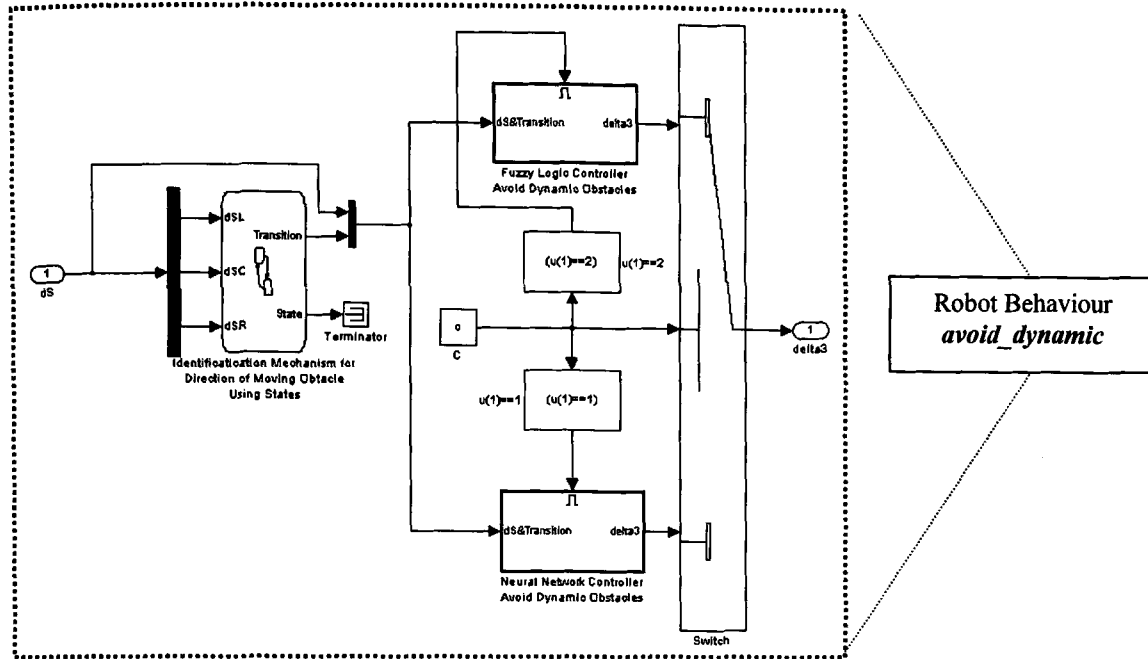


Figure 6-33 Internal structure of *avoid_dynamic* robot behaviour

6.14.1 Identification of direction of neighbour robot (moving object)

In this section a novel method for identification of direction of moving object (neighbour robot) based only on three sensory inputs is presented. The sensor arrangement on the mobile robot was discussed in section 6.7 when the sensors schematic layout was illustrated in Figure 6-8. As the robot navigates towards its target the possibility of detecting moving objects can be decomposed into six states as shown in Figure 6-34. Three states can be assigned if the robot detects the moving object left, centre or right. Two additional states can be assigned if the robot detects the moving object within the left-centre sensor intersection or within the centre-right sensor intersection. The final state is assigned if the moving object is outside the robot's sensor range area. A list of all states assigned for the identification of direction of moving object is given in Table 6-8. As all the possible states of the position of the moving object have been assigned what is remaining is to identify all possible directions of the moving object within the robot's sensor range area. Figure 6-35 shows some of the possible directions of the moving

object around the robot's sensing area. In particular Figure 6-35(a) shows five possible cases in which the moving object could enter the robot's sensing area. Case one, three and five illustrating an example in which the moving object enters the robot's sensing area at the left, centre or right side from the outside world. Case two and four showing that the moving object can enter the robot's sensing area either from the intersection of left-centre sensors or the intersection of centre-right sensors. Figure 6-35(b) and Figure 6-35(c) shows the cases in which the moving object enters and crosses the robot's sensing area either from left to right or from right to left.

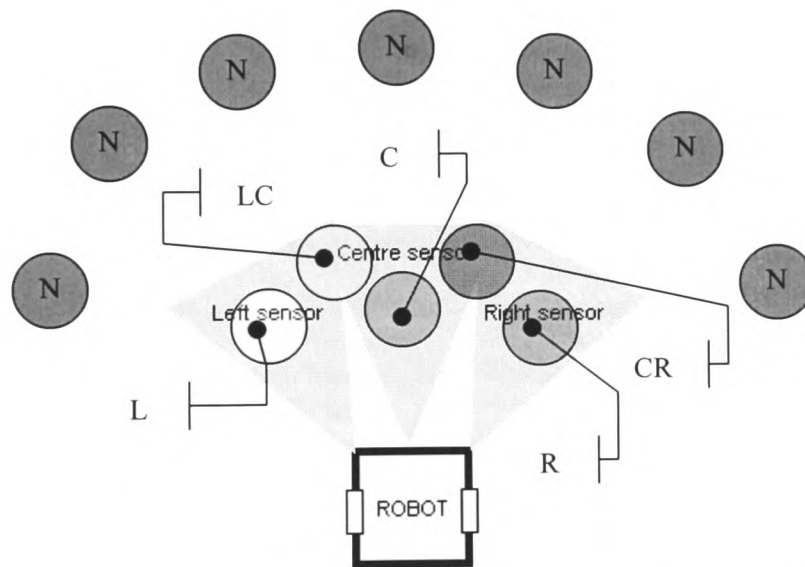


Figure 6-34 Schematic layout of possible positions (states) of moving obstacle

State	Name	Representation
N	Neutral	Obstacle outside the sensors range
L	Left	Obstacle within the left sensor
C	Centre	Obstacle within the centre sensor
R	Right	Obstacle within the right sensor
LC	LeftCentre	Obstacle within the intersection of left and centre sensor
CR	CentreRight	Obstacle within the intersection of centre and right sensor

Table 6-8 States used to identify the direction of moving obstacle

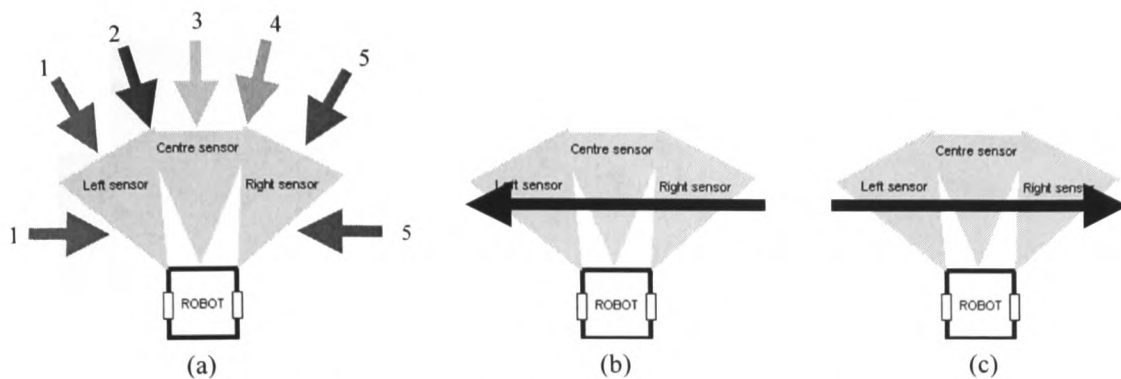


Figure 6-35 Possible directions of moving object when entering the robot's sensing area

In section 6.10.1 a supervisor-like co-ordination mechanism was modelled using FSM. The FSM used in that case was Moore FSM as all actions were associated with states. In other words when an event occurs a transition between two states is evaluated and if it is true, then one of the defined states is activated resulting in a specific control action. For the identification of the direction of moving object a Mealy FSM is designed based on the six states previously discussed. Using Mealy FSM, actions are associated with transitions rather than states. When an event occurs a transition is evaluated. The condition action is executed as soon as the condition is evaluated as true and before the transition destination has been determined to be valid. Specifying a transition action means that the action is executed when the transition is taken, provided the condition, if specified, is true. Figure 6-36 shows a Mealy FSM used to identify the direction of the moving object.

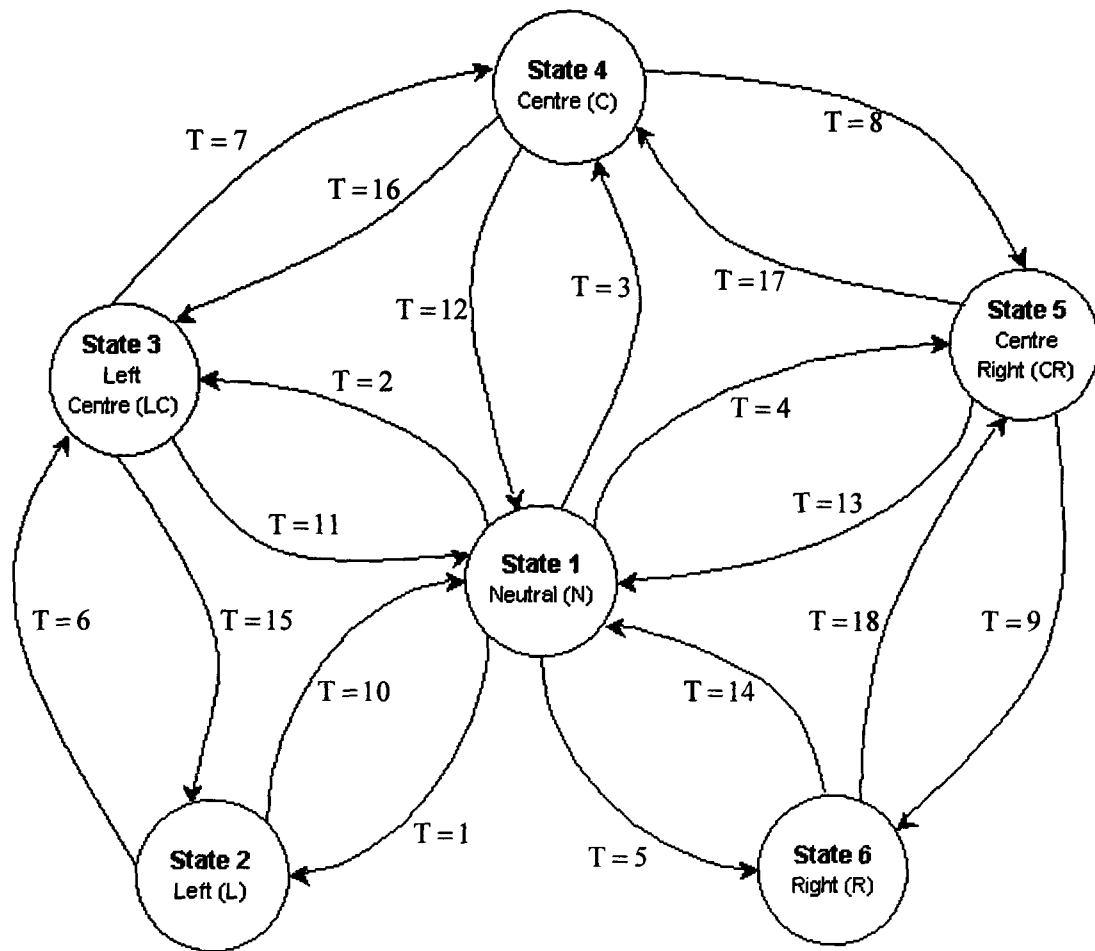


Figure 6-36 Identification of direction of moving obstacle using stateflow

From Figure 6-36 it can be observed that there are fifteen transitions associated the six states shown in Table 6-8. Every transition has been assigned a specific condition and action. A list of all the specified conditions for all transitions used in Figure 6-36 is given in Table 6-9. It can be seen from Table 6-9 that only three sensor measurements (dSL, dSC and dSR) are used to specify all transitions used in Figure 6-36. Furthermore Table 6-9 shows that the transitions actions are integer numbers from zero to eighteen. This means that when an event occurs only one transition is active (output of FSM can be between 0 to 18) if the associated condition specified is true. The transitions shown in Figure 6-36 in conjunction with the associated

conditions given in Table 6-9, covers all the possible directions of the moving object(s) around the robot's sensing area.

Transition No. (T)	Condition	Direction of moving obstacle
0 (IDLE)	$dSL=0 \& dSC=0 \& dSR=0$	Obstacle outside the sensing area
1	$dSL>0 \& dSC=0 \& dSR=0$	From outside to the left (N-L)
2	$dSL>0 \& dSC>0 \& dSR=0$	From outside to left centre (N-LC)
3	$dSL=0 \& dSC>0 \& dSR=0$	From outside to centre (N-C)
4	$dSL=0 \& dSC>0 \& dSR>0$	From outside to centre right (N-CR)
5	$dSL=0 \& dSC=0 \& dSR>0$	From outside to the right (N-R)
6	$dSL>0 \& dSC>0 \& dSR=0$	From left to the left centre (L-LC)
7	$dSL=0 \& dSC>0 \& dSR=0$	From left centre to the centre (LC-C)
8	$dSL=0 \& dSC>0 \& dSR>0$	From centre to the centre right (C-CR)
9	$dSL=0 \& dSC=0 \& dSR>0$	From centre right to the right (CR-R)
10	$dSL=0 \& dSC=0 \& dSR=0$	From left to the outside (L-N)
11	$dSL=0 \& dSC=0 \& dSR=0$	From left centre to the outside (LC-N)
12	$dSL=0 \& dSC=0 \& dSR=0$	From centre to the outside (C-N)
13	$dSL=0 \& dSC=0 \& dSR=0$	From centre right to the outside (CR-N)
14	$dSL=0 \& dSC=0 \& dSR=0$	From right to the outside (R-N)
15	$dSL>0 \& dSC>0 \& dSR=0$	From left centre to the left (LC-L)
16	$dSL>0 \& dSC>0 \& dSR=0$	From centre to the left centre (C-LC)
17	$dSL=0 \& dSC>0 \& dSR=0$	From centre right to the centre (CR-C)
18	$dSL=0 \& dSC>0 \& dSR>0$	From right to the centre right (R-CR)

Table 6-9 Transitions used for the identification of direction of moving obstacle

6.14.2 Dynamic obstacle avoidance using hybrid stateflow-fuzzy controller

In section 6.13.1 was shown that the mobile robot's reactive control consists of a direct map between input space into the output space. In this section the schematic layout of a hybrid *avoid_dynamic* robot behaviour shown in Figure 6-31 using stateflow-fuzzy encoding is presented. As shown in Figure 6-31 the *avoid_dynamic* robot behaviour has four inputs (dSL , dSC , dSR and T) and two outputs (δV_{LD}^3 and δV_{RD}^3). The purpose of the *avoid_dynamic* robot behaviour is to provide the robot with a set of specified traffic rules. This is make sure the use of a simple mechanism of "reasonable behaviour", which means that when a robot is manoeuvring to avoid a potential collision with other robots, each robot assumes that other

robots will try to avoid collisions as well. In other words, although the motion and sensing parameters may differ widely from robot to robot, each robot can safely assume that other robots operate under the same strategy or “*traffic rules*” such as in ship navigation and in air traffic collision avoidance systems. Similar to *avoid_static* robot behaviour linguistic terms such as “*far*” (F) and “*near*” (N) are defined for the left sensor (dSL), centre sensor (dSC) and right sensor (dSR). Terms such as “*from right to centre-right*” (R-CR) is used to defined the direction of the moving object within the current robot sensing range. However, terms such as “small” (S), “medium” (M) and “big” (B) are defined for the δV_{LD}^3 and δV_{RD}^3 variables in the robot’s linear left and right velocities.

The type of fuzzy model used to implement *avoid_dynamic* robot behaviour is Sugeno (Takagi and Sugeno, 1985) as it well suited to mathematical analysis and guarantees continuity of the output surface. However the most important reason for the Sugeno fuzzy model selection is that of clear relationship between input and output space. As the *avoid_dynamic* robot behaviour was implemented with a set of traffic rules a particular output of the fuzzy model can be specified upon certain value of input variables. In addition, as the *avoid_dynamic* robot behaviour is a hybrid incorporating FSM the advantage of computationally efficient Sugeno fuzzy model leads to a desirable design method. The membership functions used to model input variables are generalised bell curve (for more details see section 6.13.1) and triangular (see chapter three, section 3.2.2.3) membership functions as shown in Figure 6-37 and Figure 6-38. Table 6-10 and Table 6-11 lists the parameters defining the membership functions for the input variables. The membership functions used for the output variables δV_{LD}^3 and δV_{RD}^3 are constant membership functions listed in Table 6-12 (note that Sugeno fuzzy model output uses singletons as either constant or linear membership functions).

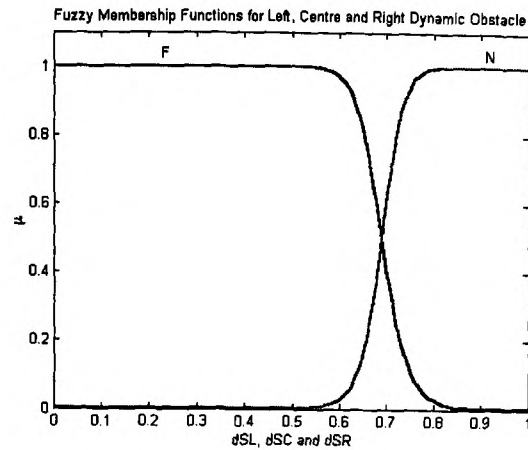


Figure 6-37 Fuzzy membership functions used for the dSL, dSC and dSR input variable in *avoid_dynamic* robot behaviour

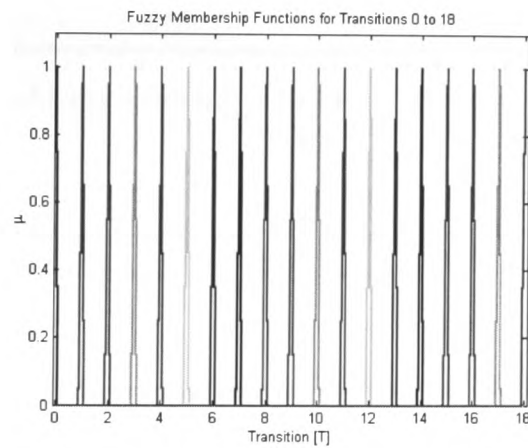


Figure 6-38 Fuzzy membership function used for T input variable of *avoid_dynamic* robot behaviour

Variables	Parameters	Far (F)	Near (N)
dSL, dSC and dSR	a	0.5	0.5
	b	9.1	10.7
	c	0.19	1.19

Table 6-10 Parameters of fuzzy membership functions of dSL, dSC and dSR for *avoid_dynamic* input variables

Variable	Term	a	b	c
T	IDLE	-0.09	0	0.1
	N-L	-0.9	1	1.1
	N-LC	1.9	2	2.1
	N-C	2.9	3	3.1
	N-CR	3.9	4	4.1
	N-R	4.9	5	5.1
	L-LC	5.9	6	6.1
	LC-C	6.9	7	7.1
	C-CR	7.9	8	8.1
	CR-R	8.9	9	9.1
	L-N	9.9	10	10.1
	LC-N	10.9	11	11.1
	C-N	11.9	12	12.1
	CR-N	12.9	13	13.1
	R-N	13.9	14	14.1
	LC-L	14.9	15	15.1
	C-LC	15.9	16	16.1
	CR-C	16.9	17	17.1
	R-CR	17.9	18	18.1

Table 6-11 Parameters of fuzzy membership functions of T for *avoid_dynamic* input variable

Variables	Term	Value
δV_{LD}^3 and δV_{RD}^3	B	0.5
	M	0.25
	S	0

Table 6-12 Singletons used of δV_{LD}^3 and δV_{RD}^3 for *avoid_dynamic* output variables

When the mobile robot has detected a moving object (another robot) it must change either its speed or steering angle to avoid the object. The fuzzy rules used for the dynamic obstacle avoidance incorporating traffic rules are listed in Table 6-13. All the rules were obtained based on traffic rules followed by humans when driving a car. More specifically the traffic rules are design based on the scenario that traffic lights are out of order and therefore drivers have to use common sense rules to avoid collisions. For example, if a robot is navigating towards its target and another moving robot has been detected on the right side, then the control action of the

robot will be to stop (or decrease speed) and let the other robot to pass through. An opposite scenario could be that the neighbour robot has been detected on the left side of the robot, then the robot's control action will be to continue its mission knowing that the other robot will follow the same traffic rules and therefore stop. For instance in rule 30, a moving object has been detected in which its direction appears to be from the right side of the robot to the centre (R-CR), in addition the centre robot sensor indicates that moving obstacle is "near", then the robot according to the traffic rules should stop to allow the other robot to pass. For this particular situation the control commands to the robot's motors should be zero left and right linear velocity ($\delta V_{LD}^3 = 0.5$ and $\delta V_{RD}^3 = 0.5$ if $V_D = 0.5$) as shown in Figure 6-39.

Rule No.	dSL	dSC	dSR	T	δV_{LD}^3	δV_{RD}^3	Action
1	Far	-	-	NL	Medium	Medium	Slow down
2	Near	-	-	NL	Medium	Big	Slow down and turn right
3	-	-	-	N-LC	Medium	Big	Slow down and turn right
4	-	-	-	NC	Medium	Small	Turn left little
5	-	-	-	N-CR	Big	Small	Turn left fast
6	-	-	Far	N-R	Medium	Medium	Slow down
7	-	-	Near	N-R	Big	Big	STOP
8	-	Far	-	L-LC	Medium	Small	Turn left little
9	-	Near	-	L-LC	Big	Big	STOP
10	-	Far	-	LC-C	Medium	Small	Turn left little
11	-	Near	-	LC-C	Big	Big	STOP
12	-	-	Far	C-CR	Small	Small	Continue
13	-	-	Near	C-CR	Medium	Small	Turn left little
14	-	-	Far	CR-R	Small	Small	Continue
15	-	-	Near	CR-R	Big	Small	Turn left fast
16	Far	-	-	L-N	Small	Small	Continue
17	Near	-	-	L-N	Small	Medium	Turn right little
18	-	-	-	LC-N	Small	Small	Continue
19	-	-	-	C-N	Small	Small	Continue
20	-	-	-	CR-N	Small	Small	Continue
21	-	-	Far	R-N	Small	Small	Continue
22	-	-	Near	R-N	Medium	Small	Turn left little
23	Far	-	-	LC-L	Small	Small	Continue
24	Near	-	-	LC-L	Medium	Big	Slow down and turn right
25	-	Far	-	C-LC	Medium	Medium	Slow down
26	-	Near	-	C-LC	Small	Big	Turn right fast

27	-	Far	-	CR-C	Medium	Medium	Slow down
28	-	Near	-	CR-C	Big	Big	STOP
29	-	Far	-	R-CR	Medium	Medium	Slow down
30	-	Near	-	R-CR	Big	Big	STOP
31	-	-	-	IDLE	Small	Small	Continue

Table 6-13 List of fuzzy rules used in hybrid stateflow-fuzzy *avoid_dynamic* robot behaviour to model “traffic rules”

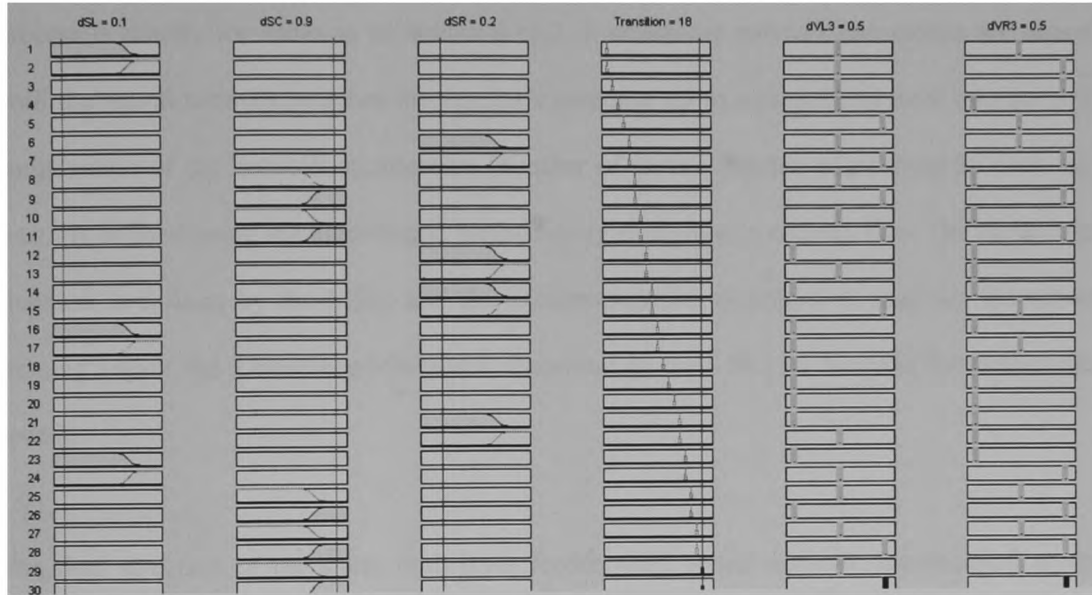


Figure 6-39 Values of δV_{LD}^3 and δV_{RD}^3 when rule 30 is activated

In the following section the *avoid_dynamic* robot behaviour is modelled using a hybrid stateflow-neural encoding. Both behavioural encoding methods (stateflow-fuzzy and stateflow-neural) are tested in chapter seven where the overall testing of the CAROS control architecture is presented.

6.14.3 Dynamic obstacle avoidance using hybrid stateflow-neural controller

This section describes the development of the *avoid_dynamic* robot behaviour using stateflow-neural networks instead of stateflow-fuzzy logic presented in section 6.14.2. The schematic layout of the hybrid stateflow-neural encoding is shown in Figure 6-31. The design

methodology followed in section 6.13.2 in which a fuzzy model was used to train a neural network is adopted in this section. Using the proposed method from Figure 6-29 the fuzzy model for dynamic obstacle avoidance derived in section 6.14.2 is used as a teacher to train the neural network to model the robot behaviour *avoid_dynamic*. During the training, the teacher (fuzzy model) and the neural network are both exposed to the training vector drawn from the environment, which consists of four input signals (dSL, dSC, dSR and T). The training process is exactly the same as in section 6.13.2 in which the network parameters are adjusted until the neural network matches the teacher's response up to a predefined error tolerance. The construction of the network architecture (number of layers, number of neurons on each layer, e.t.c.) is defined using the algorithmic methodology discussed in chapter five. The performance function is defined by the MSE, and the backpropagation technique is used for the network training (again the Levenberg-Marquardt algorithm is used also to increase the convergence speed).

The final structure of the static multilayer feedforward neural network incorporated in the *avoid_dynamic* robot behaviour is shown in Figure 6-40. It can be seen that the network has two hidden layers with 10 neurons in the first and six neurons in the second. In comparison with the neural network structure derived to model the *avoid_static* robot behaviour the network is considerably larger due to the increase in number of the input variable used (an extra input (T) has been added). However, using the design methodology from chapter five (initially small size network is selected and gradually increased) the second hidden layer is much smaller in size than the first hidden layer (only six neurons are used) providing advantage of better computational efficiency.

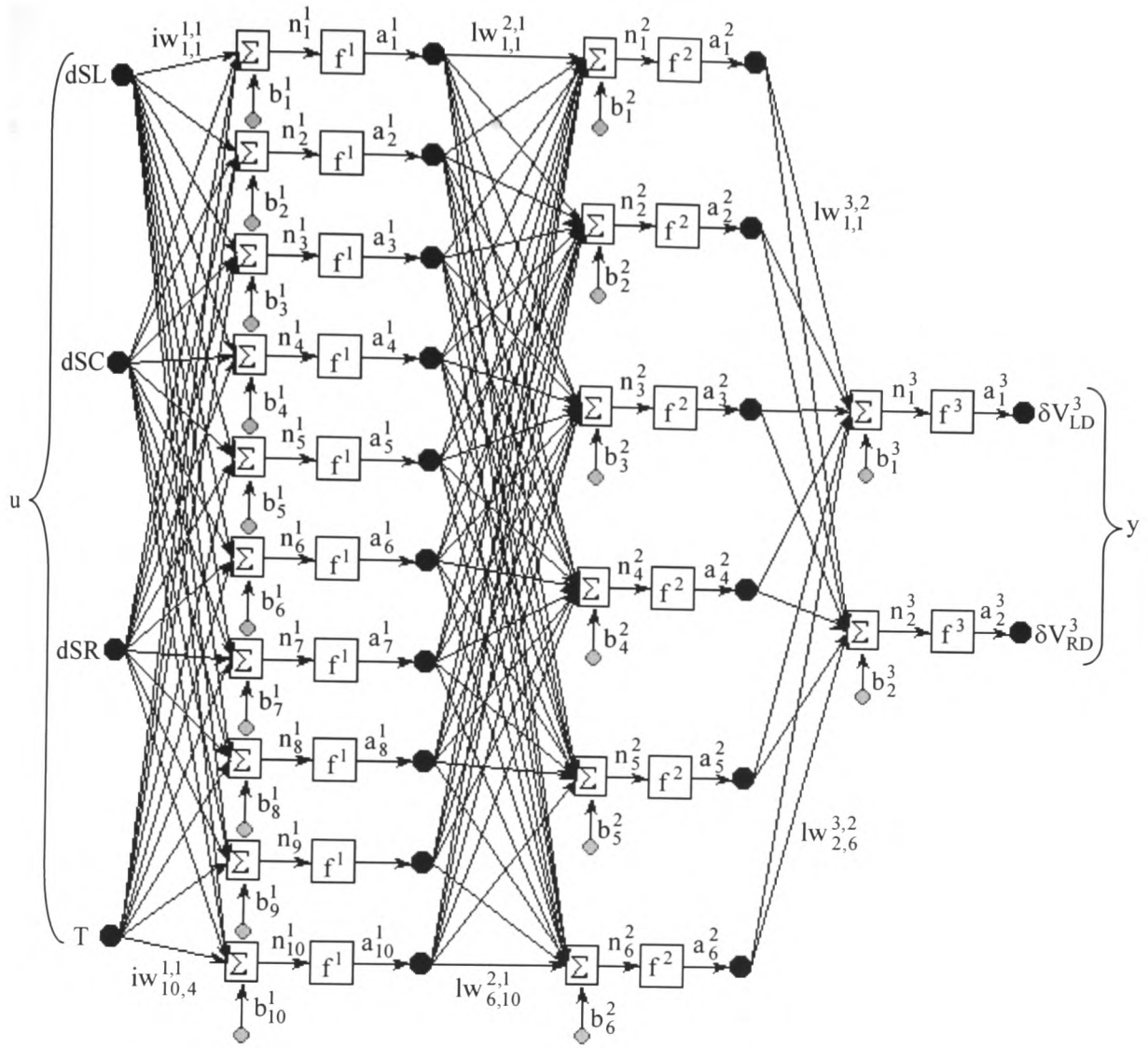


Figure 6-40 Structure of static NN used in hybrid stateflow-neural *avoid_dynamic* robot behaviour

The input and output to the network are column vectors as shown in Equations (6.23) and (6.24).

$$u = \begin{bmatrix} dSL \\ dSC \\ dSR \\ T \end{bmatrix} \quad (6.23)$$

$$y = \begin{bmatrix} \delta V_{LD}^3 \\ \delta V_{RD}^3 \end{bmatrix} \quad (6.24)$$

The weight matrix \mathbf{W} , a bias vector \mathbf{b} , and an output vector \mathbf{a} of each layer are given in Equations (6.25), (6.26), (6.27) and (6.28).

$$\mathbf{b}^1 = \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \\ b_4^1 \\ b_5^1 \\ b_6^1 \\ b_7^1 \\ b_8^1 \\ b_9^1 \\ b_{10}^1 \end{bmatrix} \quad \mathbf{b}^2 = \begin{bmatrix} b_1^2 \\ b_2^2 \\ b_3^2 \\ b_4^2 \\ b_5^2 \\ b_6^2 \end{bmatrix} \quad \mathbf{b}^3 = \begin{bmatrix} b_1^3 \\ b_2^3 \end{bmatrix} \quad (6.25)$$

$$\mathbf{IW}^{1,1} = \begin{bmatrix} iw_{1,1}^{1,1} & iw_{1,2}^{1,1} & iw_{1,3}^{1,1} & iw_{1,4}^{1,1} \\ iw_{2,1}^{1,1} & iw_{2,2}^{1,1} & iw_{2,3}^{1,1} & iw_{2,4}^{1,1} \\ iw_{3,1}^{1,1} & iw_{3,2}^{1,1} & iw_{3,3}^{1,1} & iw_{3,4}^{1,1} \\ iw_{4,1}^{1,1} & iw_{4,2}^{1,1} & iw_{4,3}^{1,1} & iw_{4,4}^{1,1} \\ iw_{5,1}^{1,1} & iw_{5,2}^{1,1} & iw_{5,3}^{1,1} & iw_{5,4}^{1,1} \\ iw_{6,1}^{1,1} & iw_{6,2}^{1,1} & iw_{6,3}^{1,1} & iw_{6,4}^{1,1} \\ iw_{7,1}^{1,1} & iw_{7,2}^{1,1} & iw_{7,3}^{1,1} & iw_{7,4}^{1,1} \\ iw_{8,1}^{1,1} & iw_{8,2}^{1,1} & iw_{8,3}^{1,1} & iw_{8,4}^{1,1} \\ iw_{9,1}^{1,1} & iw_{9,2}^{1,1} & iw_{9,3}^{1,1} & iw_{9,4}^{1,1} \\ iw_{10,1}^{1,1} & iw_{10,2}^{1,1} & iw_{10,3}^{1,1} & iw_{10,4}^{1,1} \end{bmatrix} \quad (6.26)$$

$$\mathbf{LW}^{2,1} = \begin{bmatrix} lw_{1,1}^{2,1} & lw_{1,2}^{2,1} & lw_{1,3}^{2,1} & lw_{1,4}^{2,1} & lw_{1,5}^{2,1} & lw_{1,6}^{2,1} & lw_{1,7}^{2,1} & lw_{1,8}^{2,1} & lw_{1,9}^{2,1} & lw_{1,10}^{2,1} \\ lw_{2,1}^{2,1} & lw_{2,2}^{2,1} & lw_{2,3}^{2,1} & lw_{2,4}^{2,1} & lw_{2,5}^{2,1} & lw_{2,6}^{2,1} & lw_{2,7}^{2,1} & lw_{2,8}^{2,1} & lw_{2,9}^{2,1} & lw_{2,10}^{2,1} \\ lw_{3,1}^{2,1} & lw_{3,2}^{2,1} & lw_{3,3}^{2,1} & lw_{3,4}^{2,1} & lw_{3,5}^{2,1} & lw_{3,6}^{2,1} & lw_{3,7}^{2,1} & lw_{3,8}^{2,1} & lw_{3,9}^{2,1} & lw_{3,10}^{2,1} \\ lw_{4,1}^{2,1} & lw_{4,2}^{2,1} & lw_{4,3}^{2,1} & lw_{4,4}^{2,1} & lw_{4,5}^{2,1} & lw_{4,6}^{2,1} & lw_{4,7}^{2,1} & lw_{4,8}^{2,1} & lw_{4,9}^{2,1} & lw_{4,10}^{2,1} \\ lw_{5,1}^{2,1} & lw_{5,2}^{2,1} & lw_{5,3}^{2,1} & lw_{5,4}^{2,1} & lw_{5,5}^{2,1} & lw_{5,6}^{2,1} & lw_{5,7}^{2,1} & lw_{5,8}^{2,1} & lw_{5,9}^{2,1} & lw_{5,10}^{2,1} \\ lw_{6,1}^{2,1} & lw_{6,2}^{2,1} & lw_{6,3}^{2,1} & lw_{6,4}^{2,1} & lw_{6,5}^{2,1} & lw_{6,6}^{2,1} & lw_{6,7}^{2,1} & lw_{6,8}^{2,1} & lw_{6,9}^{2,1} & lw_{6,10}^{2,1} \end{bmatrix} \quad (6.27)$$

$$LW^{3,2} = \begin{bmatrix} lw_{1,1}^{3,2} & lw_{1,2}^{3,2} & lw_{1,3}^{3,2} & lw_{1,4}^{3,2} & lw_{1,5}^{3,2} & lw_{1,6}^{3,2} \\ lw_{2,1}^{3,2} & lw_{2,2}^{3,2} & lw_{2,3}^{3,2} & lw_{2,4}^{3,2} & lw_{2,5}^{3,2} & lw_{2,6}^{3,2} \end{bmatrix} \quad (6.28)$$

The final output y of the *avoid_dynamic* robot behaviour using stateflow-neural encoding can be calculated as follows:

$$y = \text{purelin} \left(LW^{3,2} \text{tansig} \left(LW^{2,1} \text{tansig} \left(\underbrace{IW^{1,1}u + b^1}_{n^1} \right) + b^2 \right) + b^3 \right) \quad (6.29)$$

$\underbrace{\hspace{10em}}_{a^3}$
 $\underbrace{\hspace{5em}}_{a^2}$
 $\underbrace{\hspace{2em}}_{a^1}$
 $\underbrace{\hspace{2em}}_{n^1}$
 $\underbrace{\hspace{2em}}_{n^2}$
 $\underbrace{\hspace{2em}}_{n^3}$

6.15 Analysis of *avoid_trap* robot behaviour

The purpose of this kind of robot behaviour is to avoid possible trap situations for the mobile robot(s) during navigation. A trap situation is a common problem in single robot as well in multiple robot navigation. Many researchers such as (Lee and Wang, 1994), (Zhang *et al*, 1997), (Lin and Wang, 1997) and (Ng and Trivedi, 1998) have successfully attempted to detect, identify and solve possible trap situations. (Lin and Wang, 1997) classified all trap situations into three categories. The first trap situation may occur when the robot neither moves nor turns but just keeps waiting endlessly. This case can happen if the sensor readings of left, centre and right are equal (considering static obstacles). The robot can neither turn right nor left and eventually reduces its speed to zero. Consequently, the robot is trapped. The second trap situation can occur when the mobile robot has fallen into an infinity loop (robot performs the same motion iteratively). In the third trap situation, several robots meet and wait for each other to pass. Consequently, the robots are trapped. In this chapter the third trap situation is considered. The first and second trap situations are not considered as the *avoid_static* robot

behaviour overcomes successfully the first trap situation without any modification, whereas the second trap situation may only occur if the robot is navigating in the environment containing corridors (mainly indoor navigation). As previously mentioned the trap situation occurs when several robots meet and wait for each other to pass and there is a possibility for all robots to remain stationary (waiting for each other) for long time or even forever and also a potential danger of possible robot collision (robots are too close of each other). A schematic layout of the *avoid_trap* robot behaviour is shown in Figure 6-41. The behavioural encoding is fuzzy logic, which can be easily transformed into a neural network using the design methodology described in sections 6.13.2 and 6.14.3. As shown in Figure 6-41 the *avoid_trap* robot behaviour has three inputs (dSL, dSC and dSR) and two outputs (δV_{LD}^4 and δV_{RD}^4). As discussed previously the purpose of the *avoid_trap* robot behaviour is to provide the robot with a set of specified rules. Similar to *avoid_dynamic* robot behaviour linguistic terms such as “far” (F) and “near” (N) are defined for the left sensor (dSL), centre sensor (dSC) and right sensor (dSR). Terms such as “small” (S), “medium” (M) and “big” (B) are defined for the δV_{LD}^4 and δV_{RD}^4 variables in robot’s linear left and right velocities. The type of fuzzy model used to implement *avoid_trap* robot behaviour is Sugeno. The membership functions and parameters used to model the input and output variables are the same as in *avoid_dynamic* robot behaviour (Figure 6-37, Table 6-10 and Table 6-12). When the robot has detected moving obstacles at left, centre and right sides at the same time, then according to the linguistic terms “far” and “near” of the input variables a set of fuzzy rules shown in Table 6-14 is used to solve the possible trap situation.

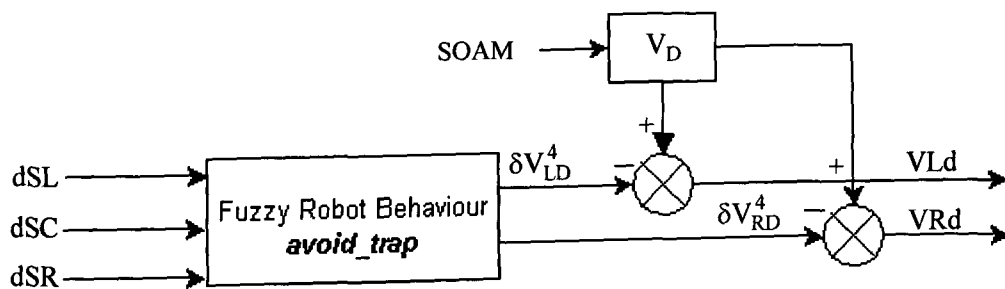


Figure 6-41 Schematic layout of *avoid_trap* robot behaviour

Rule No.	dSL	dSC	dSR	δV_{LD}^4	δV_{RD}^4	Action
1	Far	Far	Far	Medium	Big	Slow down and turn right
2	Near	Near	Near	Small	Big	Turn right fast

Table 6-14 List of fuzzy rules used in *avoid_trap* robot behaviour

As can be seen from Table 6-14 only two fuzzy rules are used for the *avoid_trap* robot behaviour. The first rule will be activated if moving objects have been detected around and far from the robot. The robot in this case will reduce its speed and it will start to turn to the right. The second rule will be activated if moving objects have been detected around and close to the robot. The robot in this case will instantly turn to the right (sharp turn). Note that the control output of this behaviour will only be released if the action co-ordinator mechanism (described in the section) has found the moving objects at the same distance from the robot. As the *avoid_trap* robot behaviour was modelled using only two fuzzy rules the development of neural network encoding of this type of behaviour is not unnecessary. However, when the trap behaviour is complex the design methodology used in sections 6.13 and 6.14 should be followed.

6.16 Co-ordination and parallel switching of robot behaviours using stateflow

In section 6.11 it was shown that the implementation of complex behaviour generation for artificial systems can be overcome by decomposing the global task(s) into simpler, well-specified behaviours which are easier to design and can be tuned independently of each other. The behaviours developed in this thesis were implemented as a set of fuzzy rules, as models of neural networks and also as hybrid solutions integrating fuzzy, neural and FSM for efficient integration of planning and reactive control. In this section the proposed action co-ordinator mechanism (see Figure 6-3 for overview of CAROS) used in CAROS control architecture is presented with a focus on, and attention to the design, co-ordination and action selection of the elementary behaviours.

Figure 6-42 shows a schematic layout of the action co-ordinator mechanism (ACM) used in CAROS. Its main components include the non-linear switch mechanism and the global state identification mechanism. The task of the ACM is to decide *when* and *how* to switch between the set of behaviours. The operation of the ACM is described as follows: ACM and behaviours are sharing the same type and number of inputs generated from the robot's working environment. Suppose that both (ACM and behaviours) are exposed to a given vector (sensory data) from the robot's environment. The global state identification mechanism (GSIM) within the ACM makes a decision as to which robot behaviours need to be activated according to the current state of the system. In particular, as shown in Figure 6-43, the GSIM is implemented by means of FSM. It utilises a stateflow model based on a Mealy FSM approach. As can be seen from Figure 6-43 the GSIM consists of four global states representing the possible robot states during its mission (note that the number of states depends on particular navigation task or control problem).

At the beginning (initialisation) of the robot navigation task the GSIM is automatically located at state one ("No obstacles"). Then the system can jump to any of the three remaining states according to the specified condition of the transitions illustrated in Table 6-15. For instance, if during the navigation the path of the robot is obstructed by static obstacles then state two is active ("Avoid static"). If the path of the robot is obstructed by dynamic obstacles (other robots) then state three ("Avoid dynamic") is activated. State four ("Avoid trap") will be active if three neighbour robots are found to be around (left, front and right) the robot within the same distance. If the robot has detected both static and dynamic obstacles then priority is given to the closest object. When all conflicts have been resolved the robot's global state re-enters state one ("No obstacle"). The task of the GSIM is to solve possible conflicts during the robot navigation task, to identify the robot's global state and to nominate which behaviour (or local controller) should be active at a particular time. The mechanism in which is responsible to release the output of the nominated behaviour from the GSIM is the non-linear switch mechanism (NSM).

The NSM has as inputs the control outputs of the behaviours and also the output of the GSIM (see Figure 6-43). The output of the NSM, and therefore the output of the ACM, is a vector carrying the control output of a specific behaviour.

This action co-ordination draws its advantages from the competitive tasks architecture in which once a functional module is selected the module is activated and the previous one deactivated. However, the deactivated module (behaviour) is not totally switched off, it continues to receive sensory data in parallel with the ACM. Thus, the behaviour exercises a monitoring activity, which consists in receiving data from sensors and calculating the values of the selection parameters, which depend on it.

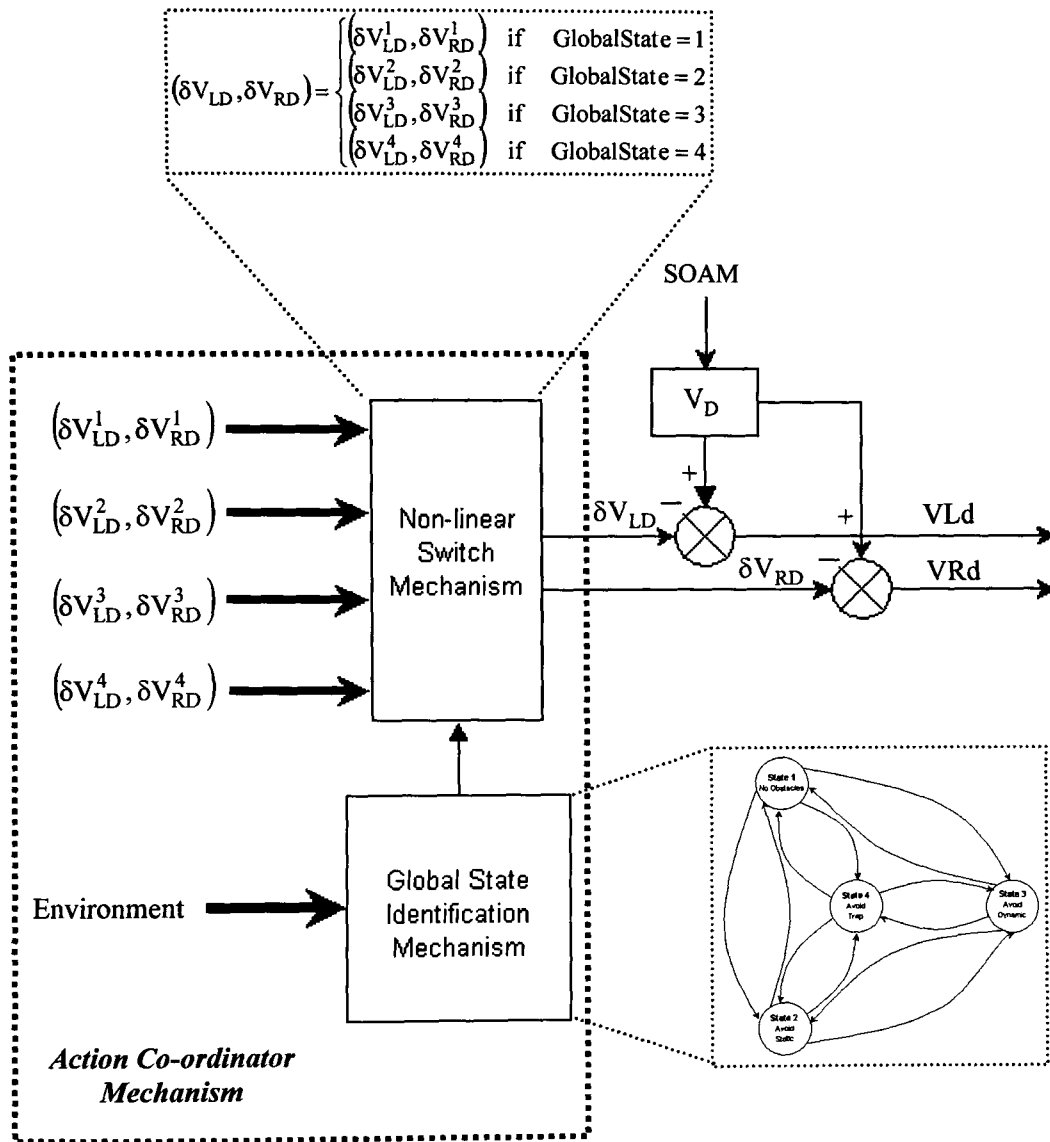


Figure 6-42 Schematic layout of the action co-ordinator mechanism

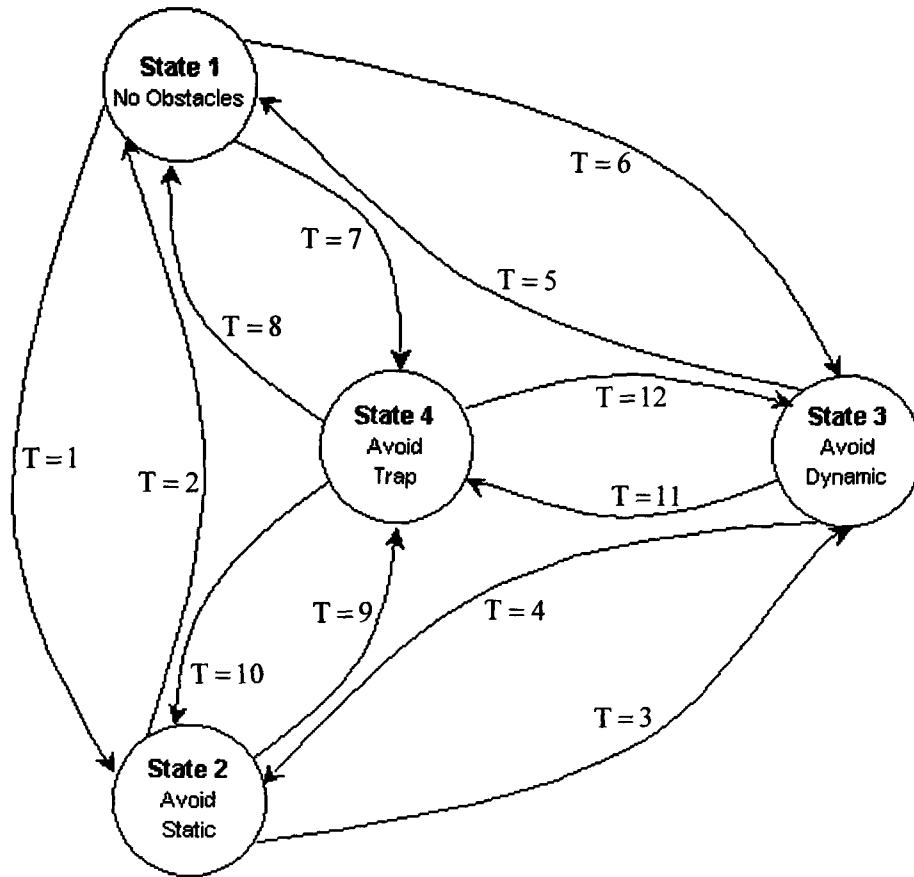


Figure 6-43 Global state identification mechanism

T	Condition
1	$\max(\max(dSL, dSC), dSR) < \max(\max(sSL, sSC), sSR)$
2	$(\max(\max(dSL, dSC), dSR) = 0) \& (\max(\max(sSL, sSC), sSR) = 0)$
3	$\max(\max(dSL, dSC), dSR) \geq \max(\max(sSL, sSC), sSR) \& (\max(\max(sSL, sSC), sSR) > 0)$
4	$\max(\max(dSL, dSC), dSR) < \max(\max(sSL, sSC), sSR)$
5	$(\max(\max(dSL, dSC), dSR) = 0) \& (\max(\max(sSL, sSC), sSR) = 0)$
6	$(\max(\max(dSL, dSC), dSR) \geq \max(\max(sSL, sSC), sSR)) \& (\max(\max(sSL, sSC), sSR) > 0)$
7	$(dSL = dSC = dSR)$
8	$(\max(\max(dSL, dSC), dSR) = 0) \& (\max(\max(sSL, sSC), sSR) = 0)$
9	$(dSL = dSC = dSR)$
10	$\max(\max(dSL, dSC), dSR) < \max(\max(sSL, sSC), sSR)$
11	$(dSL = dSC = dSR)$
12	$(\max(\max(dSL, dSC), dSR) \geq \max(\max(sSL, sSC), sSR)) \& (\max(\max(sSL, sSC), sSR) > 0)$

Table 6-15 Conditions of the transitions used in global state identification mechanism

6.17 Discussion

This chapter presents the development of a novel hybrid multi-agent based control architecture called CAROS (Co-operative Autonomous RObotic Systems) for navigation of multiple autonomous mobile robots in unknown static and/or dynamic environments. The proposed architecture takes the advantages of various control structure types thereby integrating them in a way that results in an overall increase in synergy (This aspect is demonstrated in the next chapter). This section contains discussions regarding the overall structure of the proposed control architecture and also comparison with other control architectures in terms of specific architecture characteristics discussed in previous chapters.

The design and implementation of complex control systems for autonomous mobile robots is a difficult task and still continues to challenge researchers. This challenge lies in the development of robust, flexible and modular control systems that are capable of coping with the dynamics of the real world. The new approach proposed in this thesis is a hybrid control system that takes the advantages of various control structures. In particular, the proposed control architecture draws its design from the competitive tasks architecture, the production rules architecture, the connectionist architecture, the dynamic system architecture, the multi-agent architecture and the subsumption architecture. The ongoing research for the identification of what are the “best” characteristics of each control approach in terms of requirements/properties is still an open question. In section 6.2, (see Figure 6-1), an analysis is presented based on the Quality Functional Deployment tool (QFD) for the identification of the relationship of the requirements/properties of control architecture versus the control architecture specifications. The analysis was based on fourteen requirements when design of control architecture is considered and the analysis presented clearly shows that the solution to the design and implementation of complex control systems for autonomous mobile robots is hybrid. However, the sufficient number of requirements/properties when this type of analysis is taking place is a question of interest. On the other hand the control architecture specifications taken place in the

analysis is also a question of interest. For instance, is the number of specifications such as reasoning, processing, control and integration method of the control architecture being considered sufficient for the analysis, in order the control architecture to achieve its best performance? Nevertheless the answer is not straightforward as other specifications may be added due to the dramatically progress in control engineering community.

Table 6-16 shows a comparison between the proposed control architecture and several other control structures developed in the past decade for navigation of autonomous mobile robots. The comparison demonstrates that the CAROS control architecture offers several advantages over the others. The control architecture has been designed for both single and multiple robot navigation in an unknown static and/or dynamic environment. The reasoning of the control architecture is both deliberative and reactive. Reactive reasoning is necessary so that the vehicle can navigate safely and can take actions in real-time. The reactive behaviours are in charge of the robot's specific domain. They are modelled using fuzzy, neural and hybrid behavioural encoding. They process the information from the sensors in order to apply primitive reactions such as obstacle avoidance behaviour. The deliberative system comprising finite state machines is responsible for high-level planning and for solving conflicts among behaviours that try to access the robot's actuators at the same time. Therefore the integration method among behaviours is selection. The processing can be achieved in a centralised and decentralised manner using the controller-agent concept from the field of multi-agent systems. The hierarchical and behavioural-based control structure is used, as both offer significant advantages. For example, hierarchy of behaviours offers an efficient approach to synthesis of behavioural capabilities necessary for autonomous navigational tasks. Its practical utility lies in the hierarchical decomposition of overall behaviour into sub-systems that are activated only when needed. The hybrid approach presented in this thesis provides a suitable framework for situated adaptation in single and multiple robot navigation.

6.18 Summary

This chapter presents the development of a proposed novel hybrid multi-agent based control architecture called CAROS for navigation of multiple autonomous mobile robots. The proposed architecture was designed for single/multiple robot navigation in both static and dynamic unknown environments. The need for hybrid solutions in design of complex control systems was demonstrated through an analysis using the Quality Function Deployment (QFT) tool. The analysis has clearly indicated that the most promising solution in control systems architecture design is the hybrid approach. At the beginning of the chapter an overview of the proposed architecture was presented highlighting its main characteristics. The proposed control architecture takes the advantages of various control structure types. In particular, the control architecture takes its design from competitive tasks architecture, production rules architecture, connectionist architecture, dynamic system architecture, multi-agent architecture and subsumption architecture.

The main objective is to achieve successful navigation of both single and multiple mobile robots in an environment populated by stationary and moving objects. In order to achieve this goal, the design of the architecture was based on a number of requirements/properties such as modularity, robustness, fault tolerance, distribution, reactivity, adaptability, planning, co-operation, easy of application, uncertainty, optimal control, learning and efficiency (see section 6.2). The final configuration of the control architecture utilises reactive, deliberative, distributed and centralised control approaches and uses artificial intelligence as well as modular hierarchical structure. Multi-agent systems take care of the distributed control utilising the controller-agent concept, which is relatively new field in control engineering. Controller-agents are operating in some restricted part of the operating regime producing or solving problems in terms of request from a well-defined supervisor (Co-ordination object). Centralised processing and deliberative reasoning is tackled using supervisory techniques incorporating finite state machines and non-linear switching mechanisms.

The implementation of the main robot behaviours was achieved by decomposing the global tasks into simpler well-defined behaviours. The behavioural encoding is based on fuzzy logic, neural networks and finite state machines design methodologies. An algorithm methodology was proposed for supervised learning of neural network using fuzzy logic as a teacher. As real-time motion planning in an unknown environment involves collision avoidance of static as well of moving robots a novel scheme was presented for identification of the direction of moving robots. Sensor modelling and sensor sensitivity was also considered in order to form modular reactive layer (different sensors have been modelled). Prior to the testing of the architecture in the next chapter a comparison was made with other recent approaches based on control architecture specifications such as reasoning, control, processing, integration, behavioural encoding, robot navigation and operating environment (see chapter discussions). The proposed architecture has been shown to satisfy the most of the essentials specifications (see Table 6-16).

The contributions of this chapter are: Development of a novel hybrid multi-agent based control architecture called CAROS for navigation of multiple autonomous mobile robots in an unknown environment. Identification of the most important requirements/properties of control architecture versus the main control architecture specifications using the Quality Function Deployment (QFT) tool. Comparison of the proposed architecture with recent approaches. An algorithmic methodology for supervised learning of neural network architecture using fuzzy logic as a teacher for behavioural encoding. Behavioural encoding using hybrid solutions (fuzzy/stateflow and neural/stateflow) for robot navigation. Novel approach for identification of direction of moving neighbour robots using finite state machines. Modular approach of modelling three types of sensor and sensor sensitivity.

The next chapter presents the results of testing of the proposed control architecture for navigation of single and multiple mobile robots based on the full non-linear model of the mobile robot derived in chapter four and the different speed control laws derived in chapter five. The

control strategy and architecture presented in this chapter for mobile robot navigation is validated through simulation studies. In particular, single and multiple robot navigation in both static and/or dynamic environment is presented.

6.19 References

ABDELHAMEED, M. M. 1999. Adaptive Neural Network Based Controller for Robots. *Mechatronics*, **9** (2), pp. 147-162.

AGUIRRE, E. and GONZALEZ, A. 2000. Fuzzy Behaviors for Mobile Robot Navigation: Design, Coordination and Fusion. *International Journal of Approximate Reasoning*, **25** (3), pp. 255-289.

ARKIN, R. C. 1989. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*, **8** (4), pp. 92-112.

ARKIN, R. C. 1998. *Behaviour-Based Robotics*. USA: MIT Press. 0-262-01165-4.

ARKIN, R. C. and BALCH, T. 1998. Co-operative Multiagent Robotic Systems. In: eds. D. KORTENKAMP, R. P. BONASSO, and R. MURPHY. ed. *Artificial Intelligence and Mobile Robots. Case Studies of Successful Robot Systems*. USA: AAAI Press/The MIT Press, pp. 277-296.

ARONOV, B., DE BERG, M., VAN DER STAPPEN, A. E., SVESTKA, P. and VLEUGELS, J. 1999. Motion Planning for Multiple Robots. *Discrete & Computational Geometry*, **22** (4), pp. 505-525.

ASTOLFI, A. 1999. Exponential Stabilization of a Wheeled Mobile Robot Via Discontinuous Control. *Journal of Dynamic Systems Measurement and Control-Transactions of the ASME*, **121** (1), pp. 121-126.

BOEL, R. K., DE SCHUTTER, B., NIJSSE, G., SCHUMACHER, J. M. and VAN, S. J. H. 1999. Approaches to Modelling, Analysis and Control of Hybrid Systems. *Journal of Robotic Systems*, **40** (4), pp. 16-27.

BOSSERT, J. L. 1991. *Quality Function Deployment*. New York: Marcel Dekker, Inc. 0-8247-8378-6.

BRAAE, M. and RUTHERFORD, D. A. 1978. Fuzzy Relations in a Control Setting. *Kybernets*, **7** (3), pp. 185-188.

BROOKS, R. A. 1986. A Robust Layered control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, **RA-2** (1), pp. 14-23.

BROOKS, R. A. 1990. *Challenges for Complete Creature Architectures*. 1st International Conference on Simulation of Adaptive Behaviour. pp. 434-443, MIT Press.

BRUMITT, B., STENTZ, A. and HEBERT, M. 2002. Autonomous Driving With Concurrent Goals and Multiple Vehicles: Experiments and Mobility Components. *Autonomous Robots*, **12** (2), pp. 135-156.

- CANUDAS DE WITT, C. and SORDALEN, O. J. 1992. Exponential Stabilization of Mobile Robots with Nonholonomic Constraints. *IEEE Transactions on Automation and Control*, **37** (11), pp. 1791-1797.
- CHATTERJEE, R. and MATSUNO, F. 2001. Use of Single Side Reflex for Autonomous Navigation of Mobile Robots in Unknown Environments. *Robotics and Autonomous Systems*, **35** (2), pp. 77-96.
- CHUNG, J., RYU, B. S. and YANG, H. S. 1998. Integrated Control Architecture Based on Behavior and Plan for Mobile Robot Navigation. *Robotica*, **16**, pp. 387-399.
- CONTICELLI, F., BICCHI, A. and BALESTRINO, A. 2000. *Observability and Nonlinear Observers for Mobile Robot Localization*. 6th IFAC Symposium on Robot Control. pp. 115-120, Vienna, Austria.
- DIAMANTOPOULOS, A., ROBERTS, G. N. and HARWOOD, D. J. 2000. *Robot Motion Planning in Environments Populated by Shrinking and Expanding Obstacles*. EUREL, European Advanced Robotics Systems, Masterclass and Conference - Robotics 2000 .
- EREN, H. and FUNG, C. C. 1997. Position Estimation of Mobile Robots Based on Coded Infrared Signal Transmission. *IEEE Transactions on Instrumentation and Measurement*, **46** (6), pp. 1280-1283.
- EVERETT, H. R. 1995. *Sensors for Mobile Robots: Theory and Applications*. USA: A K Peters, Ltd. 1-56881-048-2.
- FUJIMORI, A., TERAMOTO, M., NIKIFORUK, P. N. and GUPTA, M. M. 2000. Cooperative Collision Avoidance Between Multiple Mobile Robots. *Journal of Robotic Systems*, **17** (7), pp. 347-363.
- GAT, E. 1992. *Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robot*. Proceedings of the National Conference on Artificial Intelligence. pp. 809-815, Menlo Park, CA.
- GOODRIDGE, S. G., KAY, M. G. and LUO, R. C. 1996. Multilayered Fuzzy Behavior Fusion for Real-Time Reactive Control of Systems With Multiple Sensors. *IEEE Transactions on Industrial Electronics*, **43** (3), pp. 387-394.
- HOU, M. and MULLER, P. C. 1992. Design of Observers for Linear Systems with Unknown Inputs. *IEEE Transaction on Automation and Control*, **37** (6), pp. 871-875.
- HWANG, K. S., JU, M. Y. and HSU, S. P. 1999. Polymorphic Autonomous Architecture for Mobile Robot Navigation. *Journal of Information Science and Engineering*, **15** (5), pp. 737-761.
- JETTO, L., LONGHI, S. and VITALI, D. 1999. Localization of a Wheeled Mobile Robot by Sensor Data Fusion Based on a Fuzzy Logic Adapted Kalman Filter. *Control Engineering Practice*, **7** (6), pp. 763-771.
- KURZ, A. 1996. Constructing Maps for Mobile Robot Navigation Based on Ultrasonic Range Data. *IEEE Transactions on Systems Man and Cybernetics Part B- Cybernetics*, **26** (2), pp. 233-242.

- LEE, E. T. 1995. Applying Fuzzy-Logic to Robot Navigation. *Kybernetes*, **24** (6), pp. 38-43.
- LEE, P. S. and WANG, L. L. 1994. Collision-Avoidance by Fuzzy-Logic Control for Automated Guided Vehicle Navigation. *Journal of Robotic Systems*, **11** (8), pp. 743-760.
- LIN, C. H. and WANG, L. L. 1997. Intelligent Collision Avoidance by Fuzzy Logic Control. *Robotics and Autonomous Systems*, **20** (1), pp. 61-83.
- MAAREF, H. and BARRET, C. 1995. *Fuzzy Help in Mobile Robot Navigation*. IEEE Int. Conf. on Robotics & Automation. pp. 865-871,
- MALONE, T. W. and CROWSTON, K. 1994. The Interdisciplinary Study of Co-ordination. *ACM Computing Surveys*, **26** (1), pp. 87-119.
- MAMDANI, E. H. and ASSILEAN, S. 1975. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Machine Studies*, **7** (1), pp. 1-13.
- MATARIC, M. J. 1992. Integration of Representation Into Goal-Driven Behavior-Based Robots. *IEEE Transactions on Robotics and Automation*, **8** (3), pp. 304-312.
- MIN, B. K., CHO, D. W., LEE, S. J. and PARK, Y. P. 1997. Sonar Mapping of a Mobile Robot Considering Position Uncertainty . *Robotics and Computer-Integrated Manufacturing*, **13** (1), pp. 41-49.
- MUTAMBARA, A. G. O. and DURRANT-WHYTE, H. F. 2000. Estimation and Control for a Modular Wheeled Mobile Robot. *Ieee Transactions on Control Systems Technology*, **8** (1), pp. 35-46.
- NEHMZOW, U. 1999. *Mobile Robotics: A Practical Introduction*. UK: Springer-Verlag London Ltd. 1852331739.
- NG, K. C. and TRIVEDI, M. M. 1998. A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multirobot Convoying. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, **28** (6), pp. 829-840.
- NOREILS, F. R. 1993. Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment. *The Int. Journal of Robotics Research*, **12** (1), pp. 79-98.
- NOVAKOVIC, B., SCAP, D. and NOVAKOVIC, D. 2000. An Analytic Approach to Fuzzy Robot Control Synthesis. *Engineering Applications of Artificial Intelligence*, **13** (1), pp. 71-83.
- PARKER, L. E. 1998. ALLIANCE: An Architecture for fault Tolerant Multirobot Cooperation. *IEEE Transactions on Robotics and Automation*, **14** (2), pp. 220-240.
- ROUMELIOTIS, S. I. and BEKEY, G. A. 1997. *An Extended Kalman Filter for Frequent Local and Infrequent Global Sensor Data Fusion*. Proceedings of SPIE (The International Society of Optical Engineering). Sensor Fusion and Decentralised Control in Autonomous Robotics Systems. pp. 11-22, Pittsburgh, Pennsylvania.
- SASIADEK, J. Z. and HARTANA, P. 2000. *Odometry and Sonar Data Fusion for Mobile Robot Navigation*. 6th IFAC Symposium on Robot Control. pp. 531-536, Vienna, Austria.

- SGOUROS, N. M. 2001. Qualitative Navigation for Mobile Robots in Indoor Environments. *Applied Artificial Intelligence*, **15** (3), pp. 237-251.
- SONG, K. T. and CHANG, C. C. 1999. Navigation Integration of a Mobile Robot in Dynamic Environments. *Journal of Robotic Systems*, **16** (7), pp. 387-404.
- SVESTKA, P. and OVERMARS, M. H. 1998. Coordinated Path Planning for Multiple Robots. *Robotics and Autonomous Systems*, **23** (3), pp. 125-152.
- TAKAGI, T. and SUGENO, M. 1985. Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics*, **15** (1), pp. 116-132.
- TZAFESTAS, S. G. and STAMOU, G. B. 1994. *A Fuzzy Path Planning Algorithm for Autonomous Robots Moving in an Unknown Environment*. EURISCON 2nd European Robotics Intelligent Systems & Control Conf. pp. 140-149,
- VAN BREEMEN, A. J. N. 2001. *Agent-Based Multi-Controller Systems: A Design Framework for Complex Control Problems*. Ph.D. Thesis. Twente University Press.
- VAN BREEMEN, A. J. N. and DE VRIES, T. J. A. 2000. *An Agent-based Framework for Designing Multi-controller Systems*. Proceedings of the Fifth International Conference on the Practical Applications of Intelligent Agents and Multi-agent Technology. pp. 219-235, Manchester, UK.
- VAN DEN BOSCH, P. P. J. and HEEMELS, M. 1999. Hybrid Systems: Modelling Embedded Controllers. *Journal of Robotic Systems*, **40** (4), pp. 28-34.
- WU, S. F., MEI, J. S. and NIU, P. Y. 2001. Path Guidance and Control of a Guided Wheeled Mobile Robot. *Control Engineering Practice*, **9** (1), pp. 97-105.
- XU, H. and VAN BRUSSEL, H. 1997. A Behaviour-Based Blackboard Architecture for Reactive and Efficient Task Execution of an Autonomous Robot. *Robotics and Autonomous Systems*, **22** (2), pp. 115-132.
- YANG, S. X. and MENG, M. 2000. An Efficient Neural Network Approach to Dynamic Robot Motion Planning. *Neural Networks*, **13** (2), pp. 143-148.
- YAVUZ, H. and BRADSHAW, A. 2002. A New Conceptual Approach to the Design of Hybrid Control Architecture for Autonomous Mobile Robots. *Journal of Intelligent & Robotic Systems*, **34** (1), pp. 1-26.
- ZALAMA, E., GOMEZ, J., PAUL, M. and PERAN, J. R. 2002. Adaptive Behavior Navigation of a Mobile Robot. *IEEE Transactions on Systems Man and Cybernetics Part a-Systems and Humans*, **32** (1), pp. 160-169.
- ZHANG, M. L., PENG, S. X. and MENG, Q. H. 1997. Neural Network and Fuzzy Logic Techniques Based Collision Avoidance for a Mobile Robot. *Robotica*, **15**, pp. 627-632 .

7

Evaluation of the Control Architecture

7.1 Introduction

Chapter six introduced a novel hybrid multi-agent based control architecture called CAROS. The control architecture has been implemented in MATLAB/Simulink using the full non-linear model of the MIABOT V2 mobile robot, derived and described in chapter four and including its speed control laws, which were derived in chapter five. This chapter investigates the validity of the control architecture when applied to the problem of navigation of single/multiple autonomous mobile robots. Simulation results are presented to show the effectiveness of the proposed strategy for navigation and obstacle avoidance in both single and multiple robot navigation considering static and dynamic environments. The algorithmic methodology, regarding modelling of sensor sensitivity and modelling and identification of local controllers/behaviours, presented in chapters five and six, is also validated based on a number of simulations of mobile robot navigation tasks incorporating different sensors and local controllers for each of the robot behaviours described in chapter six.

The remainder of this chapter is organised as follows: Section 7.2 describes the simulation setup and the main features of the simulator. The simulation results are divided into three parts as shown in section 7.3. In particular Part I (section 7.3.1) illustrates results of single mobile robot navigation using the CAROS control architecture whereas sections 7.3.2 and 7.3.3 illustrate results involving two to five mobile robots. A discussion based on results presented in this chapter follows in section 7.4. Section 7.5 presents the chapter's summary.

7.2 Simulation setup

The simulator developed for the proposed control architecture has been implemented in MATLAB 6.0 on a 1300 MHz PC under Windows NT/98 environment. The simulation of Simulink models involved the numerical integration of sets of ordinary differential equations (ODEs). Simulink provides a number of solvers for the simulation of such equations. Due to the diversity of the dynamic system behaviour, some solvers may be more efficient than others at solving a particular problem. The solver used in CAROS simulator incorporates fixed-step of 0.05 using the fourth-order Runge-Kutta formula (ode4). In order to speed up the execution of the simulator, all Simulink models have been compiled into a C code. The computations are therefore faster, with a 2X-to-6X increase in performance, for all models that use built-in Simulink blocks. The workspace utilised in the simulator is represented by two-dimensional grid map with 73x73 pixels (3.6m x 3.6m environment). The height and width of each grid is 5cm (one pixel). Throughout the simulations the units for time is seconds (sec), for distance is meter (m) and for velocity is m/sec.

7.3 Division of results

To examine the performance of the control architecture and the navigation strategy described in chapter six, a set of simulating experiments were devised and organised into three main parts. Each part is divided into a group of results as shown in Table 7-1. A brief discussion highlighting the purpose of each part is given in the following.

Division of Results			
	Part I	Part II	Part III
GROUP	A1	B1	C1
	A2	B2	C2
	A4	-	C3
	A4	-	C4
	-	-	C5
	-	-	C6

Table 7-1 Division of results

Part I investigates the effectiveness of the proposed CAROS control architecture with a single mobile robot operating in an unknown environment populated by static obstacles (the first simulations consider the case of robot navigation without obstacles). In particular issues such as, demonstrating system performance, independencies amongst integrated subsystems, implementation of global and local control strategies, static obstacle avoidance strategy, travel time due to the different controller configurations and effectiveness of different sensor sensitivity utilisation will be addressed.

Part II investigates the effectiveness of CAROS control architecture in navigation tasks involving two mobile robots operating in an unknown static and dynamic environment. The purpose of these tests is to demonstrate that the control strategy chosen for navigation of multiple mobile robots is efficient and also observes the robustness of the designed control system architecture against desired requirements, such as, supervision, decision-making and co-ordination of internal control structures.

Part III investigates the effectiveness of CAROS control architecture in more complex navigation tasks, involving multiple mobile robots (results are presented for up to five robots) operating in a restricted workspace. These results show mobile robots are able to achieve

independent targets avoiding a large number of static obstacles in the workspace, as well as other mobile robots.

7.3.1 Part I: Results

Simulation groupings for Part I, are given in Table 7-2. In the next sections results of each group are presented followed by a summary of the performance of the proposed system.

Part I: Division of Results	
Group	Representation
A1	Single robot navigation with observations based on various control signals (no obstacles)
A2	Single robot navigation with observations based on the robot's trajectory (no obstacles)
A3	Single robot navigation with observations based on sensor sensitivity (static obstacles)
A4	Single robot navigation with observations based on static obstacle avoidance behaviour (static obstacles)

Table 7-2 Division of results (Part I)

7.3.1.1 Group A1: Results

Results presented in Group A1 concentrate on single robot navigation with observations based on various control signals (i.e. desired versus actual robot velocity, desired versus actual robot orientation etc.) in an environment free from both static and dynamic obstacles. In particular the effectiveness of the controller-agent concept used in chapter six for implementation of local control strategy is validated through several tests.

Figure 7-1 to Figure 7-3 depict navigation tasks in which the mobile robot has to reach a target point (x_T, y_T) marked by T from an initial position (x_0, y_0) marked by S . In total 9 experiments were considered as shown in Table 7-3. In particular the results of each experiment are presented using two graphical plots. Figure 7-1(a) for instance depicts on the left hand side the main plot illustrating the robot's trajectory (x_S, y_S) , initial point, target point, virtual points one and two (VP1, VP2) and the predefined region (ϵ) . On the right hand side of Figure 7-1(a)

are six smaller graphs illustrating (from right to left) a comparison of robot's left and right desired velocity (V_{Ld}, V_{Rd}) versus actual left and right velocity (V_L, V_R), the robot's actual velocity (V_S) versus the robot's desired velocity (V_D), the robot's desired orientation (ϑ_D), actual orientation ϑ_S and orientation at target point ϑ_T , the robot's actual left and right motor voltage V_r, V_l , and the internal state of the robot as a result of the co-ordinated work by the controller-agents. The topside of all sub-plots shows the global state of the robot.

Group A1				
Exp	(x_0, y_0)	(x_T, y_T)	ϑ_T	Caption
1	(0,0)	(2,2)	0°	Figure 7-1(a)
2	(0,0)	(2,2)	45°	Figure 7-1(b)
3	(0,0)	(2,2)	90°	Figure 7-1(c)
4	(0,0)	(2,2)	135°	Figure 7-2(a)
5	(0,0)	(2,2)	180°	Figure 7-2(b)
6	(0,0)	(2,2)	-135°	Figure 7-2(c)
7	(0,0)	(2,2)	-90°	Figure 7-3(a)
8	(0,0)	(2,2)	-45°	Figure 7-3(b)
9	(0,1)	(2,1)	180°	Figure 7-3(c)

Table 7-3 Initial, target co-ordinates and desirable target angle for the mobile robot

From Figure 7-1 to Figure 7-3 it can be observed that the mobile robot reaches the control objective in every case despite the different requirement of orientation ϑ_T at target point. When all controller-agents are activated within their operational regime, the transition between the controller-agents is smooth and the robot adapts its behaviour according to the current internal state. The robot's trajectory is smooth and free from oscillation. The control effort (motor voltage) is within the design requirements, and in all cases the motor's input voltages do not exceed 6V. The comparison of robot's desired versus actual velocity reinforce the robustness and stability performance of the co-ordination of the controller-agents. The same conclusion can be drawn from observations on desired versus actual orientation and orientation at target point achieved by the mobile robot. The internal robot's state demonstrates that the sequencing and

co-ordination among controller-agents emerges from the interaction of the SOAM with the environment. Thus a controller-agent is activated when its context becomes true and deactivated when its context becomes false. The transition is smooth and does not affect robot dynamics and therefore the robot's trajectory remains smooth during the navigation task. At the lower level the speed control law used is PI. Results show that the control law developed in chapter five successfully controls the robot's speed despite the transition and interaction among several controller-agents. However in order to establish the performance of each speed controller developed in chapter five (PI, fuzzy and neural) during a navigation task, experiments 1 to 4 from Table 7-3 were repeated with observations based on ISE, IAE and ITAE performance criteria as shown in Table 7-4.

Performance criteria of PI, fuzzy and neural speed controller for left and right wheel									
<i>Performance criteria of PI speed controller</i>			<i>Performance criteria of Fuzzy Logic speed controller</i>			<i>Performance criteria of Neural Networks speed controller</i>			
Exp	ISE	IAE	ITAE	ISE	IAE	ITAE	ISE	IAE	ITAE
1	0.005475	0.06187	0.1845	0.005528	0.06108	0.1842	0.005549	0.0686	0.2304
	0.01707	0.07773	0.1847	0.0173	0.07641	0.1833	0.01738	0.08694	0.2019
2	0.007247	0.08572	0.3632	0.007343	0.08474	0.3622	0.007374	0.09188	0.4112
	0.01854	0.09319	0.2963	0.01881	0.09228	0.2971	0.01888	0.1033	0.3164
3	0.007682	0.09875	0.5263	0.008582	0.1069	0.6006	0.008493	0.1137	0.6547
	0.02017	0.126	0.6349	0.01948	0.1115	0.5043	0.0196	0.1224	0.519
4	0.00692	0.0763	0.2855	0.007002	0.07575	0.2865	0.007036	0.08343	0.3383
	0.01874	0.1005	0.3505	0.01902	0.09919	0.3499	0.01914	0.1105	0.3706

Table 7-4 Comparison of performance criteria between three different speed controllers

From Table 7-4 it can be seen that PI and fuzzy speed controllers have a marginal advantage against the neural controller (red shaded numbers show the best performance). In particular the PI controller has the smallest ISE error of all controllers, whereas the fuzzy controller has the smallest IAE and ITAE error. However, as the difference between the errors is less than 1% the PI speed controller will be used for the rest of the tests in this chapter. In addition it was shown in chapter five that the execution time of PI controller is considerable smaller compared to other

speed controllers (three times faster than the neural controller and six times faster than the fuzzy controller).

7.3.1.2 Group A2: Results

The effectiveness of the controller-agent concept for implementation of local control strategy was validated through a number of navigation tasks as demonstrated in Group A1 results. The start and the target points were considered to be the same (except experiment 9, Group A1). In order to further validate the effectiveness of the controller-agent concept, 20 tests were performed with different start points, target points and desired orientation at target points. Table 7-5 shows the parameters used in robot navigation for Group A2 simulation results.

Group A2				
Exp	(x_0, y_0)	(x_T, y_T)	θ_T	Caption
1	(2.5,0)	(0.5,2)	135°	Figure 7-4(a)
2	(2.5,0)	(0.5,2)	45°	Figure 7-4(b)
3	(2.5,0)	(0.5,2)	-45°	Figure 7-4(c)
4	(2.5,0)	(0.5,2)	-135°	Figure 7-4(d)
5	(2,0)	(2,2)	-90°	Figure 7-4(e)
6	(2,0)	(2,2)	0°	Figure 7-4(f)
7	(2,0)	(2,2)	180°	Figure 7-4(g)
8	(2.5,0)	(0,0)	0°	Figure 7-4(h)
9	(2.5,0)	(0,0)	180°	Figure 7-4(i)
10	(2.5,0)	(0,0)	-90°	Figure 7-4(j)
11	(2.5,0)	(0,0)	90°	Figure 7-4(k)
12	(2.5,2.5)	(0.5,0.5)	0°	Figure 7-4(l)
13	(2.5,2.5)	(0.5,0.5)	-135°	Figure 7-4(m)
14	(2.5,2.5)	(0.5,0.5)	180°	Figure 7-4(n)
15	(2.5,2.5)	(0.5,0.5)	-45°	Figure 7-4(o)
16	(0,2.5)	(2,0)	45°	Figure 7-4(p)
17	(0,2.5)	(2,0)	90°	Figure 7-4(q)
18	(0,2.5)	(2,0)	0°	Figure 7-4(r)
19	(0,2.5)	(2,0)	-135°	Figure 7-4(s)
20	(0,2.5)	(2,0)	180°	Figure 7-4(t)

Table 7-5 Parameters used in robot navigation in Group A2

From Figure 7-4 it can be observed that the mobile robot reaches the control objective in every case despite the different start and target points, and the different desirable orientation at target point. The robot reaches the target with the desirable approach angle with smooth trajectory.

7.3.1.3 Group A3: Results

The purpose of the results presented in this section is to identify the best performance between the three types of sensor sensitivity developed in chapter six for reactive robot navigation.

Figure 7-5 to Figure 7-7 depict three different navigation tasks where the robot has to reach the target point from a starting location in an environment populated by static obstacles using three different types of sensors. Again as in Group A1 each set of results are presented as a composite set of graphical outputs. Figure 7-5(a) for instance depicts on the left hand side, the main plot illustrating the robot's trajectory, initial and target point. On the right hand side, of Figure 7-5(a) the plot is divided into six sub-plots illustrating the following. The first, second and third sub-plots show the sensor sensitivity of the left, centre and right sensor respectively. The fourth sub-plot shows the robot's desired orientation (θ_D), actual orientation θ_S and orientation at target point θ_T . The fifth sub-plot shows the robot's actual velocity (V_S) versus the robot's desired velocity (V_D). The sixth sub-plot shows the robot's global state during the navigation task. Comparison of the robot's left and right desired velocity (V_{Ld}, V_{Rd}) versus actual left and right velocity (V_L, V_R) is omitted as the simulation results from Group A1 showed satisfactory performance. The same principle is followed for the robot's actual left and right motor voltage (V_r, V_l), in which graphical representation are also omitted (simulation results in Group A1 showed control effort within the design requirements).

In Figure 7-5 the robot has to reach the target point in an environment, which contains 8 static obstacles. The same navigation task was repeated three times (see Figure 7-5(a), (b) and (c)), in

order to simulate all three types of sensor. Figure 7-5(a) shows a navigation task of the robot incorporating infrared sensor for detection and avoidance of static obstacles. The result clearly shows that the mobile robot is able to reach the target point successfully without any collision with the static obstacles. However, as can be observed from Figure 7-5(a), the mobile robot turns away from the obstacles at a very close distance. The average velocity is considerably high, the distance travelled is short and the time taken to reach the target is significantly small. Figure 7-5(b) depicts the same navigation task but with ultrasonic sensor. The result again is satisfactory and shows that the mobile robot is able to reach the target point successfully without any collision with the static obstacles. However, average velocity, travel time and distance travelled are different compared, to the navigation task incorporating infrared sensor. From Figure 7-5(b) it can be observed that the mobile robot turns away from the obstacles at a far distance resulting in a different trajectory. As can be seen from sub-plot number 5 in Figure 7-5(b) the robot reduced its velocity by 0.2m/s when the obstacle avoidance behaviour was activated. Although distance travelled and time taken to complete the mission is marginally higher than the one with infrared sensor, the robot took a safer trajectory avoiding the main cluster of static obstacles. The result of the third type of sensor (no-shape) is depicted in Figure 7-5(c). The path followed by the robot in this case is more or less equivalent to that using the ultrasonic sensor. The main difference compared with the two other sensors is that the average velocity is considerably lower resulting in an increase in travel time.

To examine further the effectiveness of different types of sensor sensitivity, two additional navigation tasks were completed considering the same environment configuration but with a higher number of static obstacles. Figure 7-6 depicts a navigation task in which the robot has to reach the target point in an environment, which contains 11 static obstacles. In Figure 7-7 a total number of 15 static obstacles populate the environment. Again the use of infrared sensor sensitivity produces short distance travelled and high average robot velocity. However, the results this time show a different trajectory generated by the robot in the case of ultrasonic and

no-shape sensor utilisation. The mobile robot with ultrasonic sensor sensitivity generated safe trajectory away from the main cluster of static obstacles, whereas the no-shape sensor sensitivity generated trajectory safe but very close to the static obstacles. Therefore, as safety is an important issue in mobile robot navigation the ultrasonic sensor sensitivity will be used for the rest of the tests in this chapter (distance travelled and time taken to complete the mission is marginally higher than the infrared sensitivity).

Table 7-6 shows the comparison made between the different types of sensor sensitivity based on three different navigation tasks.

Group A3										
$(x_0, y_0) = (0,0)$, $(x_T, y_T) = (2.5, 2.5)$ and $\theta_T = 45^\circ$										
Exp	Infrared sensor			Ultrasonic sensor			No-shape sensor			Caption
	Distance travelled	Time taken	Collision	Distance travelled	Time taken	Collision	Distance travelled	Time taken	Collision	
1	3.6606	10.56	No	4.3476	12.8	No	4.3395	13.64	No	Figure 7-5
2	3.6652	10.6	No	4.7924	14.26	No	4.1640	12	No	Figure 7-6
3	3.6776	10.7	No	3.7976	12.18	No	4.1328	15.82	No	Figure 7-7

Table 7-6 Comparison between different types of sensor sensitivity

7.3.1.4 Group A4: Results

In chapter six it was suggested that the aim of intelligent robotics research is to develop mobile robots capable of navigating autonomously in an unstructured and/or unexplored environment. In order to ensure the robot's system safety it is necessary that the robot is able to navigate without colliding with obstructions in its environment. Both fuzzy logic and neural networks behavioural encoding of the *avoid_static* robot behaviour developed in chapter six are tested in this group of simulations. Both behavioural encoding are exposed in the same environment, where the mobile robot has to complete a specified navigation task. The comparison measure between the two behavioural encoding is the distance travelled and time taken for the mobile

robot to successfully complete its mission (elapsed times are not considered here as already shown in chapter five fuzzy encoding is much slower than neural encoding).

In total 6 simulated experiments took place as depicted from Figure 7-8 to Figure 7-13. Each figure illustrates a navigation task for the mobile robot using both fuzzy and neural encoding. Again each result is presented as a composite set of graphical outputs. Figure 7-8(a) for instance depicts on the left hand side, the main plot illustrating the robot's trajectory, initial and target point. On the right hand side the plot is divided into three sub-plots, illustrating sensory information, robot's orientation and robot's velocity respectively. Again the topside of the sub-plots contains the global state of the mobile robot.

In this group for every simulated navigation task, different shapes and positions of the static obstacles have been tested. Figure 7-8 depicts a navigation task in which a circle-shape obstacle has been placed in front of the robot in order to obstruct its path towards the target. As Figure 7-8(a) and Figure 7-8 (b) show, the mobile robot was able to reach the target successfully without colliding with the static obstacle. The trajectories of both fuzzy and neural encoding are almost identical. Figure 7-9 depicts a navigation task in which a wall-shape obstacle has been placed in front of the robot. Again the robot was able to successfully find the target without any collision with the static object. The robot's trajectory is smooth and free from any signs of oscillation.

In Figure 7-10 the static environment becomes more complex as both a U-shape and a semicircle-shape static objects obstruct the robot's navigation path. Since there are no obstacles around the start position, at the start the robot mainly reflects the behaviour, *go_to* (no-obstacles global state one is activated), so that it moves towards the target at a high speed (V_{NOM}). When the robot approaches the U-shape object, state two is active (avoid static) and it automatically

decelerates in order to find a way to avoid the static object. When the robot departs from the U-shape object, state one is activated (no obstacles). The robot then moves towards the target. When the robot approaches the semicircular object, its state changes, decelerates, and successfully avoids the object. When the robot has passed the static obstacle, behaviour *go_to* is activated, so that it reaches the target according to the local control strategy defined in phase A and B. Again as in the previous navigation task the robot trajectory generated by the fuzzy and neural behavioural encoding are similar.

In Figure 7-11 an environment configuration with two static obstacles is illustrated. The first static obstacle is a corridor-shape and is located in front of the robot's start position. The second static obstacle is a circle-shape and is located in front of the corridor-shape object. Assume that this environment configuration is unknown to the robot in which its navigation task is to reach the target point from the start position. At the beginning of the mission, the robot's global state is one and therefore the robot moves towards the target activating the *go_to* behaviour. At this stage the *go_to* behaviour calculates the difference between the current heading angle of the robot and the angle towards the target. When the robot approaches the corridor-shaped object it decelerates and moves straight ahead as only left and right obstacles appear to obstruct the robot's path. The robot then detects the circular-shape object, and both fuzzy and neural static obstacle avoidance behaviours, successfully navigate the robot out of the corridor, around the circular object and then towards the target. As can be observed from Figure 7-11, when the robot moves around the circular object its deceleration is much lower than its deceleration through the corridor. The fuzzy and neural encoding navigates the mobile robot with similar trajectories.

Figure 7-12 to Figure 7-13 depicts robot navigation in an environment populated by static obstacles. In particular, Figure 7-12 illustrates the robot's workspace populated by 21 static obstacles whereas, in Figure 7-13 the robot's workspace is populated by 26 static obstacles (5

obstacles were added in Figure 7-13 in order to obstruct deliberately the robot's trajectory shown in Figure 7-12). The results of both figures show that the robot can automatically control its velocity according to the sensory information extracted from the environment. The robot when is close to the static obstacles, it reduces its velocity, whereas while has bypassed the obstacles, it moves towards the target with higher velocity. It can be observed that, using only ultrasonic sensors to acquire dynamic information, the robot can successfully reach the target by efficient co-ordination between the multiple robot behaviours. As in the previous navigation tasks both fuzzy and neural encoding navigates the mobile robot with a similar trajectories.

The results are summarised and presented in Table 7-7. As can be seen from Table 7-7, fuzzy logic behavioural encoding produces marginally better results over neural encoding in terms of distance travelled during the navigation tasks (red shaded numbers show the best performance). On the other side of the spectrum neural networks behavioural encoding shows better result over fuzzy encoding in term of time taken to complete each mission. For the rest of the tests, neural networks behavioural encoding for static obstacle avoidance will be used, as the elapsed time of neural encoding is considerably lower than fuzzy encoding, and the difference between the two behavioural methods is marginal in terms of distance travelled and time taken for the robot to complete the navigation task.

Group A4							
$(x_0, y_0) = (0,0)$, $(x_T, y_T) = (2.5, 2.5)$ and $\theta_T = 45^\circ$							
Exp	Fuzzy logic			Neural network			Caption
	Distance travelled	Time taken	Collision	Distance travelled	Time taken	Collision	
1	4.3444	12.06	No	4.3458	12.13	No	Figure 7-8
2	3.9468	11.02	No	3.9497	11.05	No	Figure 7-9
3	4.7291	13.34	No	4.7289	13.31	No	Figure 7-10
4	4.2413	17.59	No	4.2217	17.16	No	Figure 7-11
5	4.4206	16.3	No	4.4209	15.7	No	Figure 7-12
6	3.8645	17.16	No	3.8721	16.11	No	Figure 7-13

Table 7-7 Comparison between fuzzy and neural static obstacle avoidance behaviour

7.3.1.5 Part I: Outcome of results

The main outcome of Part I results can be summarised as follows. The control architecture was observed to exhibit robustness, adaptability and flexibility when tested in single robot navigation in an unknown environment populated by static obstacles. The mobile robot achieved every control objective and its trajectory was smooth despite the interaction between several behaviours and the presence of unexpected obstacles.

The effectiveness of SOAM using the controller-agent concept was validated by a number of tests. In all cases the robot achieved the control objectives defined both in A and B local control strategies.

Three speed controllers were tested and the results show that PI and fuzzy speed controllers have a marginal advantage against the neural controller. In particular the PI controller has the smallest ISE error of all controllers, whereas the fuzzy controller has the smallest IAE and ITAE error.

Three different type of sensor sensitivity were tested according to the distance travelled and time taken for the robot to complete the mission. In all cases it was possible to navigate the mobile robot towards the target without any collision with the static objects. Infrared sensor sensitivity produced the lowest values for distance and time taken for the robot to complete its mission.

Finally, the fuzzy logic and neural networks behavioural encoding for static obstacle avoidance developed in chapter six were tested in an environment containing different configurations of static obstacles. Fuzzy behavioural encoding produced marginally better result over neural encoding in terms of distance travelled during the navigation tasks. The neural networks behavioural encoding showed a better result over fuzzy encoding in terms of time taken for the

robot to complete each mission. The control architecture was successfully adapted to the environmental conditions, by adjusting the robot's control parameters (i.e. speed) and to activation of appropriate behaviour.

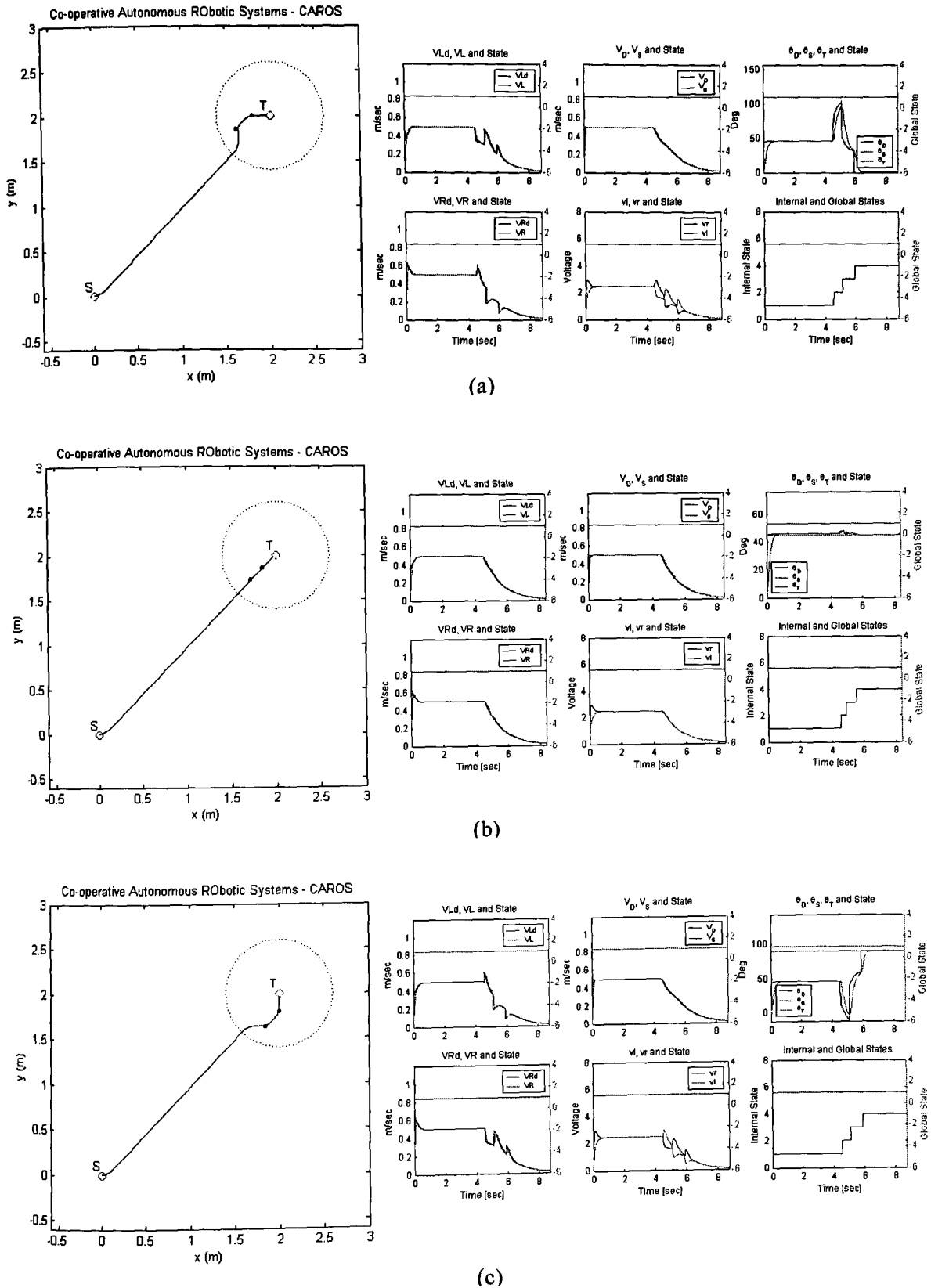


Figure 7-1 Results A1

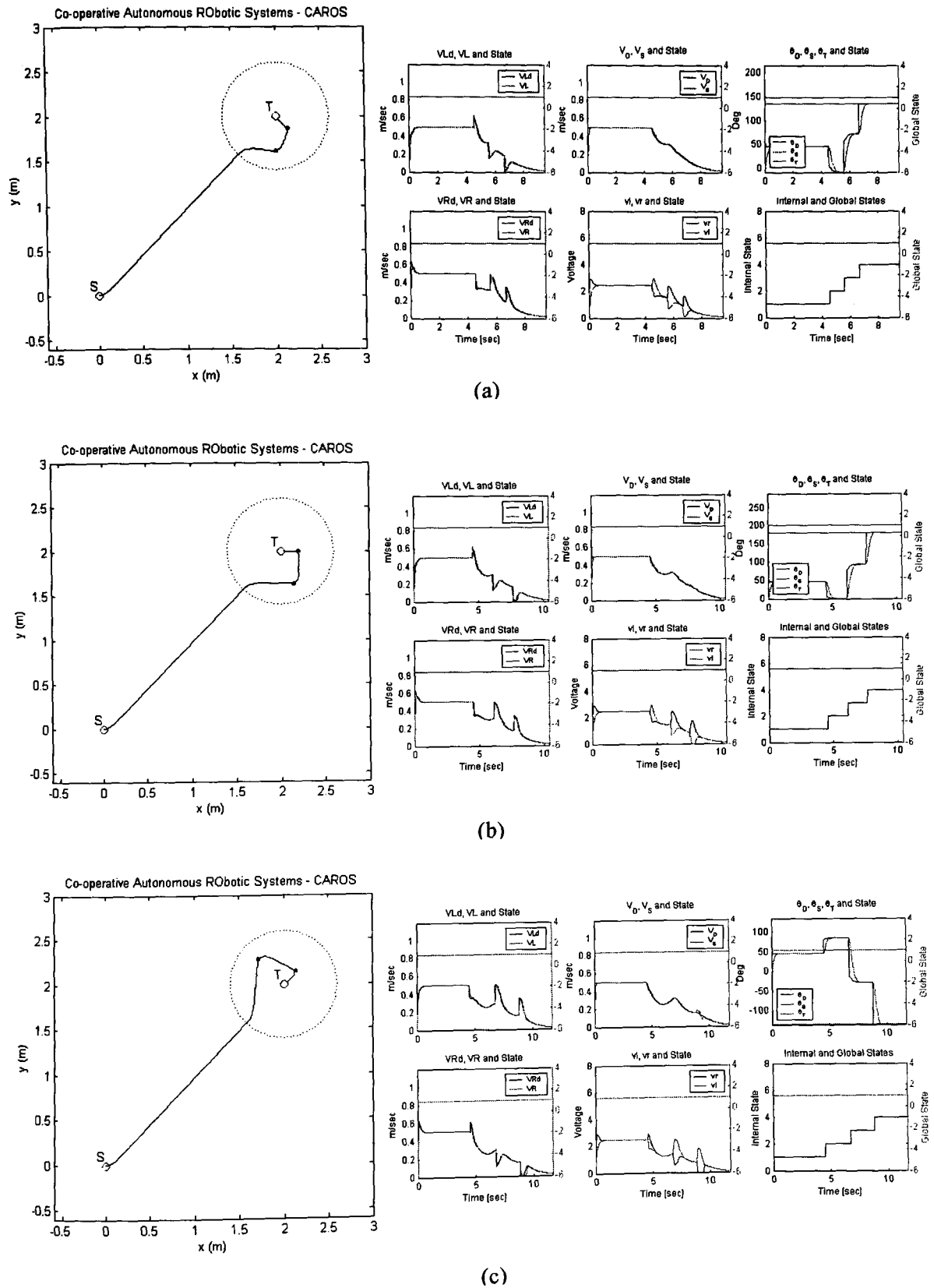


Figure 7-2 Results A1

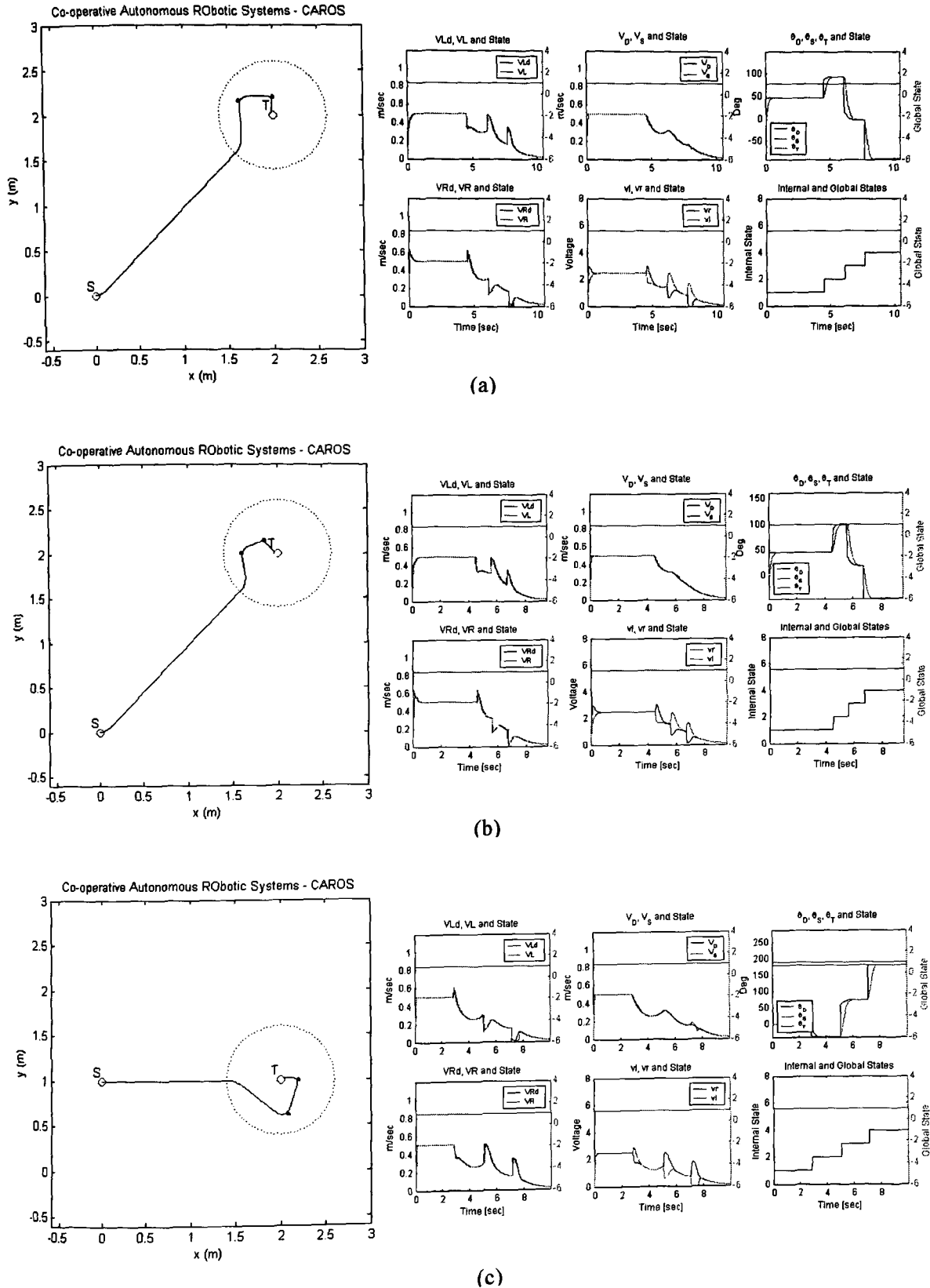


Figure 7-3 Results A1

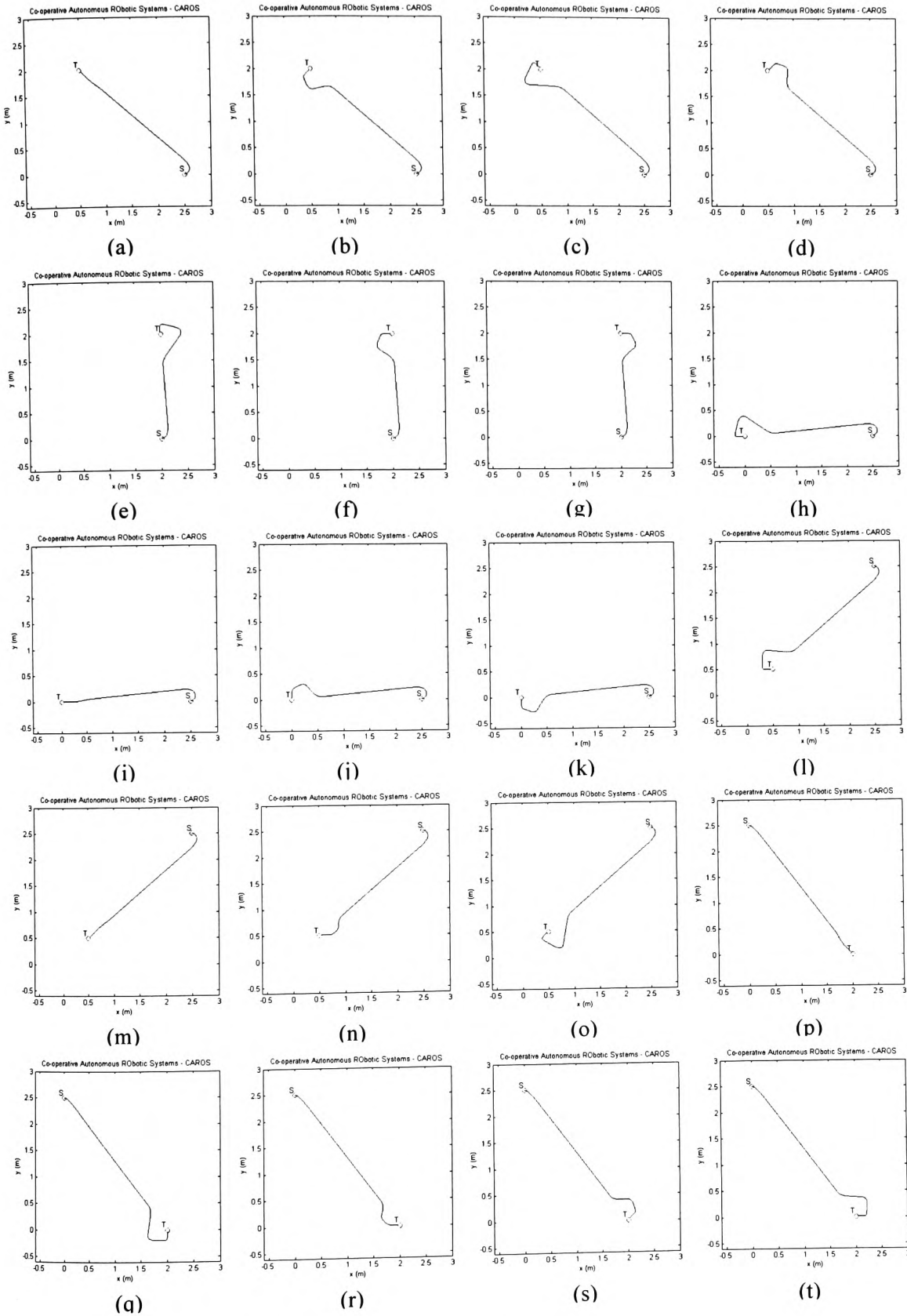
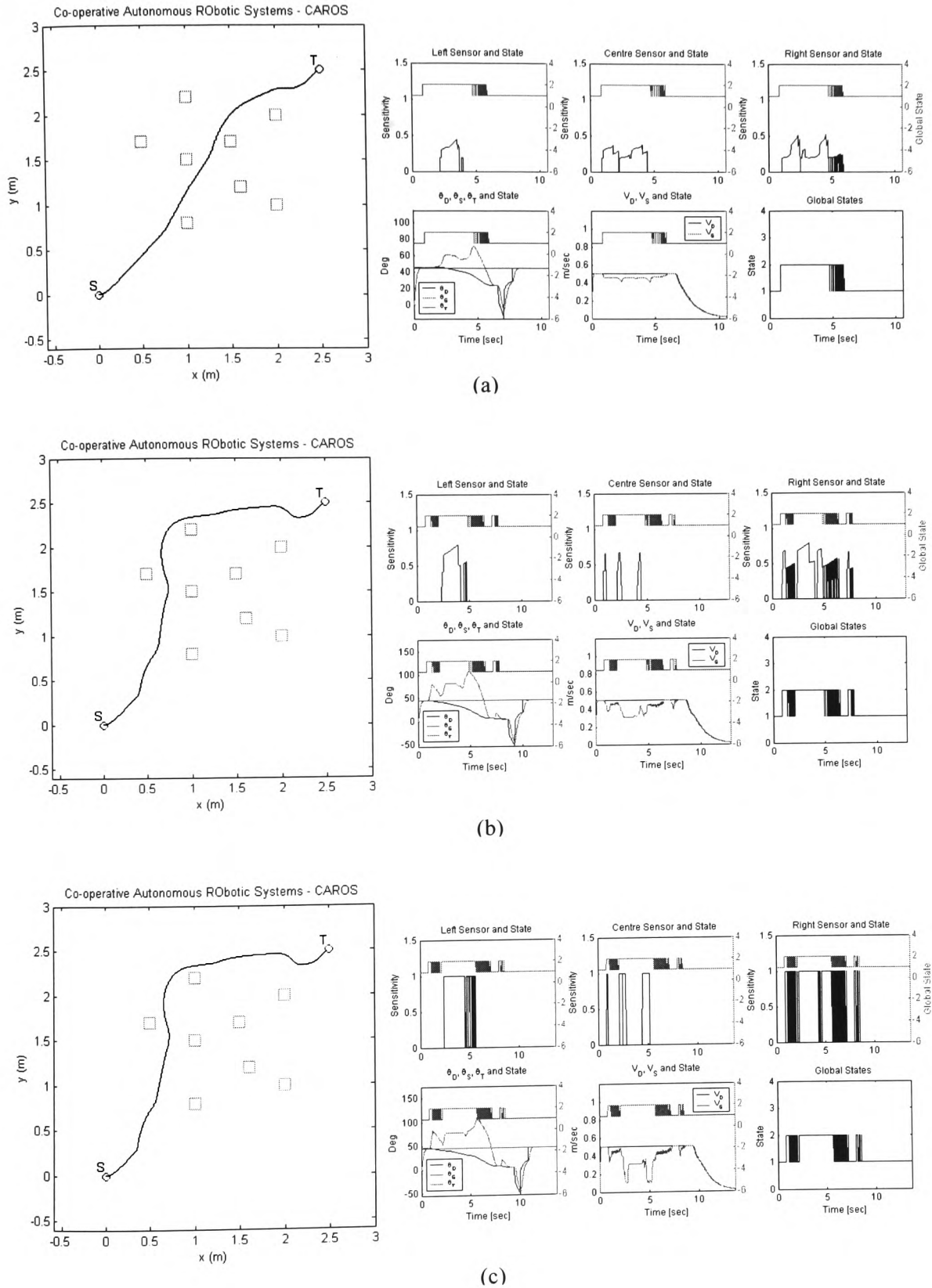


Figure 7-4 Results A2



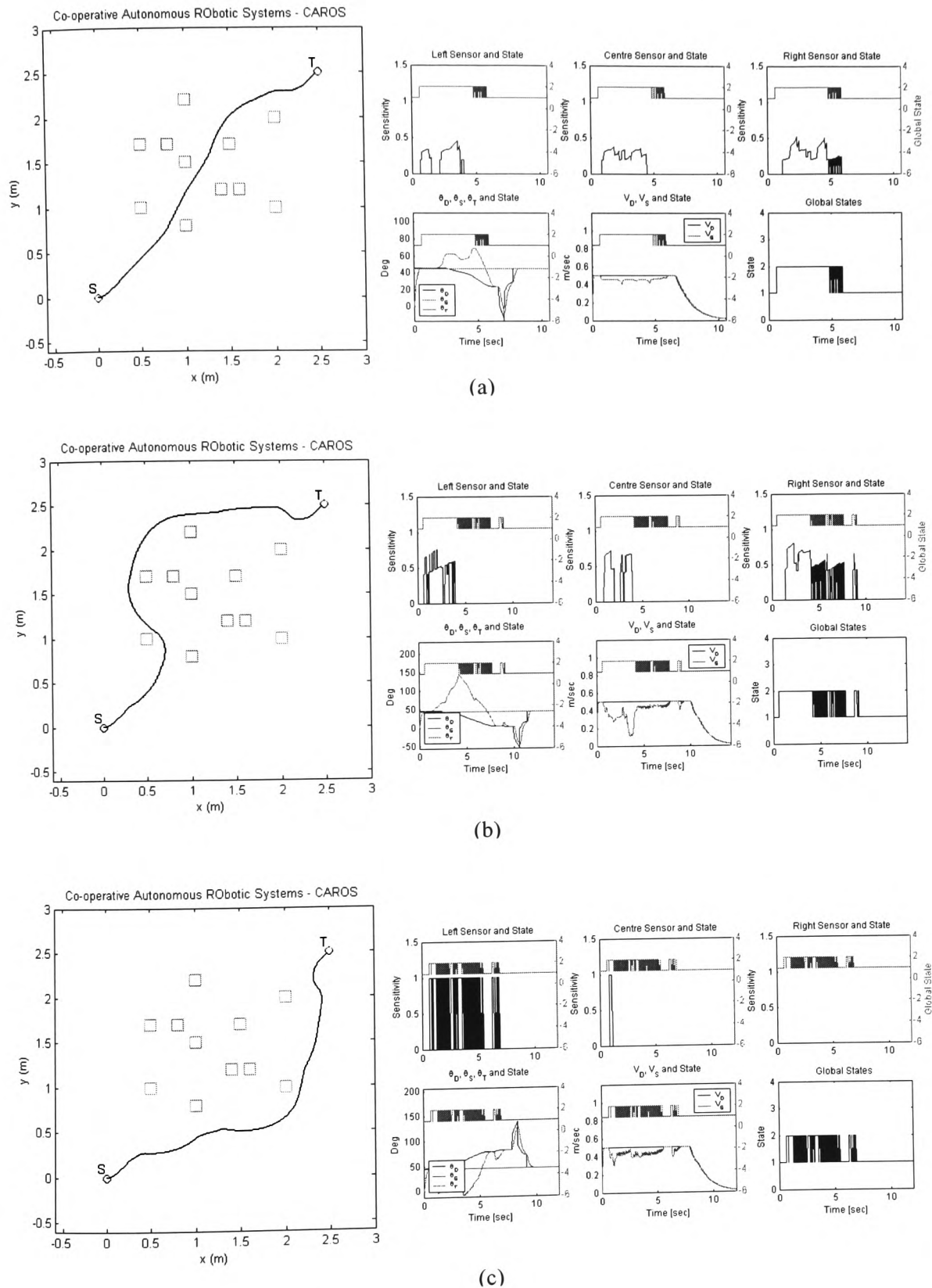


Figure 7-6 Results A3: (a) Infrared (b) Ultrasonic (c) No-shape

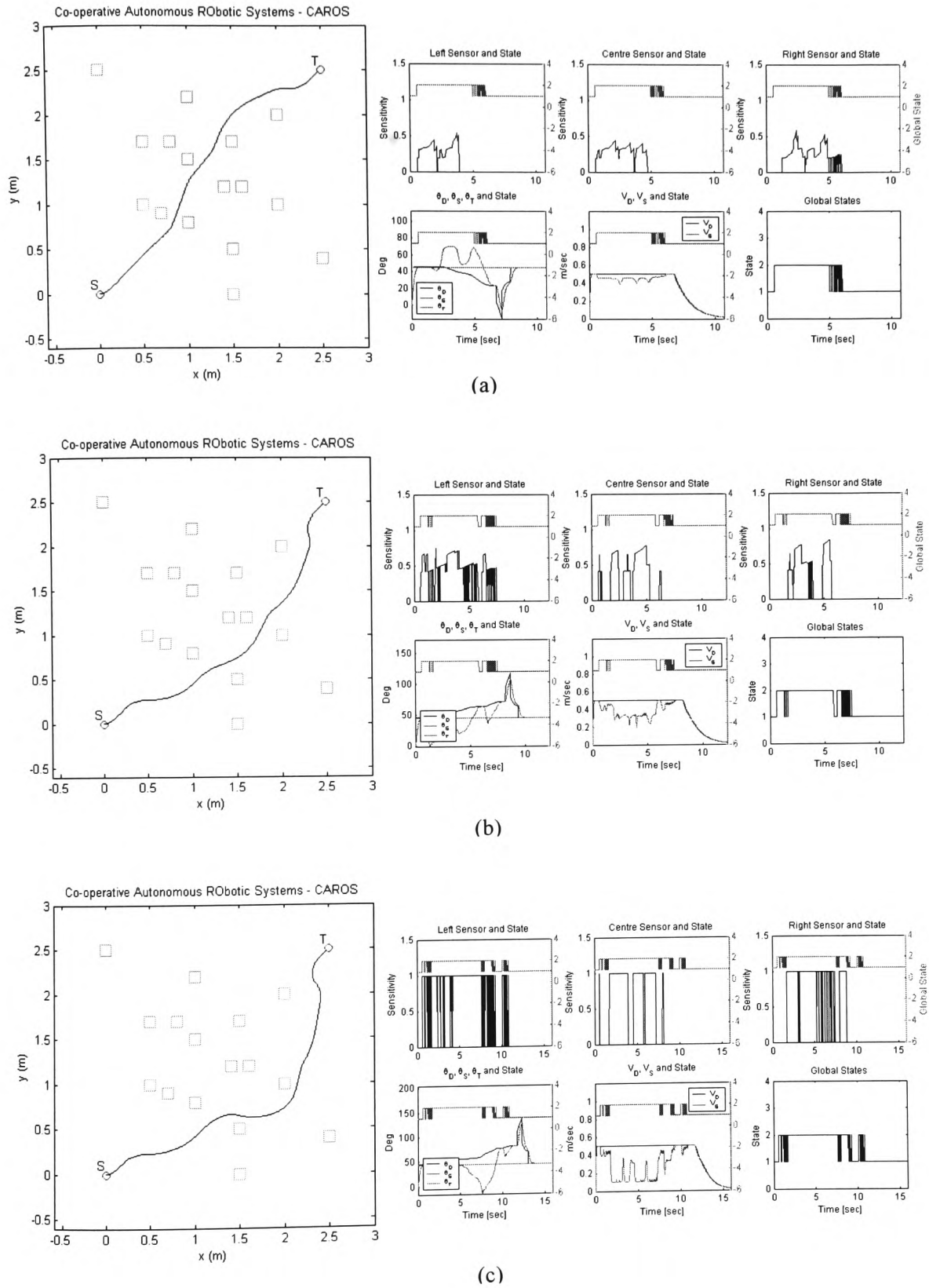
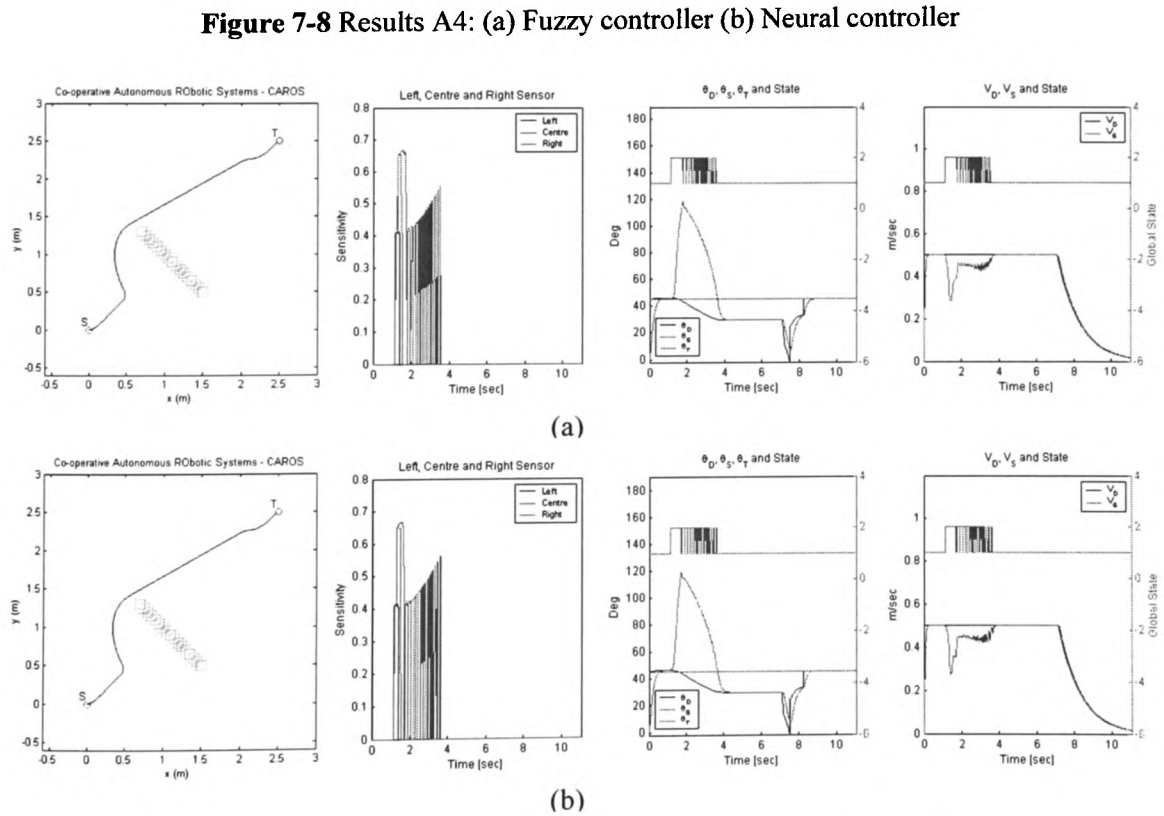
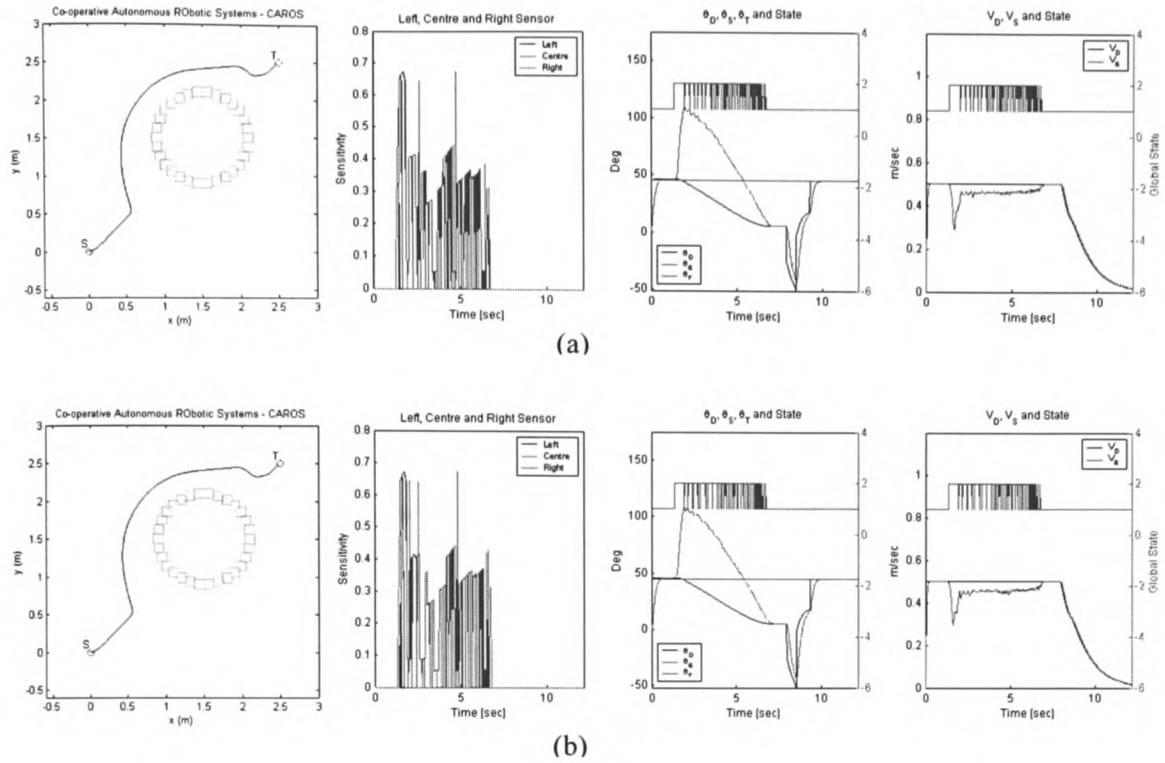


Figure 7-7 Results A3: (a) Infrared (b) Ultrasonic (c) No-shape



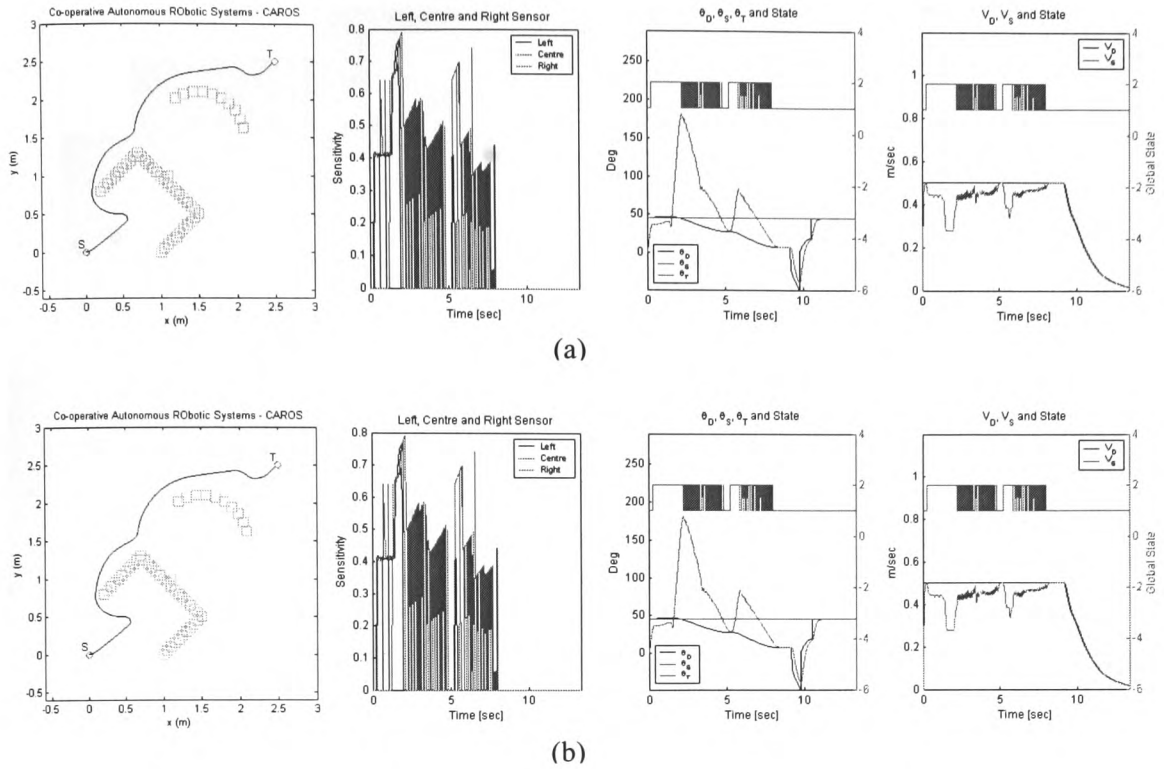


Figure 7-10 Results A4: (a) Fuzzy controller (b) Neural controller

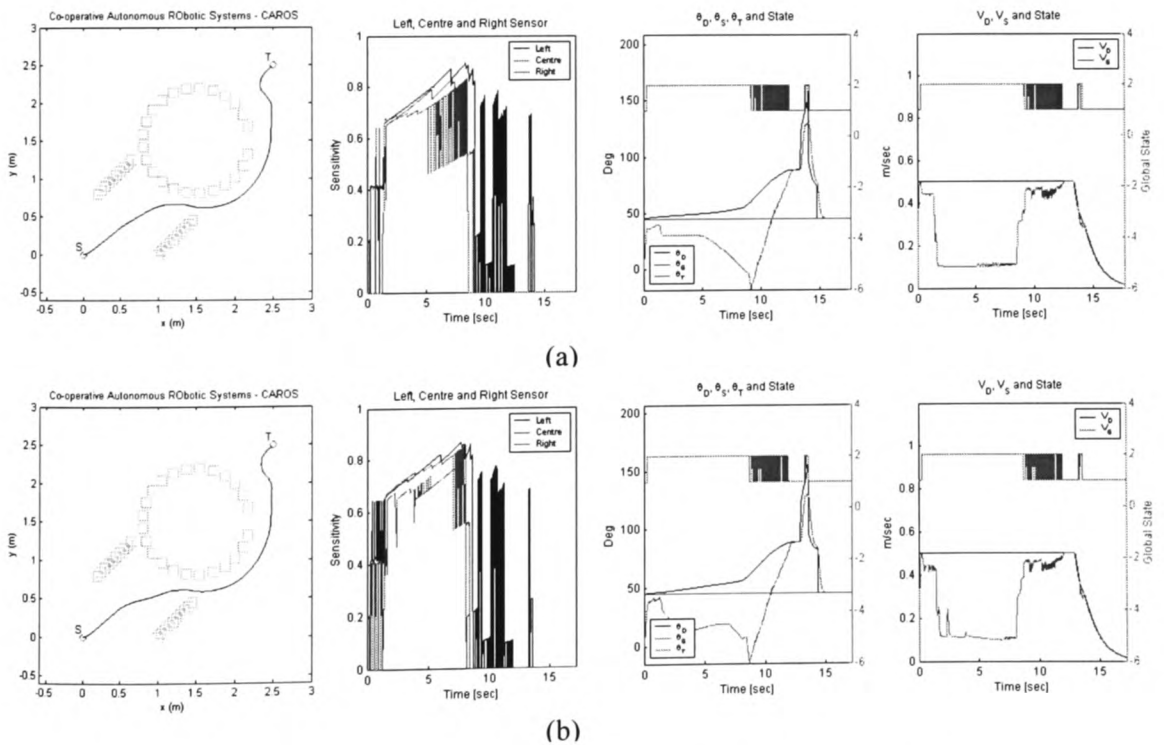


Figure 7-11 Results A4: (a) Fuzzy controller (b) Neural controller

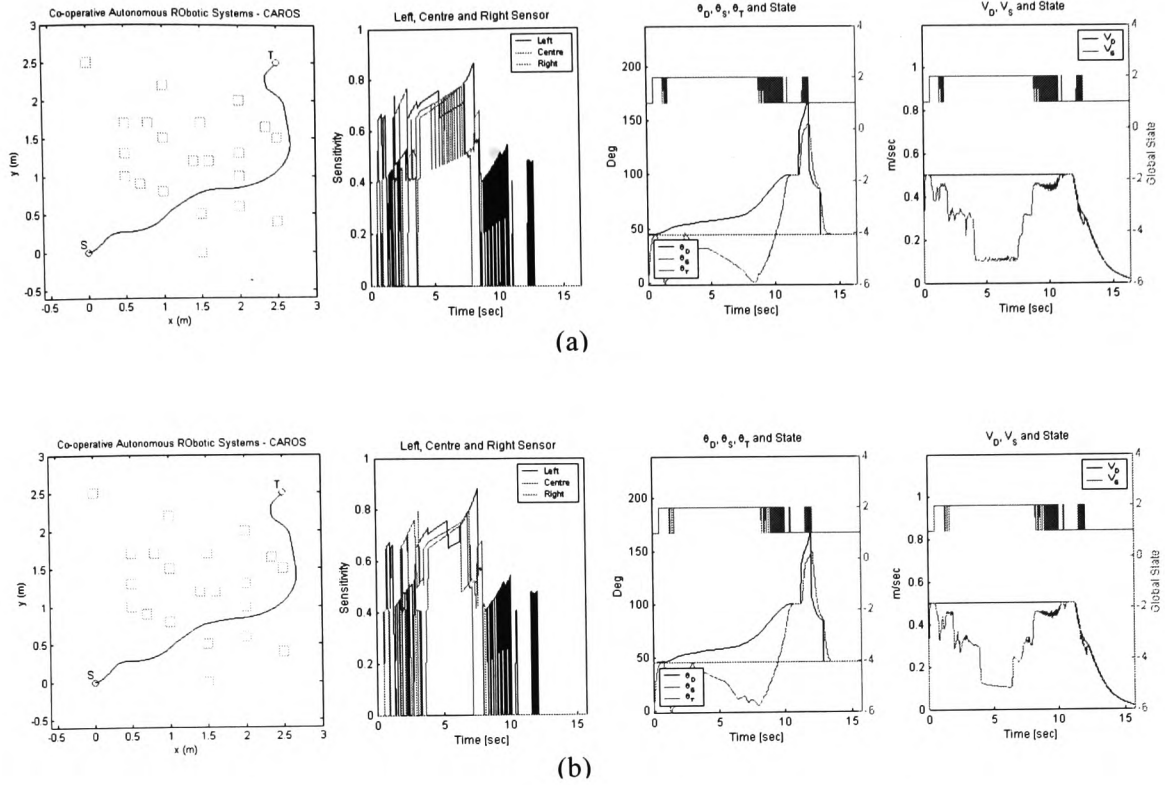


Figure 7-12 Results A4: (a) Fuzzy controller (b) Neural controller

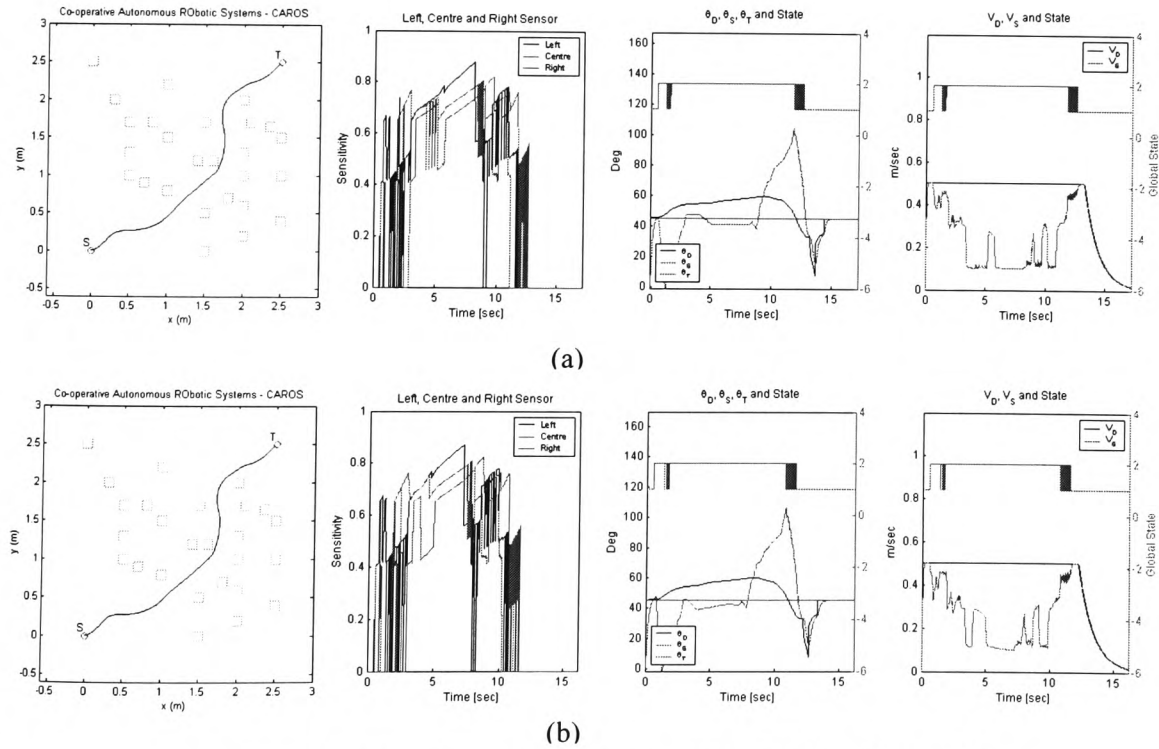


Figure 7-13 Results A4: (a) Fuzzy controller (b) Neural controller

7.3.2 Part II: Results

The results presented in this section investigate the effectiveness of CAROS control architecture in navigation tasks involving two mobile robots operating in an unknown static and dynamic environment. Division of simulation results in Part II, is given in Table 7-8. In the next sections results of each group are presented followed by a summary of the performance of the proposed system.

Part II: Division of Results	
Group	Representation
B1	Navigation of two robots with observations based on dynamic obstacle avoidance behaviour (no static obstacles)
B2	Navigation of two robots with observations based on co-ordination of robot behaviours (static and dynamic obstacles)

Table 7-8 Division of results (Part II)

7.3.2.1 Group B1: Results

Results presented in Group B1 concentrate on navigation of two mobile robots with observations based on dynamic obstacle avoidance behaviour developed in chapter six. In particular the effectiveness of the main features of the proposed hybrid behaviours (stateflow-fuzzy and stateflow-neural) for dynamic obstacle avoidance in an unknown and unstructured environment is validated through a number of tests. The results are used to compare the performance of the proposed hybrid behaviours and also to observe the robustness of the control system architecture against the desired requirements, such as supervision, decision-making and co-ordination of internal control structures (subsystems and behaviours).

Figure 7-14 to Figure 7-19 depicts a number of navigation tasks where two mobile robots have to reach independent target points (x_T, y_T) marked by T from an initial position (x_0, y_0) marked by S . Each experiment is divided into four figure plots illustrating both hybrid

behavioural encoding. For instance, the stateflow-fuzzy behavioural encoding is illustrated in Figure 7-14(a), which depicts on the top-left hand side the main plot of both robots A and B, illustrating their trajectories (x_s, y_s) , initial points and target points. On the right hand side of Figure 7-14(a) the plot is divided into six sub-plots illustrating the following. The first three of these show control parameters for robot A, and sub-plot 4 to 6 shows control parameters for robot B. The observed control parameters for each robot are, sensory data (left, centre and right sensor), comparison of robot's actual velocity (V_s) versus the robot's desired velocity (V_D) , and the distance between the two robots during the navigation task. Figure 7-14(b) illustrates the same experiment and information as above but using the stateflow-neural behavioural encoding.

Figure 7-14 to Figure 7-19 depicts results of navigation of two mobile robots (A and B) in the same environment. The robots have to reach independent targets from their initial positions without any collision between them. In order to demonstrate the effectiveness of the proposed system, start and target points are different for every experiment as show in Figure 7-14 to Figure 7-19. In all navigation tasks no collision between the mobile robots occurred. Each mobile robot, without prior knowledge about the direction and velocity of the other robot, successfully solved the conflict when the other robot was found to obstruct its path. In all figures the control parameters illustrate that when the global identification mechanism decides to activate different behaviour, the mobile robots effectively adjust their velocities producing smooth trajectories. The concept of traffic rules used in chapter six to model robot behaviour proved to be an efficient technique for multiple robot navigation without communication. In addition the efficiency of the proposed method for identification of direction of moving object developed in chapter six is established through the results. As the robots trajectories are almost on a straight line the travel time and distance travelled by both robots is low and close to optimal.

Table 7-9 shows a comparison between the stateflow-fuzzy and stateflow-neural behavioural encoding. It can be seen that stateflow-fuzzy behavioural encoding produces better result over stateflow-neural encoding in terms of distance travelled during the navigation tasks (red shaded numbers show the best performance). Stateflow-neural behavioural encoding shows better result over stateflow-fuzzy encoding in term of time taken to complete each mission. Although the performance of both behaviours is similar, stateflow-fuzzy behavioural encoding is chosen for the rest of the tests because it controls better the robot's velocity than the stateflow-neural behavioural encoding. This is desirable for a mobile robot navigating in an unknown environment containing moving obstacles.

Group B1								
Exp	Robot	Stateflow-fuzzy			Stateflow-neural			Caption
		Distance travelled	Time taken	Collision	Distance travelled	Time taken	Collision	
1	A	3.5289	9.86	N	3.5273	9.84	No	Figure 7-14
	B	3.6686	10.84	N	3.6825	10.63	No	
2	A	2.8303	8.63	N	2.8309	8.6	No	Figure 7-15
	B	2.9870	8.98	N	2.9879	8.93	No	
3	A	2.0503	6.85	N	2.0502	6.84	No	Figure 7-16
	B	2.7902	9.15	N	2.7959	9.09	No	
4	A	2.0497	7.14	N	2.0499	7.1	No	Figure 7-17
	B	2.4706	9.22	N	2.4722	9.24	No	
5	A	2.3630	7.68	N	2.3589	7.87	No	Figure 7-18
	B	2.4702	8.35	N	2.4703	8.32	No	
6	A	2.2685	7.36	N	2.2681	7.41	No	Figure 7-19
	B	2.4880	7.92	N	2.4902	7.93	No	

Table 7-9 Comparison between stateflow-fuzzy and stateflow-neural hybrid dynamic obstacle avoidance behavioural encoding

7.3.2.2 Group B2: Results

Real-time motion planning in an unknown and unstructured environment involves collision avoidance of static as well as moving objects (robots). However, strategies suitable for navigation in a stationary environment cannot be translated as strategies for dynamic

environment. In chapter six it was shown that the implementation of complex behaviour generation for artificial systems can be overcome by decomposing the global task(s) into simpler, well-specified behaviours which are easier to design and can be tuned independently of each other. In this group of results the co-ordination of these well-defined behaviours using the action co-ordinator mechanism developed in chapter six is tested. The neural networks behavioural encoding is used for the *avoid_static* robot behaviour, whereas the stateflow-fuzzy logic behavioural encoding is used for the *avoid_dynamic* robot behaviour.

Figure 7-20 to Figure 7-22 depicts a number of navigation tasks where two mobile robots have to reach independent target points (x_T, y_T) marked by T from an initial position (x_0, y_0) marked by S in an unknown environment populated by static and dynamic obstacles. A total number of nine experiments were considered. The result of each experiment is presented as two figure plots illustrating the robots trajectories and their control signals (velocity, distance to other robot and global state). For instance, the experiment illustrated in Figure 7-20(a), on the left hand side illustrates robots A and B trajectories (x_S, y_S) , initial points and target points. On the right hand side of Figure 7-20(a) the plot is divided into six sub-plots illustrating the following. The first three show control parameters for robot A and sub-plots 4 to 6 show control parameters for robot B. The control parameters for each robot are, comparison of robot's actual velocity (V_S) versus the robot's desired velocity (V_D) , the robot's global state and the distance between the two robots during the navigation task. Note that as in Group B1 results, the robot's desired orientation at the target point is shown in the legend in the main figure plot illustrating the robots trajectories.

Figure 7-20(a) to Figure 7-20(c) depicts results of three experiments of navigation tasks of two mobile robots (A and B) in the same environment populated by static obstacles. The robots have to reach independent targets from their initial positions without any collision with static

obstacles or themselves. In order to demonstrate the effectiveness of the proposed control architecture, start and target points are the same for the first three experiments but the number of static obstacles increases considerably in order to deliberately obstruct the robots paths, as shown in Figure 7-20(b) and Figure 7-20(c). In all three navigation tasks no collision between the mobile robots and the static occurred. Considering Figure 7-20(a) to Figure 7-20(c) it can be also observed that each mobile robot successfully modifies its velocity according to the current global state. The action co-ordinator mechanism activates the appropriate behaviour, and the robots reach their target with smooth and close to optimal trajectories.

Figure 7-21(a) to Figure 7-21(c) depicts results of navigation tasks of two mobile robots (A and B) in the same environment operating in a narrow corridor or road. In Figure 7-21(a) the robots depart from their initial position and automatically decelerate when they approach the left and right static obstacle (global state is two). Then according to the neural behaviour encoding for static obstacle avoidance the mobile robots move straight and keep on the road (between walls). When the two robots are meeting each other, their global state changes from two (avoid static) to three (avoid dynamic) and the stateflow-fuzzy behavioural encoding for dynamic obstacle avoidance is activated. Then the mobile robots proceed according to the traffic rules developed in chapter six for dynamic obstacle avoidance.

This example was given in order to demonstrate how a car driving on a road avoids another car moving in an opposite direction. Both drivers slow down and turn left in order to avoid each other. When the mobile robots solve the conflict between them automatically make reasonable movement to keep in the middle of the road. The robots then move directly towards the targets with a desired approach angle.

Figure 7-21(b) and Figure 7-21(b) depict the same scenario as above but this time the corridor is obstructed. In the first case (Figure 7-21(b)) there is a blockage on the left hand side and in the second case (Figure 7-21(c)) the corridor is blocked in both sides. Only a very narrow exit is available to the robots when they get trapped. The results show that the robots were able to find the path towards their targets.

Figure 7-22(a) to Figure 7-22(c) depict a scenario of navigation tasks where two mobile robots have to reach independent targets from their initial positions without any collision. In order to demonstrate and verify the effectiveness of the proposed system, start and target points are different for every simulated experiment. Initially the robots are in a mixed environment (static and dynamic), without any prior knowledge of each other or of the obstacles. The robots are therefore in state one (no obstacles). Once the robots receive a command to move towards their targets, they try to approach the targets while avoiding static obstacles and each other. From Figure 7-22(a) to Figure 7-22(c) it can be seen that during the navigation tasks each other and static obstacles obstruct the robots path. The global state of both robots shows that the action co-ordinator mechanism solves the conflict for each robot by adjusting the robots velocities accordingly. When the global state of each robot is one (no obstacles) then the robots accelerate until they reach the velocity threshold. Both robots reach the targets with smooth trajectory without any collisions. In all three navigation tasks the proposed control architecture achieve a sufficiently high control performance once the control objective of each robot has been reached.

Table 7-10 summarises the collision avoidance results from Group B2.

Group B2				
Exp	Robot	Collision Avoidance Result		Caption
		Collision with Static Obstacle	Collision with Dynamic Obstacle	
1	A	No	No	Figure 7-20(a)
	B	No	No	
2	A	No	No	Figure 7-20(b)
	B	No	No	
3	A	No	No	Figure 7-20(c)
	B	No	No	
4	A	No	No	Figure 7-21(a)
	B	No	No	
5	A	No	No	Figure 7-21(b)
	B	No	No	
6	A	No	No	Figure 7-21(c)
	B	No	No	
7	A	No	No	Figure 7-22(a)
	B	No	No	
8	A	No	No	Figure 7-22(b)
	B	No	No	
9	A	No	No	Figure 7-22(c)
	B	No	No	

Table 7-10 Collision avoidance result of a mixed environment containing both static and moving obstacles

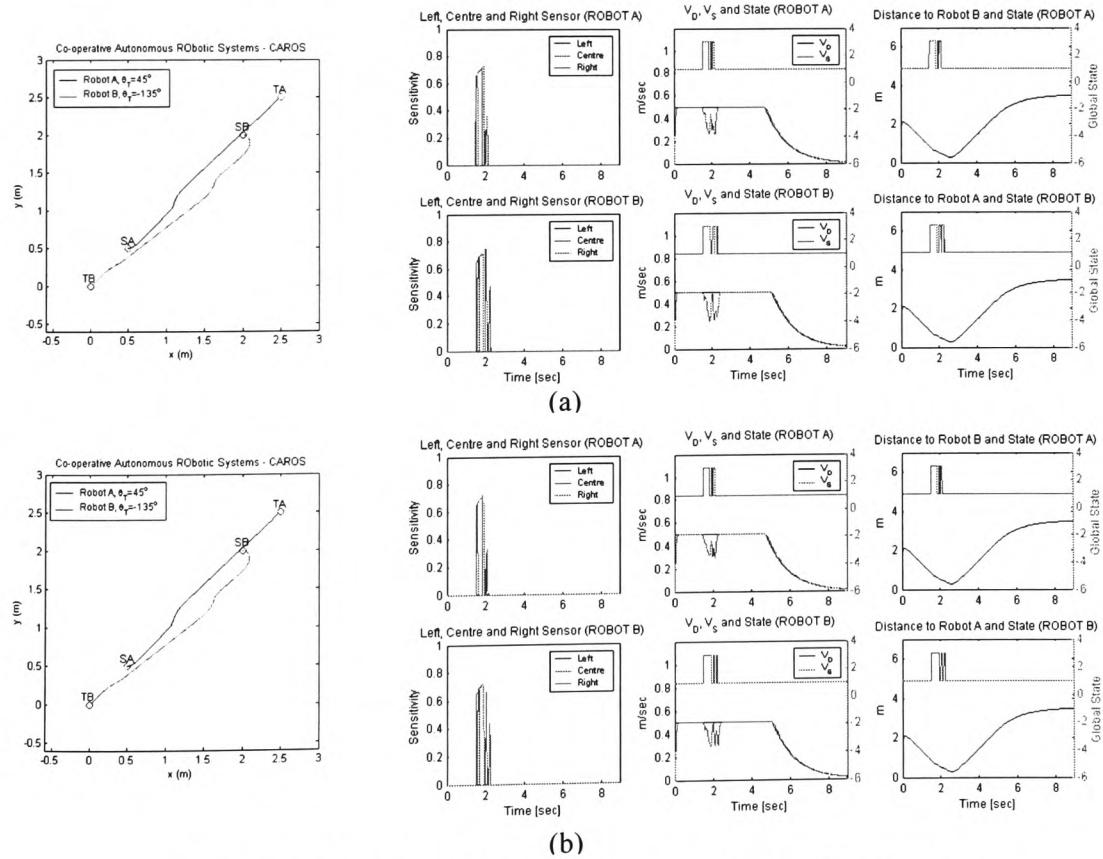
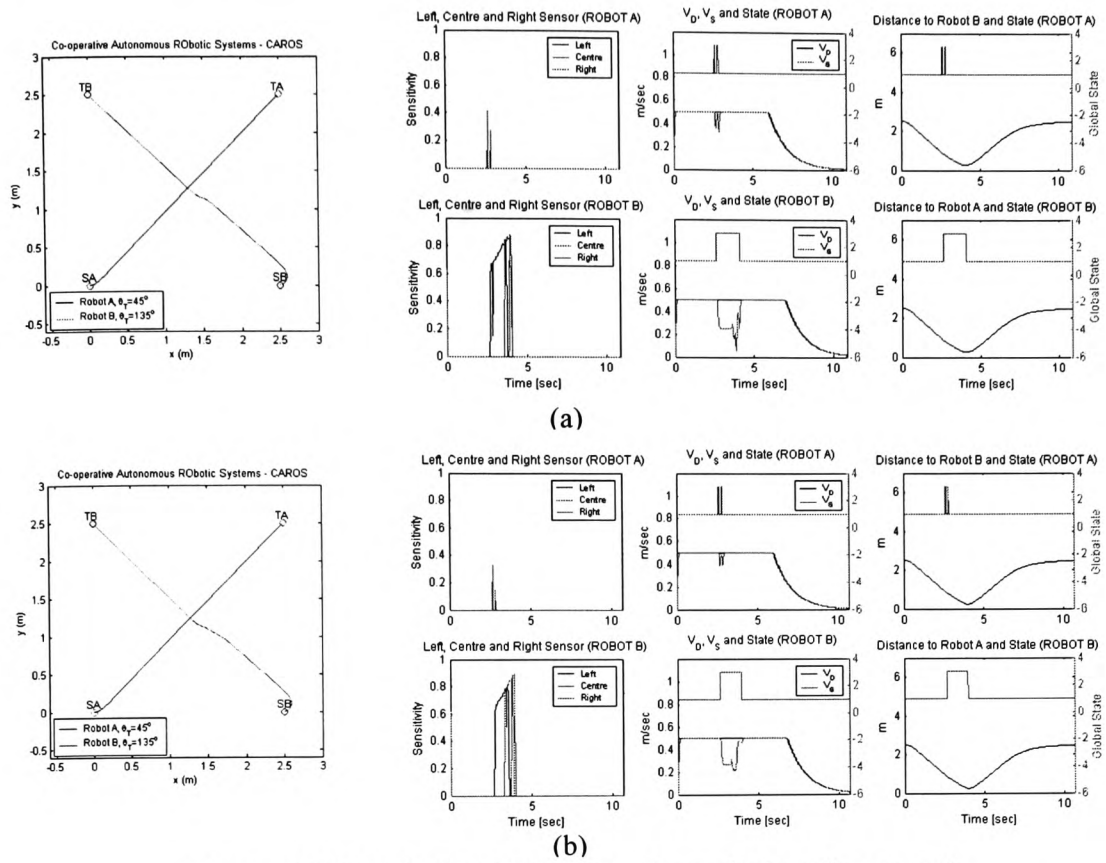
7.3.2.3 Part II: Outcome of Results

Part II investigated the effectiveness of the proposed CAROS control architecture in navigation tasks involving two mobile robots operating in an unknown both static and dynamic environments. The results demonstrate that the control strategy chosen for navigation of multiple mobile robots is efficient. The robustness of the designed control system architecture against desired requirements, such as supervision, decision-making and co-ordination of internal control structures (subsystems) was observed. Both mobile robots achieved every control objective and their trajectory was smooth despite the interaction between several behaviours and presence of unexpected obstacles.

The effectiveness of the proposed method for identification of direction of moving object defined in chapter six was established. The robots found to perform sufficient navigation with travel time and distance travelled by both robots considerably low and very close to optimal.

The effectiveness of the action co-ordinator mechanism developed in chapter six for co-ordination and parallel switching between robot behaviours was validated by several tests. In all tests the proposed control architecture achieve a sufficiently high control performance once the control objective of each robot was reached.

The stateflow-fuzzy logic and stateflow-neural networks behavioural encoding for dynamic obstacle avoidance developed in chapter six were tested in an environment containing different configurations of static and dynamic obstacles. Stateflow-fuzzy behavioural encoding produces better result over stateflow-neural encoding in terms of distance travelled during the navigation tasks. The stateflow-neural behavioural encoding shows better result over stateflow-fuzzy encoding in terms of time taken to complete each mission. The control architecture successfully adapted to the environmental conditions by adjusting the robot's control parameters (i.e. velocity) and taking actions (i.e. activation of appropriate behaviour). The stateflow-fuzzy behaviour encoding controlled better the robot's velocity.



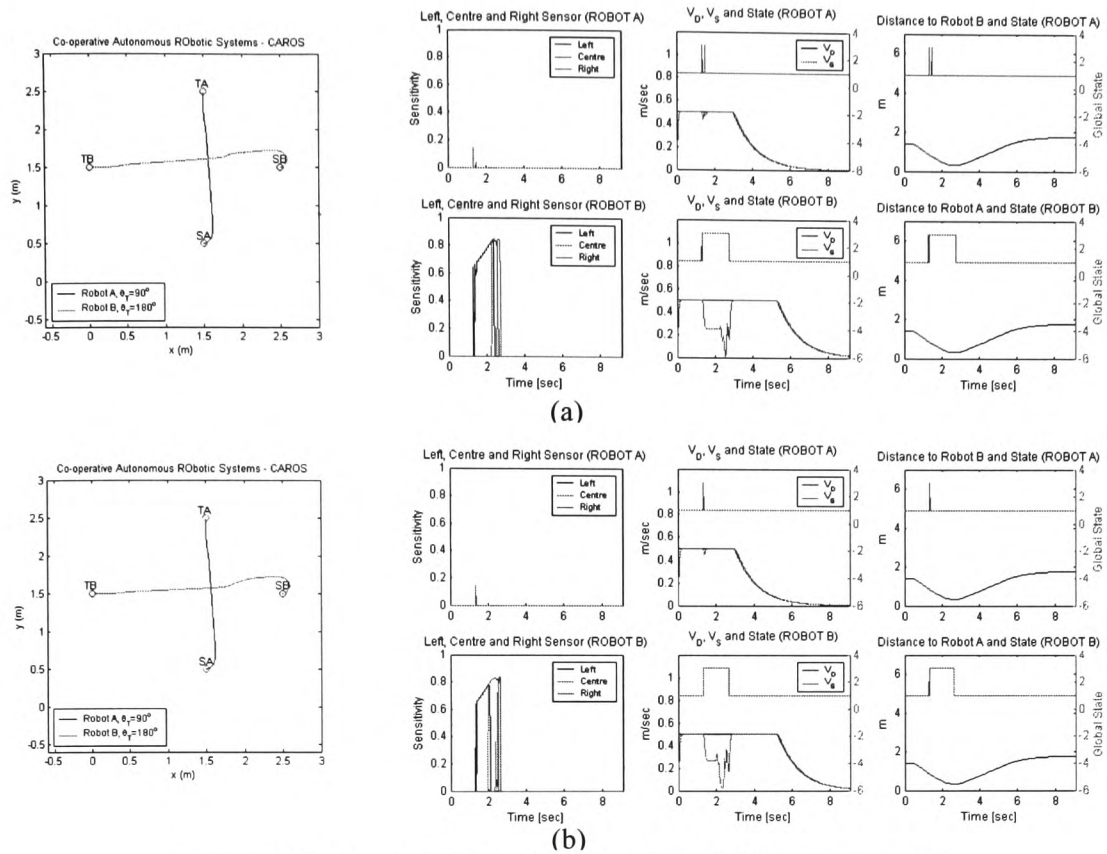


Figure 7-16 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural

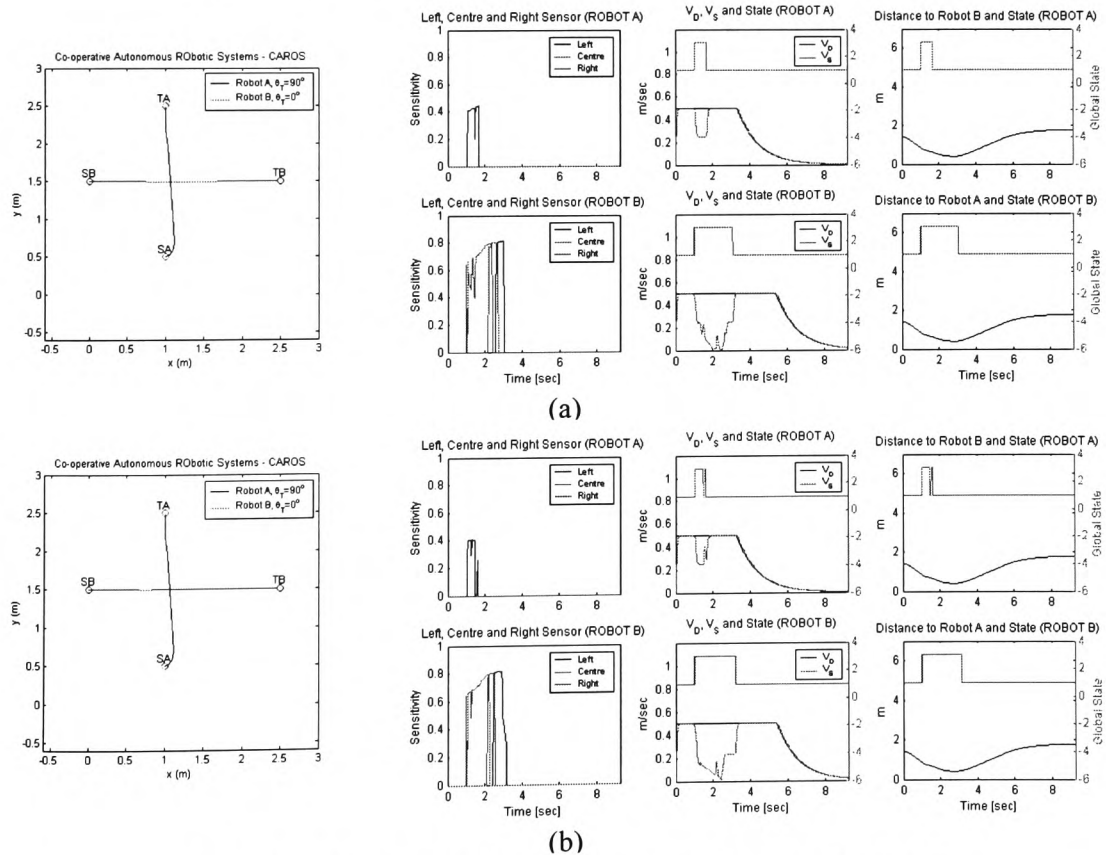


Figure 7-17 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural

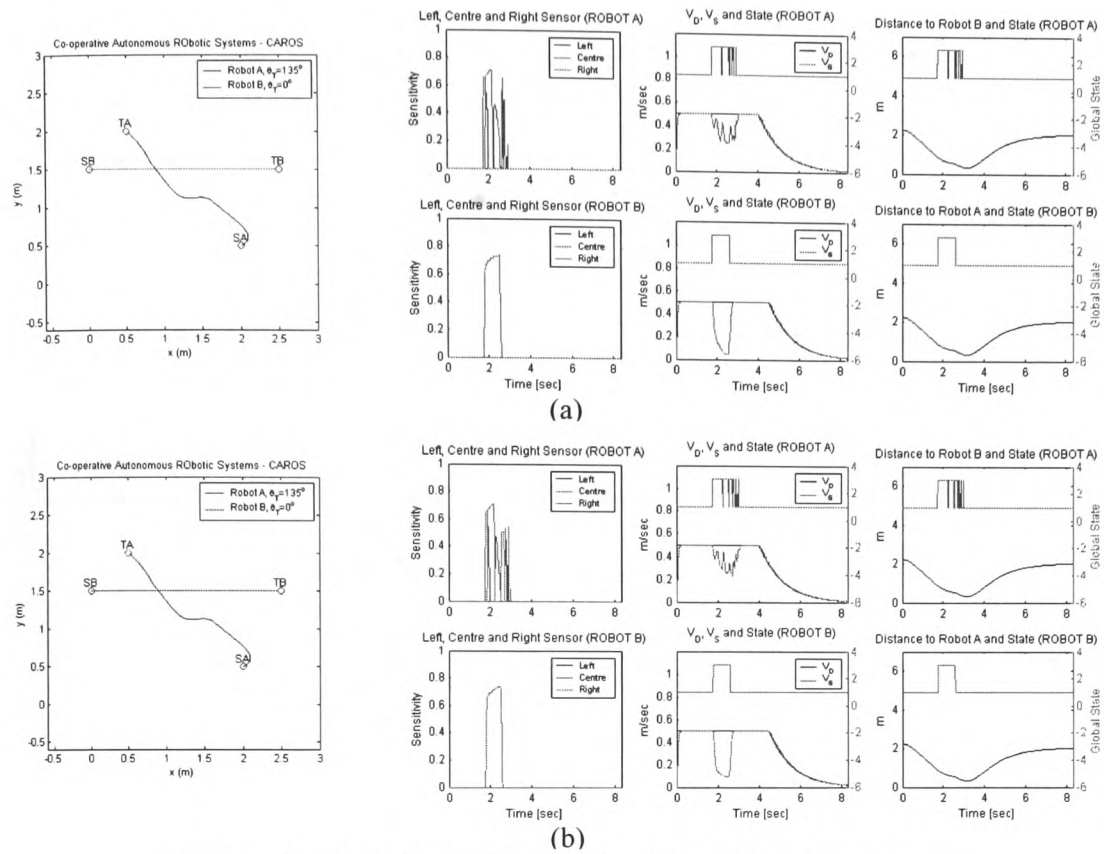


Figure 7-18 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural

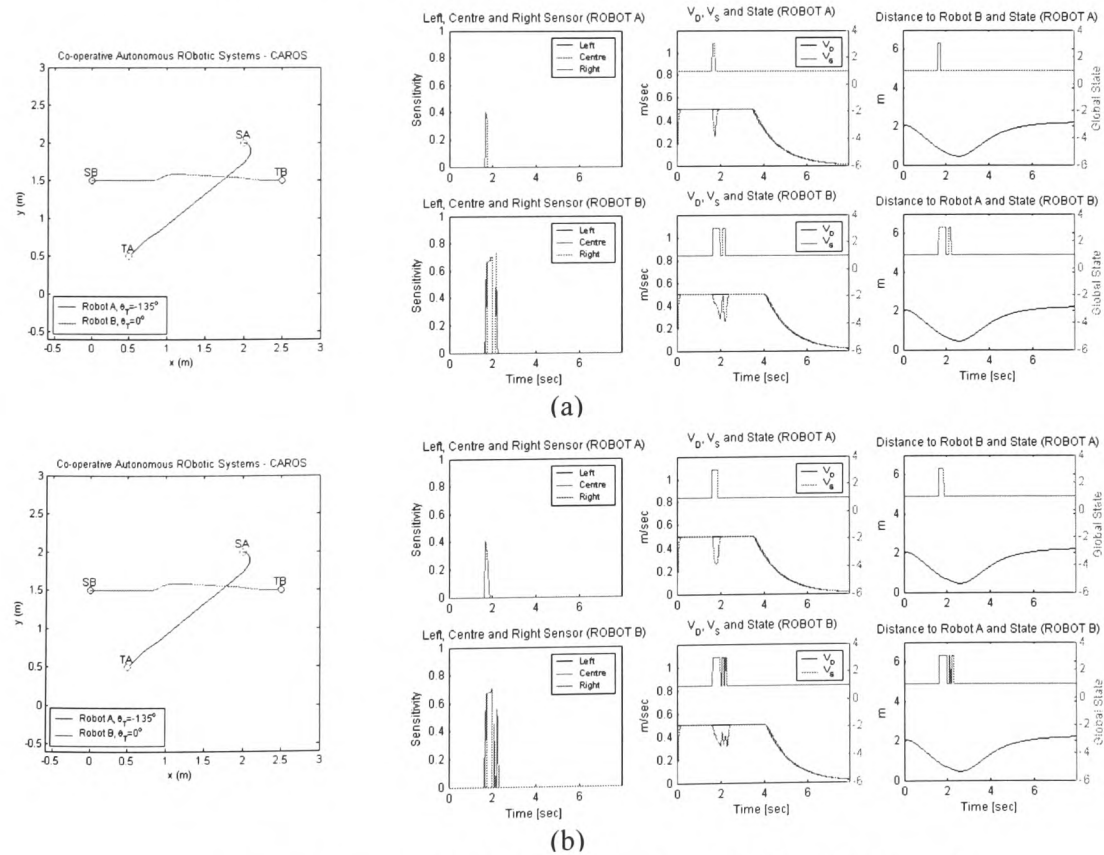


Figure 7-19 Results B1: (a) Stateflow-fuzzy (b) Stateflow-neural

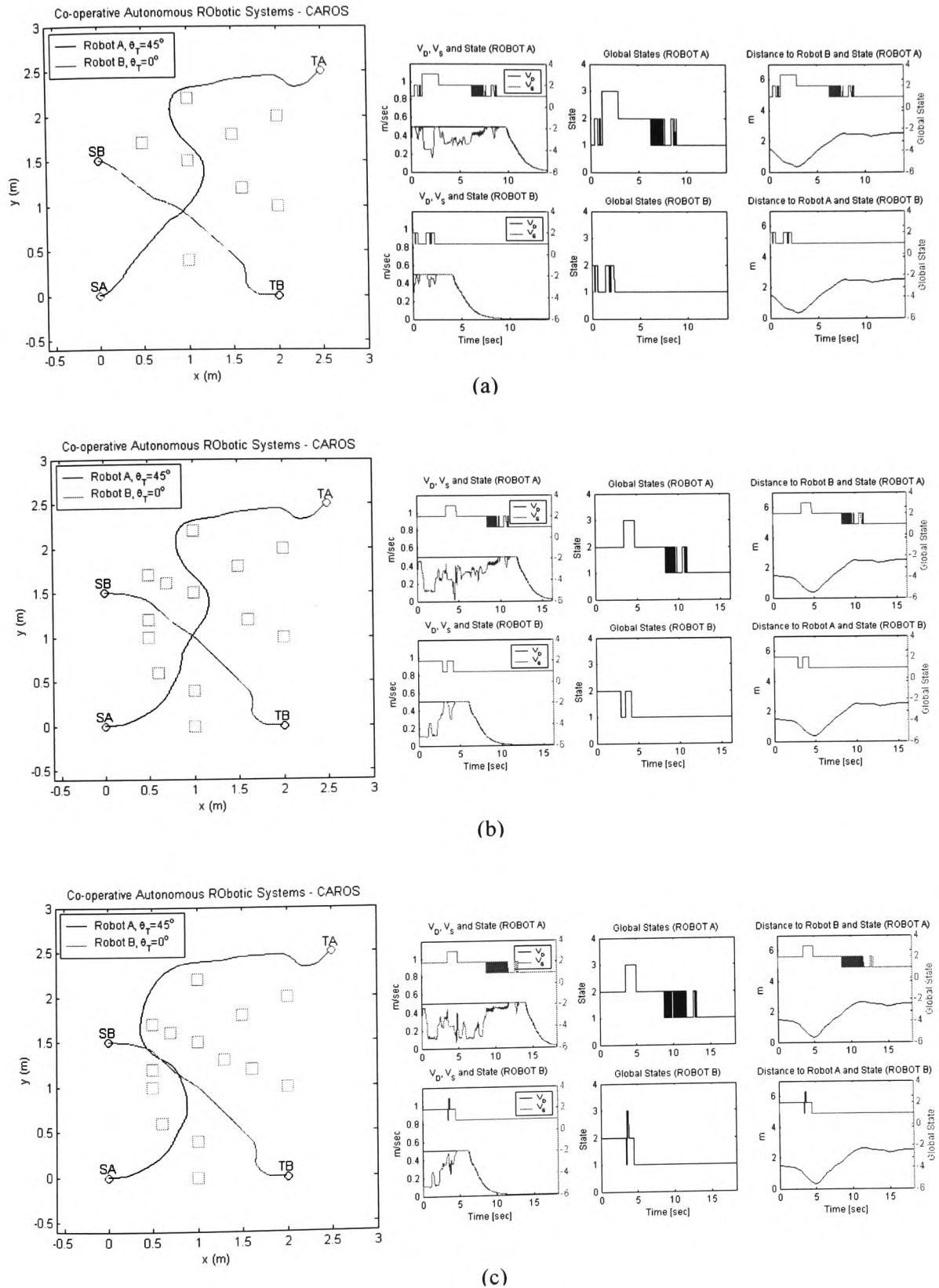


Figure 7-20 Results B2

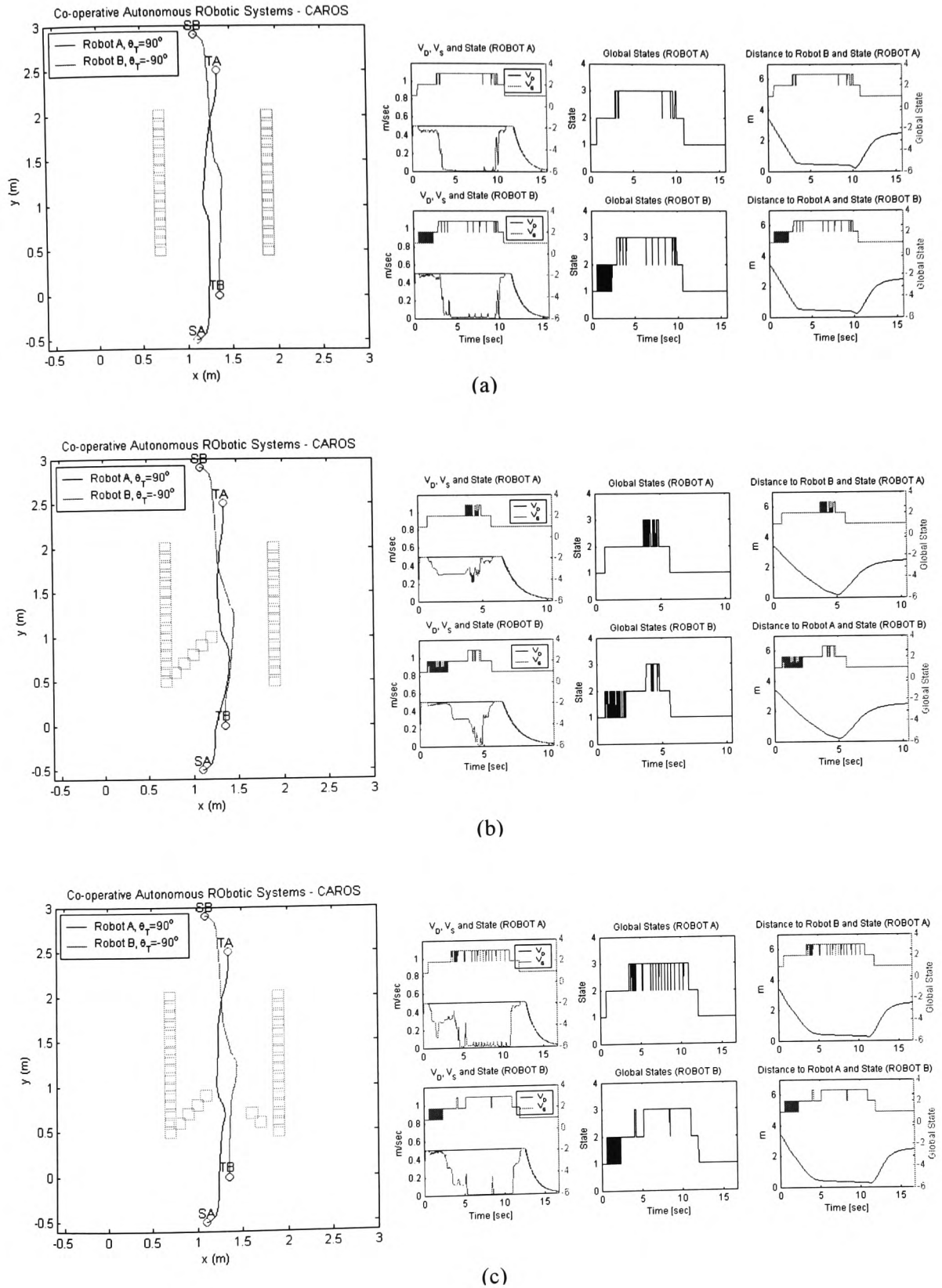


Figure 7-21 Results B2

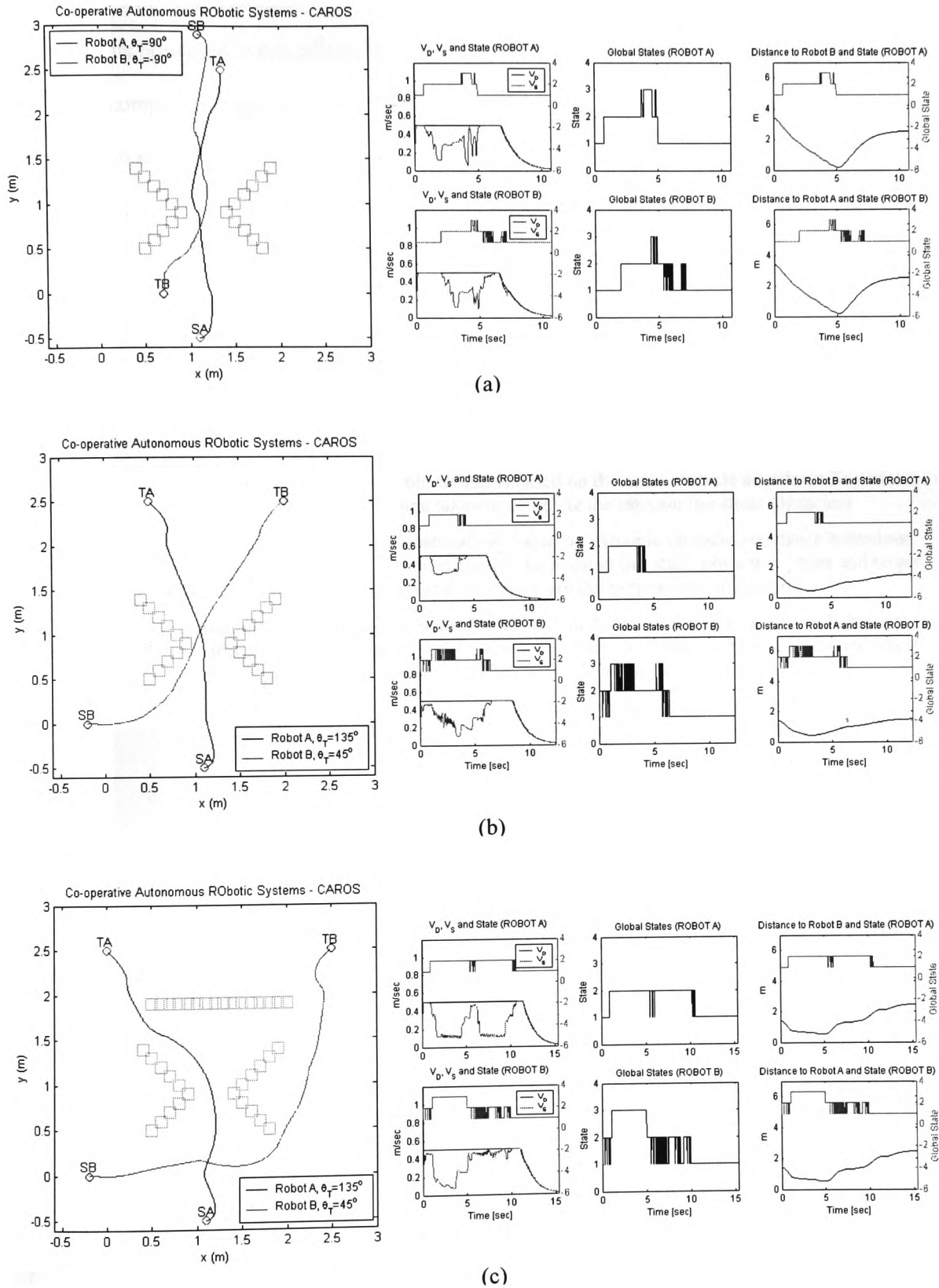


Figure 7-22 Results B2

7.3.3 Part III: Results

The results presented in this section investigate the effectiveness of CAROS control architecture in more complex navigation tasks involving multiple mobile robots (results are presented for up to five robots) operating in a restricted workspace. Part III is divided into six groups of results, there are defined by Table 7-11. Each robot incorporates the control subsystems shown in Table 7-12 for all tests. In the next section results of each group are presented followed by a summary of the main outcome of the performance of the proposed system.

Part III: Division of Results	
Group	Representation
C1	Navigation of three robots with observations based on dynamic obstacle avoidance behaviour, distance travelled and time taken to complete the mission (no static obstacles)
C2	Navigation of four robots with observations based on dynamic obstacle avoidance behaviour, distance travelled and time taken to complete the mission (no static obstacles – start and target positions of the first three robots are the same as in Group C1)
C3	Navigation of five robots with observations based on dynamic obstacle avoidance behaviour, distance travelled and time taken to complete the mission (no static obstacles – start and target positions of the first four robots are the same as in Group C2)
C4	Navigation of three robots with observations based on static and dynamic obstacle avoidance behaviour, distance travelled and time taken to complete the mission (start and target positions are the same as in Group C1)
C5	Navigation of four robots with observations based on static and dynamic obstacle avoidance behaviour, distance travelled and time taken to complete the mission (start and target positions are the same as in Group C2)
C6	Navigation of five robots with observations based on static and dynamic obstacle avoidance behaviour, distance travelled and time taken to complete the mission (start and target positions are the same as in Group C3)

Table 7-11 Division of results (Part III)

Subsystems used in robot navigation			
<i>Sensors</i>	<input checked="" type="checkbox"/> Ultrasonic	<input type="checkbox"/> Infrared	<input type="checkbox"/> No-shape
<i>Static Obstacle Avoidance</i>	<input type="checkbox"/> Fuzzy logic	<input checked="" type="checkbox"/> Neural networks	
<i>Behavioural Encoding</i>			
<i>Dynamic Obstacle Avoidance</i>	<input checked="" type="checkbox"/> Stateflow-fuzzy	<input type="checkbox"/> Stateflow-neural	
<i>Behavioural Encoding</i>			
<i>Speed Control</i>	<input checked="" type="checkbox"/> PI	<input type="checkbox"/> Fuzzy logic	<input type="checkbox"/> Neural networks

Table 7-12 Subsystems used in all mobile robots in Part III

7.3.3.1 Group C1: Results

Results presented in sections 7.3.3.1, 7.3.3.2 and 7.3.3.3 concentrate on navigation of three, four and five mobile robots with observations based on dynamic obstacle avoidance behaviour. In particular the effectiveness of the proposed control architecture is tested with three, four and five mobile robots operating in the same size of working environment as in Part I and Part II but without static obstacles.

Figure 7-23(a) to Figure 7-23(c) depicts three navigation tasks where three mobile robots have to reach independent target points (x_T, y_T) marked by T from an initial position (x_0, y_0) marked by S in an unknown environment free of static obstacles. The results of Group C1 are summarised in Table 7-13. Each result is divided into two figure plots illustrating both the robots trajectories and their control signals (distance to other robots and global state). For instance, Figure 7-23(a), depicts on the left hand side the main plot illustrating the trajectories of three mobile robots A, B and C (x_S, y_S) , initial points and target points. On the right hand side of Figure 7-23(a) the plot is divided into six sub-plots illustrating the following. The first three sub-plots show the distance of each robot with respect to others, sub-plots 4 to 6 show the global state of each robot. The robots orientation at the target point is shown in the legend of the main figure plot illustrating the robots trajectories.

Group C1					
Exp	Robot	Distance travelled	Time taken	Collision	Caption
1	A	3.5546	10.71	No	Figure 7-23(a)
	B	3.7009	11.35	No	
	C	2.5509	8.61	No	
2	A	2.5477	9.39	No	Figure 7-23(b)
	B	2.7868	9.62	No	
	C	2.5477	7.83	No	
3	A	2.5409	9.08	No	Figure 7-23(c)
	B	2.7864	9.73	No	
	C	2.5481	7.83	No	

Table 7-13 Collision avoidance result of an environment containing three mobile robots

As can be observed from Figure 7-23(a) to Figure 7-23(c) in all three navigation tasks no collision between the mobile robots occurred. The distance travelled and time taken of each robot to complete the mission is recorded for comparison with the results from Group C2 presented in the next section.

7.3.3.2 Group C2: Results

The results of this section are compared with the simulation results from Group C1.

Figure 7-24(a) to Figure 7-24(c) depicts three navigation tasks where the four mobile robots have to reach independent target points (x_T, y_T) marked by T from an initial position (x_0, y_0) marked by S in an unknown environment free of static obstacles. In particular each experiment is divided into two figure plots illustrating both the robots trajectories and their control signals (distance to other robots and global state). For instance, the results illustrated in Figure 7-24(a), depicts on the left hand side the main plot of four mobile robots A, B, C and D illustrating their trajectories (x_S, y_S) , initial points and target points. On the right hand side of Figure 7-24(a) the plot is divided into eight sub-plots illustrating the following. First four subplots show the distance of each robot in respect to other robots whereas sub-plot 5 to 8 shows the global state of each robot. Again as in the previous section the robots desired orientation at the target point is shown as a legend in the main figure plot illustrating the robots trajectories.

As can be observed from Figure 7-24(a) to Figure 7-24(c) in all three navigation tasks no collision between the mobile robots was reported. The distance travelled and time taken of each robot to complete the mission have been compared with the simulation results from Group C1.

The results of this group are summarised in Table 7-14. As can be seen from Table 7-14 the recorded travelled distance of each robot in navigation tasks involving four robots is shorter

than the recorded travelled distance of each robot in navigation tasks involving three robots. However, the time taken for each robot to complete its mission is smaller in the case of navigation involving three robots. The delay in time taken for each robot to complete its mission is a result of longer negotiations between robots. The concept of traffic rules used to model the robots behaviour for dynamic obstacle avoidance, means that conflicting mobile robots will decelerate and will negotiate with each other for longer time to decide which has a priority (note that the negotiation taking place between robots is without inter-communication). The improved result in travelled distance for each robot (for four mobile robots) is because the navigation task with three robots requires fewer negotiations with other robots and therefore resulting in longer distance by finding the easier way to reach the target. Whereas for four robots, each robot has more negotiation with other robots and therefore executes shorter distance by finding its way towards the target after the majority of the conflicts have been solved.

Group C2					
Exp	Robot	Distance travelled	Time taken	Collision	Caption
1	A	↓ 3.5351 (-0.0195)	↑ 11.47 (0.76)	No	Figure 7-24(a)
	B	↓ 3.6899 (-0.011)	↑ 12.9 (1.55)	No	
	C	↓ 2.5519 (-0.0001)	↑ 8.67 (0.06)	No	
	D	2.4949	9.14	No	
2	A	↓ 2.4873 (-0.0604)	↑ 11.41 (2.02)	No	Figure 7-24(b)
	B	↓ 2.7964 (-0.0096)	↑ 10.54 (0.92)	No	
	C	↓ 2.5502 (-0.0043)	↑ 8.68 (0.85)	No	
	D	3.5302	10.26	No	
3	A	↓ 2.5432 (-0.0023)	↑ 9.12 (0.04)	No	Figure 7-24(c)
	B	↓ 2.7856 (-0.0008)	↑ 10 (0.27)	No	
	C	↓ 2.5426 (-0.0055)	↑ 9.1 (1.27)	No	
	D	2.9086	9.54	No	

Table 7-14 Collision avoidance result of an environment containing four mobile robots

In the next group of results, experiments 2 and 3 from this section are repeated using five mobile robots.

7.3.3.3 Group C3: Results

The results of this section are compared with the results from the last two experiments from Group C1.

Figure 7-25(a) and Figure 7-25(b) depicts two navigation tasks in which five mobile robots have to reach independent target points (x_T, y_T) marked by T from an initial position (x_0, y_0) marked by S in an unknown environment free of static obstacles. In particular each experiment is divided into two figure plots illustrating both the robots trajectories and their control signals (distance to other robots and global state). The experiment illustrated in both Figure 7-25(a) and Figure 7-25(b), depicts on the top side the main plot of five mobile robots A, B, C, D and E illustrating their trajectories (x_S, y_S) , initial points and target points. On the bottom side of Figure 7-25(a) and Figure 7-25(b) the plots are divided into ten sub-plots illustrating the following. First five sub-plots show the distance of each robot in respect to other robots whereas sub-plot 6 to 10 shows the global state of each robot.

As can be observed from Figure 7-25(a) and Figure 7-25(b) in both navigation tasks no collision between the mobile robots was reported as the distance plots between the five mobile robots shows value always greater than zero. The distance travelled and time taken of each robot to complete the mission have been compared with the results from Group C1.

The results of Group C3 are summarised in Table 7-15. As can be seen from Table 7-15 the recorded travelled distance of each robot in navigation tasks involving five robots is higher than the recorded travelled distance of each robot in navigation tasks involving three robots. In addition, the time taken for each robot to complete its mission is also greater in the case of navigation involving five robots. The delays in both distance travelled and time taken for each robot to complete its mission are acceptable which is a result of longer negotiations between

robots in a restricted workspace. The results in this section demonstrated once again the efficiency and robustness of the control architectures. All mobile robots achieved every control objective and their trajectories are smooth in spite of the interaction with other robots.

Group C3					
Exp	Robot	Distance travelled	Time taken	Collision	Caption
1	A	↓ 2.5553 (-0.0076)	↑ 9.46 (0.09)	No	Figure 7-25(a)
	B	↑ 2.8328 (0.046)	↑ 11.01 (1.39)	No	
	C	↑ 2.7230 (0.1753)	↑ 8.77 (0.94)	No	
	D	↑ 4.3738 (1.8789)	↑ 15.79 (5.53)	No	
	E	3.3467	11.01	No	
2	A	↑ 2.5537 (0.0128)	↑ 9.12 (0.04)	No	Figure 7-25(b)
	B	↑ 2.8066 (0.0202)	↑ 10 (0.27)	No	
	C	↑ 2.5530 (0.0049)	↑ 9.1 (1.27)	No	
	D	↓ 2.9232 (-0.0628)	→ 9.54 (-)	No	
	E	3.5729	14.37	No	

Table 7-15 Collision avoidance result of an environment containing five mobile robots

The next group of results demonstrates the efficiency of the proposed control architecture in a highly dynamic environment.

7.3.3.4 Group C4: Results

As previously mentioned real-time motion planning in an unknown and unstructured environment involves collision avoidance of static as well as moving obstacles (other robots). However, strategies suitable for navigating two mobile robots in a dynamic environment are not necessary suitable for strategies for navigation of three or more mobile robots. This section and sections 7.3.3.5 and 7.3.3.6 demonstrate the control architecture's applicability to multiple robot navigation operating in an environment populated by static and dynamic obstacles. In particular the navigation tasks of the three, four and five mobile robots considered in Group C1, C2 and C3 are repeated with the addition of static obstacles within the robots working environment.

Figure 7-26(a) to Figure 7-26(b) depicts three navigation tasks where three mobile robots have to reach independent target points an initial in an unknown environment populated by static and dynamic obstacles. The graphical representation of each result is the same as in Group C1.

As can be observed from Figure 7-26(a) to Figure 7-26(b) in all three navigation tasks no collision between the mobile robots occurred. In addition no collision between the robots and the static obstacles occurred..

Table 7-16 summarises the collision avoidance results of all three navigation tasks performed by the robots. The distance travelled and time taken of each robot to complete the mission is similar to those in Group C1

Group C4						
Exp	Robot	Distance travelled	Time taken	Collision Avoidance Result		Caption
				Static Obstacle	Dynamic Obstacle	
1	A	3.9929	11.41	No	No	Figure 7-26(a)
	B	4.0655	11.43	No	No	
	C	3.5375	10.59	No	No	
2	A	2.9435	9.23	No	No	Figure 7-26(b)
	B	3.0882	10.13	No	No	
	C	3.6906	11.56	No	No	
3	A	2.6016	9.51	No	No	Figure 7-26(c)
	B	3.0912	10.13	No	No	
	C	3.6940	12.32	No	No	

Table 7-16 Collision avoidance result of a mixed environment containing static obstacles and three mobile robots

7.3.3.5 Group C5: Results

Figure 7-27(a) to Figure 7-27(b) depicts three navigation tasks in which four mobile robots have to reach independent target points an initial in an unknown environment populated by static and dynamic obstacles. The graphical representation of each result is the same as in Group C2.

As can be observed from Figure 7-27(a) to Figure 7-27(b) in all three navigation tasks no collision between the mobile robots occurred. Again no collision between the mobile robots and the static obstacles occurred.

Table 7-17 summarises the collision avoidance results of all three navigation tasks performed by the robots. The distance travelled and time taken of each robot to complete the mission is similar to those in Group C2.

Group C5						
Exp	Robot	Distance travelled	Time taken	Collision Avoidance Result		Caption
				Static Obstacle	Dynamic Obstacle	
1	A	4.0108	11.41	No	No	Figure 7-27(a)
	B	3.8778	13.5	No	No	
	C	3.0163	10.5	No	No	
	D	3.0570	9.5	No	No	
2	A	3.0555	9.5	No	No	Figure 7-27(b)
	B	2.9387	9.83	No	No	
	C	3.6510	11.78	No	No	
	D	3.9986	11.41	No	No	
3	A	4.3139	14.93	No	No	Figure 7-27(c)
	B	3.1175	10.52	No	No	
	C	3.6802	13.25	No	No	
	D	3.0688	12.34	No	No	

Table 7-17 Collision avoidance result of a mixed environment containing static obstacles and four mobile robots

7.3.3.6 Group C6: Results

Figure 7-28(a) to Figure 7-28(b) depicts two navigation tasks where five mobile robots have to reach independent target points an initial in an unknown environment populated by static and dynamic obstacles. The graphical representation of each result is the same as in Group C3.

As can be observed from Figure 7-28(a) to Figure 7-28(b) in both navigation tasks no collision between the mobile robots was reported. However, for this particular example the mobile robots have to pass each other very close. In addition no collision between the mobile robots and the static obstacles occurred.

Table 7-18 summarises the collision avoidance results of both navigation tasks performed by the robots. The distance travelled and time taken of each robot to complete the mission is considerably higher than those in Group C3. For all tests considered activation of *avoid_trap* robot behaviour did not occur, which means that the implementation of the standalone robot behaviours is sufficient to successfully navigate the robots.

Group C6						
Exp	Robot	Distance travelled	Time taken	Collision Avoidance Result		Caption
				Static Obstacle	Dynamic Obstacle	
1	A	3.0634	11.76	No	No	Figure 7-28(a)
	B	2.9357	10.16	No	No	
	C	3.0880	11.31	No	No	
	D	4.4927	15.24	No	No	
	E	4.2751	15.06	No	No	
2	A	2.5564	10.33	No	No	Figure 7-28(b)
	B	2.8977	10.49	No	No	
	C	3.1094	12.29	No	No	
	D	2.9668	11.41	No	No	
	E	3.7085	16.05	No	No	

Table 7-18 Collision avoidance result of a mixed environment containing static obstacles and five mobile robots

7.3.3.7 Part III: Outcome of results

Part III has investigated the effectiveness of the proposed CAROS control architecture in navigation tasks involving three to five mobile robots operating in an unknown both static and dynamic environments. The results in Part III demonstrated that the control strategy chosen for

navigation of multiple mobile robots is efficient and also re-established the robustness of the control system architecture against the desired requirements, such as supervision, decision-making and co-ordination of internal control structures (subsystems). The mobile robots were exposed in a complex and highly dynamic environment and successfully achieved every control objective, as their trajectories were smooth despite the interaction between several behaviours and presence of unexpected static and dynamic obstacles.

The effectiveness of the action co-ordinator mechanism developed in chapter six for co-ordination and parallel switching between robot behaviours was validated. In all experiments the control objective of each robot was successfully reached demonstrating that the proposed control architecture achieve a good control performance. The mobile robots were found to navigate with travel time and distance travelled low and close to optimal.

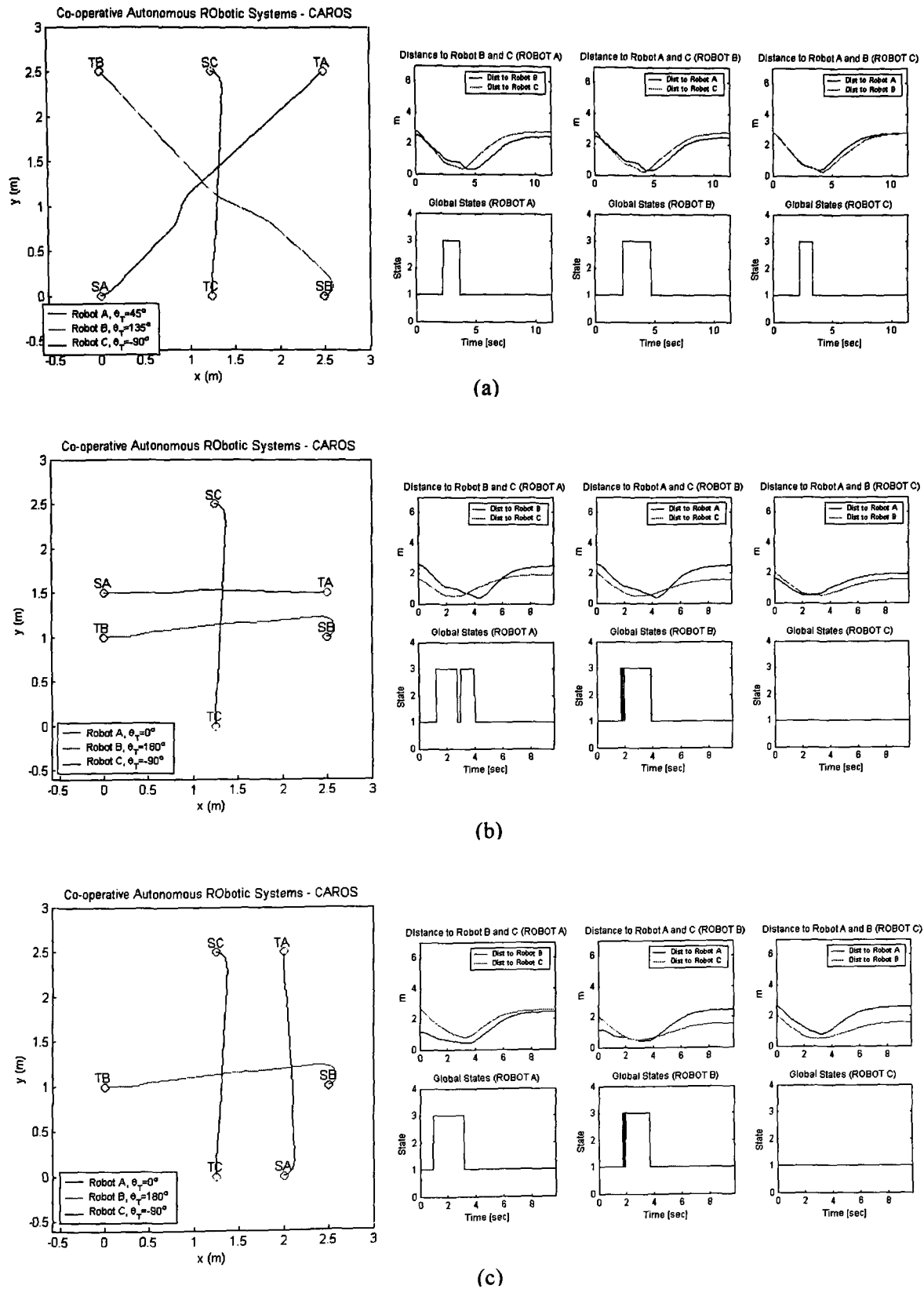


Figure 7-23 Results C1

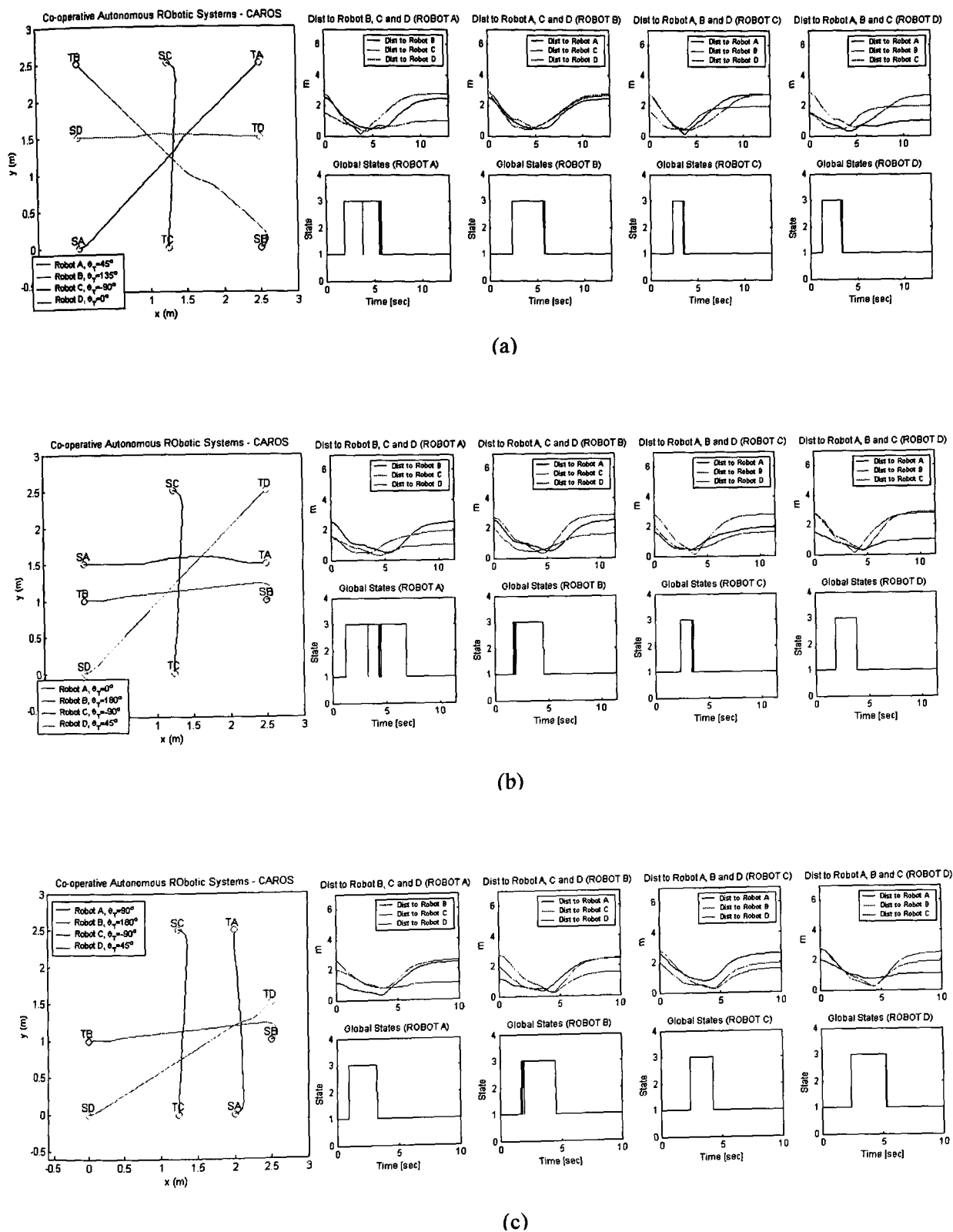


Figure 7-24 Results C2

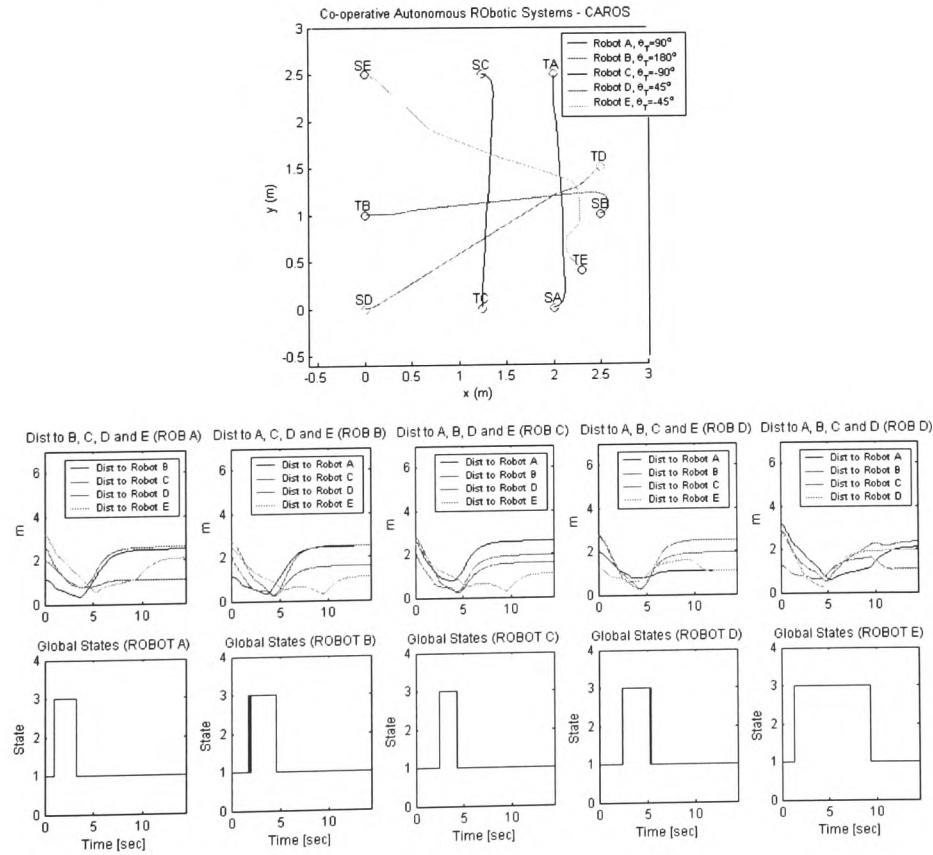
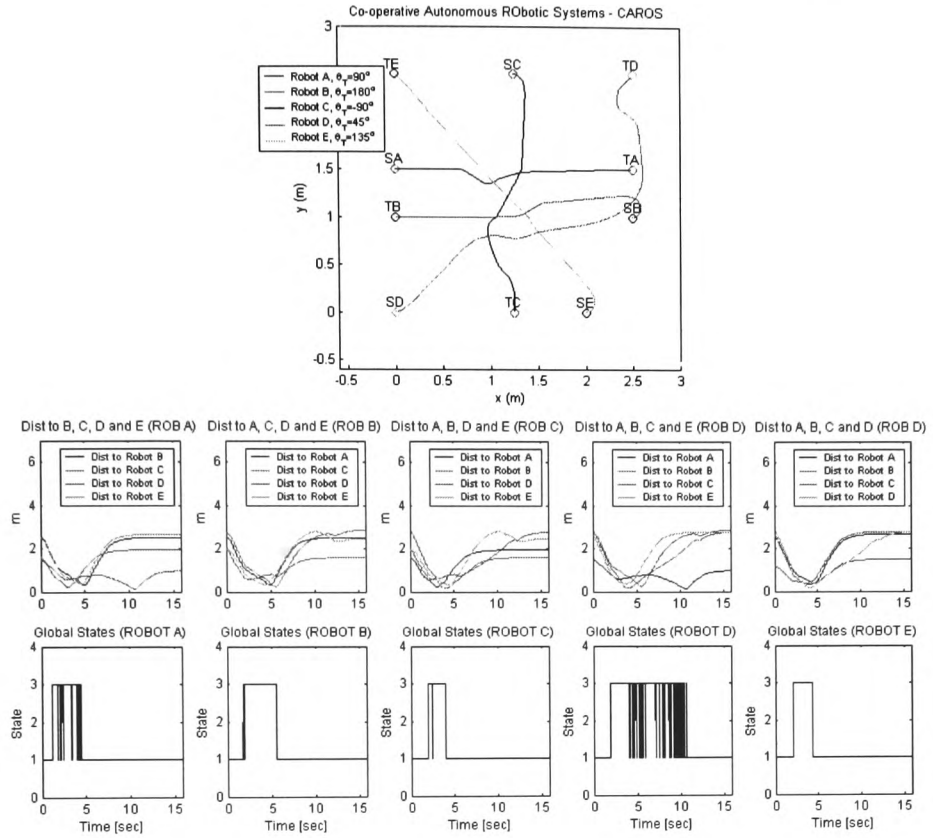


Figure 7-25 Results C3

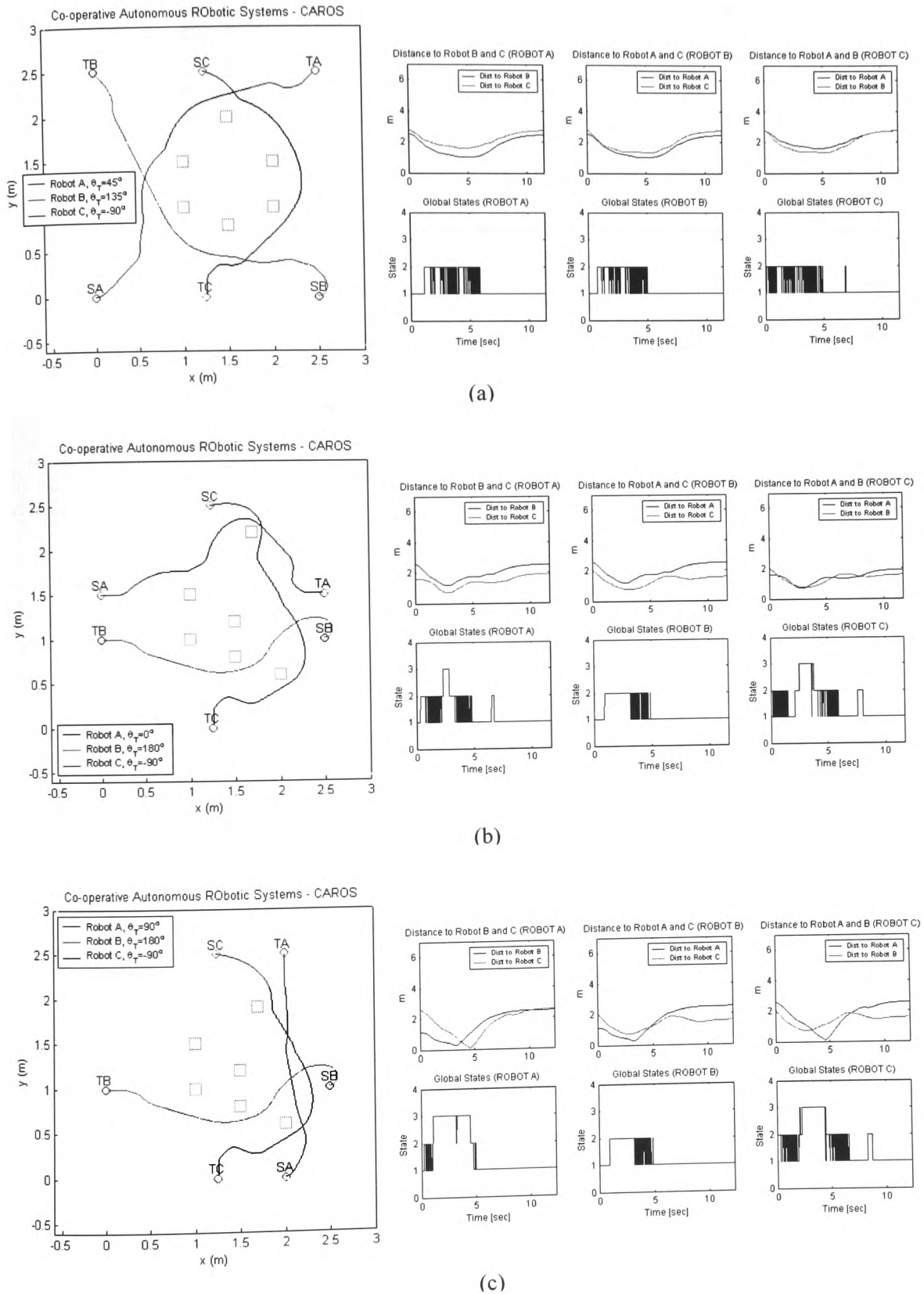
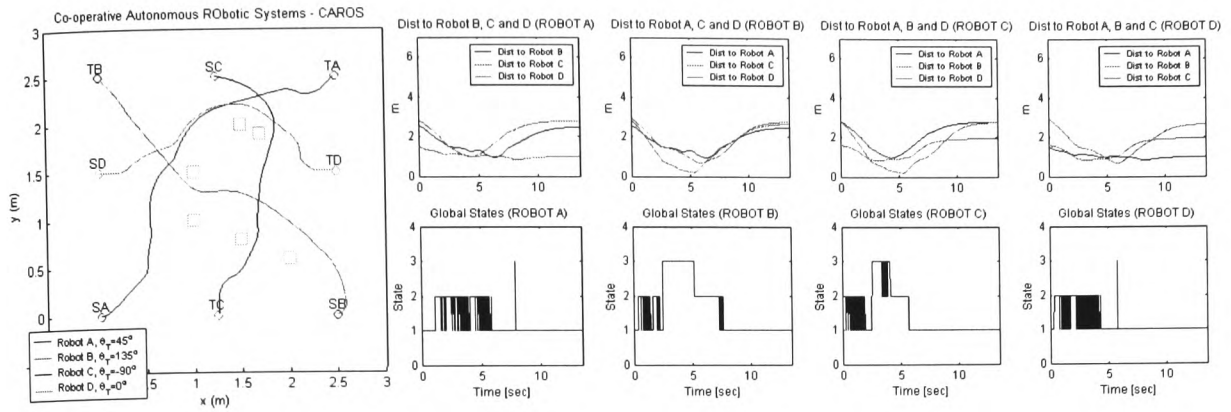
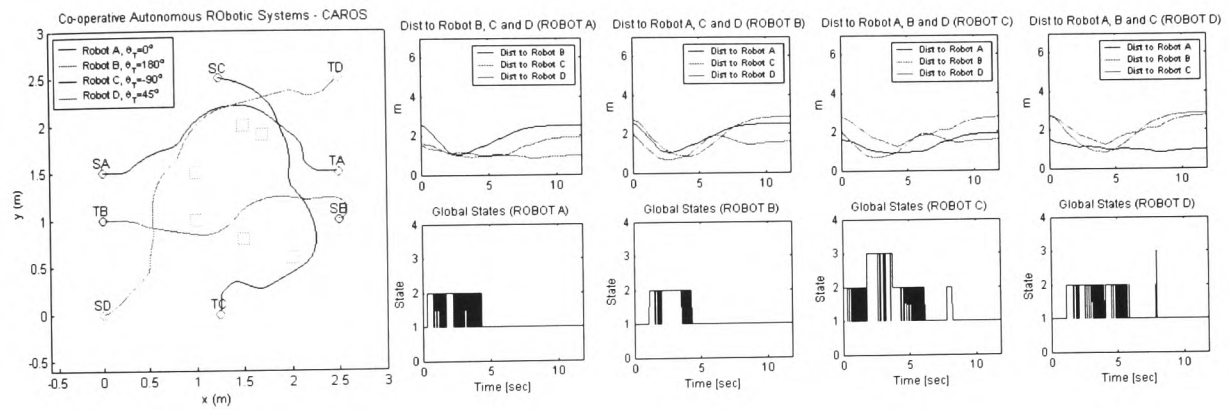


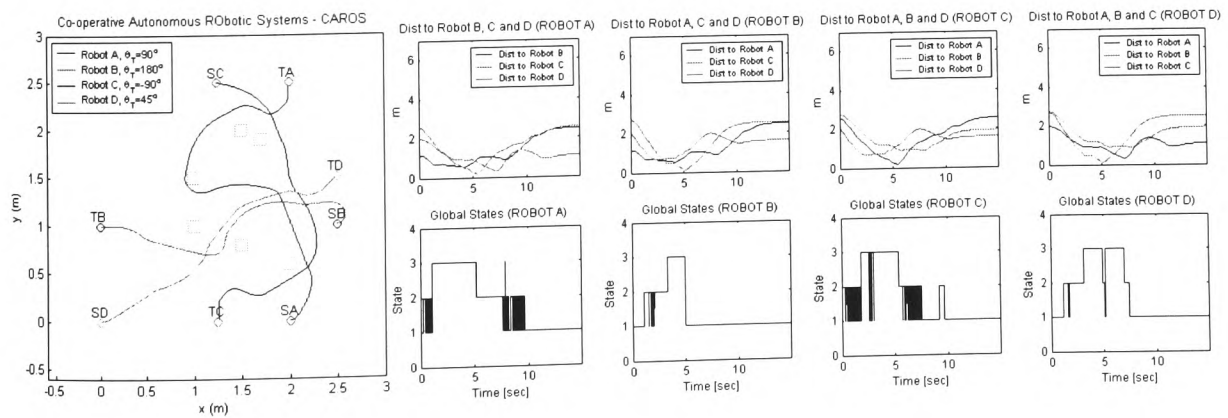
Figure 7-26 Results C4



(a)



(b)



(c)

Figure 7-27 Results C5

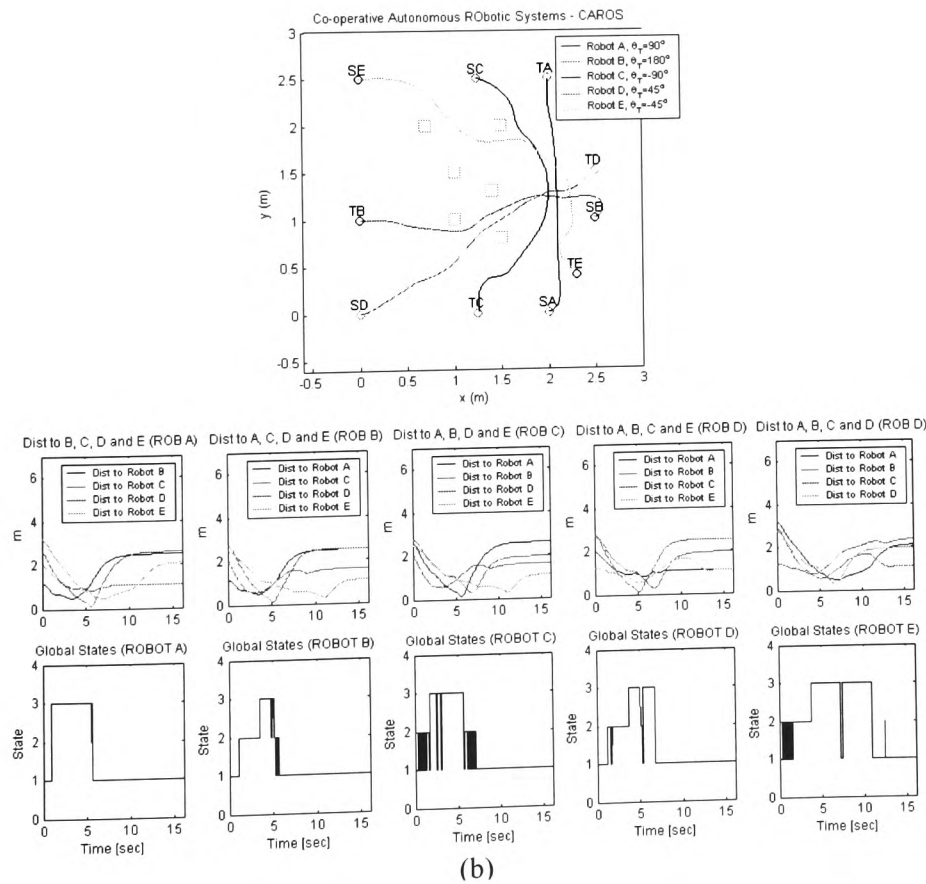
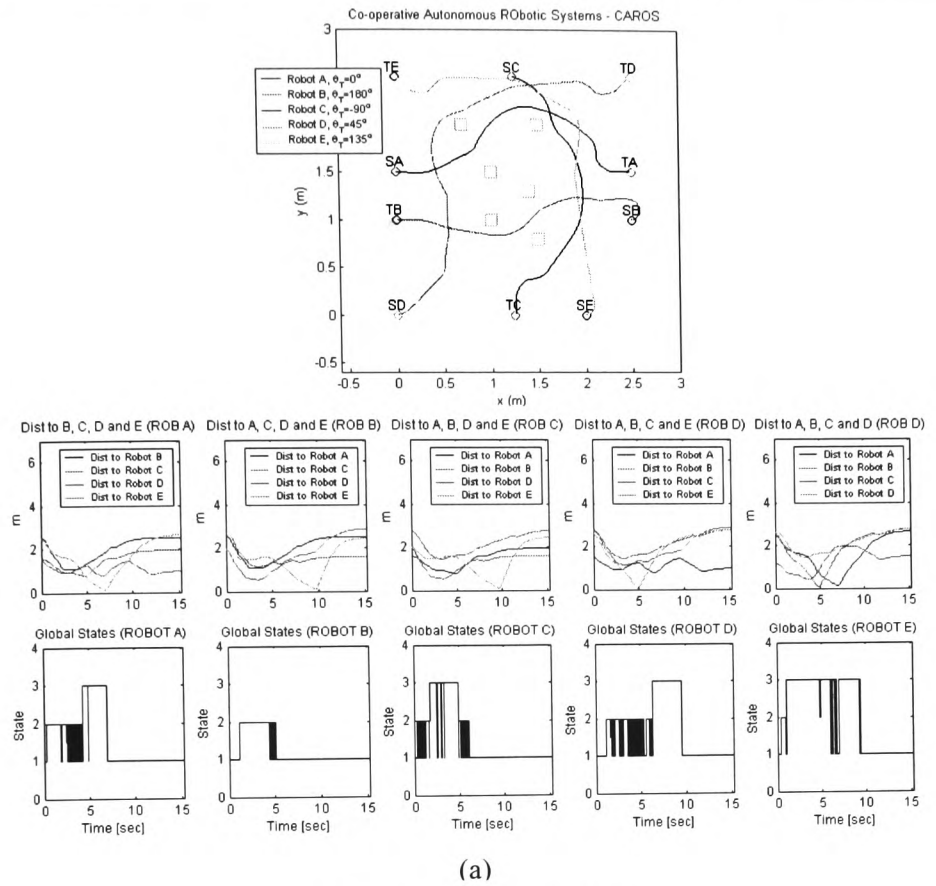


Figure 7-28 Results C6

7.4 Discussion

The main outcomes of all results presented in this chapter were discussed after the completion of the three parts. This section will give some general discussions based on the overall control architecture performance presented in this chapter.

The control architecture was tested incrementally in order to verify its overall control performance and the performance of each subsystem. The subsystem by subsystem testing of the control architecture proved useful in the achievement of system performance while issues related to subsystems integration and co-ordination were successfully validated.

Results show that the control architecture's modularity, distribution, reactivity and behaviour based structure provided the overall control system with robustness in all cases of navigation tasks utilising either single or multiple mobile robots. The experiments carried out for navigation of autonomous mobile robots proved that the modular design of the control architecture eased the testing for performance by allowing individual and integrated subsystem testing, thereby reducing the problems inherent in the design, integration and performance test of such complex systems. The results presented not only reveal the advantages of the proposed control architecture, but also identify answers to other issues in mobile robot navigation such as performance of sensor sensitivity, performance of proposed method for identification of direction of moving object and performance of behavioural encoding utilising fuzzy, neural or hybrid solutions.

It was shown that using the concept of behaviour-based control, a complex task can be easily completed implementing several simple behaviours. The experiments have demonstrated that in a complex navigation task potential difficulties may arise from the quantitative formulation of reactive behaviour as well as from the need for an efficient co-ordination and integration of conflicts and competition among different types of behavioural encoding. The results show that

the study undertaken in chapter six for the design of the action co-ordinator mechanism using finite state machines, the implementation of the controller-agent concept from the field of multi-agent systems and the use of fuzzy logic and neural networks for behavioural encoding successfully compensates and overcomes the aforementioned difficulties.

Results show that the proposed hybrid system, using only three sensors, can be successfully applied to multiple robot navigation in complex and highly dynamic environments by efficiently co-ordinating multiple types of robot behaviours and local controllers organised by means of agents.

7.5 Summary

This chapter presents evaluation of the proposed novel hybrid multi-agent control architecture developed in this thesis for navigation of multiple autonomous mobile robots. The control architecture has been verified and validated when applied to the problem of navigation of single/multiple autonomous mobile robots using the full non-linear model of the MIABOT V2 mobile robot derived and described in chapter four.

In Part I the effectiveness of the proposed CAROS control architecture in a single mobile robot navigation operating in an unknown environment populated by static obstacles was considered. The proposed control architecture was observed to exhibit robustness, adaptability and flexibility when tested in single robot navigation in an unknown environment populated by static obstacles. The mobile robot achieved every control objective and its trajectory was smooth despite the interaction between several behaviours and the presence of unexpected obstacles. The effectiveness of SOAM using the controller-agent concept was validated with several numbers of tests. Three speed controllers were tested and the results show that PI and fuzzy speed controllers have a marginal advantage against the neural controller. The different types of sensor sensitivity developed in chapter six were tested according to the distance

travelled and time taken for the robot to complete the mission. All sensors successfully navigated the mobile robot towards the target without any collision with the static objects. Fuzzy logic and neural networks behavioural encoding for static obstacle avoidance developed in chapter six were tested in an environment containing different configurations of static obstacles. Fuzzy behavioural encoding produced marginally better result over neural encoding in terms of distance travelled during the navigation tasks. The neural networks behavioural encoding showed a better result over fuzzy encoding in terms of time taken for the robot to complete each mission.

In Part II the effectiveness of CAROS control architecture in navigation tasks involving two mobile robots operating in an unknown both static and dynamic environment was investigated. The results demonstrated that the control strategy chosen for navigation of multiple mobile robots is efficient and also the robustness of the designed control system architecture against desired requirements, such as, supervision, decision-making and co-ordination of internal control structures (subsystems) was successfully observed. The results obtained established the effectiveness of the proposed method for identification of direction of moving object defined in chapter six. The stateflow-fuzzy logic and stateflow-neural networks behavioural encoding for dynamic obstacle avoidance developed in chapter six were tested in an environment containing different configurations of static and dynamic obstacles. Stateflow-fuzzy behavioural encoding produced better result over stateflow-neural encoding in terms of distance travelled during the navigation tasks. The stateflow-neural behavioural encoding showed better result over stateflow-fuzzy encoding in terms of time taken to complete each mission. The control architecture successfully adapted to the environmental conditions.

In Part III the effectiveness of CAROS control architecture in more complex navigation tasks involving multiple mobile robots (results were presented up to five robots) operating in a restricted workspace. The mobile robots were exposed in a complex and highly dynamic

environment and successfully achieved every control objective, as their trajectories were smooth despite the interaction between several behaviours and the presence of unexpected static and dynamic obstacles. In all tests the proposed control architecture achieved a high control performance. The mobile robots were found to navigate effectively with travel time and distance travelled low and very close to optimal.

The main contribution of this chapter is the implementation and evaluation of novel hybrid multi-agent control architecture for navigation of single/multiple autonomous mobile robots in both static and/or dynamic environment.

The next chapter presents review, conclusions and future work of the thesis.

8

Review, Conclusions and Future Work

8.1 Introduction

In this chapter a review of the thesis is presented, some conclusions are drawn, the main contributions of the research are discussed and future work is recommended.

As mentioned at the introduction of the thesis, autonomous control and navigation of mobile robotic vehicles are fundamental enabling technologies for automation in a variety of operating domains ranging from industrial environments to remote planetary surfaces. Therefore navigation is a vital issue in the research of autonomous mobile robots. The navigation of an autonomous mobile robot may be considered as a task of determining a collision free path that enables the robot to travel through course, populated with obstacles, from an initial configuration to a target configuration, where configuration here refers to the spatial co-ordinate and the heading angle of the robot. The ultimate goal of the research in mobile robot systems is

to develop control strategies and architectures, which support autonomous operations in an unknown or partially known environment.

It was also mentioned at the introduction of the thesis that advance mobile robotic systems (architectures) must combine deliberative planning with reactive sensor driven operations. This has been recently accepted as advance mobile robotic systems operating in uncertain dynamic environments, combine information from several sensory sources. Prior knowledge of the domain may be incomplete, and reasoning must be deliberative in nature and fast enough to respond to unexpected events. Also, the information gained via sensory subsystems is incomplete, inaccurate and uncertain.

In this thesis a novel hybrid multi-agent oriented control architecture for navigation of multiple autonomous mobile robots operating in an unknown and unstructured environment populated by static and/or dynamic obstacles is proposed. The control architecture integrates deliberative planning and reactive control with attention focused on the design and co-ordination of robot behaviours.

The remainder of this chapter is organised as follows: Section 8.2 presents review of the research described in the thesis. Discussion and the main conclusions drawn of the research are illustrated in section 8.3. In section 8.4 the main contributions of the research described in this thesis are discussed. Suggestions for future work are presented in section 8.5.

8.2 Review of the thesis

In the introductory chapter of the thesis the motivation of conducting the research described in this thesis was given. Some potential application areas for the results of this thesis were suggested and the aim and objectives of the research carried out were highlighted. Then the main challenges and problems in mobile robot navigation with brief discussion of how they

have been approach in this thesis were discussed. An outline of the thesis was presented with a brief summary of the main contributions resulted of the research work described in the thesis.

Chapter two presented a background literature review of topics related to this thesis. The work in this thesis has shared motivations and goals with a number of related fields. The first topic to be reviewed was the field of artificial life. A number of contributions from artificial life based on simulations of multiple autonomous mobile robots relevant to this thesis were discussed. The literature review of related work based on co-operative robotics and control of multiple mobile robots (multi-agent robotics) was presented, focusing on research contributions in which mobile robots can discriminate each other from the rest of the world based on a local reactive approach. Background information of the birth and origin of the behaviour-based control was highlighted. Advantages and disadvantages of the behaviour-based decomposition control approach in contrast with the traditional functional decomposition were discussed. The role of distributed artificial intelligence and distributed systems in the development of control architectures for multiple autonomous mobile robots was presented. Hence, the more significant sub-fields of distributed artificial intelligence were discussed in more detail as they relate directly to the work of this thesis. The origin of intelligent control including its approaches was illustrated. Three methodologies in the field of intelligent control were presented two of which were used direct in this thesis. Recent research in modelling, identification and control of dynamic systems using methodologies from intelligent control was reviewed with reference on related works. Robot navigation using intelligent control methods was described. In particular chapter two has reviewed robot navigation techniques up to date, focusing on those utilising fuzzy logic and neural networks.

In chapter three the aim was to propose, justify and present the main research methodology adopted for the research work carried out in this thesis. The main research methodology was broken down into nine basic steps. Each step was presented individually focusing on

design/modelling issues following a discussion of either advantages or disadvantages when the particular method is adopted. First step concerned conventional control design, which was used in chapter five to model speed controller for the mobile robot. The main advantages of conventional control design followed by short description of its main parts were outlined in the presentation of this step. Second step refers to constrained optimisation using non-linear control design tool for tuning/optimisation of physical and control parameters in chapters four and five. The design methodology for fast robust stability testing and analysis based on interval polynomials was discussed in the third step. More specific, using the parametric robustness analysis approach (Kharitonov's Theorem) the closed-loop control system (controller and plant) was proved to be robustly stable under uncertainty in robot dynamics. Steps four, five and six discussed the main research methodology for modelling and identification of local controllers (behaviours) using fuzzy logic systems, artificial neural networks and clustering techniques. Steps seven and eight defined the research methodology regarding the design of control systems architectures of mobile robots and multi-agent systems as additional tool in development of control architectures. The final step highlighted the stateflow design tool based on finite state machines theory. Using this tool model visualisation and construction of complex reactive systems can be easily achieved. In particular this design tool was used in chapter six as supervisor-like co-ordination object, global state identification mechanism and as static model for identification of direction of moving obstacle.

In chapter four the modelling of MIABOT V2 mobile robot was presented. The robot is small, measuring 8cm^3 and is steered and driven by differential drive design utilising two DC motors enabling robot's speed up to 1.2m/s . The first order kinematic model of the robot was derived in order to understand its manoeuvrability properties and to produce information about its global description. However, as the robot behaviour was related to the framework and theory of holonomic and nonholonomic systems some discussions were made to show in which category

MIABOT V2 falls (holonomic or nonholonomic). Then the full non-linear dynamic model of the robot was established for complete description of its system's dynamics. To improve the model accuracy real experiments took place. The non-linear control design blockset based on constrained optimisation method was used for identification of several robot physical parameters. To further improve the model, wheel slippage was introduced and modelled. Contacting real experiments, two different types of slippage were considered, one due to linear robot accelerations and another due to angular robot accelerations (fast turning). The robot model found to be very close to the real dynamics of the plant considered. The linearised model of the robot was extracted and some comparisons were made to show the validity of the linearised model against the non-linear. Results were presented, and show that the linearised model is acceptable, comparing non-linear and linear response under perturbations of the same control inputs.

Chapter five concerned control, robust stability analysis and discovery of fuzzy/neural local models from observation data. In the first instance a PI speed control law was developed based on full non-linear model and design requirements of the MIABOT V2 mobile robot. PI control action was selected, as it is robust and simple to design. The tuning/optimisation of the PI control parameters was achieved using the non-linear control design tool (NCD). This design methodology was fast, easy to use and suitable for the control design, which was based on non-linear model. In addition the method produced good performance and robust control under plant uncertainty. The results of tuning show that PI control action successfully controls the plant, providing performance within the design requirements. Then a design methodology to replace the PI controller by fuzzy logic controller was proposed. Based on subtractive clustering a fuzzy logic controller was identified using the PI controller as a teacher. The fuzzy controller was modelled with dynamical behaviour by emulating the controller function to more inputs (delays of inputs and outputs). To extend this design methodology a multilayer feedforward neural network with dynamical behaviour was developed based on supervised learning. Although a

neural network is static mapping of input-output indicating that theoretically it is not feasible to control or identify dynamic system, in this chapter was shown that a neural network with dynamical behaviour could successfully control the plant. Comparison was made of all three types of controllers based on their performance criteria and execution time. It was shown that all controllers controlled the plant within the design requirements producing similar RMSE of actual and reference response. However, the execution time was found to vary dramatically between the controllers. The PI control action was the fastest whereas the neural controller was faster than the fuzzy controller. Finally the closed-loop control system was tested for robustness under uncertainty in robot dynamics. Using the parametric robustness analysis approach (Kharitonov's Theorem) the closed-loop control system of the MIABOT V2 mobile robot was tested and proved to be robustly stable under uncertainty in robot dynamics. This method proved to be easy and fast to use. To reinforce the result of the approach taken the robust stability analysis of the closed-loop control system was verified using graphical techniques.

Chapter six presented the development of a proposed novel hybrid multi-agent based control architecture called CAROS for navigation of multiple autonomous mobile robots. The proposed architecture was designed for single/multiple robot navigation in both static and dynamic unknown environments. The need for hybrid solutions in design of complex control systems was demonstrated through an analysis using the Quality Function Deployment (QFT) tool. The analysis has clearly indicated that the most promising solution in control systems architecture design is the hybrid approach. At the beginning of the chapter an overview of the proposed architecture was presented highlighting its main characteristics. The proposed control architecture takes the advantages of various control structure types, and in particular the control architecture takes its design from competitive tasks architecture, production rules architecture, connectionist architecture, dynamic system architecture, multi-agent architecture and subsumption architecture.

The main objective was to achieve successful navigation of both single and multiple mobile robots in an environment populated by stationary and moving objects. In order to achieve this goal, the design of the architecture was based on a number of requirements/properties such as modularity, robustness, fault tolerance, distribution, reactivity, adaptability, planning, co-operation, easy of application, uncertainty, optimal control, learning and efficiency.

The final configuration of the control architecture utilised reactive, deliberative, distributed and centralised control approaches and used artificial intelligence as well as modular hierarchical structure. Multi-agent systems took care of the distributed control utilising the controller-agent concept, which is relatively new field in control engineering. Controller-agents operating in some restricted part of the operating regime produced or solved problems in terms of request from a well-defined supervisor-like co-ordination object. Centralised processing and deliberative reasoning was tackled using supervisory techniques incorporating finite state machines and non-linear switching mechanisms. The implementation of the main robot behaviours was achieved by decomposing the global tasks into simpler well-defined behaviours. The behavioural encoding was based on fuzzy logic, neural networks and finite state machines design methodologies. An algorithm methodology was proposed for supervised learning of neural network using fuzzy logic as a teacher.

As real-time motion planning in an unknown environment involves collision avoidance of static as well of moving robots a novel scheme was presented for identification of the direction of moving robots. Sensor modelling and sensor sensitivity was also considered in order to form modular reactive layer (different sensors have been modelled). Prior to the testing of the architecture in chapter seven a comparison was made with other recent approaches based on control architecture specifications such as reasoning, control, processing, integration, behavioural encoding, robot navigation and operating environment. The proposed architecture was shown to satisfy the most of the essentials specifications.

In chapter seven evaluation of the proposed novel hybrid multi-agent control architecture developed in this thesis for navigation of multiple autonomous mobile robots was presented. The control architecture was successfully verified and validated when applied to the problem of navigation of single/multiple autonomous mobile robots using the full non-linear model of the MIABOT V2 mobile robot derived and described in chapter four. The results presented in this chapter were devised in to three parts and are summarised follows.

The effectiveness of the proposed CAROS control architecture in a single mobile robot navigation operating in an unknown environment populated by static obstacles was considered. In particular issues such as, demonstrating system performance, independencies amongst integrated subsystems, implementation of global and local control strategies, static obstacle avoidance strategy, travel time due to the different controllers configuration and effectiveness of different sensor sensitivity utilisation were addressed. The proposed control architecture was observed to exhibit robustness, adaptability and flexibility when tested in single robot navigation in an unknown environment populated by static obstacles. The mobile robot achieved every control objective and its trajectory was smooth despite the interaction between several behaviours and the presence of unexpected obstacles. The effectiveness of SOAM using the controller-agent concept was validated with several numbers of tests. In all cases the robot achieved the control objectives defined both in A and B local control strategies defined in chapter six. The speed controllers developed in chapter five were re-tested and the results show that PI and fuzzy speed controllers have a marginal advantage against the neural controller. In particular the PI controller has the smallest ISE error of all controllers, whereas the fuzzy controller has the smallest IAE and ITAE error. The different types of sensor sensitivity developed in chapter six were tested according to the distance travelled and time taken for each robot to complete its mission. In all cases it was possible to navigate the mobile robot towards the target without any collisions with the static objects. Infrared sensor sensitivity produced the lowest values for distance and time taken for the robot to complete its mission. However as

safety is an important issue in mobile robot navigation the ultrasonic sensor sensitivity was used for the rest of the tests as the robot was navigated towards the target avoiding the main cluster of static obstacles by incorporating this type of sensor. The fuzzy logic and neural networks behavioural encoding for static obstacle avoidance developed in chapter six was tested in an environment containing different configurations of static obstacles. Fuzzy behavioural encoding produced marginally better result over neural encoding in terms of distance travelled during the navigation tasks. The neural networks behavioural encoding showed a better result over fuzzy encoding in terms of time taken for the robot to complete each mission. The control architecture was adapted to the environmental conditions, by adjusting the robot's control parameters (i.e. speed) and to activation of appropriate behaviour.

Secondly, the effectiveness of CAROS control architecture in navigation tasks involving two mobile robots operating in an unknown both static and dynamic environment was investigated. The results demonstrated that the control strategy chosen for navigation of multiple mobile robots is efficient. The robustness of the designed control system architecture against desired requirements, such as supervision, decision-making and co-ordination of internal control structures (subsystems) was observed. Both mobile robots achieved every control objective and their trajectories were smooth despite the interaction between several behaviours and presence of unexpected obstacles. The effectiveness of the proposed method for identification of direction of moving object defined in chapter six was established. The robots found to perform sufficient navigation with travel time and distance travelled by both robots considerably low and very close to optimal. The effectiveness of the action co-ordinator mechanism developed in chapter six for co-ordination and parallel switching between robot behaviours was validated by several tests. In all tests the proposed control architecture achieve a sufficiently high control performance once the control objective of each robot was reached. The stateflow-fuzzy logic and stateflow-neural networks behavioural encoding for dynamic obstacle avoidance developed in chapter six were tested in an environment containing different configurations of static and

dynamic obstacles. Stateflow-fuzzy behavioural encoding produced better result over stateflow-neural encoding in terms of distance travelled during the navigation tasks. The stateflow-neural behavioural encoding demonstrated better result over stateflow-fuzzy encoding in terms of time taken to complete each mission. The control architecture successfully adapted to the environmental conditions by adjusting the robot's control parameters (i.e. velocity) and taking actions (i.e. activation of appropriate behaviour).

Finally the effectiveness of the proposed CAROS control architecture in navigation tasks involving three to five mobile robots operating in an unknown both static and dynamic environments was considered. The results in Part III demonstrated that the control strategy chosen for navigation of multiple mobile robots is efficient and also re-established the robustness of the control system architecture against the desired requirements, such as supervision, decision-making and co-ordination of internal control structures (subsystems). The mobile robots were exposed in a complex and highly dynamic environment and successfully achieved every control objective, as their trajectories were smooth despite the interaction between several behaviours and presence of unexpected static and dynamic obstacles. The effectiveness of the action co-ordinator mechanism developed in chapter six for co-ordination and parallel switching between robot behaviours was validated. In all experiments the proposed control architecture achieve a sufficiently high control performance once the control objective of each robot was successfully reached. The mobile robots found to perform sufficient navigation with travel time and distance travelled considerably low and close to optimal.

8.3 Discussion and conclusions of the research

Throughout the thesis it has been shown that familiarities with certain scientific topics facilitate research where various complex problems can be solved. Considering the literature review in chapter two, it becomes clear that the problem of multi-robot control is challenging and also great motivation for future research. Contributions of related work show that development of

robust control with reliable behaviour with more than one robot co-existing in the same environment still remains a very difficult task. Despite the fact that behaviour-based control is one of the greatest revolutions in robot control several issues remain unanswered, which generate motivations for further research. Issues like, robot behaviour design, modularity and co-ordination is still of interest. Potential solutions to solve these issues can be found within the field of distributed artificial intelligent and distributed systems.

The field of distributed artificial intelligent and distributed systems is a very promising framework in the research area of development and control of multi-robot systems. Although the research work of distributed artificial intelligent is still in theoretical stage, it is believed that very soon practical implementations will take place. Already it is shown that the research effort regarding multi-agent systems is established in the development of more sophisticated software than traditional approaches. In addition, as lower-level processes (perception and actuation) are better understood and implemented, and as computational power increases, the high-level results of distributed artificial intelligence and distributed systems may become increasingly applicable to many practical applications including the control and development of multiple robots incorporating highly intelligent programs.

In chapter three it was shown that multi-agent systems theory is relatively new field in control and systems engineering. A special role in the theory and tools for solving complex control problems is attributed to the concept of agent. An agent represents an abstract entity that is able to solve a particular (partial) problem. Agents have the ability to be combined into a multi-agent system, such that the overall multi-agent system is able to solve a more complex problem. It was demonstrated (speed orientation adaptive mechanism in chapter six) that the concept of constructing local controllers that consist of several other controllers can be archived using multi-agent systems. Full integration and merging of multi-agent systems and control

engineering is a challenging and difficult task, because control theory has a strong mathematical foundations, whereas the field of MAS mainly is focused on abstract descriptions of the system.

Intelligent control such as fuzzy and neural approaches can be successfully implemented to solve several problems in modelling, identification and control of dynamic systems incorporating a specified behaviour as was shown in chapters five and six. There are a larger number of learning algorithms available to be used with different methodological approaches to be followed. The question is, which method should be used, and how. Therefore a development time can be dramatically increased or decreased according to a chosen approach.

Fuzzy logic and neural networks have been used in robot navigation although neural networks are relatively new. The majority of the research works in the literature consider a single mobile robot and not navigation of multiple robots. In addition much of the contributions made focus on robot navigation using kinematic models and not dynamic models. Very few works consider solving the problem of robot navigation using behaviours. Many research contributions to robot navigation are based on both fuzzy and neural methods with the attempt to solve the problem using one or two models (behaviours).

As was shown throughout the thesis (chapter five and six) one of the main advantages of fuzzy systems is that there is no need to have a mathematical model of the system. In addition, it is possible to control non-linear plants and using comprehensive linguistic rules it is possible to implement expert human knowledge and experience. However, it would be inappropriate to say that fuzzy systems can solve all control problems. Like any other design methodology the use of fuzzy systems has some drawbacks such as no global or systematic method for the transformation of the human experience into the rule base of the fuzzy system. In addition, it is not possible to show the stability of the controlled system since the model is not known and finally it is not guaranteed that rules are coherent (the possibility of mismatch between rules

exists). From the results obtained in chapter seven, it can be concluded that behaviour encoding incorporating fuzzy system is able to control the navigation of multiple mobile robots in an unknown environment populated by static and/or dynamic obstacles.

The neural networks behavioural encoding demonstrated that artificial neural networks have interesting and attractive features such as learning, selforganisation and the capability to model a large class of non-linear systems. ANN were able to learn (fuzzy systems was the teacher) and produce mapping between an input and an output space and form an associate memory that retrieves the appropriate output when presented with an unseen input. They can also generalise to produce an output when presented with previously unseen inputs. Calculations are in principle carried out in parallel resulting in speed advantages and programming can be achieved by training rather than defining explicit instructions. The results from chapter seven show that the major advantage of the ANN methodology is that it can produce learning controller for a mobile robot that can operate in an uncertain environment. A considerable drawback of ANN is that knowledge extraction and knowledge representation are difficult. This results in some kind of integration between fuzzy and neural systems. Again from the results obtained in chapter seven, it can be concluded that behaviour encoding incorporating neural networks system is able equally to control the navigation of multiple mobile robots.

It chapter four was mentioned that the design of motion control systems of many mobile robots is based only on kinematics models. In order to increase the efficiency in control architectures in robot navigation it is necessary to take into account dynamics affects. For this purpose in chapter four the dynamic model of the MIABOT V2 including its speed control law was derived with the help of the non-linear control design tool. This tool based on constrained optimisation proved to be fast, easy to use, suitable for control design of both linear and non-linear systems providing good performance. The implementation of the parametric robustness analysis approach (Kharitonov's Theorem) adopted in chapter five fro testing robust stability of the

robot's closed-loop systems found to be an easy to use method offering fast robust stability testing and analysis based on interval polynomials. Using this method the closed-loop control system of the MIABOT V2 mobile robot was proved to be robustly stable under uncertainty in robot dynamics. Three speed controllers were tested and the results show that PI and fuzzy speed controllers have a marginal advantage against the neural controller. In particular the PI controller has the smallest ISE error of all controllers, whereas the fuzzy controller has the smallest IAE and ITAE error.

The algorithmic methodology presented in chapter five for discovery of fuzzy models from observation data using subtractive clustering show that this methodology can be an effective technique when dealing with large sets of data. The principal idea is to distil natural groupings of data from a large data set thereby allowing concise representation of the model's behaviour. The advantage of using subtractive clustering compared to other clustering techniques is that computation is simply proportional to the number of data points and independent of the dimension of the problem under consideration. Having any set training data construction of a dynamic fuzzy model to control the plant can be achieved.

The design and implementation of complex control systems for autonomous mobile robots is a difficult task and still continues to challenge researchers. This challenge lies in the development of robust, flexible and modular control systems that are capable of coping with the dynamics of the real world. The new approach proposed in this thesis is a hybrid control system that takes the advantages of various control structures. The ongoing research for the identification of what are the "best" characteristics of each control approach in terms of requirements/properties is still an open question. In chapter six an analysis was presented based on the Quality Functional Deployment tool (QFD) for the identification of the relationship of the requirements/properties of control architecture versus the control architecture specifications. The analysis was based on fourteen requirements when design of control architecture is considered and the analysis

presented clearly shows that the solution to the design and implementation of complex control systems for autonomous mobile robots is hybrid. Deliberative systems are best suited for proposes where planning and long-term reasoning is essential. On the other hand reactive systems are best suited for proposes where the environment continuously changing and fast responses are essential.

The reasoning of the proposed control architecture is both deliberative and reactive as for an unknown and changing environment it seems likely that no single architecture (deliberative or reactive), will be able to cope with all different problems. Reactive reasoning was necessary so that the vehicle can navigate safely and can take actions in real-time. The reactive behaviours are in charge of the robot's specific domain. They were modelled using fuzzy, neural and hybrid behavioural encoding. The deliberative system comprising finite state machines is responsible for high-level planning and for solving conflicts among behaviours that try to access the robot's actuators at the same time. Using both hierarchical and behavioural-based control structure significant advantages can be achieved. For example, hierarchy of behaviours offers an efficient approach to synthesis of behavioural capabilities necessary for autonomous navigational tasks. Its practical utility lies in the hierarchical decomposition of overall behaviour into sub-systems that are activated only when needed. A hybrid approach similar to the one presented in this thesis should provide a suitable framework for situated adaptation in single and multiple robot navigation. Therefore hybrid models in mobile robotics systems are more flexible because they exploit the whole potential of behaviour-based systems and allow the introduction of diverse forms of knowledge. This type of control architecture combines symbolic methods of artificial intelligence, maintaining the objective of providing robustness, and real-time response.

The results presented in chapter seven show that the control architecture's modularity, distribution, reactivity and behaviour-based structure provided the overall control system with robustness in all cases of navigation tasks utilising either single or multiple mobile robots. The

experiments carried out for navigation of autonomous mobile robots proved that the modular design of the control architecture eased the testing for performance by allowing individual and integrated subsystem testing, thereby reducing the problems inherent in the design, integration and performance test of such complex systems. The results presented not only reveal the advantages of the proposed control architecture, but also identify answers to other issues in mobile robot navigation such as performance of sensor sensitivity, performance of proposed method for identification of direction of moving object and performance of behavioural encoding utilising fuzzy, neural or hybrid solutions.

It can be concluded that using the concept of behaviour-based control, a complex task can be easily completed implementing several simple behaviours. The experiments have demonstrated that in a complex navigation task potential difficulties may arise from the quantitative formulation of reactive behaviour as well as from the need for an efficient co-ordination and integration of conflicts and competition among different types of behavioural encoding. The results show that the study undertaken in chapter six for the design of the action co-ordinator mechanism using finite state machines, the implementation of the controller-agent concept from the field of multi-agent systems and the use of fuzzy logic and neural networks for behavioural encoding successfully compensates and overcomes the aforementioned difficulties.

Results show that the proposed hybrid system, using only three sensors, can be successfully applied to multiple robot (results are presented for up to five robots) navigation in complex and highly dynamic environments by efficiently co-ordinating multiple types of robot behaviours and local controllers organised by means of agents. The control architecture was observed to exhibit robustness, adaptability and flexibility when tested in single/multiplerobot navigation in an unknown environment populated by static an/or dynamic obstacles. The mobile robots achieved every control objective and their trajectories were smooth despite the interaction between several behaviours and the presence of unexpected obstacles.

Three different type of sensor sensitivity were tested according to the distance travelled and time taken for the robot to complete the mission. In all cases it was possible to navigate the mobile robot towards the target without any collision with the static objects. Infrared sensor sensitivity produced the lowest values for distance and time taken for the robot to complete its mission. The proposed method for identification of direction of moving object show that the robots were able to navigate sufficiently with travel time and distance travelled by all robots considerably low and very close to optimal.

The fuzzy logic and neural networks behavioural encoding for static obstacle avoidance were tested in an environment containing different configurations of static obstacles. Fuzzy behavioural encoding produced marginally better result over neural encoding in terms of distance travelled during the navigation tasks. The neural networks behavioural encoding showed a better result over fuzzy encoding in terms of time taken for the robot to complete each mission. The stateflow-fuzzy logic and stateflow-neural networks behavioural encoding for dynamic obstacle avoidance were tested in an environment containing different configurations of static and dynamic obstacles. Stateflow-fuzzy behavioural encoding produces better result over stateflow-neural encoding in terms of distance travelled during the navigation tasks. The stateflow-neural behavioural encoding shows better result over stateflow-fuzzy encoding in terms of time taken to complete each mission.

8.4 The main contributions of the thesis

The main contributions of the research presented in this thesis are summarised as follows:

1. Development of a novel hybrid multi-agent based control architecture called CAROS for navigation of multiple autonomous mobile robots operating in an unknown and unstructured environment populated by static and/or dynamic obstacles

The proposed hybrid control architecture is modular and draws its design from competitive tasks architecture, production rules architecture, connectionist architecture, dynamic system architecture, multi-agent architecture and subsumption architecture. A comparison between the proposed control architecture and several other control structures developed in the past decade for navigation of autonomous mobile robots was made. The comparison demonstrates that the CAROS control architecture offers several advantages over the others. The reasoning of the control architecture is both deliberative and reactive. The proposed reactive behaviours are modelled using fuzzy logic, neural networks and hybrid behavioural encoding incorporating stateflow-fuzzy logic and stateflow-neural networks. The deliberative system comprised of finite state machines. The processing can be achieved in a centralised and decentralised manner using the proposed controller-agent concept from the field of multi-agent systems. The framework of the control architecture is suitable for situated adaptation in single and multiple robot navigation.

2. Novel approach for identification of direction of moving obstacles (other robots) using finite state machines.

A novel method for identification of direction of moving objects (other robots) based on three sensory inputs using finite state machine was proposed. More specifically fifteen transitions associated with six states incorporating a Moore FSM were proposed. Every transition was assigned with a specific condition and action. Only three sensors are used to specify all transitions conditions. The specified transitions in conjunction with the associated conditions covers all the possible directions of the moving object around the robot's sensing area.

3. Novel approach for behavioural encoding using hybrid solutions such as stateflow-fuzzy and stateflow-neural for autonomous robot navigation.

A new real-time dynamic obstacle avoidance method for autonomous mobile robot navigation has been developed. The proposed novel approach incorporates either stateflow-fuzzy or stateflow-neural behavioural encoding. The applicability and effectiveness of the proposed approaches were evaluated in an environment containing different configurations of dynamic obstacles. The mobile robot successfully adapted to the environmental conditions in the presence of moving obstacles (other robots). The approach is generic and can be easily extended to accommodate more complicated dynamic environments.

4. Proposed a design methodology for developing integrated solutions for autonomous mobile robotic systems and classification of the main design methodology (properties) of control systems architectures for autonomous mobile robots.

Six key issues, regarding the control architecture design of multiple autonomous mobile robots were proposed. These key issues include: centralised, decentralised or hybrid control, heterogeneous or homogeneous robots, co-operation with-or-without communication, (implicit or explicit communication), making agents (robots) work as a team, multiple mobile robots path planning, and learning. When comparison between different types of control architectures is undertaken, this comparison should be made based on a number of important properties. Classification among these properties was proposed including: modularity, robustness, fault-tolerant, distribution, reactivity, adaptability, planning, co-operation, easy of adaptation, uncertainty, optimal control, goal-oriented, learning and efficiency.

Less significance contributions are summarised as follows:

1. Literature survey on approaches/methods related to the development of intelligent control architectures for navigation of multiple autonomous mobile robots.

More specifically the literature survey shows that intelligent control comprises three main approaches (methodologies), which are now widely accepted and established (fuzzy logic, neural networks and knowledge based systems). The use of these methodologies can be implemented to solve several problems in the development of intelligent control architectures for navigation of multiple autonomous mobile robots. The literature review has shown that fuzzy and neural approaches dominate the research community when mobile robot navigation is considered.

2. Modelling of MIABOT V2 mobile robot. A generic approach for optimisation and identification of parameters of physical components (i.e moments of inertia) conducting experiments and using the non-linear control design tool.

The full non-linear dynamic model of the mobile robot was established for complete description of its system's dynamics. To improve the model accuracy a generic approach for optimisation and identification of parameters of physical components (i.e moments of inertia) conducting real experiments and using the non-linear control design tool was proposed. The robot model found to be very close to the real dynamics of the plant considered.

3. Comparison between PI, fuzzy and neural controllers based on their performance criteria (ISE, IAE and ITAE) and execution time. An algorithmic methodology for discovery of fuzzy/neural local models from observation data. Use of parametric robustness analysis approach for robust stability testing of closed-loop control system under uncertainty in robot dynamics (The approach has not been considered previously within the research community of autonomous mobile robots).

Comparison between PI, fuzzy and neural controllers was made. In particular the PI controller has the smallest ISE error of all controllers considered, whereas the fuzzy controller has the

smallest IAE and ITAE error. The execution time of PI controller is considerable smaller compared to other speed controllers considered (three times faster than the neural controller and six times faster than the fuzzy controller). A design methodology for discovery of fuzzy-neural local models from observation data was proposed. Based on subtractive clustering a fuzzy logic local controller was identified using a PI controller as a teacher. To extend this design methodology a multilayer feedforward neural network with dynamical behaviour was developed based on supervised learning using, this time, a fuzzy logic model as a teacher. Finally, the parametric robustness analysis approach (Kharitonov's Theorem) was used for testing the stability of the closed-loop control system of the MIABOT V2 mobile robot under uncertainty in robot dynamics. This method, which proved to be easy and fast method to use has not been considered previously within the research community of autonomous mobile robots.

4. Identification of the relationship of the most important requirements/properties of control architecture versus the main control architecture specifications using the Quality Function Deployment (QFT) tool.

An analysis is presented based on the Quality Functional Deployment tool (QFD) for the identification of the relationship of the requirements/properties of control architecture versus the control architecture specifications. The analysis was based on fourteen requirements when design of control architecture is considered and the analysis presented clearly shows that the solution to the design and implementation of complex control systems for autonomous mobile robots is hybrid.

5. Modular approach for modelling three types of sensor and sensor sensitivity.

Three different types of sensor sensitivity (infrared, ultrasonic and no-shape) were evaluated according to the distance travelled and time taken for the each mobile robot to complete the

mission. In all cases it was possible to navigate the mobile robot towards the target without any collision with the static objects. Infrared sensor sensitivity produced the lowest values for distance and time taken for the each robot to complete its mission. The approach is both modular and generic. New sensors or new sensor sensitivity can be easily added to different applications (different robot) and environments.

8.5 Suggestions for future work

After the completion of the research the results obtained open several possible avenues for future work. This work is summarised in the following.

In order to further improve navigation performance, the present study can be extended to implement a vision system for determining some sub-goals and building a local map. On-line learning also is an important issue to be considered as it has promising features for design and implementation of more flexible and adaptive systems. More specifically, automatic generation of behaviours is an area that has not been explored enough. Research could be focused not only on basic behaviours, but also on the process of developing complex emergent behaviours by learning. The concepts of basic and emergent behaviours are relative, because emergent behaviours can be fused recursively with other behaviours to produce new generations of emergent behaviours. A improvement will be to develop control architectures which are able to cope with this recursive evolution process.

Investigations into some adaptations in the proposed control architecture can be a potential area for further research. Instead of having a fixed hierarchy of priorities for behaviours, both variable and time varying priority schemes are certainly worthwhile for further investigation. Further research must be devoted to develop planners that work directly with a library of skills, instead of working with pre-programmed plan units. This library of skills should be dynamic and must allow the incorporation of new learned skills.

In order to cope with real situations, dynamic objects should be detected and distinguish among static. Despite the work done on this topic, the problem of distinguishing between static and dynamic obstacles is still unsolved.

Design methodologies for agents and multi-agent systems are certainty becoming a research issue. The aim of a unique and generally agreed on definition of agents and multi-agent control architectures have the potential for further research on a single programming language to be used worldwide. Given this statement, maybe the important issue is not only to agree on a unique definition of agents and their internal structure, but also to develop uniform interfaces and communications standards that allow different and possibly heterogeneous agents to interact.

A

MIABOT V2 Mobile Robot

A.1 Introduction

The robots are manufactured by Merlin Systems Corporation Ltd (Merlin Systems Corporation, 2002). Each robot is fully autonomous measuring (8cm x 8cm x 8cm). Commands can be sent to the robots from the host PC via a debug cable, but obviously for football match purposes the communications link is used to send complete strategic instructions to each robot. Each MIABOT package comes as a fully autonomous development robot supplied complete with 'C' compiler and ISP programmer.

A.2 Main features

The main structure of the robots are built around the RISC based Atmel (AT90S8515) CPU, of which some features are mentioned below, but full specifications can be found on the ATMEL website (Atmel, 2002).

- 4 MIPS operation.

- 8K bytes flash programmable RAM.
- 512 bytes SRAM.

All software for the robots operation is stored in an onboard EPROM.

The Robots other features include:

- 2 re-chargeable NIMH battery packs.
- RF receive communications module (Frequency can be selected by exchanging between 418 and 433Mhz modules).
- Easy access modular style case. (For easy battery change etc.)
- On-board systems programmer (enables programs to be initially downloaded to the robots EPROM via the supplied cable).
- On-board dual channel H-Bridge: gives the ability to drive 2 DC motors in either direction.
- Drive train with achievable speeds of up to 1.2m/s (Each wheel having it's own encoder (opto-feedback) for position and speed feedback).

A.3 Drive train

The drive train Figure A-1 consists of two motors (DC 4.5 - 12V), produced by Swallow Systems Ltd (Swallow Systems, 2002), with worm gear drives to each wheel enabling accurate stop positions. The overall width of the drive train is approximately 80mm. The wheels are 32mm in diameter and equipped with O-rings to reduce pitch slip. Each motor shaft is monitored via a combination of phototransistors and infrared LED's (Shaft Encoders), which provide feedback for the ATMEL microprocessor. The speed of the robots can also be

controlled, using the shaft encoders (phototransistors) to interrupt the microprocessor. The microprocessor gathers all the information, and each motor is driven accordingly.

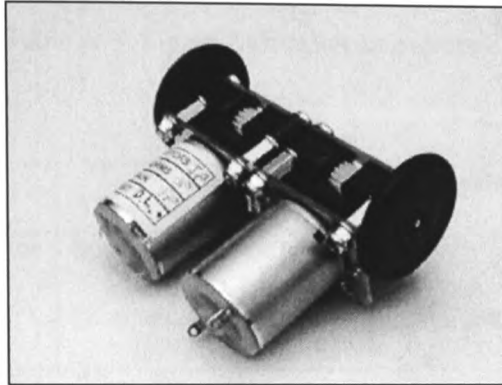


Figure A-1 Drive train overview

A.4 DC motors

The DC motor Figure A-2 used in the drive train it is a Mabuchi RC-280SA-20120 motor giving 29g.cm torque at 6400 rpm taking 0.57A from a 6V supply at maximum efficiency. The stall torque from a 6V supply is 175g.cm at a current of 2.85A. This is a powerful beast. These motors are able to make the robot to achieve speeds over 1 m/s.

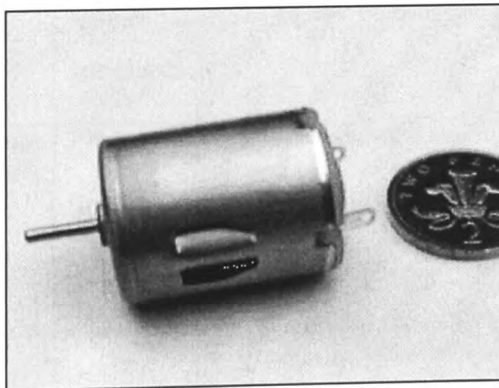
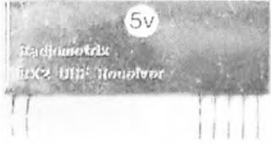


Figure A-2 MIABOT's DC motor overview

A.5 MIABOT control board

The MIABOT control board is designed using the latest surface mount technology to create a compact feature packed general-purpose control board. The specifications of MIABOT's control board are listed in Table A-1. Figure A-3 shows an overview of the control board.

Feature	Description
4 - Way Battery Connector	Socket provided to run the robot via the onboard batteries or from another power source (such as a bench supply)
2 - Way Charger Socket	When the switch is in the charge position power supplied to the charge socket will recharge the onboard batteries.
Opto Encoders	Open collector opto inputs, which are connected, to interrupt capture pins on the MCU. These inputs are normally connected to the outputs from the drive train encoders.
LED1 & LED2	Status LED's, software programmable but by default used to indicate power on and activity on the RX line.
CPU	Atmel AT90s8515 Microcontroller (MCU)
Motor Output	2 Channel DC motor outputs. 12v with max. (stall) current of 3A per channel.
RX Module	<p>Plug n slot for wireless communications module.</p> <p>Specifications: 418Mhz or 413Mhz.</p> 


TX Module	Optional plug in socket for wireless transmission module. Specifications: 418Mhz or 433Mhz. <u>Note:</u> The RX and TX unit must be on separate frequencies if both are used. 
Programmer/Debugger/SPI	Socket can be used for general purpose I/O (5) as well as programming, and debugging (bi-directional communications via cable) and SPI (Serial Peripheral Interface).
H-Bridge	Power drive for the motors.
Expansion Port	15 Way header for general purpose I/O (13).
Switch	On, Off or Charge

Table A-1 Specifications of MIABOT's control board

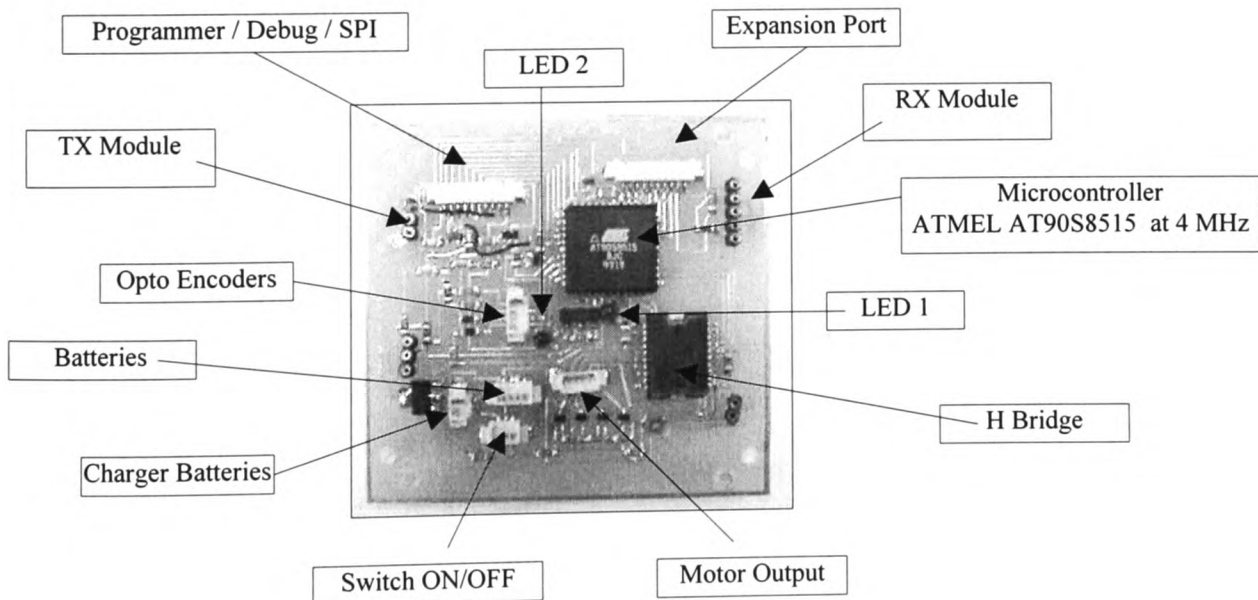


Figure A-3 Control board overview

A.6 The software

Each robot is supplied with the GNU 'C' compiler, download utility (DOS & Windows 95,98 only), editor and other development tools. Basic (but extensible) motion control software is supplied with the robot. Corresponding sample code is provided to run on the PC to control the robot (QBASIC and Visual C++).

A.7 PC transmitter

Each robot can be driven via the debug cable or wireless communications. To communicate with the robot the PC Transmitter unit Figure A-4, which plugs into the 9-Way RS232 port is used. Specific transmitter frequency is required (418Mhz or 433Mhz) to match the RX module on the robot. A single PC transmitter can be used to communicate with multiple robots by sending out communications packets each with a unique ID. The software on each robot can be then tuned to only respond to commands with the correct ID. The commands are sent from the host computer with a protocol communication shown below:

Preamble	Start byte	Command	Robot Id	Data command	End byte
----------	------------	---------	----------	--------------	----------

Preamble: Used to synchronise the emitter and the transmitter

Start byte: Start

Command: Type of command (forward, stop, long curve, etc)

Robot Id: 0, 1, 2, 3 (0 to control all the robots)

Data command: Parameters of the command

End Byte End

To simplify the transmission, the byte commands are coded in ASCII, for example 'E' = 69 in decimal or 0100 0101 in binary. The usually commands used is shown below in Table A-2.

	ASCII	Decimal
Preamble:		000
Start byte:	[091
Command	L	108
Robot Id:	0, 1, 2, 3	048, 049, 050, 051
Left wheel		
Speed		0...255
Distance		0...255
Distance		0...255
Direction	0 or 1	048 or 049
Right wheel		
Speed		0...255
Distance		0...255
Distance		0...255
Direction	0 or 1	048 or 049
End Byte]	093

Table A-2 Command coded

Note that distances for both wheels are coded with two bytes, 0...65536. When the data needs more than a byte, it is real problem because they need to be decomposed and rebuilt. For integer is simple but for float it is more complicated.

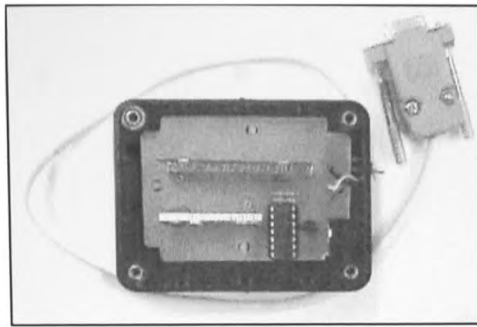


Figure A-4 PC transmitter unit overview

A.8 References

ATMEL. 2002. [WWW]. <http://www.atmel.com> (30 January 2002)

MERLIN SYSTEMS CORPORATION. 2002. [WWW]. <http://www.merlinsystemscorp.co.uk> (30 January 2002)

SWALLOW SYSTEMS. 2002. [WWW]. <http://www.swallow.co.uk> (30 January 2002)

B

SIMULINK Block Diagrams

B.1 Introduction

This appendix contains the main block diagrams that were designed in Simulink for the modelling and simulation of the CAROS control architecture described in this thesis.

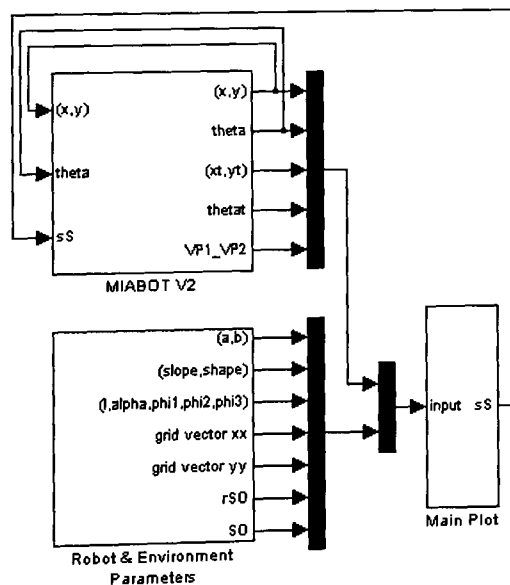


Figure B-1 Implementation of CAROS in MATLAB/Simulink-based simulator with one robot

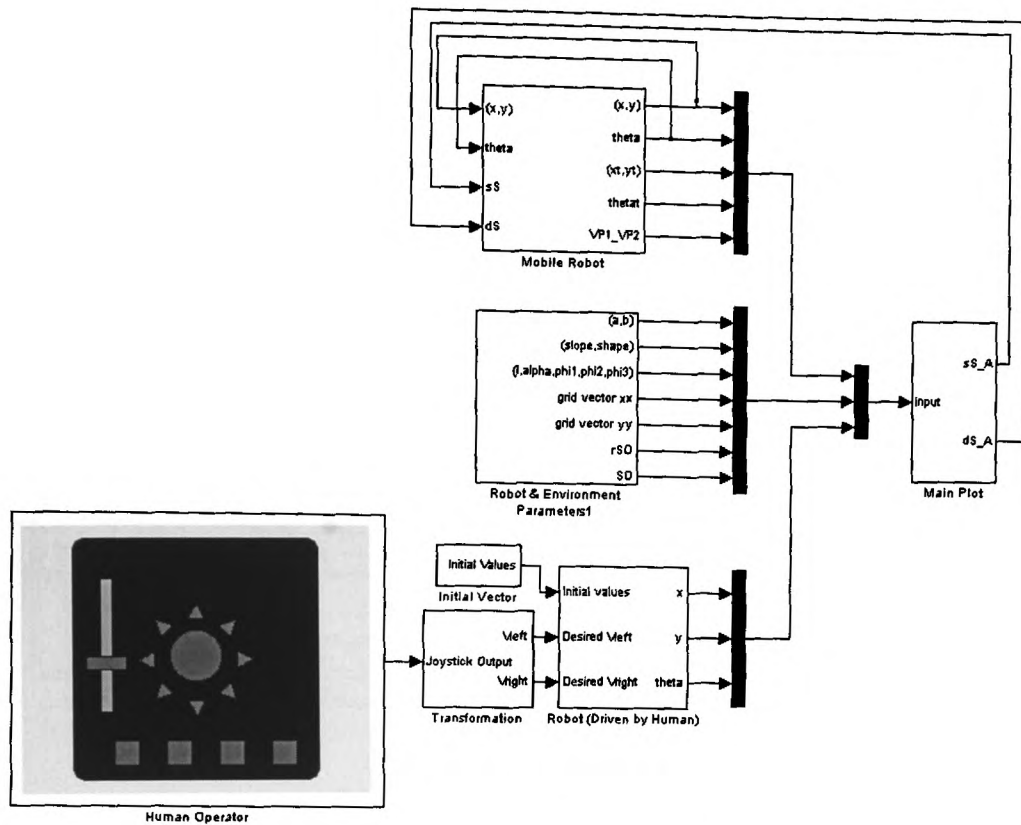


Figure B-2 Implementation of CAROS in MATLAB/Simulink-based simulator with two robots (one robot is driven by Human)

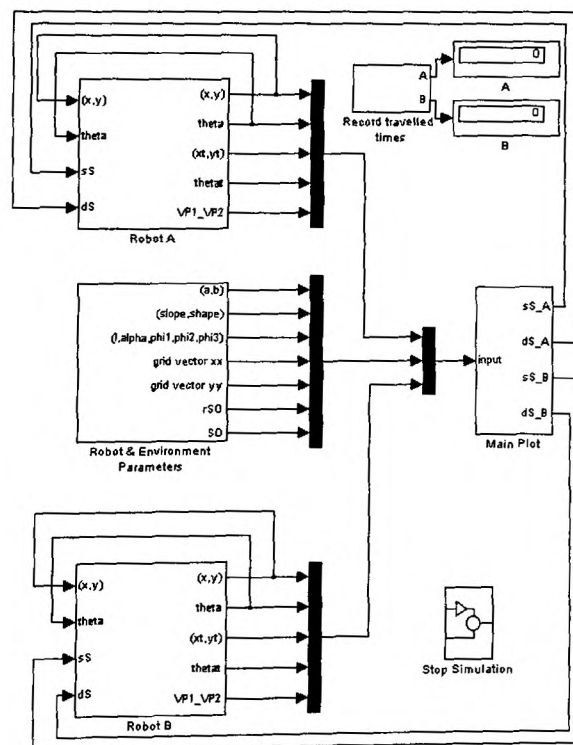


Figure B-3 Implementation of CAROS in MATLAB/Simulink-based simulator with two robots

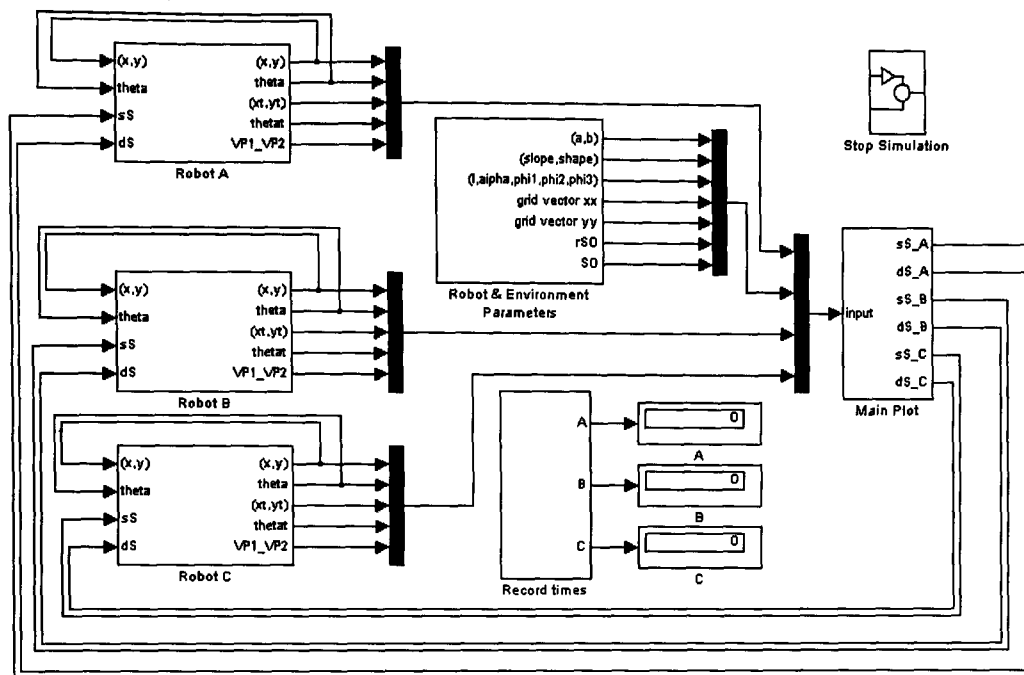


Figure B-4 Implementation of CAROS in MATLAB/Simulink-based simulator with three robots

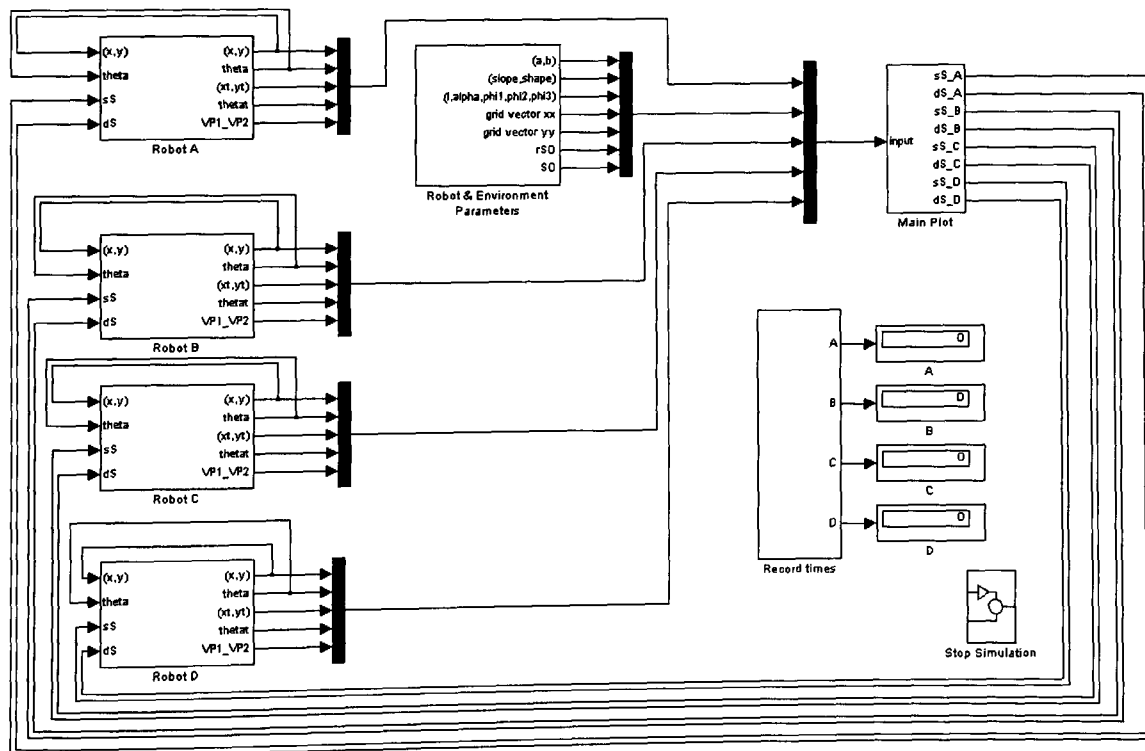


Figure B-5 Implementation of CAROS in MATLAB/Simulink-based simulator with four robots

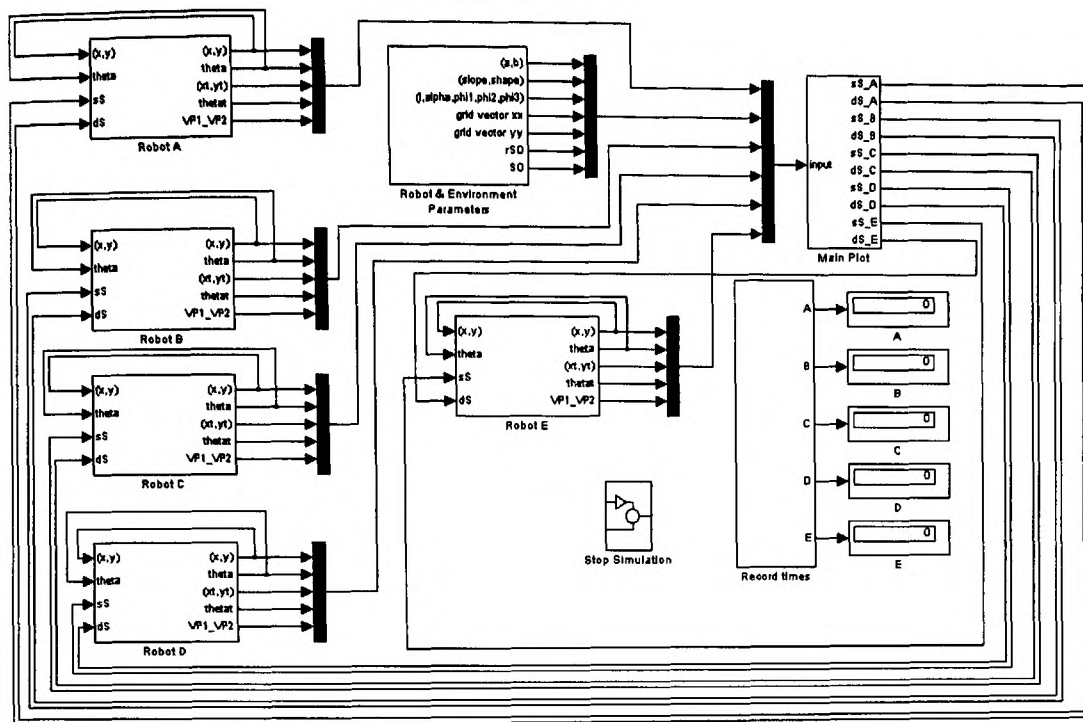


Figure B-6 Implementation of CAROS in MATLAB/Simulink-based simulator with five robots

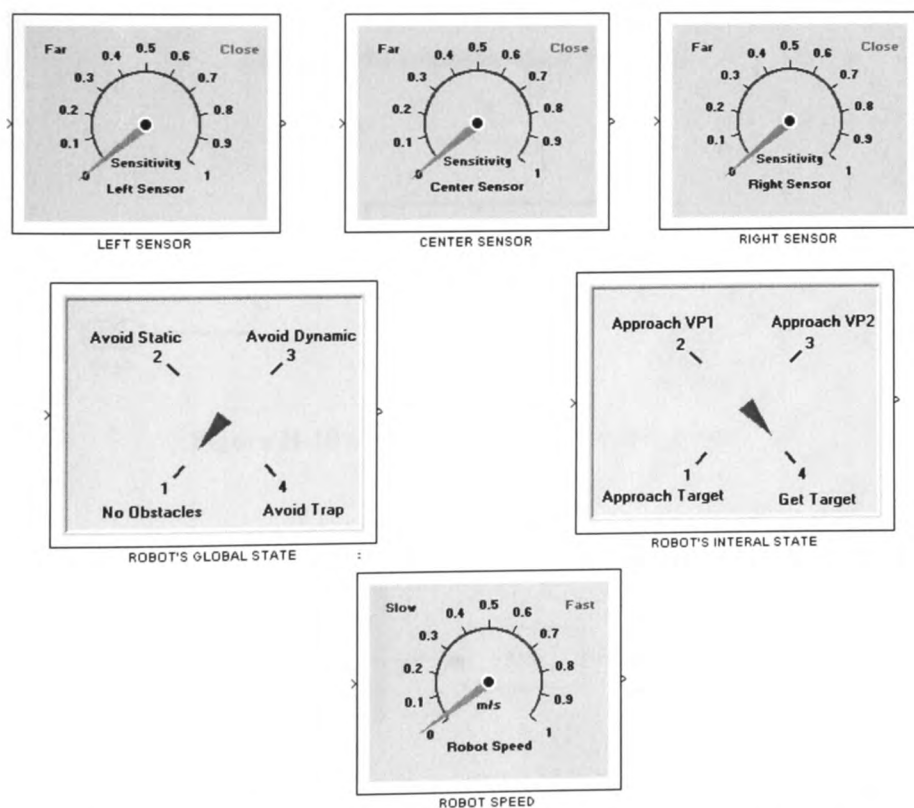
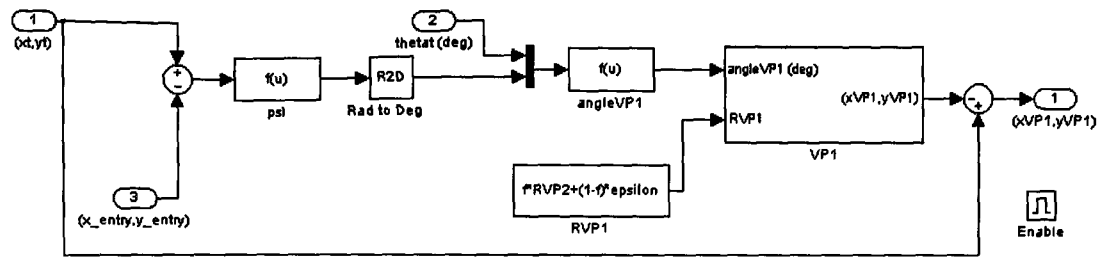
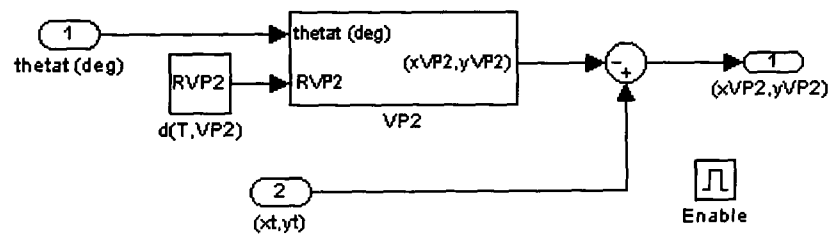
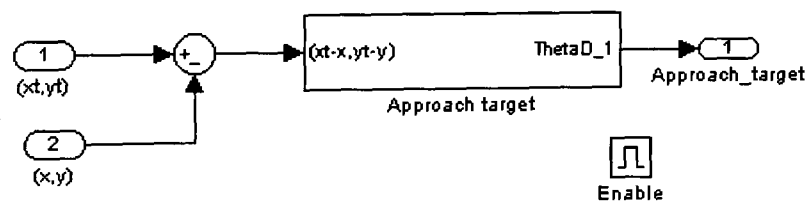
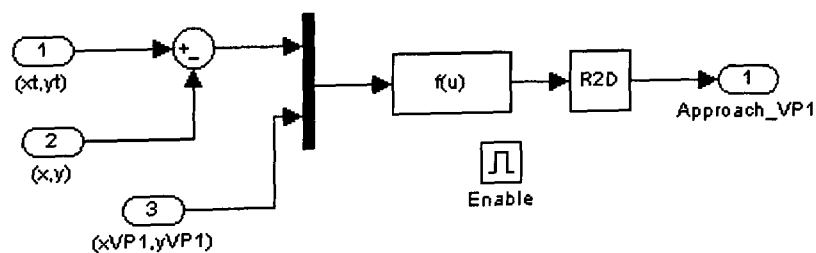


Figure B-7 Gauges used to read sensor sensitivity, robot speed and both global and internal states

Figure B-8 Controller-agent *find_VP1*Figure B-9 Controller-agent *find_VP2*Figure B-10 Controller-agent *approach_target*Figure B-11 Controller-agent *approach_VP1*

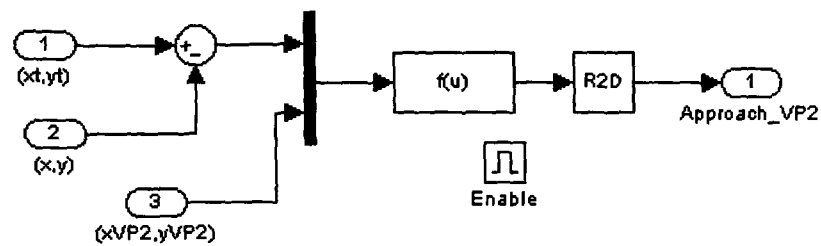


Figure B-12 Controller-agent *approach_VP2*

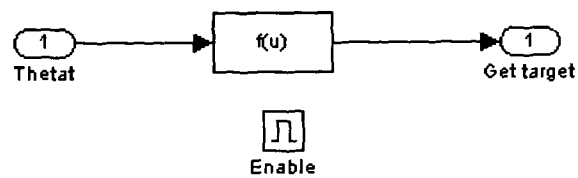


Figure B-13 Controller-agent *get_target*

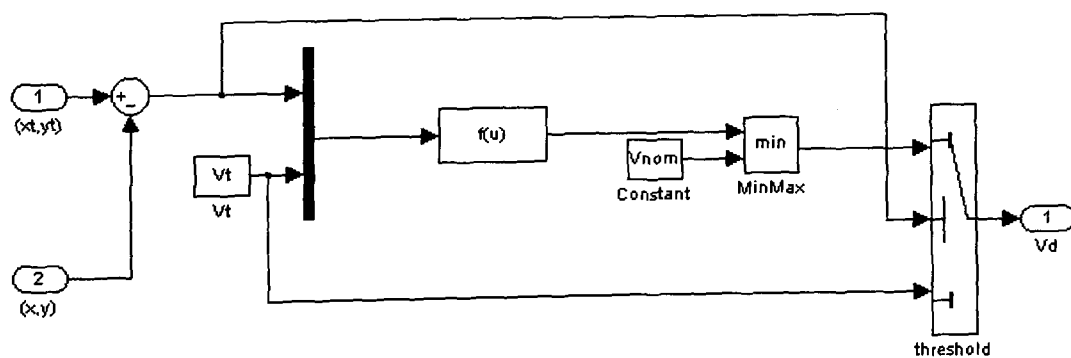


Figure B-14 Local controller *adjust_speed*

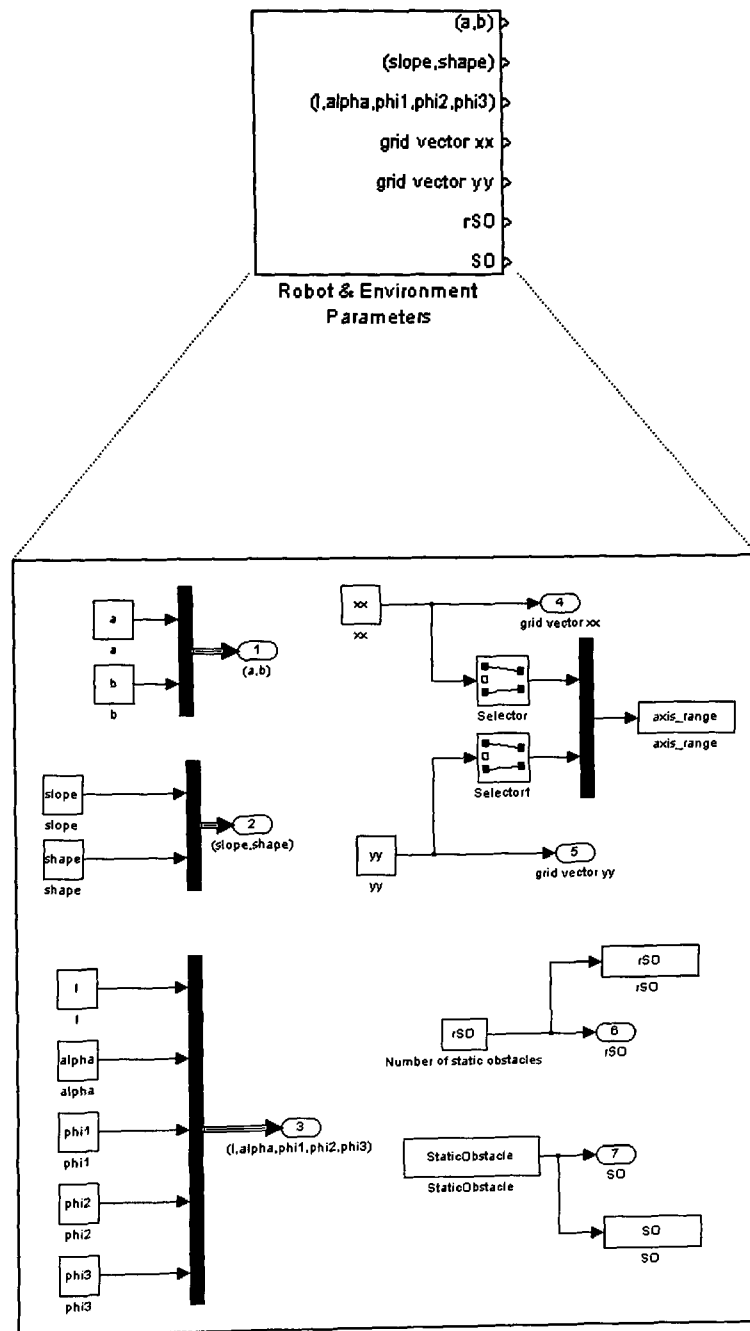


Figure B-15 Modelling parameters for mobile robot sensors and environment

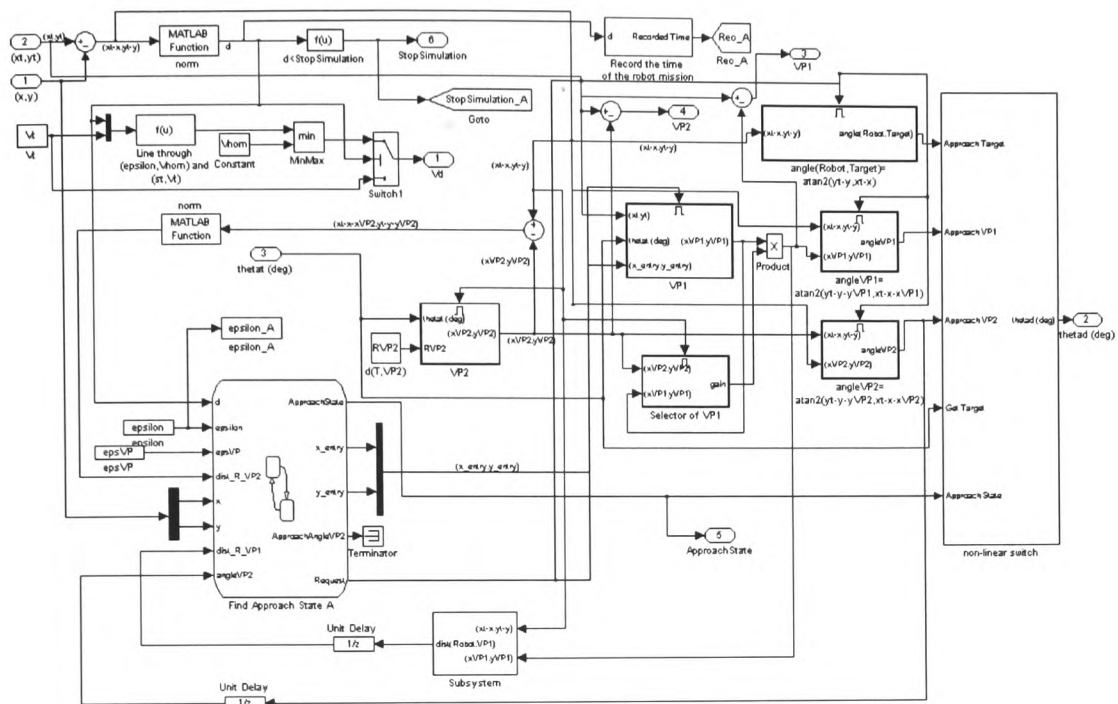


Figure B-16 Global view of speed and orientation adaptive mechanism

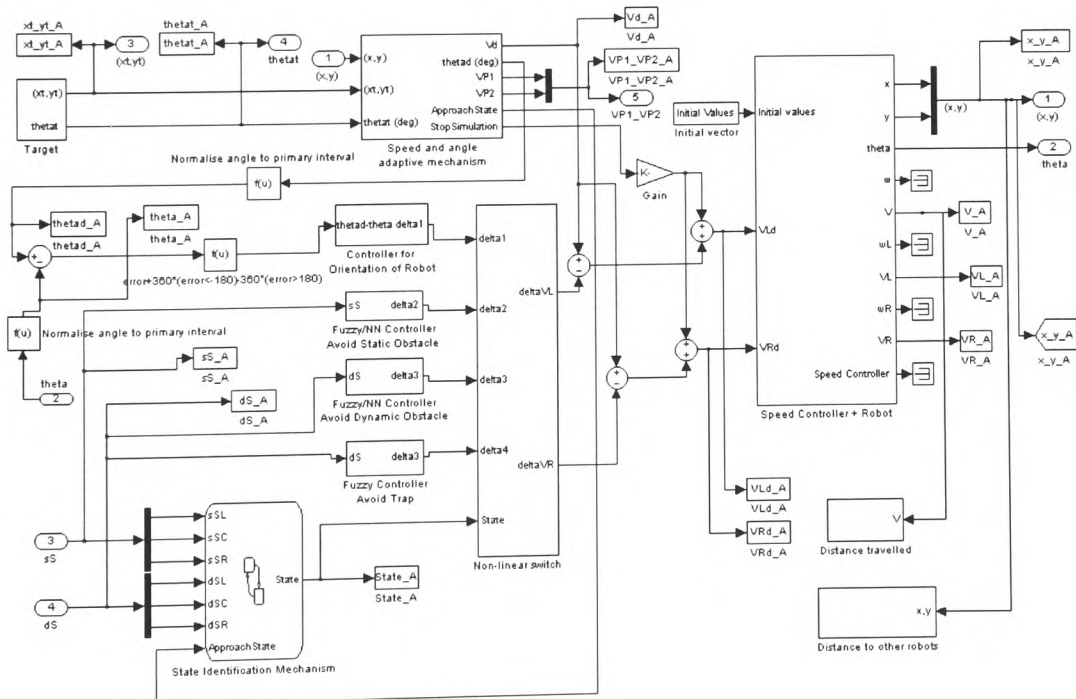


Figure B-17 Global view of co-ordination between robot behaviours and action co-ordinator mechanism

C

Conventional Control Design

C.1 The basic control system structure

A control system consists of subsystems and processes (plants) assembled for the main purpose to control the output of the process. In the simplest form a control system provides an output response for a given input as shown in Figure C-1.

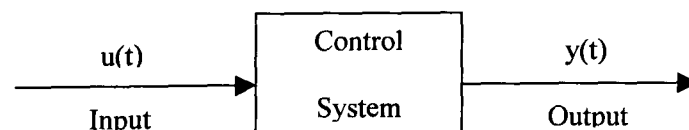


Figure C-1 Input-output block diagram of a control system

C.1.1 Description of the input and output

In a general control system the input represents the desired response and the output is the actual response. Figure C-2 shows the input and output of a general control system. The transient response shown in Figure C-2 indicates a number of the most commonly used performance

criteria such as rise time, settling time, steady state, etc. (more details can be found in (Dutton *et al*, 1997)). It can be observed that mainly two factors make the output different from the input. The first factor is the instantaneous change of the input against the gradual change of output (transient response) and the second factor is the accuracy of the final transient response in respect with the input command (steady state error).

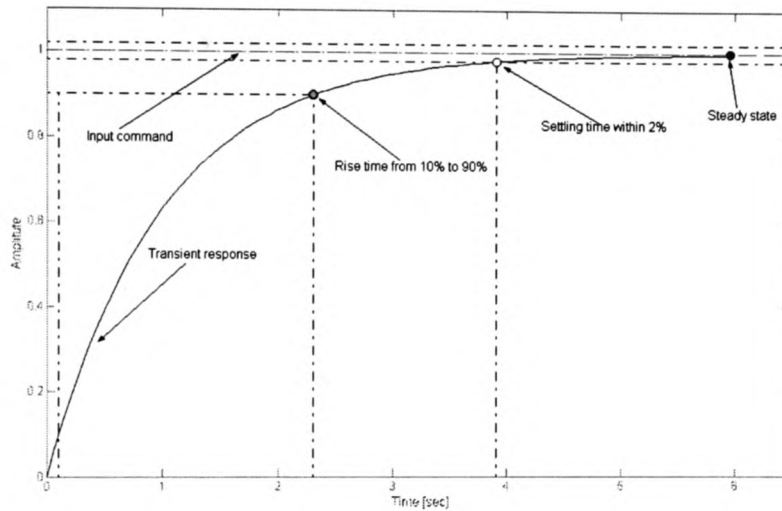


Figure C-2 Description of input output of a general control system

C.2 Linear and non-linear control systems

Systems with linear, time invariant (LTI) behaviour can be understood very well using frequency-domain techniques. Unfortunately many control systems include regions of non-linear operation and have significant parameter variation. The definition of linear and non-linear control system is given in the following paragraphs (Ellis, 2000).

Definition C-1 (Linear Time-Invariant System): Consider a system with an input $r(t)$ and output $c(t)$. The system is LTI if the following three criteria are satisfied. *Homogeneity*: If $r(t)$

generates $c(t)$, then $k \times r(t)$ generates $k \times c(t)$. *Superposition*: If $r_1(t)$ generates $c_1(t)$ and $r_2(t)$ generates $c_2(t)$, then $r_1(t) + r_2(t)$ generates $c_1(t) + c_2(t)$. *Time invariance*: If $r(t)$ generates $c(t)$, then $r(t - \tau)$ generates $c(t - \tau)$. Note, that the first and second criteria define linearity, whereas, the third defines the time invariance. Examples of linear¹ system include addition, subtraction, scaling by a fixed constant, integration, differentiation, time delay and sampling.

Definition C-2 (Non-Linear Time-Invariant System): Consider a system with an input $r(t)$ and output $c(t)$. If the system violates one or more of the three criteria listed in Definition 3-1 then the system is said to be non-linear time-invariant (Non-LTI). The most common non-linear behaviours are gains that vary as a function of operating conditions.

C.3 PID Control

PID stands for a Proportional (present control error), Integral (past control error) and Derivative (future control error) control, and, is the most commonly used standard form of dynamic compensation in most practical applications such as process control, motor drives, magnetic and optic memories, automotive, flight control, instrumentation etc.

Despite a lot of research and the huge number of different solutions proposed, different sources estimate the share taken by PID controllers is between 90% and 99% (Astrom and Hagglund, 2001). Some of the reasons include (Reznik *et al*, 2000): PID controllers are robust and simple to design. There exists a clear relationship between PID and system response parameters. As a PID controller has only three terms, plant operators have a deep knowledge about the influence of these parameters and the specified response characteristics on each other. Many PID tuning techniques have been elaborated during recent decades, which facilitates the operator's task.

¹ No practical control system is completely linear and most vary over time.

PID control could benefit from the advances in technology due to their flexibility (tuning and self-tuning).

PID controllers (Figure C-3) are derived from the best properties of PD and PI controllers. The proportional controls action is employed to reduce the settling time and the rise time of the plant response during transient conditions, the derivative control action is employed to reduce the overshoot and the oscillations of the plant response during transient conditions, and the integral control action is employed to eliminate the steady state error during steady state conditions.

Generally, the PID control law can be formulated according to:

$$G_c(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s \right) \quad (C.1)$$

Where K_p , T_i and T_d are proportional gain, integral time and derivative time respectively. If an instantaneous error signal $e(t)$ is the input to the PID controller, then the output $u(t)$ is the calculated control effort from the controller and is given, in time domain, by:

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int_0^t e(t) dt + T_d \frac{de(t)}{dt} \right] \quad (C.2)$$

Another common form of the PID control action is a rearrangement of (C.1) to:

$$G_c(s) = K_p + \frac{K_i}{s} + K_d s \quad (C.3)$$

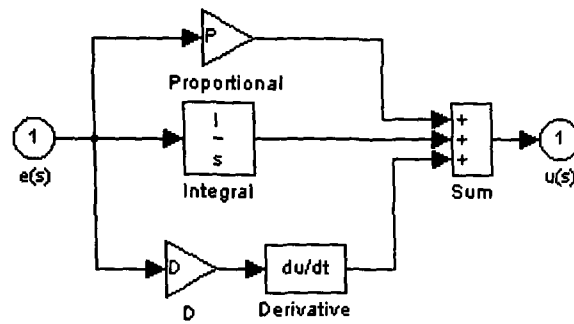


Figure C-3 Conventional PID Controller

As mentioned earlier PID controllers are derived from PD and PI controllers. Consequently PD controller can be formulated as:

$$u(t) = K_p \left[e(t) + T_d \frac{de(t)}{dt} \right] \quad (C.4)$$

The output signal of this controller is equal to the sum of two signals: the signal obtained by multiplying the input signal by a constant gain K_p and the signal obtained by differentiating and multiplying the input signal by T_d . Similarly to the PD controller, the PI controller produces as its output a weighed sum of the input signal and its integral. This controller is formulated as follows:

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int_0^t e(t) dt \right] \quad (C.5)$$

C.4 References

ASTROM, K. J. and HAGGLUND, T. 2001. The Future of PID Control. *Control Engineering Practice*, **9** (11), pp. 1163-1175.

DUTTON, K., THOMPSON, S. and BARRACLOUGH, B. 1997. *The Art of Control Engineering*. USA: Addison Wesley Longman. 0-201-17545-2.

ELLIS, G. 2000. *Control System Design Guide: Using your Computer to Understand and Diagnose Feedback Controllers*. 2nd end. USA: Academic Press. 0-12-237465-7.

REZNIK, L., GHANAYEM, O. and BOURMISTROV, A. 2000. PID plus Fuzzy Controller Structures as the Design Base for Industrial Applications. *Engineering Applications of Artificial Intelligence*, **13** (4), pp. 419-430.

D

Constrained Optimisation using the non-linear control design tool

D.1 Introduction

In this appendix an overview of constrained optimisation and the non-linear control design tool (The MathWorks, 1997) is presented. This methodology is used both in chapters four and five for tuning/optimisation of physical and control parameters.

D.2 Overview

Optimisation techniques are used to find a set of design parameters, $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, that can be in some way be defined as optimal. In a simple case this may be the minimisation or maximisation of some system characteristic, dependent on \mathbf{x} . In a more advanced formulation the objective function, $f(\mathbf{x})$, to be minimised or maximised, may be subject to constraints in the form of equality constraints, $C_i(\mathbf{x}) = 0 \quad (i = 1, \dots, p_e)$, inequality constraints,

$C_i(\mathbf{x}) \leq 0$ ($i = p_e + 1, \dots, p$), and/or parameter bounds, x_l, x_u . A general optimisation problem is given as follows:

$$\begin{aligned} & \text{minimise } f(\mathbf{x}) \quad \mathbf{x} \in \mathcal{R}^n \\ & \text{subject to: } C_i(\mathbf{x}) = 0 \quad i = 1, \dots, p_e, \quad C_i(\mathbf{x}) \leq 0 \quad i = p_e + 1, \dots, p, \quad x_l \leq x \leq x_u \end{aligned} \quad (\text{D.1})$$

Where \mathbf{x} is the vector of design parameters ($\mathbf{x} \in \mathcal{R}^n$), $f(\mathbf{x})$ is the objective function that returns a scalar value ($f(\mathbf{x}): \mathcal{R}^n \rightarrow \mathcal{R}$), and the vector function ($C(\mathbf{x})$) returns the values of the equality and inequality constraints evaluated at \mathbf{x} ($C(\mathbf{x}): \mathcal{R}^n \rightarrow \mathcal{R}^p$). Although an efficient and accurate solution to the above problem depends on the number of constraints and design variables, the characteristics of the objective function and constraints are also associated with efficient and accurate solution. The optimisation problem is known as Linear Programming (LP) problem if both the objective function and the constraints are linear functions of the design variable. However, if the objective function and the constraints are non-linear functions of the design variable then the optimisation problem is known as Non-linear Programming (NP) problem.

D.3 Constrained optimisation

In constrained optimisation, the general aim is to transform the problem into an easier subproblem that can then be solved and used as the basis of an iterative process. Early methods were based on the translation of the constrained problem to a basic unconstrained problem by using a penalty function. In this way the constrained problem is solved using a sequence of parameterised unconstrained optimisations. These methods are now considered relatively inefficient and have been replaced by methods that have focused on the of the Kuhn-Tucker (KT) equations which are necessary conditions for optimality for a constrained optimisation problem (Kuhn and Tucker, 1951). In other words if $f(\mathbf{x})$ and $C_i(\mathbf{x})$, $i = 1, \dots, p$ are convex

functions, then the KT equations are both necessary and sufficient for a global solution point.

The KT equations for the optimisation problem in (D.1) can be expressed as follows:

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^p \lambda_i^* \cdot \nabla C_i(\mathbf{x}^*) = 0 \quad (\text{D.2})$$

$$\nabla C_i(\mathbf{x}^*) = 0 \quad i = 1, \dots, p_e \quad (\text{D.3})$$

$$\lambda_i^* \geq 0 \quad i = p_e + 1, \dots, p \quad (\text{D.4})$$

Equation (D.2) describes a cancelling of the gradients between the objective function and the active constraints at the solution point. For the gradients to be cancelled, Lagrange Multipliers (λ_i , $i = 1, \dots, p$) are necessary to balance the deviations in magnitude of the objective function and constraint gradients. Equations (D.3) and (D.4) shows that constraints that are not active are given Lagrange multipliers equal to zero, since only active constraints are included in this cancelling operation. The solution of the KT equations forms the basis to many non-linear programming algorithms in which they attempt to compute directly the Lagrange multipliers. Constrained quasi-Newton methods guarantee linear convergence by accumulating second order information regarding the KT equations using a quasi-Newton updating procedure. These methods are commonly referred to as Sequential Quadratic Programming (SQP) methods since a QP subproblem is solved at each major iteration.

D.4 Sequential Quadratic Programming (SQP)

Sequential quadratic programming (SQP) methods represent state of the art in non-linear programming. Historically the original SQP method due to (Wilson, 1963) generates a sequence of directions, each of which is the minimiser to a quadratic programming (QP) subproblem. Based on the work of (Hann, 1977), SQP method allows you to closely mimic Newton's method for constrained optimisation just as is done for unconstrained optimisation. At each major

iteration an approximation is made of the Hessian of the Lagrange function using a quasi-Newton updating method. This idea was first proposed by (Powell, 1983). Then QP subproblem is generated whose solution is used to form a search direction for a line search procedure. More information and details about SQP can be found in (Gill *et al*, 1981), (Powell, 1983), (Fletcher, 1987), (Goldsmith, 1999). To demonstrate the method the optimisation problem in (D.1) is taken as example. The main idea is the formulation of a QP subproblem based on a quadratic approximation of the Lagrangian function as follows (note that equation (D.1) is simplified by assuming that bound constraints have been expressed as inequality constraints):

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^p \lambda_i \cdot g_i(\mathbf{x}) \quad (\text{D.5})$$

The QP subproblem is obtained by linearising the non-linear constraints.

D.5 Solving the optimisation problem using the non-linear control design tool

As mentioned in the introduction of this appendix the non-linear control design tool (NCD) based on constrained optimisation is used in chapters four and five for tuning/optimisation of both physical and control parameters. NCD uses time domain constraint bounds to represent lower and upper bounds on response signals. When the optimisation procedure starts NCD transforms constraints and simulated system output into an optimisation problem in the following form:

$$\begin{aligned} & \underset{\mathbf{x}, \gamma}{\text{minimise}} \quad \gamma \\ & \text{subject to: } g(\mathbf{x}) - \omega\gamma \leq 0, \quad \mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u \end{aligned} \quad (\text{D.6})$$

Where variable \mathbf{x} is a vectorisation of the tuneable variables, while \mathbf{x}_l and \mathbf{x}_u are vectorisations of the lower and upper bounds on the tuneable variables. The vector $\mathbf{g}(\mathbf{x})$ is a vectorisation of the constraint bound error and ω is a vectorisation of weightings on the constraints. The scalar γ imposes an element of slackness into a problem which otherwise imposes that the goals be rigidly met.

The constrained optimisation problem is solved by a SQP and quasi-Newton techniques. The principal idea behind this optimisation is the minimisation of the maximum constraint violation (or minimisation of the maximum weighted constrained error).

Figure D-1 shows an overview of the constraint window used in NCD. The set-up for upper and lower bounds is defined as follows: For upper bound constraints, the difference between the constraint boundary and the simulated error defines the constraint error. The constraint error for lower bound constraints is defined as the difference between the simulated error and the constraint boundary. The subroutine of the SQP method solves a Quadratic Programming (QP) problem at each iteration. Then, the upgrade and estimate of the Hessian of the Lagrangian is achieved. At this stage a line search method is required in which a merit function is used to solve the problem. Finally, the implementation of the SQP subproblem attempts to satisfy the Kuhn-Tucker equations, which are necessary conditions for optimality of the constrained optimisation problem.

More details about the non-linear control design tool can be found in (The MathWorks, 1997)

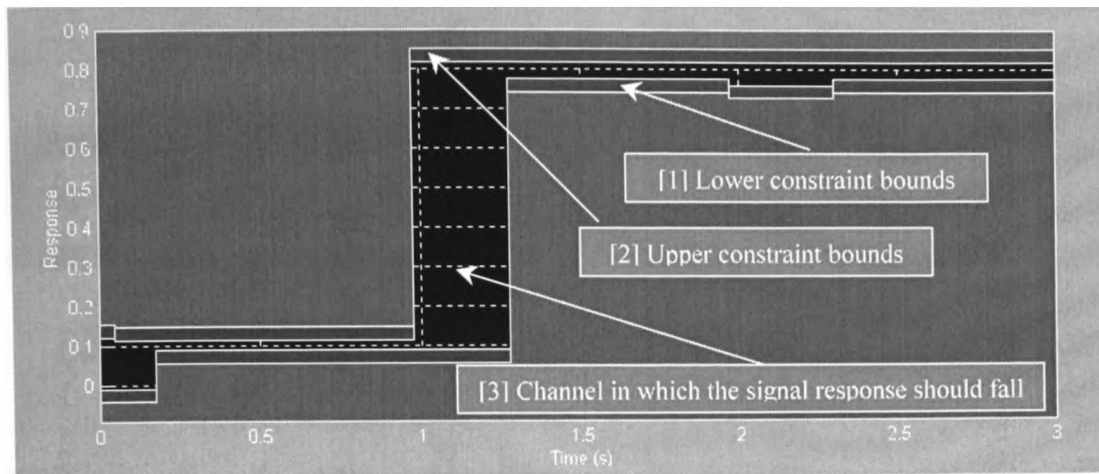


Figure D-1 Constraint window

D.6 References

FLETCHER, R. 1987. *Practical Methods for Optimisation*. 2nd edn. UK: John Wiley & Sons Ltd. 0-471-91547-5.

GILL, P. E., MURRAY, W. and WRIGHT, M. H. 1981. *Practical Optimisation*. United Kingdom: Academic Press Inc. 0-12-283952-8.

GOLDSMITH, M. J. 1999. *Sequential Quadratic Programming Methods Based on Indefinite Hessian Approximations*. PhD Thesis. Stanford University.

HANN, S. P. 1977. A Globally Convergent Method for Nonlinear Programming. *Journal of Optimisation Theory and Applications*, **22**, pp. 297-304.

KUHN, H. W. and TUCKER, A. M. 1951. *Nonlinear Programming*. Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability. University of California.

POWELL, M. J. D. 1983. Variable Metric Methods for Constrained Optimisation. In: eds. Bachem, A., Grottschel, M., and Korte, B. ed. *Mathematical Programming: The State of the Art*. USA: Springer Verlag, pp. 288-311.

THE MATHWORKS, I. 1997. *Nonlinear Control Design Blockset User's Guide*.

WILSON, R. B. 1963. *A Simplicial Method for Convex Programming*. PhD Thesis. Harvard University.

E

Robust Stability Analysis for Interval Polynomials Based on Parametric Approach

E.1 Introduction

This appendix presents definitions and theorems (Barmish, 1994) and (Bhattacharyya *et al*, 1995) related to robust stability testing of the closed-loop control system of the MIABOT V2 presented in chapter five.

E.2 Description of uncertain structure

Definition E-1 (Uncertainty Bounding Set): The uncertainty bounding set Q can be described as follows:

$$Q = \{ \mathbf{q} \in \mathbb{R}^l \mid q_i \in \mathbb{R} \quad \text{for} \quad i = 1, 2, \dots, l \} \quad (\text{E.1})$$

Note that q_i 's and therefore Q need not be connected. However, connected sets will be used since much of the results in the literature apply only to connected sets. This assumption is not restrictive because most of the physical parameters (such as viscous friction coefficients, material properties, lengths, etc) entering the uncertainty vector vary continuously over a bounded interval of the real line. Frequently, each element q_i of \mathbf{q} is described by its lower and upper bounds q_i^- and q_i^+ , respectively. Then the uncertainty set is the *box*:

$$Q = \left\{ \mathbf{q} \in \mathbb{R}^l \mid q_i^- < q_i < q_i^+ \quad \text{for } i = 1, 2, \dots, l \right\} \quad (\text{E.2})$$

Definition E-2 (Family): An uncertain function together with its uncertainty bounding set is called a *family* i.e.

$$F(., Q) = \{f(., \mathbf{q}) \mid \mathbf{q} \in Q\} \quad (\text{E.3})$$

For example, an uncertain plant $G(s, \mathbf{q})$ and its uncertainty bounding set Q form a family of plants denoted by $G(s, Q) = \{G(s, \mathbf{q}) \mid \mathbf{q} \in Q\}$. Similarly, it can be written $n(s, Q) = \{N(s, \mathbf{q}) \mid \mathbf{q} \in Q\}$ for the family of numerators and $d(s, Q) = \{D(s, \mathbf{q}) \mid \mathbf{q} \in Q\}$ for the family of denominators.

Definition E-3 (Independent Uncertainty Structure): An uncertain polynomial,

$$p(s, \mathbf{q}) = \sum_{i=0}^n a_i(\mathbf{q}) s^i \quad (\text{E.4})$$

is said to have an *independent uncertainty structure* if each component q_i of \mathbf{q} enters into only one coefficient.

Definition E-4 (Affine Linear Uncertainty Structure): An uncertain polynomial $p(s, \mathbf{q})$ is said to have an *affine linear uncertainty structure* if each coefficient function $a_i(\mathbf{q})$ is of the form:

$$a_i(\mathbf{q}) = \mathbf{a}_i^T \mathbf{q} + \beta_i \quad (\text{E.5})$$

where \mathbf{a}_i is a column vector and β_i is a scalar.

Definition E-5 (Multilinear Uncertainty Structure): An uncertain polynomial $p(s, \mathbf{q})$ is said to have a *multilinear uncertainty structure* if each function $a_i(\mathbf{q})$ is a multilinear function in the components of \mathbf{q} . That is, if all but one uncertain parameter is kept constant, then $a_i(\mathbf{q})$ is affine linear in the remaining component of \mathbf{q} .

Definition E-6 (Polynomic Uncertainty Structure): An uncertain polynomial $p(s, \mathbf{q})$ is said to have a *polynomic uncertainty structure* if each coefficient function $a_i(\mathbf{q})$ is a multivariable polynomial in the components of \mathbf{q} .

E.3 Value sets and zero exclusion condition

Definition E-7 (Value Set): The *value set* is the subset of the complex plane consisting of all values, which can be assumed by $p(j\omega, \mathbf{q})$ as \mathbf{q} ranges over Q (ω is a fixed frequency).

Theorem E-1 (Zero Exclusion Condition): A polynomial family $P(s, Q)$ having invariant degree with associated uncertainty bounding set Q , which is pathwise connected, continuous coefficient functions $a_i(\mathbf{q})$ for $i = 0, 1, 2, \dots, n$ and at least one stable member $p(s, \mathbf{q}^*)$ is robustly

stable if and only if the origin of the complex plane is excluded from the value set $P(j\omega, Q)$ at all nonnegative frequencies i.e. $0 \in p(j\omega, \mathbf{q})$ for all frequencies $\omega \geq 0$ and $\mathbf{q} \in Q$.

Definition E-8 (Robust Stability): An uncertain system with the characteristic polynomial $p(s, \mathbf{q})$ is *robustly stable* if and only if $p(s, \mathbf{q})$ is stable for all $\mathbf{q} \in Q$, where Q is the uncertainty bounding set.

Definition E-9 (Interval Polynomial Family): A family of polynomials $P(s, Q) = \{p(s, \mathbf{q}) | \mathbf{q} \in Q\}$ is said to be an *interval polynomial family* if $p(s, \mathbf{q})$ has an independent uncertainty structure, each coefficient depends continuously on \mathbf{q} and the uncertainty bounding set Q is a n -dimensional box. For brevity, is also referred to $P(s, Q)$ as an interval polynomial. Similarly, a family of uncertain plants $G(s, Q) = \{G(s, \mathbf{q}) = N(s, \mathbf{q}) / D(s, \mathbf{q}) | \mathbf{q} \in Q\}$ is said to be an *interval plant family* if both $N(s, \mathbf{q})$ and $D(s, \mathbf{q})$ are interval polynomials.

E.4 Kharitonov's theorem

Definition E-10 (Kharitonov's Polynomials): Associated with the interval polynomial family:

$$P(s, Q) = \left\{ p(s, \mathbf{q}) = \sum_{i=0}^n a_i(\mathbf{q}) s^i | \mathbf{q} \in Q \right\} \quad (\text{E.6})$$

are four fixed Kharitonov's polynomials formulated as follows:

$$K_1(s) = a_0^- + a_1^- s + a_2^+ s^2 + a_3^+ s^3 + a_4^- s^4 + a_5^- s^5 + \dots \quad (\text{E.7})$$

$$K_2(s) = a_0^+ + a_1^+ s + a_2^- s^2 + a_3^- s^3 + a_4^+ s^4 + a_5^+ s^5 + \dots \quad (\text{E.8})$$

$$K_3(s) = a_0^+ + a_1^- s + a_2^- s^2 + a_3^+ s^3 + a_4^+ s^4 + a_5^- s^5 + \dots \quad (\text{E.9})$$

$$K_4(s) = a_0^- + a_1^+ s + a_2^+ s^2 + a_3^- s^3 + a_4^- s^4 + a_5^+ s^5 + \dots \quad (\text{E.10})$$

Theorem E-2 (Kharitonov's Theorem): An interval polynomial family $P(s, Q)$ with invariant degree is robustly stable if and only if its four Kharitonov's polynomials are stable.

Definition E-11 (Kharitonov's Rectangle): Associated with the four Kharitonov's polynomials $K_1(s)$, $K_2(s)$, $K_3(s)$ and $K_4(s)$ is a rectangle (the *Kharitonov's rectangle*) whose four vertices are obtained by evaluating the four Kharitonov's polynomials at $s = j\omega_0$. Therefore given an interval polynomial family $P(s, Q)$ and a fixed frequency $\omega = \omega_0$, the value set $P(j\omega_0, Q)$ is a rectangle whose vertices are given by $K_i(j\omega_0)$ for $i=1,2,3,4$. Figure E-1 shows a generic Kharitonov rectangle. Note that the size and the position of the Kharitonov rectangle change with ω , while its sides always remain parallel to the respective real and imaginary axes. Therefore, the frequency is swept over a certain polynomial, it can be observed the motion of the Kharitonov rectangle.

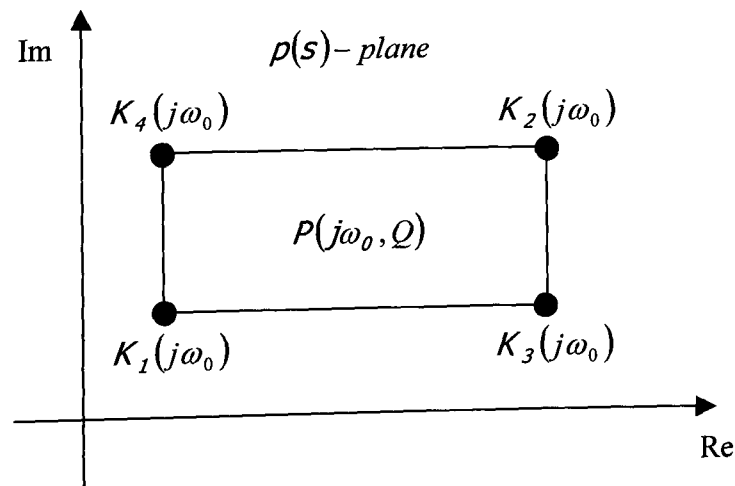


Figure E-1 The Kharitonov's rectangular for $\omega_0 \geq 0$

E.5 Robust stability testing via graphics

The Kharitonov's rectangle provides a very handy graphical means to test the robust stability of physical systems. Plots of successive Kharitonov's rectangles over the frequency interval $\omega \in [0, \infty)$ can produce observation of their motion in the complex plane. This plot together with the following theorem enables checking the stability of interval polynomials.

Theorem E-3 (Zero Exclusion for Interval Families): An interval polynomial family $\mathcal{P}(s, Q) = \{p(s, q) | q \in Q\}$ having invariant degree and at least one stable member $p(s, q^*)$ is robustly stable if and only if the origin of the complex plane is excluded from the Kharitonov's rectangle at all nonnegative frequencies i.e. $0 \notin \mathcal{P}(j\omega_0, Q)$ for all frequencies $\omega \geq 0$. In practice, there is not need to plot the Kharitonov's rectangles for all $\omega \geq 0$. A cut-off frequency $\omega_c > 0$ can be obtained such that $0 \notin \mathcal{P}(j\omega_0, Q)$ for all $\omega \geq \omega_c$. One such estimate, suggested from the classical bounds on the roots of a polynomial, provides an appropriate cut-off frequency as given by:

$$\omega_c = 1 + \frac{\max\{q_0^+, q_1^+, \dots, q_{n-1}^+\}}{q_n^-} \quad (\text{E.11})$$

for the interval polynomial $p(s, q)$ with $q_n^- > 0$ (n is the order of the polynomial). Instead of generating two-dimensional Kharitonov's rectangles, examination of the plot of the scalar function $H(\omega)$ (*Frequency Sweeping Function*) is determine if the family of polynomials \mathcal{P} is robustly stable.

Theorem E-4 (Frequency Sweeping Function for Robust Stability): Let \mathcal{P} be an interval polynomial family with interval degree, at least one stable member and associated Kharitonov's polynomials $K_1(s)$, $K_2(s)$, $K_3(s)$ and $K_4(s)$. Then with,

$$H(\omega) = \max \left\{ \begin{array}{l} \text{Re } K_1(j\omega) - \text{Re } K_2(j\omega), \\ \text{Im } K_3(j\omega) - \text{Im } K_4(j\omega) \end{array} \right\} \quad (\text{E.12})$$

it follows that \mathcal{P} is robustly stable if and only if $H(\omega) > 0$ for all frequencies $\omega \geq 0$.

E.6 References

- BARMISH, B. R. 1994. *New Tools for Robustness of Linear Systems*. New York: Macmillan Publishing Company. 0-02-306055-7.
- BHATTACHARYYA, S. P., CHAPPELLAT, H. and KEEL, L. H. 1995. *Robust Control: The Parametric Approach*. USA: Prentice-Hall Inc. 0-13-781576-X.

F

Training Algorithms

F.1 Introduction

In this appendix a training algorithm is presented to show how to choose the weights of the multilayer feedforward neural network. The neural network learns to associate a given output with a given input by adapting its weights. For this purpose a steepest descent algorithm for minimising a non-linear function is used. This algorithm for neural networks is called backpropagation. Description of the backpropagation algorithm can be also found in the contributions of (Kosko, 1992), (Jang *et al*, 1997), (Picton, 1998) and (Haykin, 1999).

F.1.1 Error definitions

To describe the training the error for a network layer (including the output layer) with n neurons is first defined as follows:

$$\mathbf{E} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} = \begin{pmatrix} (x_1 - \tilde{x}_1)^2 \\ (x_2 - \tilde{x}_2)^2 \\ \vdots \\ (x_m - \tilde{x}_m)^2 \end{pmatrix} \quad (\text{F.1})$$

If \mathbf{x} is the given output vector and $\tilde{\mathbf{x}}$ is the actual output vector the output error e is a function of the input data and the weight matrix:

$$E = [x_1 - \tilde{x}_1]^2 + [x_2 - \tilde{x}_2]^2 + \dots + [x_m - \tilde{x}_m]^2 \quad (\text{F.2})$$

This is a non-linear function in weight matrix \mathbf{W} , due to the non-linear transfer function. The training is a problem of minimising this non-linear function that also is called the loss function. The error of the hidden neurons is defined as follows:

$$e_i = \sum_{j=1}^m e_j (\text{sech}^2 u_j) w_{ji} \quad (\text{F.3})$$

The function $\text{sech}^2(x)$ is the hyperbolic secant that is the derivate of the hyperbolic tangent sigmoid (tansig) function. e_i is the error for neuron i and u is the output of the summing junction of neuron i before the application of the tansig function,

$$u_i = \sum_{j=1}^m w_{ij} x_j(k) \quad (\text{F.4})$$

F.1.2 Backpropagation to update the parameters

To demonstrate the training of a network using backpropagation the following example is used. Assume that the neural network has an input layer, a hidden layer and an output layer with n number of neurons in the whole network. Further assume that the weights are collected in the matrix W and the upgrading rule is:

$$x_i(k) = \tanh \sum_{j=1}^n w_{ij} x_j(k-1) \quad (F.5)$$

The initial value of weights is randomly guessed. It is important this value is not zero (there is a risk that the neuron will remain at the same state). Then an input is applied to the neural network and the outputs of the summing junctions are changing according to the upgrade rule. The error is then calculated for the output layer and the weights are modified to minimise the error according to the following equation:

$$\Delta w_{ij}(k) = \eta e_i (\sec h^2 u_j) x_j \quad (F.6)$$

Where the gain η is called the learning rate. A small value (e.g. 0.001) might cause a too slow change to the weights and a big value (e.g. 10) might change the weights too much.

With the new weights it is possible to calculate the errors for the hidden neurons according to Equation (F.5), and the weights between the input layer and the hidden layer is updated¹. Then another input-output pair is applied to the neural network and the same procedure is repeated once more.

¹ In the case where several hidden layers exist the updating procedure would have been done one layer at the time starting with the one closest to the output

This procedure to adapt the weights once according to a series of input-output pairs is called an epoch. The training is repeated through several epochs to make the weights better and better. It is not known how many input-output pairs it is possible to show to a neural network, but if it seems impossible to train the network to sufficient small errors, it might be because of a too small number of neurons. Also it is not known how many input-output pairs it is needed to train a neural network sufficiently.

Finally one way to sum up the backpropagation algorithm is as follows:

$$\Delta w_{nj} = -\eta_k \frac{\partial E}{\partial w_{nj}} \quad (F.7)$$

Where Δw_{nj} is the change of the weights, $\frac{\partial E}{\partial w_{nj}}$ is the gradient vector of the update functions and η is the learning rate.

F.1.3 The Levenberg-Marquardt algorithm

In order to avoid the computation of the second derivate with the backpropagation algorithm the Levenberg-Marquardt algorithm approximates the Hessian matrix by:

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} \quad (F.8)$$

and the gradient can be computed as:

$$\nabla E = \mathbf{J}^T \mathbf{e} \quad (F.9)$$

Where \mathbf{J} is the Jacobian matrix, which contains of the derivatives of the neural network errors with respect to the weights. The Jacobian matrix is much less complex to compute than the Hessian matrix. The update is performed then as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - [\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}]^{-1} \mathbf{J}^T \mathbf{e} \quad (\text{F.10})$$

Where μ is a parameter that is decreased at every successful iteration. It is only increased when the calculated change would increase the loss function. Another important aspect of μ is that it makes sure that the matrix has an inverse. The size of the Jacobian matrix is $Q \times n$, where Q is the number of training sets and n is the number of weights and biases in the neural network.

F.1.4 References

HAYKIN, S. 1999. *Neural Networks: A Comprehensive Foundation*. 2nd end. USA: Prentice-Hall, Inc. 0-13-273350-1.

JANG, J. S., SUN, C. T. and MIZUTANI, E. 1997. *Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence*. USA: Prentice-Hall, Inc. 0-13-261066-3.

KOSKO, B. 1992. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. USA: Prentice-Hall International, Inc. 0-13-612334-1.

PICTON, P. 1998. *Neural Networks*. 2nd end. USA: Palgrave. 0-333-80287-X.

G

Published Work

G.1 Introduction

The following papers have been published during the period of the research work presented in this thesis.

1. MOUZAKITIS, A. and ROBERTS, G. N. 2002. *Robust Stability of an Autonomous Mobile Robot Based on a Parametric Approach*. 15th IFAC World Congress, B'02. Barcelona, Spain.
2. MOUZAKITIS, A. and ROBERTS, G. N. 2001c. *Fuzzy Logic and Least Squares Methods as System Identification Tool for Prediction and Estimation of Dynamic Objects in Applications Which Involves Operations of Co-Operative Agents*. The 5th International Conference on Mechatronics Design and Modelling, MDM 2001. pp. 105-119, Konya, Turkey.
3. MOUZAKITIS, A. and ROBERTS, G. N. 2001b. *Autonomous Robot Behaviours for Co-operative Agents Using Fuzzy Logic and Subtractive Clustering*. International Conference in Fuzzy Logic and Technology, EUSFLAT 2001. pp. 277-280, Leicester, UK.
4. MOUZAKITIS, A. and ROBERTS, G. N. 2001a. *Modelling, Identification and Control Techniques for Intelligent Robot's Architecture and Co-operative Agents*. 3rd Workshop on European Scientific and Industrial collaboration, WESIC 2001. pp. 309-318, Enschede, The Netherlands.

5. MOUZAKITIS, A. and ROBERTS, G. N. 2000b. *A New Decentralised Control for Co-operative Agents Using Fuzzy Logic*. The 14th International Conference on Systems Engineering, ICSE 2000. pp. 436-501, Coventry, UK.
6. MOUZAKITIS, A. and ROBERTS, G. N. 2000a. *Intelligent Autonomous Mobile Robots and Co-operative Multi-Agent Systems: Research, Design, Future Developments and Improvements*. EUREL, European Advanced Robotics Systems, Masterclass and Conference - Robotics 2000. pp. 600-609, Salford, Manchester, UK.

ROBUST STABILITY OF AN AUTONOMOUS MOBILE ROBOT BASED ON A PARAMETRIC APPROACH

A. Mouzakis and G. N. Roberts

*Mechatronics Research Centre
University of Wales College, Newport
Allt-yr-yn Campus, P.O. Box 180, NP20 5XR Newport,
United Kingdom
Tel: +44(0) 1633 432487, Fax: +44(0) 1633 432442.
Email: alexandros.mouzakis@newport.ac.uk*

Abstract: In this paper the robust stability of an autonomous mobile robot based on a parametric approach is presented. The closed-loop control consists of Multi-Input / Multi-Output (MIMO) uncertain plant (mobile robot) and a MIMO Proportional Integral (PI) controller. Using Kharitonov's Theorem and Zero Exclusion Condition the closed-loop system is proved to be robustly stable in the presence of parameter variations or the system dynamics which are sensitive with respect to these parameters (Uncertainty). Simulation results are presented to demonstrate the robust analysis and to prove the robust stability of the closed-loop system. *Copyright © 2002 IFAC*

Keywords: Autonomous mobile robots, MIMO, Uncertainty, Interval polynomials, Robust control, Robust stability, Robust analysis.

1. INTRODUCTION

Almost all-dynamic systems depend on varying or uncertain parameters and this is certainly true for small mobile robots. For instance, consider the velocity of a mobile robot (i.e. due to the battery variations), or the mass of a mobile robot (i.e. adding or removing components) all these parameters may vary more or less significantly within certain bounds and they influence the system dynamics. Traditional control design approaches consider a fixed operating point in which the controller (compensator) is robust enough to stabilise the plant for different operating conditions. These approaches produce good results if the parameter variations are small or the system dynamics are not too sensitive with respect to these parameters. For significant (large) parameter variations these control design methods reach their performance limits. Robust control theory

based on interval polynomials is an effective approach when considering plant uncertainty. The interval polynomial problem was first posed by Faedo (1953) who attempted to solve it using the Routh-Hurwitz conditions. Kharitonov (1978) gave the complete solution with his theorem for real polynomials, which then he extended to the complex case. Since then many papers have been published based on parametric approach regarding robust stability of uncertain plants (Siljak, 1989; Kontogiannis and Munro, 1996).

The main objective of this paper is to show that the closed-loop system (mobile robot and controller) is robustly stable to varying or uncertain parameters. The parametric approach based on interval polynomials using Kharitonov theorem was chosen due to its simplicity and its suitability when considering uncertainty of interval polynomials.

This paper is organised into 5 sections. The control of the MIABOT V2 mobile robot is described in section 2. Section 3 presents the robust stability analysis for interval polynomials together with a number of simulations to verify the robust stability of the closed-loop system. Some discussions of the work are given in section 4. Finally, section 5 contains the conclusions of the work presented.

2. CONTROL OF MIABOT V2 MOBILE ROBOT

MIABOT V2 mobile robots shown in Fig 1 are a small sized (8cm³), two-wheeled autonomous mobile robots, which have the ability to achieve speeds up to 1-1.5m/sec by driving each wheel independently (two DC motors). A Multi-Input / Multi-Output (MIMO) Proportional Integral (PI) controller has been designed for accurate speed control.

Fig 2 shows the overall system structure of the closed-loop control. The open-loop robot model $G(s)$ consists of two inputs, and two outputs. The inputs are left and right voltages of the left and right wheel respectively. Outputs are the speed of the left and right wheel. The second-order dynamic model of the mobile robot is described in the following transfer function matrix (TFM) form:

$$G(s) = \begin{bmatrix} \frac{10.64s + 554.6}{s^2 + 108s + 2835} & \frac{2.245s + 25.42}{s^2 + 108s + 2835} \\ \frac{2.245s + 25.42}{s^2 + 108s + 2835} & \frac{10.64s + 554.6}{s^2 + 108s + 2835} \end{bmatrix} \quad (1)$$

Similarly the MIMO PI controller $G_c(s)$ is described in the following TFM form:

$$G_c(s) = \begin{bmatrix} \frac{1.354s + 103}{s} & \frac{-1.577s - 25.19}{s} \\ \frac{-1.577s - 25.19}{s} & \frac{1.354s + 103}{s} \end{bmatrix} \quad (2)$$

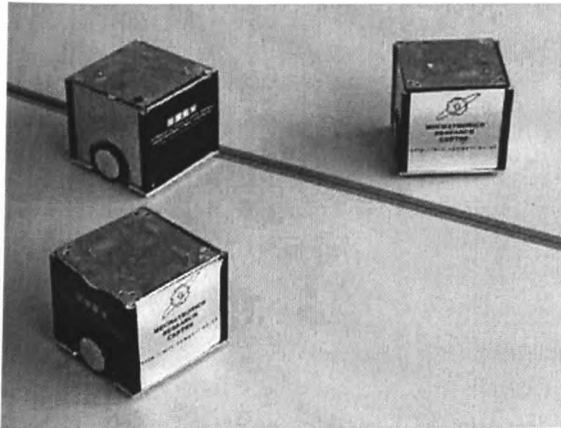


Fig. 1. MIABOT V2 mobile robots.

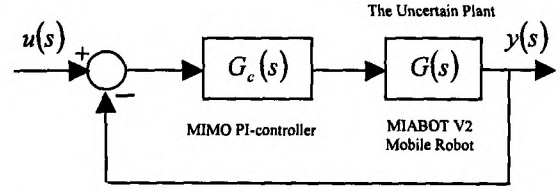


Fig. 2. Closed-loop system.

Equations (3a, 3b, 3c and 3d) describe the closed-loop system illustrated in Fig 2.

$$\frac{Y_1(s)}{U_1(s)} = \frac{10.87s^3 + 1679s^2 + 7.806e004s + 1.079e006}{s^4 + 129.7s^3 + 6265s^2 + 1.345e005s + 1.079e006} = g_{11}(s) \quad (3a)$$

$$\frac{Y_1(s)}{U_2(s)} = \frac{-13.75s^3 - 877.2s^2 - 1.135e004s - 0.01564}{s^4 + 129.7s^3 + 6265s^2 + 1.345e005s + 1.079e006} = g_{12}(s) \quad (3b)$$

$$\frac{Y_2(s)}{U_1(s)} = \frac{-13.75s^3 - 877.2s^2 - 1.135e004s - 0.01564}{s^4 + 129.7s^3 + 6265s^2 + 1.345e005s + 1.079e006} = g_{21}(s) \quad (3c)$$

$$\frac{Y_2(s)}{U_2(s)} = \frac{10.87s^3 + 1679s^2 + 7.806e004s + 1.079e006}{s^4 + 129.7s^3 + 6265s^2 + 1.345e005s + 1.079e006} = g_{22}(s) \quad (3d)$$

Fig 3 shows the transfer function matrix description expanded for the MIMO system in Fig 2.

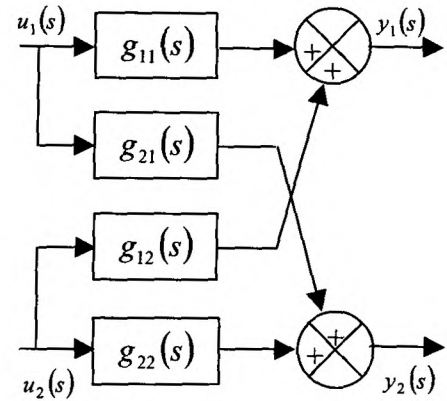


Fig. 3. The transfer function matrix description expanded for the 2x2 system (plant and controller)

The characteristic equation given in Equation (4) will be used for the testing of the robust stability of the closed-loop system given Fig 2.

$$s^4 + 129.7s^3 + 6265s^2 + 134500s + 1079000 \quad (4)$$

3. ROBUST STABILITY ANALYSIS FOR INTERVAL POLYNOMIALS

In this section definitions and theorems related to robust stability analysis for interval polynomials are given (Bhattacharyya, *et al.*, 1995). First the description of uncertainty structure is given through several definitions following by both definitions and theorems regarding value sets and zero exclusion condition. Then the Kharitonov's theorem (Barmish, 1994) is described in brief together with results of the application (closed-loop control). Finally, the robust stability of the closed-loop control is demonstrated using graphical techniques.

3.1 Description of uncertainty structure.

Definition 1 (Uncertainty Bounding Set): The uncertainty bounding set Q is the set

$$Q = \{q \in R^l \mid q_i \in R \text{ for } i = 1, 2, \dots, l\} \quad (5)$$

Note that q_i 's and therefore Q need not be connected. However, connected sets will be used since much of the results in literature apply only to connected sets. This assumption is not restrictive because most of the physical parameters (such as viscous friction coefficients, material properties, lengths, etc) entering the uncertainty vector vary continuously over a bounded interval of the real line.

Frequently, each element q_i of q is described by its lower and upper bounds q_i^- and q_i^+ , respectively. Then the uncertainty set is the box

$$Q = \{q \in R^l \mid q_i^- < q_i < q_i^+ \text{ for } i = 1, 2, \dots, l\} \quad (6)$$

Definition 2 (Family): An uncertain function together with its uncertainty bounding set is called a *family* i.e.

$$F(., Q) = \{f(., q) \mid q \in Q\} \quad (7)$$

For example, an uncertain plant $G(s, q)$ and its uncertainty bounding set Q form a family of plants denoted by $G(s, Q) = \{G(s, q) \mid q \in Q\}$. Similarly, it can be written $n(s, Q) = \{N(s, q) \mid q \in Q\}$ for the family of numerators and $d(s, Q) = \{D(s, q) \mid q \in Q\}$ for the family of denominators.

Definition 3 (Independent Uncertainty Structure): An uncertain polynomial

$$p(s, q) = \sum_{i=0}^n a_i(q) s^i \quad (8)$$

is said to have an *independent uncertainty structure* if each component q_i of q enters into only one coefficient.

Definition 4 (Affine Linear Uncertainty Structure): An uncertain polynomial $p(s, q)$ is said to have an *affine linear uncertainty structure* if each coefficient function $a_i(q)$ is of the form

$$a_i(q) = a_i^T q + \beta_i \quad (9)$$

where a_i is a column vector and β_i is a scalar

Definition 5 (Multilinear Uncertainty Structure): An uncertain polynomial $p(s, q)$ is said to have a *multilinear uncertainty structure* if each function $a_i(q)$ is a multilinear function in the components of q . That is, if all but one uncertain parameter is kept constant, then $a_i(q)$ is affine linear in the remaining component of q .

Definition 6 (Polynomial Uncertainty Structure): An uncertain polynomial $p(s, q)$ is said to have a *polynomial uncertainty structure* if each coefficient function $a_i(q)$ is a multivariable polynomial in the components of q .

3.2 Value sets and zero exclusion condition.

Definition 7 (Value Set): The *value set* is the subset of the complex plane consisting of all values which can be assumed by $p(j\omega, q)$ as q ranges over Q (ω is a fixed frequency).

Theorem 1 (Zero Exclusion Condition): A polynomial family $P(s, Q)$ having invariant degree with associated uncertainty bounding set Q , which is pathwise connected, continuous coefficient functions $a_i(q)$ for $i = 0, 1, 2, \dots, n$ and at least one stable member $p(s, q^*)$ is robustly stable if and only if the origin of the complex plane is excluded from the value set $P(j\omega, Q)$ at all nonnegative frequencies i.e.

$$0 \notin p(j\omega, q)$$

for all frequencies $\omega \geq 0$ and $q \in Q$.

Definition 8 (Robust Stability): An uncertain system with the characteristic polynomial $p(s, q)$ is *robustly stable* if and only if $p(s, q)$ is stable for all $q \in Q$, where Q is the uncertainty bounding set.

Definition 9 (Interval Polynomial Family): A family of polynomials $P(s, Q) = \{p(s, q) | q \in Q\}$ is said to be an *interval polynomial family* if $p(s, q)$ has an independent uncertainty structure, each coefficient depends continuously on q and the uncertainty bounding set Q is a n -dimensional box.

For brevity, is also referred to $P(s, Q)$ as an interval polynomial. Similarly, a family of uncertain plants $G(s, Q) = \{G(s, q) = N(s, q)/D(s, q) | q \in Q\}$ is said to be an *interval plant family* if both $N(s, q)$ and $D(s, q)$ are interval polynomials.

3.3 Kharitonov's theorem.

Definition 10 (Kharitonov Polynomials): Associated with the interval polynomial family

$$P(s, Q) = \left\{ p(s, q) = \sum_{i=0}^n a_i(q) s^i | q \in Q \right\} \quad (10)$$

are four fixed Kharitonov polynomials

$$K_1(s) = a_0^- + a_1^- s + a_2^+ s^2 + a_3^+ s^3 + a_4^- s^4 + a_5^- s^5 + \dots \quad (11)$$

$$K_2(s) = a_0^+ + a_1^+ s + a_2^- s^2 + a_3^- s^3 + a_4^+ s^4 + a_5^+ s^5 + \dots \quad (12)$$

$$K_3(s) = a_0^- + a_1^- s + a_2^- s^2 + a_3^+ s^3 + a_4^+ s^4 + a_5^- s^5 + \dots \quad (13)$$

$$K_4(s) = a_0^+ + a_1^+ s + a_2^+ s^2 + a_3^- s^3 + a_4^- s^4 + a_5^+ s^5 + \dots \quad (14)$$

Theorem 2 (Kharitonov's Theorem): An interval polynomial family $P(s, Q)$ with invariant degree is robustly stable if and only if its four Kharitonov polynomials are stable.

Definition 11 (Kharitonov Rectangle): Associated with the four Kharitonov polynomials $K_1(s)$, $K_2(s)$, $K_3(s)$ and $K_4(s)$ is a rectangle (the *Kharitonov rectangle*) whose four vertices are obtained by evaluating the four Kharitonov polynomials at $s = j\omega_0$. Therefore given an interval polynomial family $P(s, Q)$ and a fixed frequency $\omega = \omega_0$, the value set $P(j\omega_0, Q)$ is a rectangle whose vertices are given by $K_i(j\omega_0)$ for $i = 1, 2, 3, 4$.

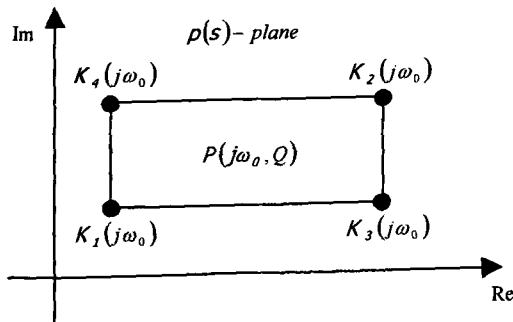


Fig. 4. The Kharitonov rectangle for $\omega_0 \geq 0$.

Fig. 4 shows a generic Kharitonov rectangle. Note that the size and the position of the Kharitonov rectangle change with ω , while its sides always remain parallel to the respective real and imaginary axes. Therefore, as we sweep the frequency over a certain polynomial, we can observe the motion of the Kharitonov rectangle.

3.4 The application (robust stability of mobile robot)

According to the definitions and theorems from subsection 3.1, 3.2 and 3.3 the robust stability of the closed-loop system of Equation (4) can be proved. Equation (4) can be written as interval polynomial in the following form:

$$p(s, q) = q_4 s^4 + q_3 s^3 + q_2 s^2 + q_1 s + q_0 \quad (15)$$

where

$$q = [q_4, q_3, q_2, q_1, q_0] \quad (16)$$

is the vector of uncertain parameters, and assume that

$$q_4 \in [1, 2], q_3 \in [123.2, 136.2], q_2 \in [5951, 6579], \\ q_1 \in [127780, 141230], q_0 \in [1025100, 1133000]$$

then the uncertainty bounding set (Definition 1) is

$$Q = \left\{ q \mid \begin{array}{l} q_0 \in [1025100, 1133000], q_1 \in [127780, 141230], \\ q_2 \in [5951, 6579], q_3 \in [123.2, 136.2], q_4 \in [1, 2] \end{array} \right\} \quad (17)$$

The above interval polynomial family is denoted by writing an interval polynomial family of the form:

$$p(s, q) = [1, 2]s^4 + [123.2, 136.2]s^3 + [5951, 6579]s^2 \\ + [127780, 141230]s + [1025100, 1133000] \quad (18)$$

$$\text{where } 0 \notin [q_4^-, q_4^+] = [1, 2] \Rightarrow$$

interval polynomial family $P(s, Q)$ has invariant degree.

From Definition 10 the four fixed Kharitonov polynomials are derived as follows:

$$K_1(s) = s^4 + 136.2s^3 + 6579s^2 + 127780s + 1025100 \quad (19)$$

$$K_2(s) = 2s^4 + 123.2s^3 + 5951s^2 + 141230s + 1133000 \quad (20)$$

$$K_3(s) = 2s^4 + 136.2s^3 + 5951s^2 + 127780s + 1133000 \quad (21)$$

$$K_4(s) = s^4 + 123.2s^3 + 6579s^2 + 141230s + 1025100 \quad (22)$$

Using Routh criterion it is easy to verify that all four Kharitonov polynomials are stable (Routh column is positive in all cases). Hence it can be concluded that the closed-loop control system is robustly stable. The same conclusions can be drawn using the Zero Exclusion Condition in subsection 3.5 below.

3.5 Robust stability testing via graphics.

The Kharitonov rectangle provides a very handy graphical means to test the robust stability of physical systems. Plots of successive Kharitonov rectangles over the frequency interval $\omega \in [0, \infty)$, can produce observation of their motion in the complex plane. This plot together with the following theorem enables checking the stability of interval polynomials.

Theorem 3 (Zero Exclusion for Interval Families):

An interval polynomial family $P(s, Q) = \{p(s, q) | q \in Q\}$ having invariant degree and at least one stable member $p(s, q^*)$ is robustly stable if and only if the origin of the complex plane is excluded from the Kharitonov rectangle at all nonnegative frequencies i.e. $0 \notin P(j\omega, Q)$ for all frequencies $\omega \geq 0$.

In practice, there is not need to plot the Kharitonov rectangles for all $\omega \geq 0$. A cut-off frequency $\omega_c > 0$ can be obtained such that $0 \notin P(j\omega, Q)$ for all $\omega \geq \omega_c$. One such estimate, suggested from the classical bounds on the roots of a polynomial, provides an appropriate cut-off frequency as given by

$$\omega_c = 1 + \frac{\max\{q_0^+, q_1^+, \dots, q_{n-1}^+\}}{q_n^-} \quad (23)$$

for the interval polynomial $p(s, q)$ with $q_n^- > 0$ (n is the order of the polynomial).

Instead of generating two-dimensional Kharitonov rectangles, examination of the plot of the scalar function $H(\omega)$ (Frequency Sweeping Function) is determine if the family of polynomials P is robustly stable.

Theorem 4 (Frequency Sweeping Function for Robust Stability): Let P be an interval polynomial family with interval degree, at least one stable member and associated Kharitonov polynomials $K_1(s)$, $K_2(s)$, $K_3(s)$ and $K_4(s)$. Then with

$$H(\omega) = \max \begin{Bmatrix} \text{Re } K_1(j\omega), -\text{Re } K_2(j\omega), \\ \text{Im } K_3(j\omega), -\text{Im } K_4(j\omega) \end{Bmatrix} \quad (24)$$

it follows that P is robustly stable if and only if $H(\omega) > 0$ for all frequencies $\omega \geq 0$.

3.6 Verification of the closed-loop robust stability

To verify that the closed loop control system is robustly stable further testing using graphics is performed using the Theorem 3 and 4. The characteristic equation of the closed-loop system is

given in Equation (4). Equations (15) and (16) both provide the uncertainty vector q and the uncertainty bounding set Q . It was already shown that the interval polynomial family $P(s, Q)$ has invariant degree. In accordance with Theorem 3, the first step in the graphical test for robust stability requires that at least one polynomial in $P(s, Q)$ that is stable. Using the midpoint of each interval from Equation (17) q^* is obtained as follows:

$$q^* = (1.5, 129.7, 6265, 134505, 1079050) \quad (25)$$

then

$$p(s, q^*) = 1.5s^4 + 129.7s^3 + 6265s^2 + 134505s + 1079050 \quad (26)$$

Using the Routh criterion it is easy to verify that $p(s, q^*)$ is stable. The cut-off frequency can be calculated from Equation (22) as follows:

$$\omega_c = 1 + \frac{\max\{1133000, 141230, 6579, 136.2\}}{1} \quad (27)$$

$$= 1133001 \text{ rad/s}$$

The Kharitonov rectangles can be plotted to verify the stability of the closed loop system for frequency range $\omega \in [0, 1133001] \text{ rad/s}$. For more convenient frequency range $\omega \in [0, 100] \text{ rad/s}$ plot is shown in order the zero point in the graph to be visible. Fig 5 shows the Kharitonov rectangles for the closed-loop system. Fig 6 shows the plot of the frequency sweeping function $H(\omega)$.

Since the origin is excluded from the Kharitonov rectangles (Fig 5) it is concluded that the closed-loop control system is robustly stable. The same conclusion can be obtained from Fig 6 because it can be observed that the frequency sweeping function $H(\omega)$ is positive for all $\omega \in [0, 100] \text{ rad/s}$.

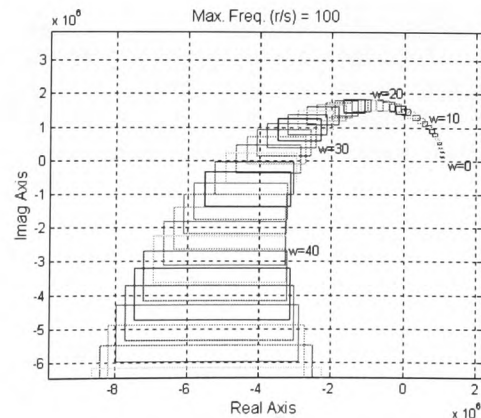


Fig. 5. Kharitonov rectangles for the controlled closed-loop control system.

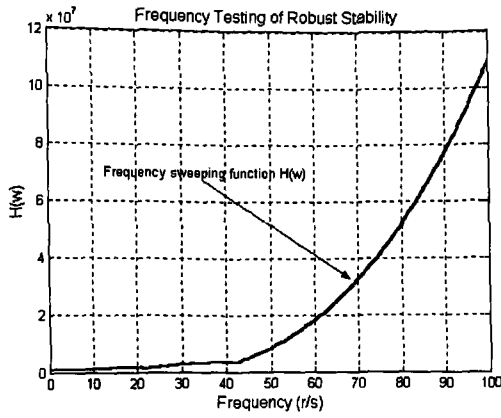


Fig. 6. A plot of $H(\omega)$ versus ω .

4. DISCUSSIONS

The uncertainty of the closed-loop system was modelled by replacing the coefficients of the closed-loop characteristic equation of the MIMO system with closed interval polynomials. Although the robust stability was proven, a question remains of how to map a closed-loop characteristic equation of system to the system's physical parameters. For example, the mobile robot for which the robust analysis took place weighs 0.5 kg. If there was a need for 10% increase of its mass (i.e. adding more sensing elements) how the coefficients of the closed-loop characteristic equation will change is of interest. To map the change of the robot's mass to the change in the coefficients of the closed-loop characteristic equation is very difficult. In order to demonstrate this, consider the modified open-loop robot transfer function matrix in Equation (28), and the closed-loop transfer function of the system in Equation (29a, 29b, 29c, 29d) resulting from the 10% increase in mass. It can be observed that the intervals used for Equation (16) do not include all the variations in coefficients resulting from a 10% increase in mass. Care must therefore be taken in selecting the most suitable interval in order to accommodate the range of the expected parameter variations. The closed-loop control system described in Equations (3a, 3b, 3c, 3d) was tested again for robust stability based on new uncertainty bounding set given in equation (30). And was found to be robustly stable.

$$G(s) = \begin{bmatrix} \frac{10.06s + 504.2}{s^2 + 102.3s + 2577} & \frac{1.659s + 23.11}{s^2 + 102.3s + 2577} \\ \frac{1.659s + 23.11}{s^2 + 102.3s + 2577} & \frac{10.06s + 504.2}{s^2 + 102.3s + 2577} \end{bmatrix} \quad (28)$$

$$\frac{Y_1(s)}{U_1(s)} = \quad (29a)$$

$$\frac{11s^3 + 1576s^2 + 7.096e004s + 9.814e005}{s^4 + 124.3s^3 + 5793s^2 + 1.223e005s + 9.814e005} = g_{11}(s)$$

$$\frac{Y_1(s)}{U_2(s)} = \quad (29b)$$

$$\frac{-13.62s^3 - 846.4s^2 - 1.032e004s}{s^4 + 124.3s^3 + 5793s^2 + 1.223e005s + 9.814e005} = g_{12}(s)$$

$$\frac{Y_2(s)}{U_1(s)} = \quad (29c)$$

$$\frac{-13.62s^3 - 846.4s^2 - 1.032e004s}{s^4 + 124.3s^3 + 5793s^2 + 1.223e005s + 9.814e005} = g_{21}(s)$$

$$\frac{Y_2(s)}{U_2(s)} = \quad (29d)$$

$$\frac{11s^3 + 1576s^2 + 7.096e004s + 9.814e005}{s^4 + 124.3s^3 + 5793s^2 + 1.223e005s + 9.814e005} = g_{22}(s)$$

$$Q = \left\{ q \begin{cases} q_0 \in [981400, 1133000], q_1 \in [122300, 141230] \\ q_2 \in [5793, 6579], q_3 \in [123.2, 136.2], q_4 \in [1, 2] \end{cases} \right\} \quad (30)$$

5. CONCLUSIONS

In this paper the robust analysis of a closed-loop MIMO system based on parametric approach was investigated. Robust stability is vital due to the dynamics of the system. To demonstrate robust stability the Kharitonov theorem was used, based on interval polynomials control theory. The closed-loop control system was shown to be robustly stable under uncertainty based on closed intervals (first arbitrary then specific). Finally the robust stability was verified using graphical techniques based on Zero Exclusion Condition.

REFERENCES

- Barmish, B.R. (1994). *New Tools for Robustness of Linear Systems*, MacMillan, New York.
- Bhattacharyya, S.P., H. Chapellat and L.H. Keel. (1995). *Robust Control: The Parametric Approach*, Prentice Hall, New Jersey.
- Faedo, S. (1953). A New Stability Problem for Polynomials with Real Coefficients. *Ann. Scuola Norm. Pisa Sci. Fis. Mat. Ser.* 3-7, 53-63.
- Kharitonov, V.L. (1978). Asymptotic stability of an equilibrium position of a family of systems of linear differential equations. *Differential Uraunen.* 14, 2086-2088.
- Kontogiannis, E., N. Munro (1996). *The Fundamental Dominance Condition for MIMO Systems with Parametric Uncertainty*. Proc. IEEE/IFAC Control '96, UK, 1202-1207.
- Siljad, D. D. (1989). Parameter Space Methods for Robust Control Design: A Guided Tour. *IEEE Transactions on Automatic Control.* 34-7, 674-688.

FUZZY LOGIC AND LEAST SQUARES METHODS AS SYSTEM IDENTIFICATION TOOL FOR PREDICTION AND ESTIMATION OF DYNAMIC OBJECTS IN APPLICATIONS WHICH INVOLVES OPERATIONS OF CO-OPERATIVE AGENTS

Alexandros MOUZAKITIS, Mechatronics Research Centre, University of Wales College, Newport, United Kingdom.

Geoff ROBERTS, Mechatronics Research Centre, University of Wales College, Newport, United Kingdom.

ABSTRACT

The paper is concerned with real-time, on-line estimation and prediction of dynamic objects using a hybrid combination of least squares and fuzzy logic. The development of the real-time algorithm is described together with experimental results. The particular application considered is robot football where by estimating positions of ball and mobile robots in x and y co-ordinates separately the algorithm is shown to have increased prediction capabilities.

1. INTRODUCTION

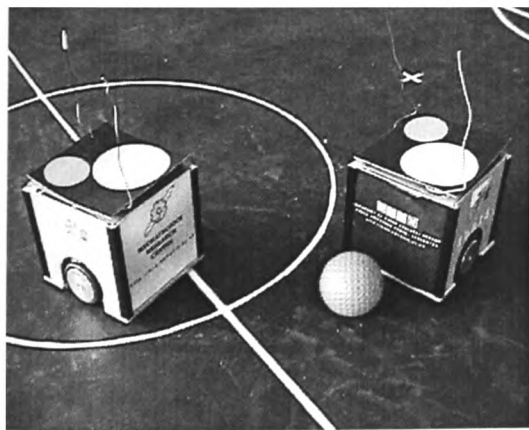
The problem of system identification is generally referred to as the determination of a mathematical model for a system or a process by observing its input-output relationships. In many cases it is necessary, or useful, to have a model available on-line while the system is in operation. The model should then be based on observation up to the current time. The need for such an on-line model typically arises since a model is required in order to take some decisions about the system. Numbers of such decisions that have to be made are given in [Ljung 1999]. However, there are two methods in which identification can be accomplished. One is off line identification, in which a record of input/output data is first observed and the model parameters are estimated based on the entire data record. The other approach is on-line identification, the parameter estimates are recursively calculated for every data set so that new data is used to correct and update the existing estimates. If the updating process can be made very fast it becomes possible to obtain parameter estimates of time varying systems with reasonable accuracy.

Traditional control designs are based on physical models of the system to be controlled. If mathematical models are difficult to be defined, i.e. due to the complexity of the system or to the kind of available information (rather vague and uncertain), cognitive modelling represents a more viable alternative. Using a cognitive based modelling approach, the aim is to design control system based on a model of the expert, who is able to specify the general properties of the system, rather than a model of the system to be controlled. In this respect, fuzzy logic has proved to be a powerful tool [Zadeh et al 1997, Roberts 1998]. In this paper a hybrid solution of both the least squares method and fuzzy logic is used for prediction and estimation of dynamic objects

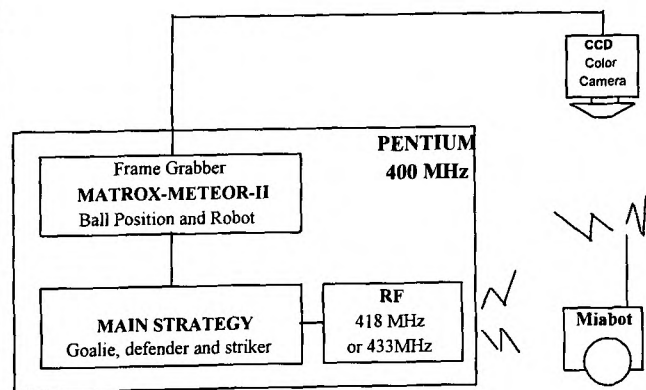
This paper is organised into seven sections. The problem statement is defined in section two. Section three presents a brief overview of the theory of least squares methods. A brief overview of how fuzzy logic operates is presented in section four. Section five contains the hybrid scheme of least squares methods and fuzzy logic. Section six shows some experimental results of the proposed approach. Section seven contains the conclusions of the overall work.

2. PROBLEM STATEMENT

In this paper it is assumed that there are number n mobile robots (Figure 1a), which much complete a predefined mission i.e. play football. CCD colour camera and frame grabber



(a)



(b)

Figure 1. (a) MIABOT V2 mobile robots (b) Overall system structure

(Figure 1b), are used to track the ball and robots positions in a x, y coordinate frame. The information are processed in the Pentium 400 MHz computer, which sends the signal commands via the serial port in to the robots receiver module. The objective is to identify the position of the ball at one time step ahead ($t+1$). To achieve this goal an improved approach for estimation and prediction is required.

3. LEAST SQUARES THEORY

This section in brief describes the operation of least squares theory. Karl Gauss first proposed least squares theory for carrying out his work in orbit prediction of planets. Least squares theory has since become a major tool for parameter estimation from experimental data. It provides a mathematical procedure by which a model can achieve a best fit to experimental data in the sense of minimum error squares. In the following least squares theory in brief is presented, but more details can be found in [Cowan and Grant 1985, Isermann et al 1992].

Suppose that is a variable y that is related linearly to a set of n variables $x = (x_1, x_2, \dots, x_n)$ that is

$$y = \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n \quad (1)$$

in which $\theta = (\theta_1, \theta_2, \dots, \theta_n)$ is a set of constant parameters. It is assumed here that θ_i is unknown and the estimation of their values required by observing the y and x at different times.

Let assume that a sequence of m observations on both y and x has been made at times t_1, t_2, \dots, t_m , and denote the measured data by $y(i)$ and $x_1(i), x_2(i), \dots, x_n(i)$, $i = 1, 2, \dots, m$. Now it is possible to relate these data by the following set of m linear equations:

$$y(i) = \theta_1 x_1(i) + \theta_2 x_2(i) + \dots + \theta_n x_n(i) \quad i = 1, 2, \dots, m \quad (2)$$

The equation (2) can be alternative and conveniently arranged into a simple matrix form

$$y = X\theta \quad (3)$$

where

$$y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(m) \end{bmatrix} \quad X = \begin{bmatrix} x_1(1) & \dots & x_n(1) \\ x_1(2) & & x_n(2) \\ \vdots & & \vdots \\ x_1(m) & & x_n(m) \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

To be able to estimate the n parameters θ_i , it is necessary that $m \geq n$. If $m = n$, then θ can be solved uniquely from the equation (3) by

$$\hat{\theta} = X^{-1}y \quad (4)$$

provided that X^{-1} , exists. The estimate of θ is denoted by $\hat{\theta}$. However when $m > n$ it is generally not possible to determine a set of θ_i exactly satisfying all m equations (2) because the data maybe complicated by random measurements noise error in the model or a combination of both. Therefore one way to determine θ is on the basis of least error squares.

Define an error vector $e = (e_1, e_2, \dots, e_m)^T$ and let

$$e = y - X\theta \quad (5)$$

then let $\hat{\theta}$ in such a way that the criterion J is minimised.

$$J = \sum_{i=1}^m e_i^2 = e^T e \quad (6)$$

To carry out the minimisation the criterion becomes

$$J = (y - X\theta)^T (y - X\theta) = y^T y - \theta^T X^T y - y^T X\theta + \theta^T X^T X\theta$$

Differentiating J with respect to θ and equating the result to zero the conditions on the estimate $\hat{\theta}$ that minimises J can be determined. Thus

$$\left. \frac{\partial J}{\partial \theta} \right|_{\theta=\hat{\theta}} = -2X^T y + 2X^T X\hat{\theta} = 0, \text{ this yields} \quad (7)$$

$$X^T X\hat{\theta} = X^T y$$

from which $\hat{\theta}$ can be solved as

$$\hat{\theta} = (X^T X)^{-1} X^T y \quad (8)$$

The result of equation (8) is called least squares estimator of θ .

4. FUZZY THEORY

This section of the paper gives a brief description of the basic concepts of fuzzy theory including fuzzy sets, operations on fuzzy sets and inference engine [Zimmermann 1991, Wang 1997].

Fuzzy sets

In the classical theory a crisp set A is a collection of objects or elements x taken in a universal set U . The characteristic function $\mu_A : x \in U \rightarrow \{0,1\}$ declares which elements of U are members of the set and which are not. If x belongs to A then $\mu_A(x) = 1$ otherwise $\mu_A(x) = 0$.

For a fuzzy set, the characteristic function allows various degrees of membership for each object or element. When \tilde{A} is a fuzzy set and x is a relevant element, the proposition " x is a member of \tilde{A} " is not necessarily true or false, but it may be true only to some degree, the degree to which x is actually member of \tilde{A} . According to this introductory discussion the formal definition of a fuzzy set is stated as follows:

Let U be a universal set, then the fuzzy set \tilde{A} in U is defined by a set of pairs:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | x \in U\} \quad (9)$$

where the $\mu_{\tilde{A}}(x)$ is called the membership function of x in \tilde{A} . U is defined as Universe of discourse and \tilde{A} is defined as the supremum of $\mu_{\tilde{A}}(x)$ over U .

Operations on fuzzy sets

In fuzzy theory the most common crisp operators, such as union, intersection and complement are used, dependent on particular application. According to fuzzy set theory for any two fuzzy set \tilde{A} and \tilde{I} defined in U with membership functions being $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{I}}(x)$ respectively the following applies:

1) membership function $\mu_{\tilde{C}}(x)$ of their intersection $\hat{C} = \tilde{A} \cap \tilde{I}$ is pointwise defined by

$$\mu_{\tilde{C}}(x) = \min\{\mu_{\tilde{A}}(x), \mu_{\tilde{I}}(x)\}, \quad x \in U \quad (10)$$

2) membership function $\mu_{\hat{N}}(x)$ of their union $\hat{N} = \tilde{A} \cup \tilde{I}$ is pointwise defined by

$$\mu_{\hat{N}}(x) = \max\{\mu_{\tilde{A}}(x), \mu_{\tilde{I}}(x)\}, \quad x \in U \quad (11)$$

3) membership function $\mu_{\alpha\hat{A}}(x)$ of the complement of a fuzzy set \tilde{A} is defined by

$$\mu_{\alpha\hat{A}}(x) = 1 - \mu_{\tilde{A}}(x), \quad x \in U \quad (12)$$

Fuzzy inference

Fuzzy inference is a reasoning method using fuzzy theory in which the human knowledge is expressed using linguistic rules. Membership functions, assigned with linguistic variables, are used to fuzzify physical quantities. Fuzzified inputs are inferred to a fuzzy rule base which characterises the relationship between fuzzy inputs and fuzzy outputs. For instance, a simple fuzzy control rule relating the input a to the output b maybe expressed in the following form:

IF a is C **THEN** b is D

where C and D are fuzzy values defined on the universes of X and Y , respectively. The response of each fuzzy rule is weighted according to the degree of membership of its input conditions. The role of the fuzzy inference is to provide a set of common actions according to fuzzified inputs. The next step is the defuzzification method to convert the fuzzy values into a crisp output value of the fuzzy system. In other words defuzzification is the process of mapping from a space of inferred fuzzy control actions to a space of non-fuzzy (crisp) control actions. The defuzzification method used for this work is the centroid method and is expressed in the form as follows:

$$b^* = \frac{\sum_{i=1}^n \mu_D(c_i) * c_i}{\sum_{i=1}^n \mu_D(c_i)} \quad (13)$$

where b^* is a crisp output of the fuzzy system, n is the number of control rules associated with the fuzzified inputs, and c_i is the centroid of the membership function associated with each linguistic value in the output space.

5. HYBRID SCHEME OF LEAST SQUARES METHODS AND FUZZY LOGIC

This section presents an improved method for estimation and prediction of dynamic object using the hybrid scheme of least squares and fuzzy logic. It is important to state that the work in this paper differs to work done by [Lee and Wang 1994] in terms of how and when λ coefficient changes. λ coefficient is greater than 0 and smaller than 1 and it is used to place more or less weight on recent data. The prediction of the ball position in (t+1) requires that $x+1$ and $y+1$ be estimated. A sequential algorithm that is able to closely track time-varying parameters is called a real-time algorithm. In this section such a real time algorithm is presented. An exponential weighting scheme is used to place heavier emphasis on more recent data. As a result, the parameter tracking capability is greatly increased. In [Lee and Wang 1994] the prediction error $error = \sqrt{(x_i - x_p)^2 + (y_i - y_p)^2}$ is defined and then according to its magnitude the λ parameter either increases or decreases. However, by manipulating the errors (x error and y error) separately a more reliable result is produced. This means that two controllers are required for adjusting λ , one for the x and another for the y. In the following, the algorithm is presented for dynamic object prediction.

In section three it was shown that equation (3) can be used to simplify a set of linear equations. Let matrices y , X and θ denote the system's observations, the time related with system's observations and the parameters to be estimated. The three matrices can be defined as follows.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_m \end{bmatrix} \quad X = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_m & 1 \end{bmatrix} \quad \theta = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \quad (14)$$

The error vector for each solution θ is the same as in equation (5). Then consider the error function

$$J_m = \sum_{i=1}^m \lambda^{m-i} e_i^2, \quad 0 < \lambda < 1 \quad (15)$$

in which later squared errors are given more weight than earlier ones. According to [Hsia 1977] matrix θ can be estimated as follows.

$$\hat{\theta}_{m+1} = \hat{\theta}_m + \gamma_{m+1} P_m \chi_{m+1} [y_{m+1} - \chi_{m+1}^T \hat{\theta}_m] \quad (16)$$

The parameters γ_{m+1} , P_m , χ_{m+1} are defined as follows.

$$\gamma_{m+1} = \frac{1}{1 + \chi_{m+1}^T P_m \chi_{m+1}} \quad (19)$$

$$P_m = \frac{1}{\lambda} (X_m^T X_m)^{-1} \quad (20)$$

$$\chi_{m+1} = \begin{bmatrix} x_m \\ 1 \end{bmatrix} \quad (21)$$

It can be observed from equation (15) that the smaller the λ , the heavier are the weights assigned to the more recent data. This implies that the algorithm is more capable of tracking the parameter variations. For the estimation and prediction of dynamic objects in an x,y co-ordinate frame the above algorithm is used to estimate both x and y. The estimate $\hat{\theta}_{m+1}$ from (16) produces the unknown parameters ϑ_1 and ϑ_2 so the linear equation $y = \vartheta_1 x + \vartheta_2$ can be solved.

Figure 2 shows the overview of the proposed approach. Observations are taken in x and y co-ordinates representing the object position. The real time algorithm takes as inputs the value of x or y and λ . Then the algorithm's output is \hat{x} , $\hat{x}+1$, λ_m which denotes the estimate of x, the predicted value of x and the value of the current λ . Then two blocks are used, one to calculate the new λ value and another to calculate the predicted error. Fuzzy logic was employed as a control tool for the adjustment of the λ values. Figure 3 shows the 3-D surface of the fuzzy logic controller. The inputs to the fuzzy inference engine are prediction error and current value of λ . The output is the new value of λ for the next estimation and prediction step.

The 3-D surface demonstrates how new λ is derived according to the input parameters. If the estimation error is big and the current λ is high then the new value of λ is small. That means that if the previous estimate was inaccurate and only way to get more accurate the next prediction step is to reduce λ . Finally the block (predicted_error_xy) in figure 2 finds the Euclidean distance between the current and predicted position of the object. This method has the advantage of adjusting different values of λ for x and y according to their estimation errors.

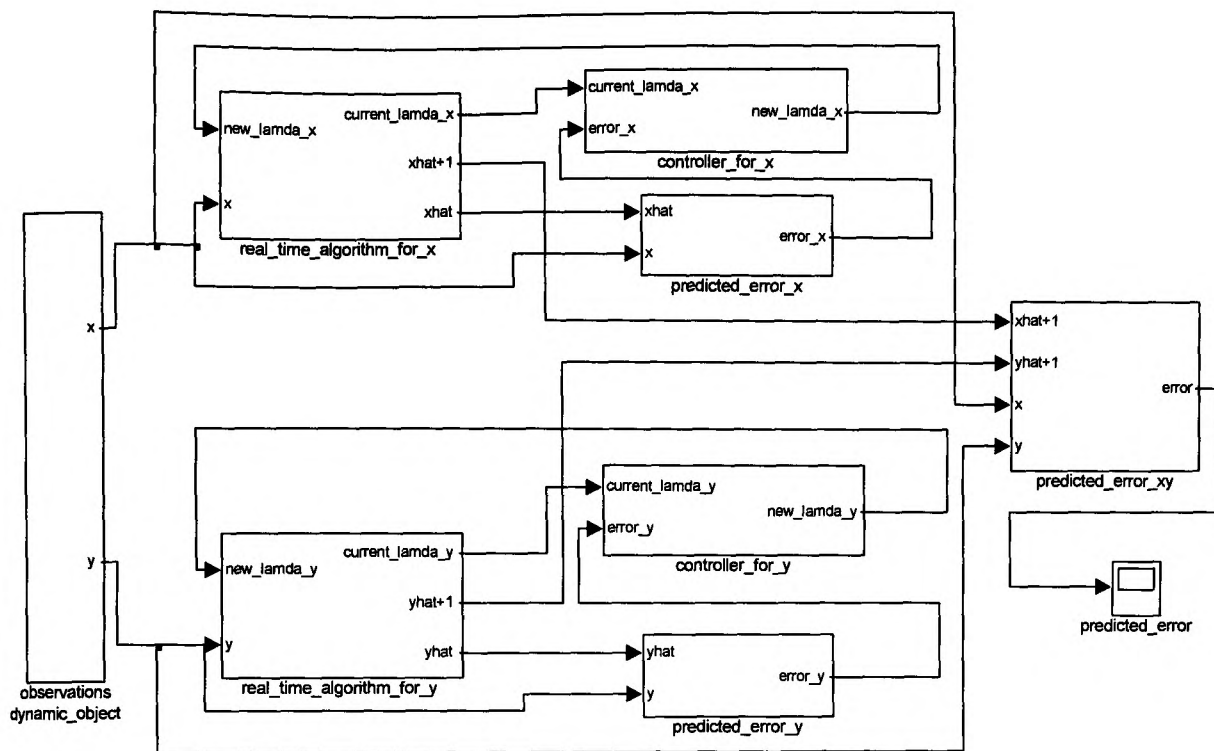
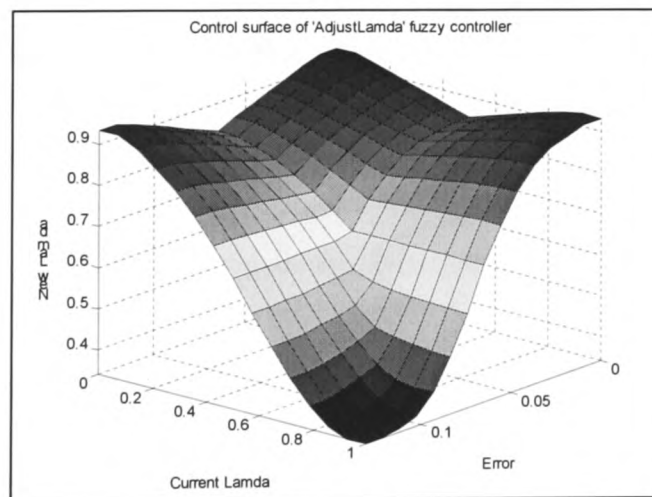


Figure 2. Overview of the proposed identification approach

Figure 3. Control surface of "Adjust λ " fuzzy controller

6. EXPERIMENTAL RESULTS

To test and analyse the proposed approach for estimation and prediction of dynamic objects, simulation and experimental studies were performed. Simulation was carried out using the MATLAB development software. The task set was for the mobile robot to move with a random trajectory (the same trajectory could also represent the ball's movement). It was assumed that this represented a course of the mobile robot when avoiding obstacles. The velocity varies due to accelerations and decelerations of the mobile robot.

Figure 4a shows the velocity during the robot's motion, the desired trajectory (to be estimated and predicted, figure 4b), and the x and y co-ordinates of the trajectory, figure 4c and 4d. It can be seen that the x and y co-ordinates of the trajectory are different in terms of local minima. The y co-ordinate of the trajectory is more difficult to predict compared to the x co-ordinate due to its non-monotonic characteristics and the appearance of larger number of local minima. For this reason the experiments were carried out on y co-ordinate only for this stage of simulation.

Figure 5 shows results with the λ coefficient having constant value of 0.98. The absolute values of estimated errors are illustrated in figure 5a, the square errors are shown in figure 5b (equation 15 with a window of 6 observations), the minimisation of the error function is shown in figure 5c and the value of the correction term is illustrated in figure 5d. It is clear that the estimated error is large when the error function (J_m) is high. It can be concluded that the mean rate of the correction term remains the same throughout the simulation period.

Figure 6 shows results of the real-time algorithm using fuzzy logic to control the λ coefficient. It can be seen that when λ is small then the correction term is small and vice versa (Figure 6a and 6b). Observing figure 5a and figure 6a the λ value increases when the error is small and λ value decreases when the error is large. These changes in λ coefficient and corresponding changes in the correction term reduces the estimation error producing more accurate position prediction.

Figure 7a shows both estimated errors with and without the fuzzy controller. It can be observed that the error between those two estimations is very small. Figure 7b shows the absolute value of difference between the estimation errors.

Finally figure 8a shows the real trajectory derived by the mobile robot and that predicted using the hybrid least square and fuzzy logic method. Again they are very close and figures 8b, 8c and 8d are included to demonstrate both trajectories in order to show the accuracy of the prediction method. It can be observed that the predicted path is very accurate at the local minima point where the mobile robot changes trajectory.

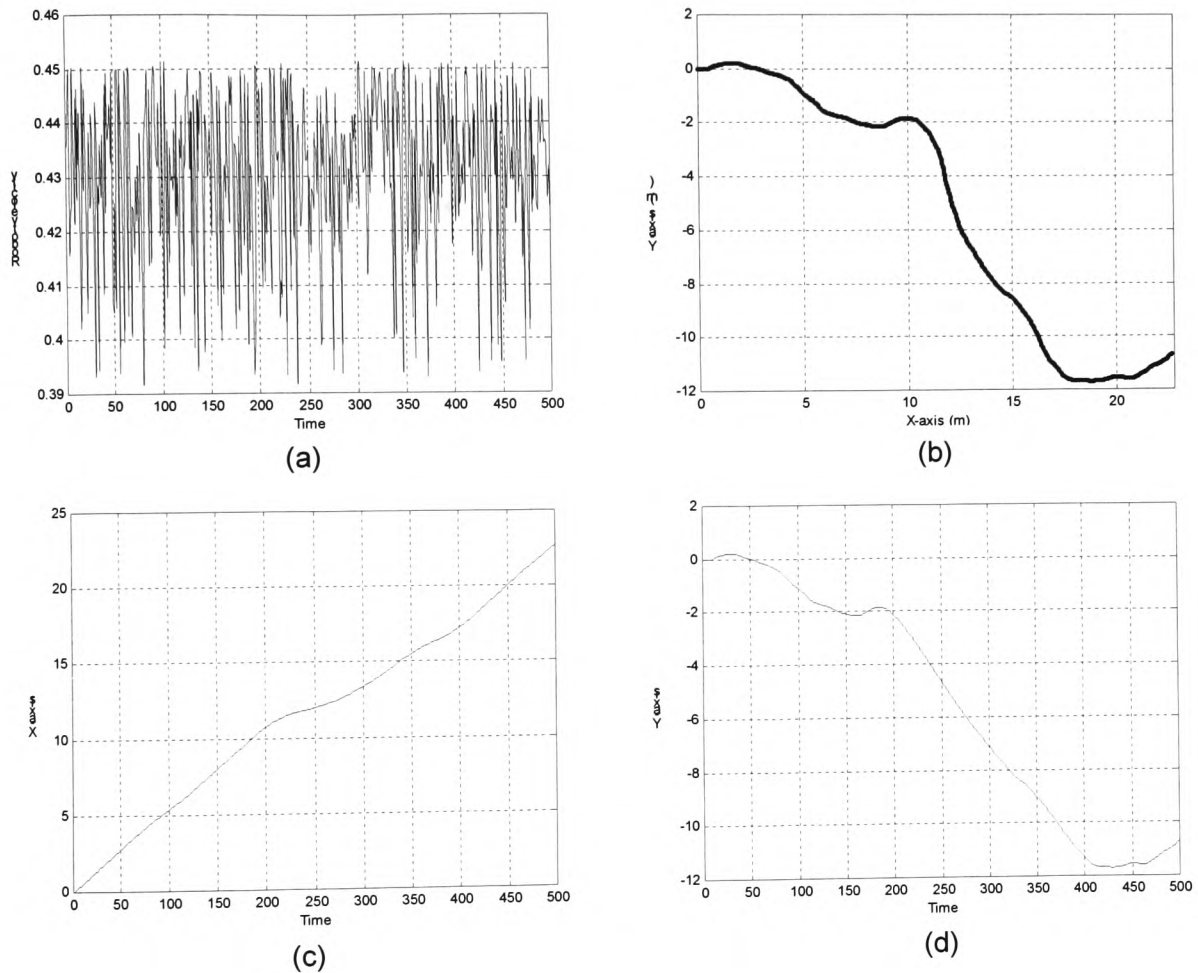


Figure 4. (a) Velocity of the center point of MIABOT V2 mobile robot (b) Random trajectory of the mobile robot to be estimated (c) Trajectory on x-axis (d) Trajectory on y-axis

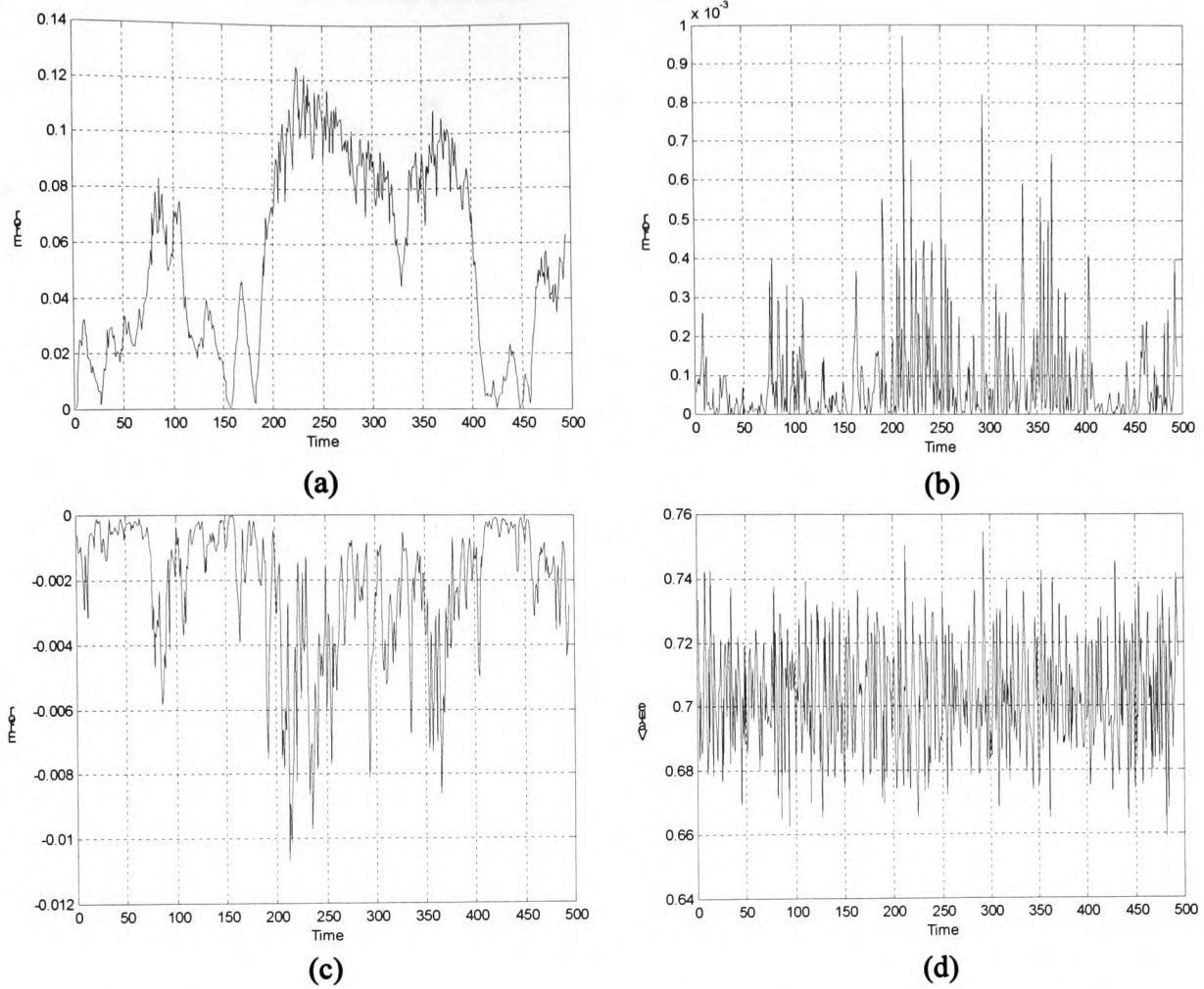


Figure 5. (a) Estimated error with λ constant (b) Square errors (c) Error function minimisation (d) Correction term with λ constant

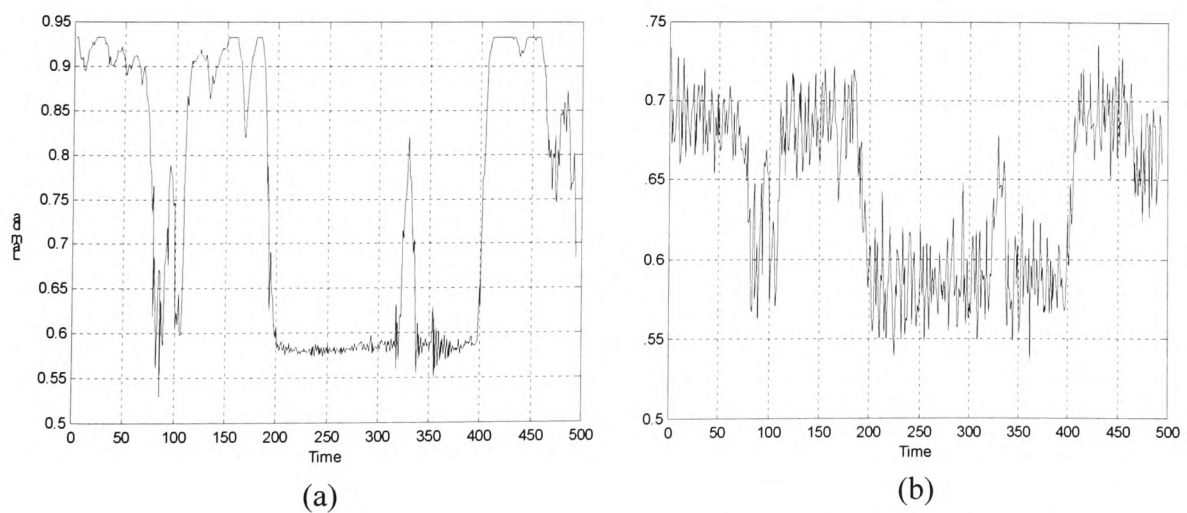


Figure 6. (a) New λ coefficient depending on error and current λ value (b) Correction term (P_m) with variable λ coefficient

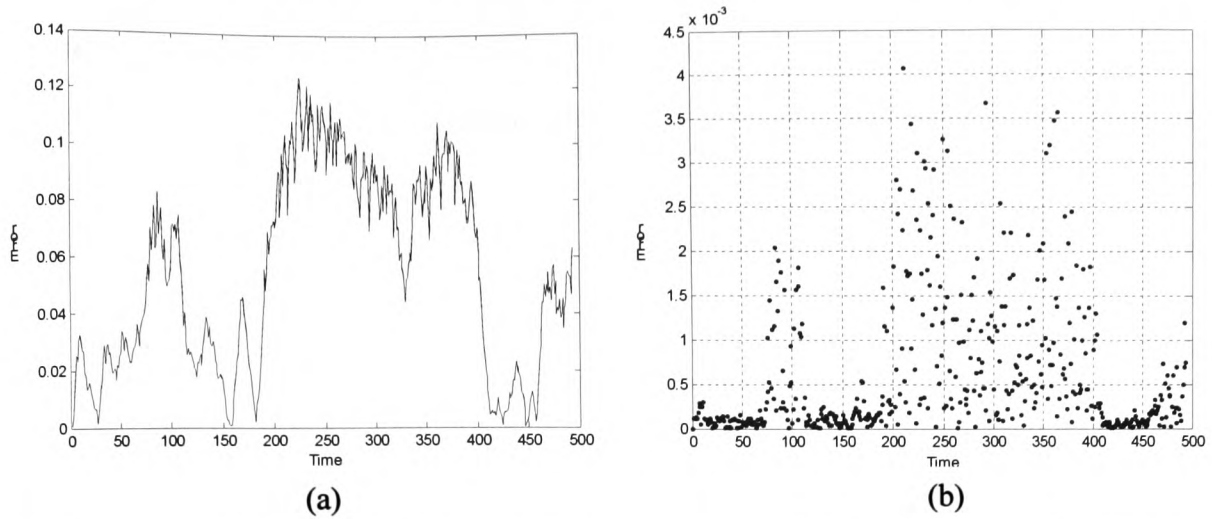


Figure 7. (a) Estimated error with and without the fuzzy controller to adjust λ (b) Difference in estimation error using fuzzy controller to adjust λ

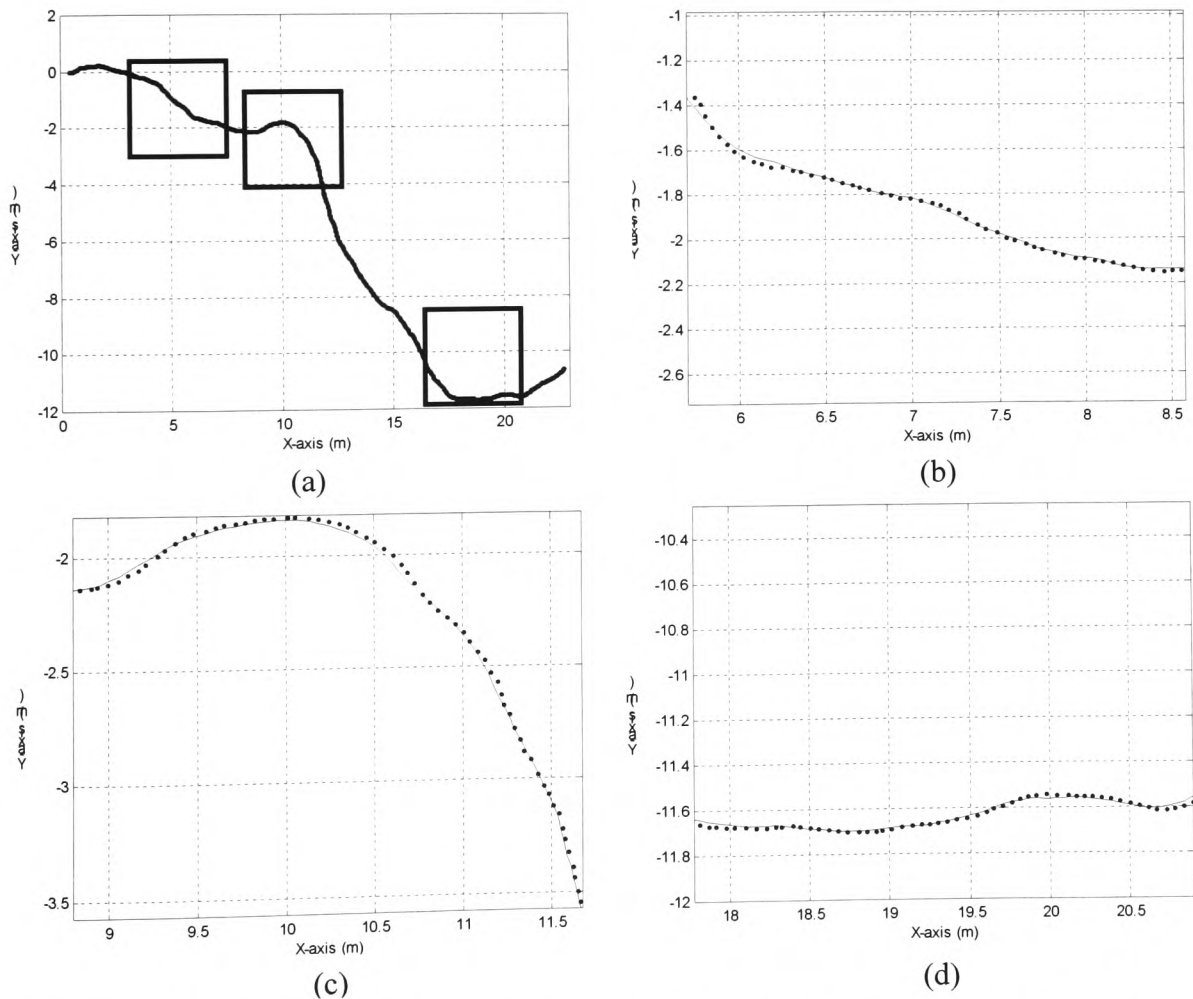


Figure 8. (a) Real trajectory (solid line) against predicted (dotted line) (b) Top left side zoom showing both trajectories (c) Zoom at the center (d) Bottom right side zoom

7. CONCLUSIONS

This paper presents a hybrid method of least squares and fuzzy logic for estimation and prediction of dynamic objects. The method is to be implemented in the robot football system at University of Wales College, Newport for estimation and prediction of ball and mobile robot positions. The paper described in brief the operation of least squares and the function of fuzzy logic. The real-time least squares algorithm based on minimisation of the error function and the adjustment of the λ coefficient was also presented. The approach presented uses fuzzy logic to adjust respective λ coefficients for x and y co-ordinates based on their estimation errors.

The task considered was to estimate and predict the random trajectory, which was generated by the mobile robot moving with non-constant velocity. Experiments concentrated on the observations of the y co-ordinate of the trajectory due to its non-monotonic characteristics and the appearance of large number of local minima. The results clearly indicate that the method is extremely accurate. In fact for this particular application the improvement obtained when using the proposed hybrid method is marginal. This is due to the fact that the sample period used was very small. However, in real application there is often a conflict between sample time and accuracy and the next stage of the work will investigate how the prediction accuracy is affected by increasing in sample period.

REFERENCES

1. **Ljung, L., (1999)**, System Identification: Theory for the User, Prentice-Hall, Inc., United States of America.
2. **Roberts, G. N., (1998)**, "Intelligent Mechatronics", Computing and Control Engineering Journal, Vol. 9, No. 6, pp 257-264.
3. **Zadeh, A. G, Fahim, A. and Elgindy (1997)**, "Neural Network and Fuzzy Logic Applications to Vehicle Systems: Literature Survey", International Journal of Vehicle Design, Vol. 18, No. 2, pp 132-193.

4. **Lee, P. S. and Wang, L. L., (1994)**, "Collision-Avoidance by Fuzzy-Logic Control for Automated Guided Vehicle Navigation", Journal of Robotic Systems, Vol. 11, No. 8, pp 743-760.
5. **Cowan, C. F. N. and Grant, P. M., (1985)**, Adaptive Filters, Prentice-Hall, Inc., United States of America.
6. **Isermann, R., Lachmann, K-H. and Marko, D., (1992)**, Adaptive Control Systems, Prentice-Hall, Inc., United States of America.
7. **Zimmermann, H. J., (1991)**, Fuzzy Set Theory and its Applications, Kluwer academic Publishers, United States of America.
8. **Wang, L. X., (1997)**, Course in Fuzzy Systems and Control, Prentice-Hall, Inc., United States of America.
9. **Hsia, T. C., (1977)**, System Identification: Least-Squares Methods, Lexington Books, United States of America.

Autonomous Robot Behaviours for Co-operative Agents using Fuzzy Logic and Subtractive Clustering

Alexandros Mouzakitis

Mechatronics Research Centre
University of Wales College, Newport
Allt-yr-yn Campus, P.O. Box 180
Newport, NP20 5XR
United Kingdom
Tel: +44 (0) 1633 432487
Fax: +44 (0) 1633 432442
e-mail: alexandros.mouzakitis@newport.ac.uk

Geoff Roberts

Mechatronics Research Centre
University of Wales College, Newport
Allt-yr-yn Campus, P.O. Box 180
Newport, NP20 5XR
United Kingdom
Tel: +44 (0) 1633 432441
Fax: +44 (0) 1633 432442
e-mail: geoff.roberts@newport.ac.uk

Abstract

Intelligent autonomous robots and multi-agent systems, having different skills and capabilities for specific subtasks, have the potential to solve problems more efficiently and effectively. In this paper both fuzzy logic (FL) and subtractive clustering (SC) are used for the design of autonomous robot behaviours. The design procedure is conducted in two stages: first subtractive clustering is applied to extract fuzzy model from experimental data; then adaptive neuro-fuzzy inference system (ANFIS) is applied to improve the fuzzy model performance. This technique produces good result (0.01% root mean square error) and has the advantage of being closer to natural human language, by describing the robot behaviours using a set of linguistic rules.

Keywords: Robot Behaviours, Subtractive Clustering, Neuro-fuzzy Modelling.

1 Introduction

An autonomous robot is defined as a physical device, which performs a predefined task in a dynamic and unknown environment without any kind of external help. Having the ability to sense the state of the environment the robot is able to perform its control actions with the help of what is called

control architecture. In most multi-agent type architectures the control is divided into vertical functional modules or behaviours where each of the behaviour is responsible for a well-defined task.

Classical control theory is based on mathematical models that describe the behaviour of the plant or system under consideration. The main idea of fuzzy control [4] is to build a model of a human control expert who is capable of controlling the plant without thinking in mathematical model terms.

Fuzzy logic modelling techniques can be classified into three categories, namely the linguistic (Mamdani-type), the relational equation, and Takagi, Sugeno and Kang (TSK). In linguistic models, both the antecedent and the consequence are fuzzy sets while in the TSK model the antecedent consists of fuzzy sets but the consequence is made up of linear equations. Fuzzy relational equation models aim at building the fuzzy relation matrices according to the input-output process data. To model a nonlinear system Jang [2] has introduced an adaptive neuro-fuzzy inference system (ANFIS) based on TSK model.

In this paper subtractive clustering is used to derive a fuzzy model from experimental data for the design of autonomous robot behaviours. Then ANFIS is used as a post-processor to improve model capability and efficiency.

2 Subtractive Clustering Method

In order to obtain a set of R rules avoiding the problems inherent to grid partitioning, e.g., rule explosion, subtractive clustering is applied [1]. This technique, is employed since it allows a scatter input-output space partitioning.

Subtractive clustering is an extension of the mountain clustering method [3]. Mountain clustering is relatively simple and effective. However, its computation grows exponentially with the dimension of the problem because the method must evaluate the mountain function over all grid points. Using subtractive clustering data points (not grid points) are considered as the candidates for cluster centers. The computation is simply proportional to the number of data points and independent of the dimension of the problem under consideration.

Consider a collection of n data points $\{x_1, \dots, x_n\}$ in an M -dimensional space. Without the loss of generality, the data points are assumed to have been normalised within a hypercube. Since each data point is a candidate for cluster centers, a density measure at data x_i is defined as:

$$D_i = \sum_{j=1}^n e^{-a\|x_i - x_j\|^2} \quad (1)$$

where $a = \frac{4}{r_a^2}$ and $r_a > 0$.

Hence, a data point will have a high density value if it has many neighboring data points. The radius r_a defines a neighbourhood; data points outside this radius contribute only slightly to the density measure.

After computing the density measure for each point, the one with the higher density is selected as the first cluster center. Let x_{c_1} be the centre of the first group and D_{c_1} its density. Then, the density measure for each data point x_i is revised by the formula:

$$D_i = D_i - D_{c_1} e^{-\beta\|x_i - x_{c_1}\|^2} \quad (2)$$

where $\beta = \frac{4}{r_b^2}$ and $r_b > 0$.

The radius r_b represents the radius of the neighbourhood for which significant density measure reduction will occur. The radius for reduction of density should be to some extent higher than the neighbourhood radius to avoid closely spaced clusters. The value is typically, $r_b = 1.5r_a$. Since the points closer to the cluster center will have their density measure strongly reduced, the probability for those points to be chosen as the next cluster is lower. This procedure is carried out iteratively, until the stopping criteria are reached. The algorithm is:

```

if  $D_k > \varepsilon^{up} D_{c_1}$ 
    Accept  $x_k$  as the next cluster center and continue
else if  $D_k < \varepsilon^{down} D_{c_1}$ 
    Reject  $x_k$  and end the clustering process
else
    Let  $d_{min}$  be the shortest distance between  $x_k$  and
    all previously found cluster centers
    if  $\frac{d_{min}}{r_a} + \frac{D_k}{D_{c_1}} \geq 1$ 
        Accept  $x_k$  as the next cluster center and
        continue
    else
        Reject  $x_k$  and set the density at  $x_k$  to 0
        Select the data point with the next highest
        density as the new  $x_k$  and re-test
    end if
end if

```

Here, ε^{up} specifies a threshold above which the point is selected as a center, and ε^{down} specifies the threshold below which the point is definitely rejected. Typically, $\varepsilon^{up} = 0.5$ and $\varepsilon^{down} = 0.15$. If the density measure fails in the gray region, then checking of data points is required to identify where they provide a good trade-off between having a significant density measure and being sufficiently far from existing clusters.

At the end of clustering procedure, a set of fuzzy rules will have been obtained. Each cluster will represent a rule. However, since the clustering procedure is conducted in a multidimensional space, fuzzy sets must be obtained. As each axis of the multidimensional space refers to a variable, the

centers of the membership functions for that variable are obtained by projecting the center of each cluster in the corresponding axis. As for the widths, they are obtained on the basis of the neighbourhood radius r_a , defined while performing subtractive clustering. Since Gaussian membership functions are used, their standard deviations are computed as:

$$\sigma_{ij} = r_a \frac{\max(x_{kj}) - \min(x_{kj})}{\sqrt{8}}, k = 1, \dots, N \quad (3)$$

3 Adaptive Neuro-Fuzzy Inference System

Jang [2] introduced the ANFIS (Adaptive neuro-fuzzy inference system). Figure 1 provides an example of a simple fuzzy inference system (FIS) represented in an ANFIS network. In the ANFIS architecture, FIS is described in a layered, feed-forward network structure where some of the parameters are represented by adjustable nodes (represented as rectangular entities in the figure) and the others as fixed nodes (represented as spherical entities in the figure). The raw inputs are fed into the layer 1 nodes that represent the membership functions. The parameters in this layer are called premise parameters and they are adjustable. The second layer represents the T-norm operators that combine the possible input membership grades in order to compute the firing strength of the rule. In the basic ANFIS method these parameters are not adjustable. The third layer implements a normalisation function to the firing strengths producing normalised firing strengths. The fourth layer represents the consequent parameters that are adjustable. The fifth layer represents the aggregation of the outputs performed by weighted summation. This is not adjustable.

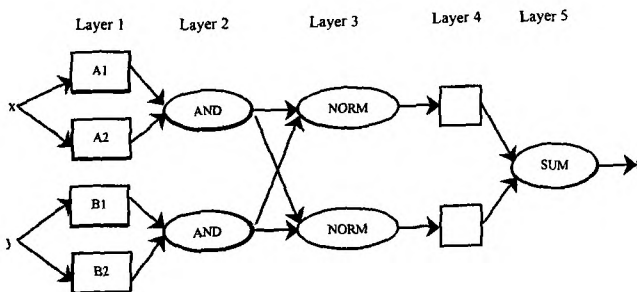


Figure 1: An ANFIS structure for a simple FIS

4 Problem Statement

In section 2 and 3 an algorithmic methodology was described to identify fuzzy control strategies using any n -dimensional input-output space. In this paper the proposed algorithmic methodology is applied to identify an autonomous robot's (MIABOT V2, Figure 2) control strategy to avoid objects (neighbour robots). The robot's control architecture is multi-agent type and consists of multiple controllers (fuzzy behaviours). Each of the behaviours is designed to perform a particular task such as to orientate the mobile robot towards the goal, to avoid neighbour robots (agents) and either to find or reach the goal.

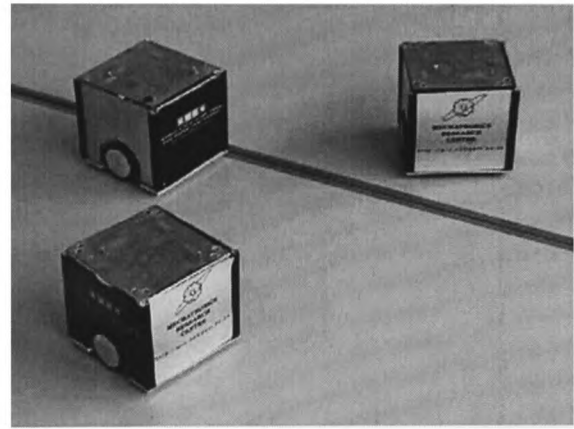


Figure 2: MIABOT V2 mobile robots

5 Simulation Results

In this section experimental results are presented to show the validity of the proposed algorithmic methodology. The strategy of the control action is based on three inputs: speed of the robot u , distance d and angle θ between the robot and the neighbour robots (obstacles). Hence the model has three input variables and two output (left velocity UL , right velocity UR) variables subtractive clustering is used to take input-output training data and generate a Sugeno-type fuzzy inference system that models the data behaviour. Of the original 209 experimental data points, 176 data points are used as training data and 33 data points as checking data. Figure 3 shows the input output data.

Using subtractive clustering the result for the first and second output is shown in figure 4 (top) and figure 5 (top). Checking data are compared with observation data and the root mean square error

(RMSE) is 0.0066 and 0.0610 respectively. To improve the model's accuracy ANFIS is employed. Figure 4 (Bottom) and figure 5 (Bottom) illustrate the new model in which in both cases the RMSE has been reduced to 0.0019 and 0.0133 respectively. Finally figure 6 illustrates at which point the training and checking error settles.

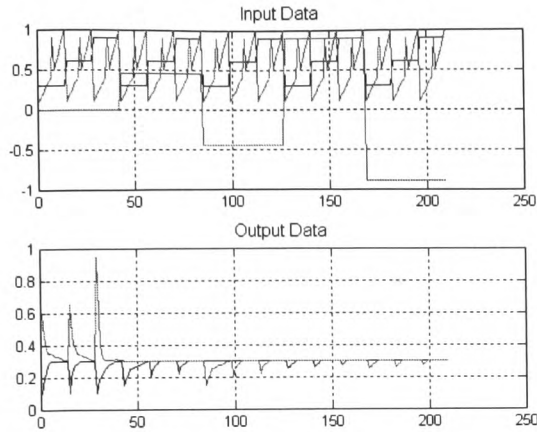


Figure 3: Input-output experimental data points

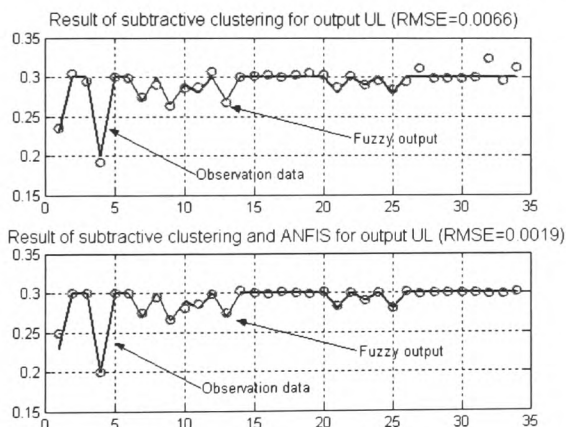


Figure 4: Fuzzy model for output (UL)

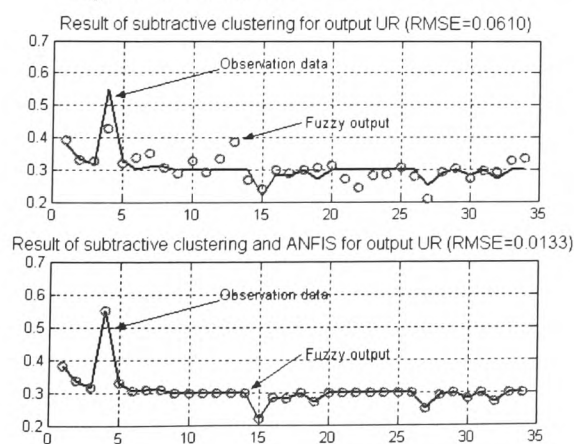


Figure 5: Fuzzy model for output (UR)

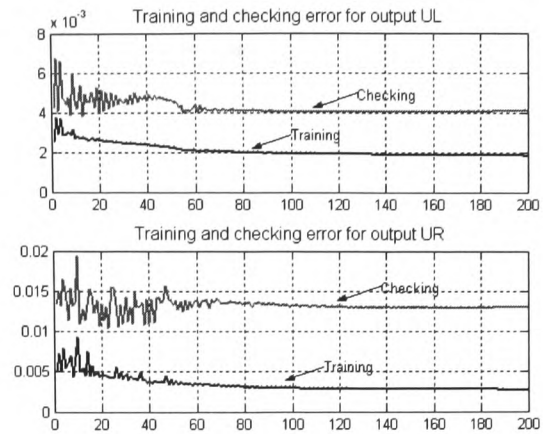


Figure 6: Settle of training and checking error

6 Conclusions

In this paper, a hybrid method of subtractive clustering and neuro-fuzzy modelling using ANFIS was presented. The proposed method can be applied to construct an identification method of fuzzy control strategies based on availability of input/output mapping data. In the first stage subtractive clustering was applied in order to obtain a fuzzy model of the robot behaviour. Then, an adaptive neuro-fuzzy inference system was used to improve the model's accuracy. Simulation results show that the proposed method can produce very accurate results (RMSE 0.0019 and 0.013 respectively).

References

- [1] Chiu, S. L. (1994). Fuzzy Model Identification Based on Cluster Estimation. *Journal of Intelligent and Fuzzy Systems*, 3 (2), pp. 267-278.
- [2] Jang, R. J.-S. (1993). ANFIS: Adaptive-Neuro-Network-Based Fuzzy Inference System. *IEEE Transactions on System, Man Cybernetics*, 23 (3), pp. 665-684.
- [3] Yager, R. R. and Filev, D. P. (1994). Generation of Fuzzy Rules by Mountain Clustering. *Journal of Intelligent and Fuzzy Systems*, 2 (3), pp. 209-219.
- [4] Zadeh, A. G., Fahim, A. and Elgindy, M. (1997). Neural Network and Fuzzy Logic Applications to Vehicle Systems: Literature Survey. *International Journal of Vehicle Design*, 18 (2), pp. 132-193.

MODELLING, IDENTIFICATION AND CONTROL TECHNIQUES FOR INTELLIGENT ROBOT'S ARCHITECTURE AND CO-OPERATIVE AGENTS

Alexandros Mouzakitis and Geoff Roberts

Mechatronics Research Centre
University of Wales College, Newport
Allt-yr-yn Campus, P.O. Box 180, NP20 5XR Newport,
United Kingdom
Tel: ++44 (0) 1633 432487, Fax: ++44 (0) 1633 432442.
email: alexandros.mouzakitis@newport.ac.uk

Abstract

Intelligent autonomous mobile robots and multi-agent systems, having different skills and capabilities for specific subtasks, have the potential to solve problems more efficiently and effectively than single agents. This paper contributes with the proposal of new hybrid control architecture for autonomous robots and co-operative agents. The proposed architecture is a combination of subsumption architecture, competitive tasks architecture and production systems architecture. The state identification mechanism, CAROS algorithm, decision making mechanism and action co-ordinator mechanism define the structure of the proposed architecture. As a tool for this research work the autonomous mobile robots MIABOT V2 are used both as real examples and simulation models. Simulation results are presented to show the validity of the proposed architecture and also to show how the mobile robot performs according to various dynamic environment changes.

1 Introduction

An autonomous robot is defined as a physical device, which performs a predefined task in a dynamic and unknown environment without any kind of external help. Having the ability to sense the state of the environment the robot is able to perform its control actions with the help of what is called control architecture. Control architecture for autonomous robots have been previously proposed or developed by various researchers (Muller, 1997). In general mobile robot control architectures have been classified in two major categories: hierarchical and behaviour-based. Traditional artificial intelligence approach to robot control is based on functional decomposition into hierarchical levels. In this control architecture the problem is divided into vertical functional modules. Each vertical functional module is responsible for a well-defined task, where each level must pass its output onto the next level in order for the robot to be able to complete a predefined task. The behaviour-based approach is an alternative to the traditional approach. Instead of having a number of vertical functional levels, the behaviour-based

method tackles the control problem by thinking of it as a number of horizontal arranged layers (Brooks, 1986). In (Xu and Van Brussel, 1997) advantages of the behaviour-based architecture and weaknesses of the traditional approach are discussed in more detail.

This paper concentrates on modelling new hybrid control architecture for autonomous robots and co-operative agents. The proposed control architecture is hybrid of subsumption architecture, competitive tasks architecture and production systems architecture. The control architecture consists of five independent modules, each of which is responsible for a particular task. The state identification mechanism produces the states of the system as true or false according to the input sensory data, then the CAROS algorithm having the required states of the system selects the appropriate controller. The function modules (or controllers) have been designed using Fuzzy Logic (FL). The control output of the system adjusts left and right velocities of the mobile robot MIABOT V2, the kinematics model of which is used for the simulation.

This paper is organised into six sections. The related work on control of co-operative robots and design in control architecture is described in section 2. Section 3 defines the problem statement. The modelling of the new hybrid control architecture is described and discussed in section 4. Section 5 presents experimental results of the proposed control architecture. Finally section 6 contains the conclusions of the work presented.

2 Related Work

Within the robotics community, research into multi-robot systems or co-operative robotics has increased dramatically due to wide range of applications such as planetary exploration, map making and trash collecting. Most of the research has focused on either group architecture, resource conflicts, origins of co-operation, learning, and control or geometric problems (Cao *et al*, 1997). (Fukuda and Nakagawa, 1987), introduced a new project in the field of co-operative robotics called CEBOT. Their work is based on co-ordination among mobile multi-robot systems with emphasis on communication mechanisms, which can be used to support co-ordinated behaviour. (Asama *et al*, 1989), developed ACTRESS a multi-robot system designed for heterogeneous agents based on communication issues. The robots act independently, but if the need arises, they negotiate with other robots to form a co-operative group to solve the problem. (Arkin, 1992), presents research concerned with sensing, communication, and social organisation for multiple mobile robots that are supposed to forage and retrieve objects in the hostile environment. (Noreils, 1993), proposed a three-layered control architecture for semi-heterogeneous robots that included a planner level, a control level, and a functional level. The first (planner level) was the high-level decision-maker. However many of the recent co-operative robotics systems, in contrast to the earliest works, are based on the behaviour-based approach (Brooks, 1986). For instance (Mataric, 1992) developed behaviours for multi-agent systems using subsumption architecture. In terms of co-operation and communication, most of the above work has fallen along the two ends of the spectrum. It either uses extensive explicit communication and co-operation, or almost none at all. In systems that are co-operative by design, two or more robots are aware of each other's existence, and can sense and recognise each other directly or through communication. This type of research explores explicit co-operation, usually through the use of direct communication (Parker, 1993). The other category includes work on implicit co-operation, in which the robots usually do not recognise each other but indirectly co-operate by having identical or at least compatible tasks. Such work includes (Kube and Zhang, 1992). The design of such control architectures is complex due to the system's complexity. (Ferber, 1999), has categorised the main types of

architectures based on multi-agent platforms. The main agent architectures are categorised according to type of architecture, approach, type of component, subordination structure, coupling structure and constitution. In (Ferber, 1999), there are nine possible architectures from which the control designer can choose. The research work in this paper concerns the design and modelling of control architecture for autonomous robots and co-operative agent using implicit communication based on production rules architecture (Fuzzy Logic), competitive tasks architecture and subsumption architecture.

3 Problem statement

In this work it is assumed that there are number n mobile robots. The robots, without any prior knowledge of the environment must reach two independent objects, from an initial position, in the shortest possible travel time, without colliding with each other.

The robots are moving in a 2-dimensional completely uncertain and unknown environment with non-constant velocity. The unknown environment is assumed to be filled by moving and/or stationary neighbour robots. Ultrasonic sensors provide information about the neighbouring robots and their relative positions. It is assumed that the ultrasonic sensors obtain the information in terms of polar co-ordinates (d, θ) with error-free measure. This information will be the distance to the nearest robot (d_{robot}) and the direction to the nearest robot (θ_{robot}) in robot co-ordinates $(x_{r,n}, y_{r,n})$. An additional input provides information about the direction and distance to the goal. The robot also receives a signal from a beacon placed on top of the goals and it is assumed that this signal is always receivable. The output signals from the control architecture are the commands for the speed control of the two wheels of the mobile robot (u_L and u_R).

The main objective is to design multi-agent behaviour-based control architecture for the mobile robot in order to be able to adjust its motion according to the dynamic input sensory data.

4 Modelling the control architecture

This section presents the modelling of the new hybrid control architecture, which draws its design from competitive tasks architecture, production rules architecture and subsumption architecture. Figure 1 shows the overview of the control architecture.

As can be seen from figure 1 the control architecture consists of the following parts: Sensory data, state identification mechanism, CAROS algorithm, action co-ordinator mechanism, decision making mechanism the environment and the model of the MIABOT V2 mobile robot. The main characteristics/advantages of the proposed architecture can be summarised in the following:

1. Coupling structure is variable. This structure introduces more flexibility in the choice of behaviour selection.
2. Subordination structure is hierarchical. This structure represents the most of the new intelligent control multi-agent type architectures.
3. Type of component is based on rules, task and primitive actions. This type of component integrates the advantages of the type of component of the subsumption, competitive tasks and production rules architectures.

4. Constitution is predefined. Almost all the multi-agent architectures follow the same constitution.
5. Low computational complexity. Each of the behaviour only consists of maximum 15 rules and minimum of 3.
6. Easy to upgrade. Extra behaviour can be added into the control system with only minor modifications in state identification mechanism and CAROS algorithm.

In the following subsections each individual part of the proposed control architecture is described with emphasis on modelling and identification issues. The control architecture was developed and simulated under the MATLAB SIMULINK software environment.

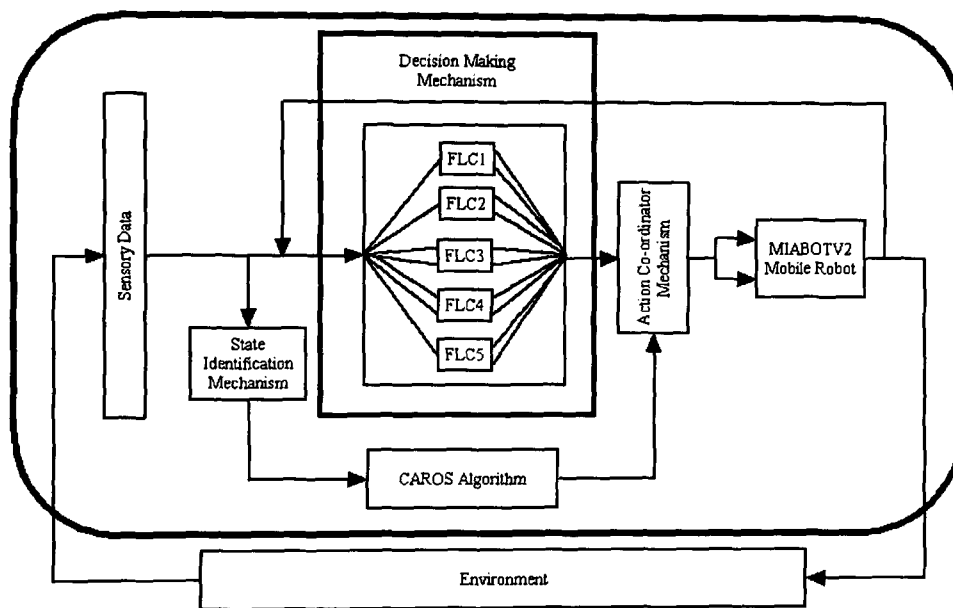


Figure 1 Overview of the control architecture

4.1 Sensory data

This mechanism performs data fusion, which produces the required inputs for the control architecture. The main sensory data are the distance and the angle for the neighbour robots and the goal. As stated earlier this information can be derived using ultrasonic sensors and beacons.

4.2 State identification mechanism (SIM)

This mechanism is responsible for identification of system's states. The number of required states is not unique but depends on a particular application and designer choice. The SIM has an input of sensory data from the robot's environment and outputs of the required states as true or false represented as 1 and 0 respectively. Figure 2 shows the MATLAB pseudo-code showing the operation of the SIM mechanism.

The SIM can be programmed with predefined sampling time. This allows the control architecture to know its current state and decide the control output according to this predefined sampling time. For the threshold value, a precise designer's decision is important in order to produce a result that at least is correct within an order of magnitude. For instance, each state must be assigned as true or false only if it satisfies the threshold value, which has been set by the designer. Thus the threshold value should be as close as possible to the system specific optimum value.

```

sensor_data_1=input(1,1);
sensor_data_2=input(2,1);
sensor_data_3=input(3,1);
...
sensor_data_n=input(n,1);

output=[A,B,C,..n];

if sensor_data_1<= threshold (detect the goal)
    state A is true;
else
    state A is false;
end
...
if sensor_data_n<= threshold (reach goal)
    state N is true;
else
    state N is false;
end

```

Figure 2 MATLAB pseudo-code showing the State Identification Mechanism

4.3 CAROS algorithm

This is the main algorithm that decides which behaviour (or controller) is to be activated according to the current state of the system. For different application, modification of the CAROS algorithm is required. The control strategy for this research work, including the control flowchart of the CAROS algorithm can be found in (Mouzakitis and Roberts, 2000). The CAROS algorithm navigates the mobile robots towards the goal avoiding collisions with neighbour robots either stationary or moving. Pseudo-code of the CAROS algorithm is illustrated in Figure 3.

4.4 Action co-ordinator mechanism (ACM)

The ACM is responsible for releasing the control output of the functional module, which the CAROS algorithm has selected. The inputs into this module are the control outputs of the functional modules (controllers) and also the output from the CAROS algorithm. Each functional module produces two outputs, each of which is responsible for the left and right velocity respectively. The output of the CAROS algorithm is a crisp number from 1 to n . For instance, output value 2 would represent the label for the functional module numbered as 2, then the control output of this particular module will be used as the command for the differential drive of the mobile robot. This kind of action co-ordination draws its advantages from the competitive tasks architecture in which once a functional module is selected the module is activated and the previous one deactivated. However the deactivated module is not totally switched off, it continues to receive sensory data in parallel with SIM. Thus, the module it exercises a monitoring activity,

which consists in receiving data from sensors and calculating the values of the selection parameters, which depend on it. The local model of ACM is given in Figure 4.

```

state_A=input(1,1);
state_B=input(2,1);
state_C=input(3,1);
...
state_n=input(n,1);

output=[1,2,3,4,...,n];

if state A true (neighbour robots found)
    if state B true (moving robots found)
        control action=3; (FLC: AVOID MOVING NEIGHBOUR)
    else
        if state C true (stationary robots found)
            control action=4; (FLC: AVOID STATIC NEIGHBOUR)
        end
    end
else
    if state D true (robot detects the goal)
        if state E true (robot orientates the goal)
            control action=5; (FLC: GET GOAL)
            if state F true (robots reached the goal)
                control action=6; (STOP the robot)
            end
        else
            control action=2; (FLC: CHANGE ORIENTATION)
        end
    else
        control action=1; (FLC: FIND GOAL)
    end
end
end

```

Figure 3 MATLAB pseudo-code showing the CAROS algorithm

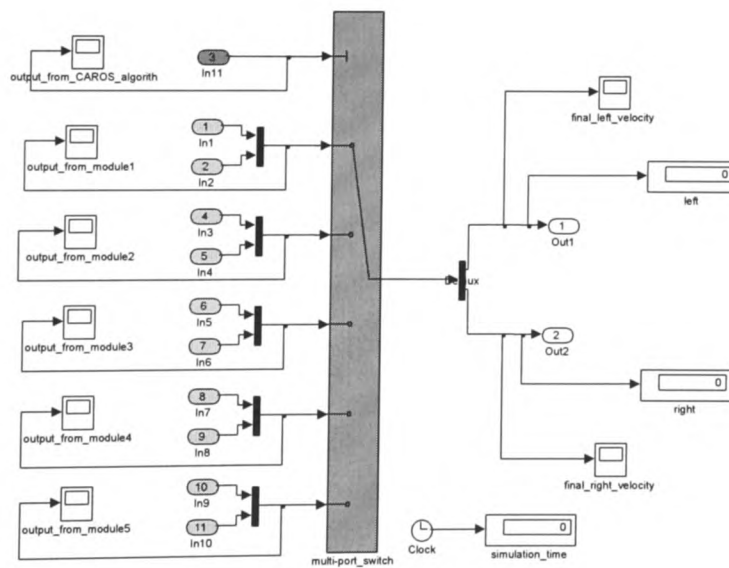


Figure 4 Local model Action Co-ordinator Mechanism

4.5 Decision-making mechanism (DMM)

The decision-making mechanism is the largest part of the proposed control architecture. Five independent control units are employed for this work. Each of the independent functional modules is responsible for a particular task. The design philosophy behind the decision-making mechanism is based on behaviour-based approach. The classical artificial intelligent approach to robot control is to divide the problem in a task-based style. The major problem with this approach was that all subsystems must work well in order for the robot to function at all. This is very easy to occur as perception and world modelling encompass problems which are very difficult to be solved. The behaviour-based approach came to solve this problem. With the new intelligent control approach several behaviours are used to control the robot. For this work, each of the behaviour (functional module) is designed to perform a particular task such as to orientate the goal, to avoid neighbour robot and either to find or reach the goal. The major difference between this work and the subsumption architecture is that the coupling structure is not fixed. The variable coupling structure of the proposed architecture allows more flexibility and modularity into the control system. The design of the functional modules is not unique. The most popular behaviour design is the one by (Brooks, 1986), although other researchers have developed behaviour on more or less the same principle (Arkin, 1989), (Mataric, 1992). For this work the principle of the production systems architecture is proposed for the design of the major functional modules. Fuzzy logic is one form of production systems architecture. This form of control architecture consists of the perception, the database with the inference engine and the rule base, and finally the execution. In terms of fuzzy logic the functional module consists of the fuzzification mechanism, the inference engine and the defuzzification mechanism which produces the crisp output as the control command. The use of fuzzy logic was chosen due to lack of mathematical model of the mobile robots (approximate kinematics model only) and also as the tool of translating human knowledge into mathematical terms. However, fuzzy logic has other advantages as well and has proved in the past that is a superior method for navigation of mobile platforms.

4.6 The mobile robot

As a tool for this research work the autonomous mobile robots MIABOT V2 are used both as real examples and simulation models. Figure 5 shows the appearance of the fully autonomous mobile robots. The driving system is a power wheeled steering system, which drives a left and a right wheel independently. The steering control of the robot is a simple procedure of sending the speed commands u_l and u_r as input to the left and right wheel respectively. The drive train consists of two motors (DC 4.5-12V), with worm gear drives to each wheel enabling accurate stop positions and achievable speeds up to 1m/s. The wheels are 32mm in diameter and equipped with an O-ring to reduce pitch slip. Each motor shaft is monitored via a combination of phototransistors and infrared LED's (shaft encoders), which provide feedback for the ATMEL microprocessor.

To simulate an approximate models in terms of kinematics equations the first order model of the mobile robot MIABOT V2 is required. Let point C denote the centre of the mobile robot. Let \dot{x}_C be the time derivative of the global x position of point C and \dot{y}_C the time derivative of the global y position of the point C . Let $\dot{\theta}$ be the time derivative of the global orientation angle. Also, let $R=2b$, where b is defined as the half distance between the two wheels. Let u_l be the forward velocity of the left wheel

and u_r the forward velocity of the right wheel. Then the first order model equations of the mobile robot are defined as follows:

$$\dot{x}_c = \frac{l}{2} * \cos \vartheta * u_l + \frac{l}{2} * \cos \vartheta * u_r \quad (1)$$

$$\dot{y}_c = \frac{l}{2} * \sin \vartheta * u_l + \frac{l}{2} * \sin \vartheta * u_r \quad (2)$$

$$\dot{\vartheta} = -\frac{l}{R} * u_l + \frac{l}{R} * u_r \quad (3)$$

5 Experimental results

In this section experimental results are presented to show the validity of the proposed architecture. Experiments were carried out for 600 seconds, (which is the travel time for the mobile robot to reach the goal). In order to clearly illustrate the operation of the control architecture, experimental results presented in this paper use a window period of 1 second only, with noise measurements every 100ms (however, the control architecture is capable of functioning with higher or lower sampling intervals). In this experiment the control architecture allows the mobile robot to reach a goal point, from a starting one whilst avoiding neighbour robots. Control inputs are the orientation error (the angle between the current direction of the robot and the direction to goal), distance to the goal and also the angle and distance to the nearest moving or stationary neighbour mobile robot. Control outputs are left and right velocity for the mobile robot (differential drive).

As avoiding obstacles is more important than taking the shortest path to the goal, the obstacle avoidance behaviours are assigned a higher priority than *orientate_goal* and *get_goal* behaviour (all the behaviours are illustrated in Figure 1 within the decision making mechanism from FLC1 to FLC2). However *avoid_moving_neighbours* behaviour is assigned a highest priority than *avoid_stationary_neighbours* behaviour by using a threshold range of their sensory data. A simple example to illustrate the manner in which robot's behaviours are activated according to the current sensory data is shown in figure 5 and 6.

Using random sensory data to represent the inputs from the robot's working environment the states of the system were simulated. Figure 5 shows the states of the control system as true or false represented as 1 and 0 respectively. State A will be true if the mobile robot has found any neighbour within a predefined range, either as moving or stationary. If moving neighbours have been found then state B is true, otherwise if stationary neighbours have been found state C will be true. States D and E inform the control system if the mobile robot detects or orientates towards the goal. Finally, if the mobile robot has reached the goal, state F is true. Figure 6 shows the output of the CAROS algorithm having as inputs the states of the system. It is shown that CAROS output activates only one controller at a time. This gives the advantage of the coupling structure being variable. Thus more flexibility in the choice of behaviour selection is introduced. Each of the behaviours will accelerate or decelerate the mobile robot into the required direction taking either sharp or smooth turns in a left or right direction. This method shows its robustness by the fact that the states of the system have been bounded with a predefined threshold, and also from the fact that for each action an independent controller is allocated to perform the required task.

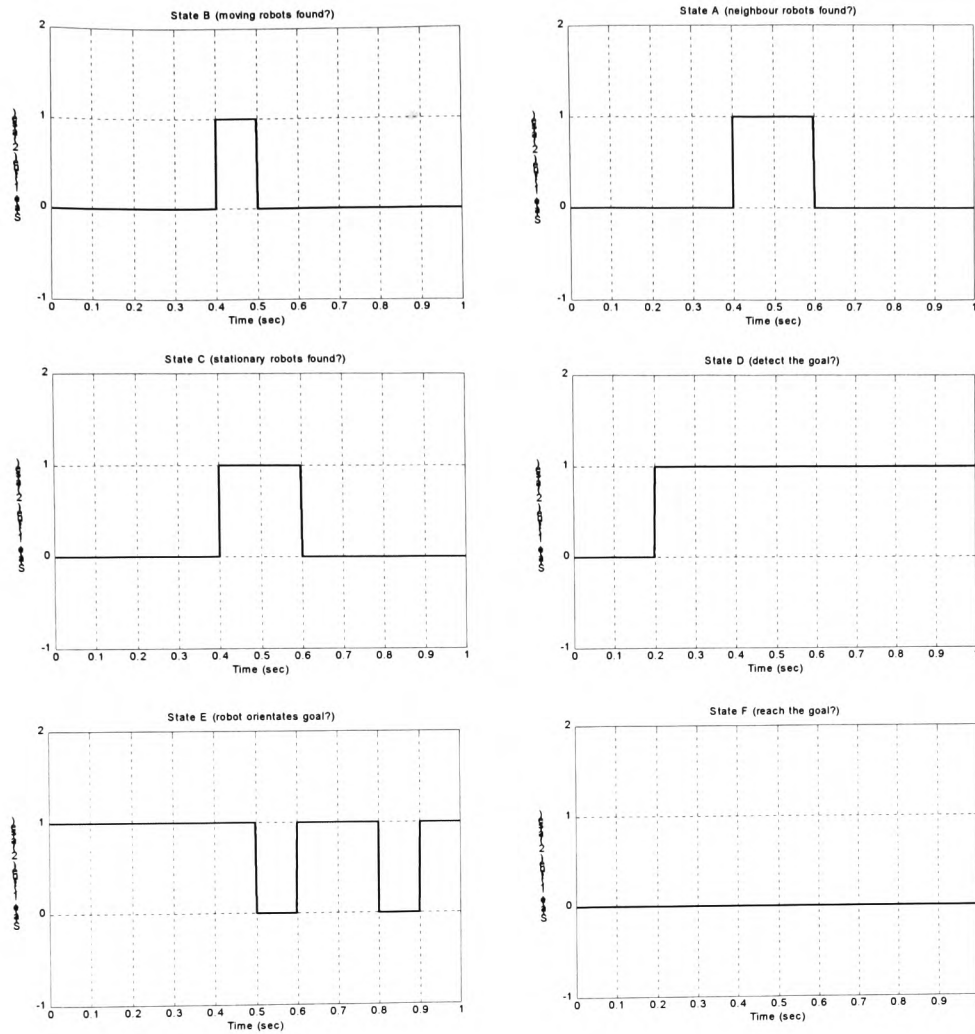


Figure 5 Output from the State Identification Mechanism

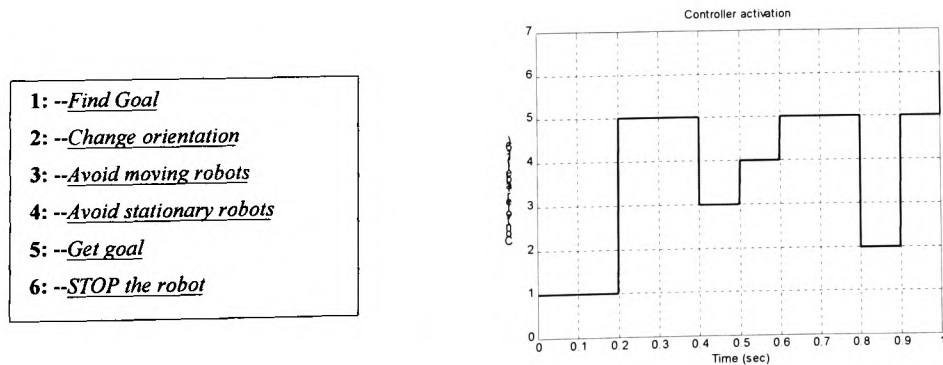


Figure 6 Output from the CAROS algorithm

- 1: --Find Goal
- 2: --Change orientation
- 3: --Avoid moving robots
- 4: --Avoid stationary robots
- 5: --Get goal
- 6: --STOP the robot

6 Conclusions and future work

This work has presented a new hybrid control architecture for autonomous mobile robots and co-operative agents. It is based on a combination of subsumption architecture, competitive tasks architecture and production systems architecture. It was shown a successful attempt to produce useful independent functional modules (behaviours), based the philosophy of the behaviour-based approach and production rules. The control architecture can be extended into higher levels of intelligence and become more distributed with the introduction of extra behaviours and more control algorithms (like CAROS) responsible for a certain number of functional modules. Although this introduces more flexibility it also creates several problems such as appearance of control conflicts. The structure and function of the new hybrid control architecture may be used for construction of complex control systems for mobile robots. Further work is under development to increase the architecture's capability for identification and prediction of moving neighbours, including methods for identification of direction-to-goal in an unknown environment taking into account nonholonomic constraints.

References

- Arkin, R. C. (1989). Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*, **8** (4), pp. 92-112.
- Arkin, R. C. (1992). Cooperation Without Communication: Multiagent Schema-Based Robot Navigation. *Robotic Systems*, **9** (3), pp. 351-364.
- Asama, H., Matsumoto, Y. & Ishida, Y. (1989). *Design of an Autonomous and Distributed Robot System: ACTRESS*. IEEE/RSJ. pp. 283-290,
- Brooks, R. A. (1986). A Robust Layered control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, **RA-2** (1), pp. 14-23.
- Cao, Y. U., Fukunaga, A. S. & Kahng, A. B. (1997). Cooperative Mobile Robots: Antecedents and Directions. *Autonomous Robots*, **4** (1), pp. 7-27.
- Ferber, J. (1999). *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. New York, USA: Addison Wesley Longman Inc. 0-201-36048-9.
- Fukuda, T. & Nakagawa, S. (1987). *A Dynamically Reconfigurable Robotic System (Concept of a system and Optimal Configurations)*. Int. Conf. on Industrial Electronics, Control, and Instrumentation. pp. 588-595,
- Kube, C. R. & Zhang, H. (1992). *Collective Robotics Intelligence*. 2nd International Conference on Simulation of Adaptive Control. pp. 460-468,
- Mataric, M. J. (1992). Integration of Representation Into Goal-Driven Behavior-Based Robos. *IEEE Transactions on Robotics and Automation*, **8** (3), pp. 304-312.
- Mouzakitis, A. & Roberts, G. N. (2000). *A New Decentralised Control for Co-operative Agents Using Fuzzy Logic*. The 14th International Conference on Systems Engineering, ICSE 2000 . pp. 436-501, Coventry, UK.
- Muller, J. P. (1997). Control Architectures for Autonomous and Interacting Agents: A Survey. *Lecture Notes in Artificial Intelligence*, **1209**, pp. 1-26.
- Noreils, F. R. (1993). Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment. *The International Journal of Robotics Research*, **12** (1), pp. 79-98.
- Parker, L. E. (1993). *Designing control Laws for Cooperative Agent Teams*. Proc. IEEE Int. Conf. on Robotics and Automation. pp. 582-587, Atlanta, GA.
- Xu, H. & Van Brussel, H. (1997). A Behaviour-Based Blackboard Architecture for Reactive and Efficient Task Execution of an Autonomous Robot. *Robotics and Autonomous Systems*, **22** (2), pp. 115-132.

A NEW DECENTRALISED CONTROL FOR CO-OPERATIVE AGENTS USING FUZZY LOGIC

A. Mouzakitis and G. N. Roberts

*Mechatronics Research Centre
University of Wales College, Newport*

Allt-yr-yn Campus, PO Box 180, Newport, NP20 5XR, UK

Tel: ++44 (0) 1633 432487. Fax: ++44 (0) 1633 432442. E-mail: alexandros.mouzakitis@newport.ac.uk

Keywords: Autonomous Mobile Robots, Fuzzy Logic, Co-operative Agents, Decentralised control

Abstract

Intelligent autonomous mobile robots and multi-agent (robot) systems, having different skills and capabilities for specific subtasks, have the ability to solve problems more efficiently and effectively than single agents. This paper considers the problem of controlling co-operative multiple agents so that a number of multiple robots will reach common or different goals (objects) without colliding with each other. The proposed system called *CAROS* (Co-operative Autonomous RObotic Systems) consists of autonomous mobile robots using decentralised control and implicit communication. To achieve cost effective agents, a fuzzy control method is used in this work to replace the conventional control. Each robot carries a fuzzy controller (FC) MIMO (Multi-input Multi-Output), of four state-input variables and two control-output variables. The four inputs into the FC are the direction and distance to the goal and also the direction and distance to the nearest robot (static or motionary). The two control-output variables are the commands for the speed control of the two wheels of the mobile robot. Simulation results are presented to show how the mobile robot performs according to various dynamic environment changes.

1 Introduction

Achieving co-operative multi-agent (robot) systems is a very difficult and challenging task. The assumption that multiple agents have the potential to solve problems more efficiently than a single agent, has attracted the attention of researchers in the areas of control engineering, computer science, psychology and others. Possible reasons for developing and designing co-operative robots include distributed action, problem decomposition and task allocation in parallel. The meaningful operation of a multi-robot system requires that serious problems such as collision avoidance and co-ordination of member of robots be solved.

The architecture design requirements [1] for a multiple robot system are complex and have several differences compared to single robot architecture. However, the effort to

build several simple, cheap, more flexible and more fault-tolerant robots rather than a single powerful robot for each separate task has led to recent research on decentralised control architectures. The development of these architectures becomes simpler and may be more reliable using a series of intelligent methods such as fuzzy logic, and neural networks. The demand for simple controllers with small amount of processing and memory onboard brings a need for the above intelligent control methods. This paper proposes a system called *CAROS* (Co-operative Autonomous RObotic Systems) in order to achieve co-operative agents in an unknown and uncertain environment. The system comprises of the three following characteristics. Firstly the number N of robots must be homogeneous (functionally and physically identical). Secondly the use of communication among the agents is implicit (robots do not communicate with each other and also these is no broadcast communication between them). Third the control method is decentralised (each robot makes its own decisions and performs only these decisions without having any connection to a central mechanism) using fuzzy logic to tackle the motion planning and control problem.

This paper is organised into seven sections. The related work on control of co-operative robots is defined in section 2. Section 3 defines the problem statement. The Fuzzy theory is discussed in section 4. Section 5 presents the fuzzy approach on multiple co-operative agents. Section 6 shows some experiments results of the proposed fuzzy system. Section 7 contains the conclusions of the overall work.

2 Related work

Co-operative robotics is a relatively new research area that began in the late 1980s. There has been much work in multiple mobile robot systems, much of which has not considered the use of fuzzy logic. Fukuda and Nakagawa [2], introduced new project in the field of co-operative robotics called CEBOT (Cellular Robotic System). Their work is based on co-ordination among mobile multi-robot systems with emphasis on communication mechanisms, which can be used to support co-ordinated behaviour. Asama [3], developed the ACTERSS (ACTor-based Robot and Equipment Synthesis System) a multi-robot system designed for heterogeneous agents, based on communication issues. The robots act independently, but if the need arises, they

negotiate with other robots to form a co-operative group to handle the problem. Arkin [4], presents research concerned with sensing, communication, and social organisation for multiple mobile robots that are able to forage and retrieve objects in a hostile environment. Noreils [5], proposed a three-layered control architecture for semi-heterogeneous robots that included a planner level, a control level, and a functional level. The first (planner) level is the high-level decision-maker. Many of the recent co-operative robotic systems, in contrast to the earlier works, are based on a behaviour-based approach [6]. For instance Mataric [7], developed behaviours for multi-agent systems using subsumption style architecture. Further reading on recent trends in control architecture for autonomous vehicles can be found in [8]

This work presents the design of a fuzzy logic control for co-operative agents. Each robot (agent) will be able to reach an independent object from an initial position, without any prior knowledge of the environment, avoiding collisions with the other agents and in the shortest possible travel time.

3 Problem statement

In this paper it is assumed that there are number n mobile robots (MIABOT V2) figure 1, which must reach two independent objects (G_1 and G_2) without colliding with each other.

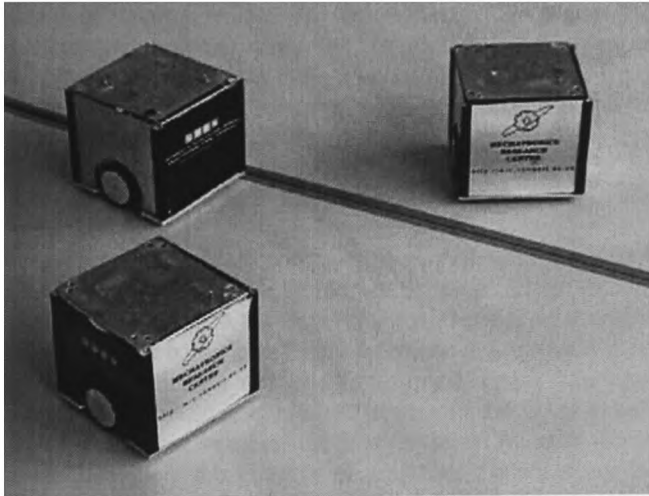


Figure 1. Photo of MIABOT V2 robots

The robots are moving in 2-dimensional completely uncertain and unknown environment with non-constant velocity. The unknown environment is assumed to be filled by motionary and/or stationary neighbour robots. The geometric configuration of the mobile robots during the co-operative task is shown in figure 2. Ultrasonic sensors provide information about the neighbouring robots and their relative positions. It is assumed that the ultrasonic sensors mounted on top of the robot obtains the information in terms of polar co-ordinates (d, a) with error-free measure. This information will be the distance to the nearest robot (d_{robot}) and the direction to the nearest robot (a_{robot}) in robot co-ordinates $(x_{r,n}, y_{r,n})$. An additional input provides information about the direction and

distance to the goal. Using a compass the robot will be able to correct the error (Δa) between its position and the goal a_{goal} . The robot also receives a signal from a beacon placed on top of the goals and it is assumed that this signal is always receivable. The output signals from the fuzzy controller are the commands for the speed control of the two wheels of the mobile robot (u_L and u_R).

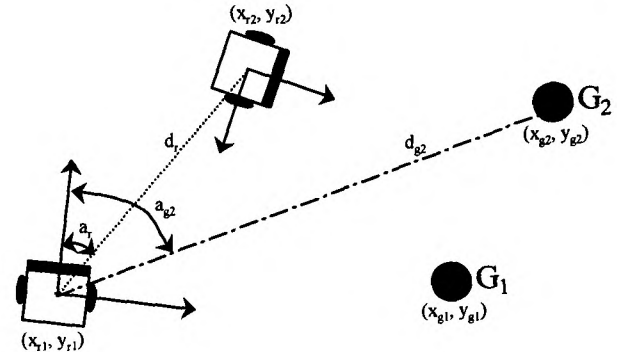


Figure 2. Geometric configuration of the mobile robots during the mission.

4 Fuzzy theory

This section of the paper gives a brief description of the basic concepts of fuzzy theory including fuzzy sets, operations on fuzzy sets and fuzzy inference [9,10].

4.1 Fuzzy sets

In the classical theory a *crisp set* A is a collection of objects or elements x taken in a *universal set* U . The characteristic function $\mu_A: x \in U \rightarrow \{0,1\}$ declares which elements of U are members of the set and which are not. If x belongs to A then $\mu_A(x)=1$ otherwise $\mu_A(x)=0$.

For a fuzzy set, the characteristic function allows various degrees of membership for each object or element. When \tilde{A} is a fuzzy set and x is a relevant element, the proposition " x is a member of \tilde{A} " is not necessarily true or false, but it may be true only to some degree, the degree to which x is actually a member of \tilde{A} .

According to this introductory discussion the formal definition of a fuzzy set is stated as follows:

Let U be a universal set, then the fuzzy set \tilde{A} in U is defined by a set of pairs:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in U\} \quad (1)$$

where the $\mu_{\tilde{A}}(x)$ is called the membership function of x in \tilde{A} , U is defined as Universe of discourse and \tilde{A} is defined as the supremum of $\mu_{\tilde{A}}(x)$ over U .

4.2 Operations on fuzzy sets

In fuzzy theory the most common crisp operators, such as *union*, *intersection* and *complement* are used, always

dependent on a particular application. According to fuzzy set theory for any two fuzzy set \tilde{A} and \cup defined in U , with membership functions being $\mu_{\tilde{A}}(x)$ and $\mu_{\cup}(x)$ respectively the

membership function $\mu_{\mathcal{C}}(x)$ of their intersection $\mathcal{C} = \tilde{A} \cap \cup$ is pointwise defined by

$$\mu_{\mathcal{C}}(x) = \min\{\mu_{\tilde{A}}(x), \mu_{\cup}(x)\}, \quad x \in U \quad (2)$$

membership function $\mu_{\tilde{N}}(x)$ of their union $\tilde{N} = \tilde{A} \cup \cup$ is pointwise defined by

$$\mu_{\tilde{N}}(x) = \max\{\mu_{\tilde{A}}(x), \mu_{\cup}(x)\}, \quad x \in U \quad (3)$$

membership function $\mu_{\tilde{A}^c}(x)$ of the complement of a fuzzy set \tilde{A} is defined by

$$\mu_{\tilde{A}^c}(x) = 1 - \mu_{\tilde{A}}(x) \quad x \in U \quad (4)$$

4.3 Fuzzy inference

The fuzzy inference method, which has been selected for this work is the fuzzy min-max method proposed by Mamdani [11]. Fuzzy inference is a reasoning method using fuzzy theory in which the human knowledge is expressed using linguistic rules. Membership functions, assigned with linguistic variables, are used to fuzzify physical quantities. Fuzzified inputs are inferred to a fuzzy rule base. The relationship between fuzzy inputs and fuzzy outputs is characterised by the rule base. For instance, a simple fuzzy control rule relating the input d to the output e maybe expressed in the following form:

$$\text{IF } d \text{ is } F \text{ THEN } e \text{ is } K \quad (5)$$

where F and K are fuzzy values defined on the universes of X and J , respectively. The response of each fuzzy rule is weighted according to the degree of membership of its input conditions. The role of the fuzzy inference is to provide a set of common actions according to fuzzified inputs. The next step is the defuzzification method to convert the fuzzy values into a crisp output value of the fuzzy system. In other words defuzzification is the process of mapping from a space of inferred fuzzy control actions to a space of non-fuzzy (crisp) control actions. The defuzzification method used for this work is the centroid method and is expressed in the form as follows:

$$e^* = \frac{\sum_{i=1}^n \mu_k(c_i) * c_i}{\sum_{i=1}^n \mu_k(c_i)} \quad (6)$$

where e^* is a crisp output value of the fuzzy system, n is the number of control rules associated with the fuzzified inputs, and c_i is the centroid of the membership function associated with each linguistic value in the output space.

5 Fuzzy approach on multiple co-operative agents

In this section the paper proposes the fuzzy approach for multiple co-operative agents. The agents move through an unknown environment in which they must avoid collision with their neighbours and must reach an allocated goal. Fuzzy logic is used to implement the direct map between the input space (sensor data) into the output space (control commands u_L and u_R).

The proposed algorithm for the co-operative agents can be decomposed in seven steps as follows:

Algorithm CAROS

Step 1: Start the robot (noisy sensor measurements) from the position R_i and set robot's speed.

Step 2: Detect the goal? (distance to the goal, signal from the beacon)

If true: go to step 3

If false: find_goal (change speed $u_i = u_r > \text{set speed}$), and then go to step 2

Step 3: Orientates the goal? (angle to the goal, measurement from the compass)

If true: go to step 4

If false: change_orientation, (change speed $u_i > u_r$ or $u_i < u_l$) and then go to step 3

Step 4: Moving robot (neighbour) found? (signal from ultrasonic sensors)

If true: avoid_moving_robot (change the speed $u_i = u_r > \text{set speed}$ or $u_i = u_r < \text{set speed}$) and then go to step 4

If false: go to step 5

Step 5: Static robot found? (signal from ultrasonic sensors)

If true: avoid_static_robot (change speed $u_i > u_r$ or $u_i < u_l$) and then go to step 3

If false: go to step 6

Step 6: get_goal (change speed $u_i = u_r > \text{set speed}$) and then go to step 7

Step 7: Reach the goal?

If true: STOP the robot (distance to the goal equal to zero)

If false: then go to step 2

The above algorithm presents an efficient and effective approach for co-operative agents moving in 2-dimensional environment. However, some questions still remain for the proposed algorithm. The algorithm only considers the nearest neighbour robot at a given time. The assumption that the robot is able to detect the status between static and moving neighbour robot is also made. The algorithm is suited to the proposed CAROS approach as all the robots are

homogeneous. Figure 3 illustrates the flowchart of the proposed algorithm for the co-operative agents.

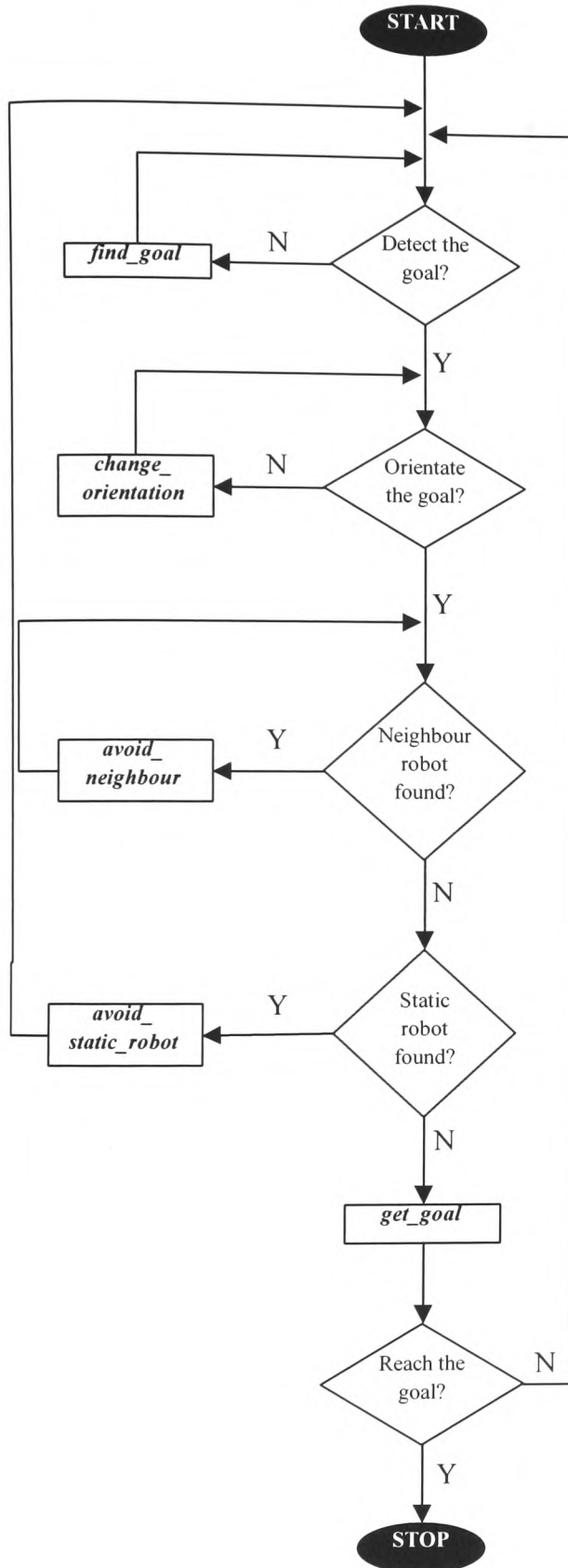


Figure 3. Flowchart of the proposed algorithm

The fuzzy control system adjusts the velocities of the two wheels of the mobile robot, which are independently controlled by a pair of DC motors. The input signals to the fuzzy system are the distances to the nearest robot *robot_close*, *robot_medium* and *robot_far*, the direction to the nearest robot *negative_left_big*, *negative_left_small*, *zero*, *positive_right_small*, and *positive_right_big*, the distance to the goal *goal_close*, *goal_medium* and *goal_far*, and finally the direction to the goal *goal_big_left*, *goal_left*, *goal_front*, *goal_right*, and *goal_big_right*. The output control signals of the fuzzy system are the two vector velocities of the left and right wheel (u_L , u_R), *slow*, *medium*, and *fast*. Very careful consideration is needed for the design of the universe of discourse of the two velocity vectors. Those two velocity vectors must be able to accelerate, decelerate and turn the robot into the required turning angle.

The flowchart in figure 3 shows the flow of the proposed control algorithm within a certain amount of time (t). The first function will 'fire' if the robot cannot detect the beacon a simple command to the two motors will increase the robot's speed. As long as the robot detects the beacon the robot's speed returns to predefined one. The second function *change_orientation* will 'fire' if the robot does not orientate the goal. The differential (ΔU) of the two velocities will orientate the robot to the goal. As the flowchart shows, the functions are closed loop to make sure that there is zero error before proceeding into the next step. The third control function will 'fire' only if a neighbour robot has been found around the agent within a certain distance and direction. As stated earlier, at this point only one agent at a time is considered, and it is also assumed that the robot is able to define its motion. The output signals of the fuzzy system are the velocity vectors for the two wheels, either positive or negative. As long as the robot finds a neighbour in a collision zone it will automatically speed up or slow down. Four of the fifteen fuzzy control rules required to generate the *avoid_neighbour* function are given as follows:

1. If (*angle_r* is *negative_left_big* {NLB}) and (*dist_r* is *close* {C}) then (u_L is *medium* {M})(u_R is *medium* {M})
2. If (*angle_r* is *negative_left_small* {NLS}) and (*dist_r* is *far* {F}) then (u_L is *fast* {F})(u_R is *fast* {F})
3. If (*angle_r* is *zero* {Z}) and (*dist_r* is *close* {C}) then (u_L is *slow* {S})(u_R is *slow* {S})
4. If (*angle_r* is *positive_right_small* {PRS}) and (*dist_r* is *medium* {M}) then (u_L is *medium* {M})(u_R is *medium* {M})

In order to improve and enhance the *avoid_neighbour* function ongoing research is being carried out. The fourth function *avoid_static_robot* is needed in case the robot finds a broken agent which is unable to complete the mission. In this case the robot must be able to avoid the neighbour and carry on with its own mission. The scenario that the robot is only able to avoid one neighbour at a time is applied in this function as well. The output control signal of the fuzzy system is the differential command of the two independent drive wheels. Four of the fifteen fuzzy control rules required to generate the *avoid_static_robot* function are given as follows:

1. If (*angle_r* is *negative_left_big* {NLB}) and (*dist_r* is *close* {C}) then (u_L is *medium* {M})(u_R is *medium* {M})

2. If (*angle_r* is *negative_left_big* {NLB}) and (*dist_r* is *far* {F}) then (*u_L* is *fast* {F})(*u_R* is *fast* {F})
3. If (*angle_r* is *negative_left_small* {NLS}) and (*dist_r* is *close* {C}) then (*u_L* is *medium* {M})(*u_R* is *slow* {S})
4. If (*angle_r* is *zero* {Z}) and (*dist_r* is *close* {C}) then (*u_L* is *fast* {F})(*u_R* is *slow* {S})
5. If (*angle_r* is *positive_right_small* {PRS}) and (*dist_r* is *close* {C}) then (*u_L* is *slow* {S})(*u_R* is *medium* {M})

The last function *get_goal* of the proposed algorithm is very similar to the first one. If the algorithm passes all the stages and the robot orientates to the goal without having any problems with other agents, then its speed increases in order to approach the goal in the shortest possible time. The algorithm is repeated until the agent reaches the goal.

6 Experimental results

To test and analyse the proposed fuzzy system for the co-operative agents, simulation and experimental studies were performed. Simulation was carried out using the Matlab Fuzzy Logic Toolbox. The two most important functions *avoid_neighbour* and *avoid_static_robot* were simulated and analysed. Figure 4 and 5 show a 3-D surface and the relationship between the two input variables *distance* and *angle* to the nearest static agent and the two output variables the velocity of the left and right wheel (*u_L*, *u_R*). It is demonstrated that according to the static robot position the velocity of both wheels is modified respectively.

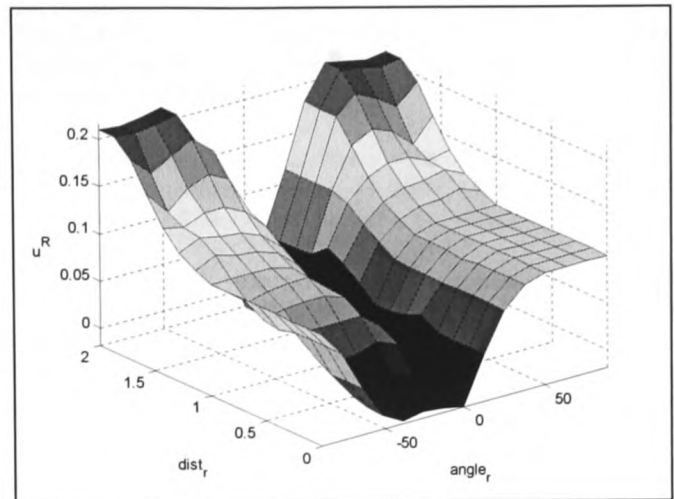


Figure 5. Output for the velocity of the right wheel according to the distance and direction of the nearest static neighbour

Figure 6 presents the relationship between the two input *distance* and *direction* to the nearest moving agent and the velocity for both left and right wheels control output.

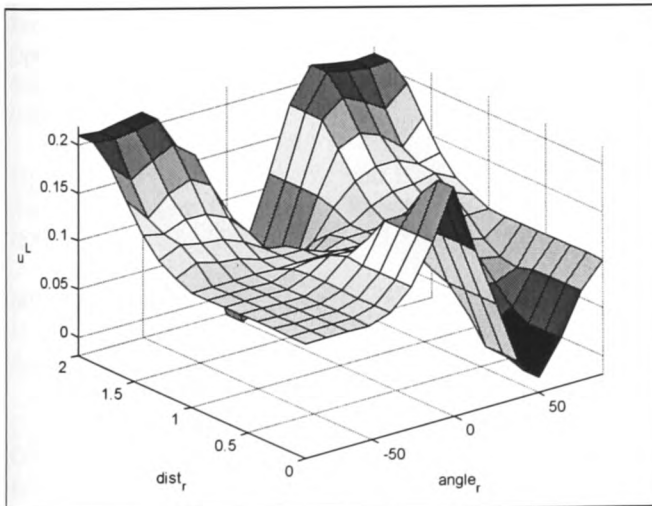


Figure 4. Output for the velocity of the left wheel according to the distance and direction of the nearest static neighbour

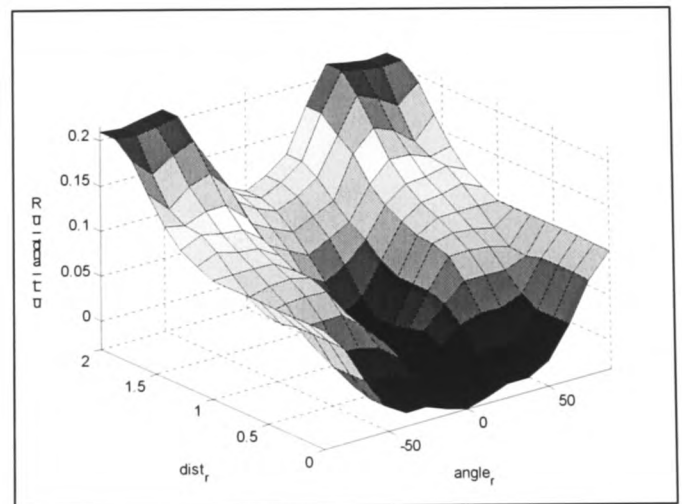


Figure 6. . Output signal for the velocity of both left and right wheel according to the distance and direction of the nearest moving agent

From figure 6, it can be observed that when the robot finds a neighbour agent in front of it, it instantly reduces its speed on both wheels. If the moving neighbour is far from the

robot trajectory the fuzzy system sends a signal to the DC motors for either acceleration or deceleration.

7 Conclusions

This paper has considered the control problem for co-operation within multiple robots. The paper proposed a new decentralised control for co-operative agent using fuzzy logic control. Each agent carries a MIMO FLC and has the ability to sense the goal and the neighbour robots. The design of the fuzzy approach is based on min-max Mamdani method. The new decentralised control was shown that is more suited to the **CAROS** approach. Simulation results were also presented to show how the two independent velocities change according to the dynamic environment.

The next phase of this work is to implement the same method but with more than one robot at a time. In order to achieve further improvement experiments with other fuzzy inference techniques will be considered for obtaining more accurate output velocities. Finally research is being carried out in order to combine the two control-output variables into a single control variable.

References

- [1] Mouzakitis, A. and Roberts, G. N., "Intelligent Autonomous Mobile Robots and Co-operative Multi-Agent Systems: Research, Design, Future Developments and Improvements," *EUREL, European Advanced Robotics Systems, Masterclass and Conference - Robotics 2000*, University of Salford, Manchester, UK, pp. 600-609, (2000).
- [2] Fukuda, T. and Nakagawa, S., "A Dynamically Reconfigurable Robotic System (Concept of a system and Optimal Configurations)," *Int. Conf. on Industrial Electronics, Control, and Instrumentation*, pp. 588-595, (1987).
- [3] Asama, H., Matsumoto, Y., and Ishida, Y., "Design of an Autonomous and Distributed Robot System: ACTRESS," *IEEE/RSJ*, pp. 283-290, (1989).
- [4] Arkin, R. C., Cooperation Without Communication: Multiagent Schema-Based Robot Navigation. *Robotic Systems*, vol. 9, pp. 351-364, (1992).
- [5] Noreils, F. R., Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment *The Int. Journal of Robotics Research*, vol. 12, pp. 79-98, (1993).
- [6] Brooks, R. A., A Robust Layered control System for a Mobile Robot *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. 14-23, (1986).
- [7] Mataric, M. J., Integration of Representation Into Goal-Driven Behavior-Based Robos *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 304-312, (1992).
- [8] Ridao, P., Batllet, J., Amat, J., and Roberts, G. N., Recent Trends in Control Architectures for Autonomous Underwater Vehicles *International Journal of Systems science*, vol. 30, pp. 1033-1056, (1999).
- [9] Zimmermann, H. J., Fuzzy Set Theory - and Its Applications Anonymous 2nd, 1991. Kluwer Academic Publishers. USA. 0-7923-9075-X.
- [10] Wang, L. X., Course in Fuzzy Systems and Control 1997. Bernard Goodwill. USA. 0-13-540882-2.
- [11] Mamdani, E. H., Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis *IEEE Transactions on Computers*, vol. 26, pp. 1182-1191, (1977).

Intelligent Autonomous Mobile Robots and Co-operative Multi-agent Systems: Research, Design, Future Developments and Improvements.

by A. Mouzakitis, and G. N. Roberts
Mechatronics Research Centre
University of Wales College, Newport
Allt-yr-yn Campus, PO Box 180, Newport, NP20 5XR, UK
Tel: +44 (0) 1633 432487. Fax: +44 (0) 1633 432442
E-mail: alexandros.mouzakitis@newport.ac.uk

Abstract - Intelligent autonomous mobile robots and multi-agent (robot) systems, having different skills and capabilities for specific subtasks, have the ability to solve problems more efficiently and effectively than single agents. This paper discusses recent research on co-operative robots, the design of their control architectures and proposes future developments and improvements by presenting design methods, evaluation of the control systems, and applications such as robot football. Different forms of co-operation are presented. In particular, the paper will show how multiple mobile robot systems architectures may be designed by examining six key issues: (1) centralised, decentralised or hybrid control, (2) heterogeneous or homogeneous robots, (3) co-operation with-or-without communication, (implicit or explicit communication), (4) making agents that work as a team, (5) multiple mobile robots path planning, and (6) learning. The paper also reviews and gives brief descriptions of the most common architectures for co-operative robots, emphasising the theory and the implementation of these techniques. Robot football provides an excellent research opportunity for integrated multiple robots systems, the robot football system at UWCN (University of Wales College, Newport) is described. The paper concludes with a discussion on future developments and improvements.

Keywords: Co-operative Mobile Robots, Design Architecture, Robot Football

1. Introduction

Achieving co-operative multi-agent (robot) systems is a very difficult and challenging task. The assumption that multiple agents have the potential to solve problems more efficiently than a single agent,

has led to recent research effort into multi-robot systems [1]. There are several reasons for developing and designing co-operative robots. Firstly, distributed action, at the same time multiple robots can be in many places [2]. Secondly, it is quite possible that many applications could be solved much more quickly if the mission could be divided across a number of robots in parallel [3]. Thirdly, building and using several simple robots can be easier, cheaper, more flexible and more fault-tolerant than having a single powerful robot for each separate task [4]. Fourthly, several problems are well suited for decomposition and allocation among multi-robot systems. The meaningful operation of a multi-robot system requires that serious problems such as collision avoidance among individual robots and co-ordination of member robots be solved [5].

This paper is organised into seven sections. The related work on co-operative robots is defined in section 2. The six key issues for designing multiple mobile robot systems architectures are discussed in section 3. Section 4 is concerned with forms of co-operation. Section 5 presents the robot football system of the UWCN. Section 6 suggests future developments and improvements. Section 7 contains the conclusions.

2. Related Work

Co-operative robotics is a new research area that began in the late 1980s. Fukuda and Nakagawa [6], introduced new project in the field of co-operative robotics called CEBOT (Cellular Robotic System). Their work is based on co-ordination among mobile multi-robot systems with emphasis on communication mechanisms, which can be used to support co-ordinated behaviour. About the same time relevant work on co-operative robotics carried out by

Beni [7], SWARM (large numbers of homogeneous robots), and Asama [8], with the ACTRESS (ACTor-based Robot and Equipment Synthetic System), multi-robot system designed for heterogeneous agents, with focus on communication issues. The robots act independently, but if the need arises, they negotiate with other robots to form a co-operative group to handle the problem. The 1990s decade begins with the works of Caloud *et al.* [9], on the GOFER architecture, and Steels [10]. The latter used the behaviour-based approach to solve the problem of co-operation between distributed agents. Research activity on co-operative robots increased dramatically with important work by [11, 12, 13, 14], the latter presents research concerned with sensing, communication, and social organisation for tasks such as foraging. Mataric [15], developed behaviours for multi-agent systems using subsumption style architecture. Noreils [16], proposed a three-layered control architecture that included a planner level, a control level, and a functional level. Similar work was carried out by [17, 18, 19, 20, 21, 22, 23]. There is more work reported in the literature, but the aforementioned is the most significant within the research community. All the above research activity has attempted to solve the same problem (co-operation among robots) by adopting different techniques. The solution to the problem is not straightforward, so many of the researchers prefer to perform simulation rather than physical implementation. Many of the recent co-operative robotic systems, in contrast to the earlier works, are based on a behaviour-based approach [24]. The emphasis on the connection between intelligence and environment is strongly associated with the behaviour-based approach but is not intrinsic to multiple robot systems.

Shen [25], presents a system architecture for robot football in order that the robot players can recognise and track objects in real-time, navigate in a dynamic field, collaborate with team-mates, and strike the ball in the correct direction.

3. Architecture Design for Multiple Mobile Co-operative Robot System

The architecture design requirements for a multiple mobile co-operative robot system is complex and have several differences compared to single robot architecture. According to Hayes-Roth [26], robot architecture refers to the interconnection topology between components with the specific functionality and interface of each component, in which perception, reasoning and action occur. However, from another point of view, multiple robot architecture of a co-operative robotic system provides the infrastructure upon which collective behaviours are implemented, and determines the capabilities and limitations of the system [4].

In the following subsections the six key issues, regarding the architecture design of multiple mobile co-operative robot system are discussed.

3.1 Centralised, Decentralised or Hybrid Control

The first decision that has to be made is whether the control architecture will be centralised, decentralised or one form of hybrid. There is no specific law or any particular restriction of which control form is the best. This is because the control architecture that is suitable for a given task may not be flexible and suitable for another.

In centralised control architecture, decisions are made in a central mechanism or in a single control agent, and afterwards transmitted to the executive components (robots). Due to the complexity of the hierarchical planning system the development of these architectures is difficult because it is not easy to determine the suitability of this concept in advance [19].

With decentralised control architecture each robot makes its own decisions and performs only these decisions without having any connection with a central mechanism or single control agent. This is very important for a multiple mobile co-operative robot system, because problems such as fault-tolerance, the difficulty of adding new features into the system, and re-implementation of the system are automatically avoided. The research literature has been dominated with works on decentralised control architecture, avoiding the centralised approach. Related work on a decentralised approach has been carried out by Parker [27], which proposes a fully distributed, behaviour-based architecture called ALLIANCE.

Hybrid control architecture - often designers decide to adopt a hybrid solution of the decentralised and centralised approach. If the problem lies where the decentralised system needs an internal central agent a hybrid solution has to be adopted. There are few reports of work on co-operative robot in the literature in which the control architecture is hybrid. An example is the work of Noreils [16] in which the architecture is composed of, three levels: functional, control, and planner level. The last level is the level at which complex operations such as planning and co-operation are carried out.

3.2 Heterogeneous or Homogeneous Robots

A second step in the design of the architecture for multiple mobile co-operative robots is the selection between heterogeneous and homogeneous robot characteristics. The robots should be either homogeneous or heterogeneous, depending on the task being undertaken. This decision is very important because any underestimation or overestimation of the problem could lead to wrong design with undesired results. The following paragraphs explain and give appropriate definitions

of the terms homogeneous and heterogeneous with reference to the related works of several other researchers.

A homogeneous robot team consists of a number of robots, which have the same skills and capabilities. So far, most of the research projects involve homogeneous robot teams. This choice makes the design process much easier, because it minimises the complexity of task allocation.

Co-operation may involve a team of robots provided with different skills (the mechanical design may be different also), referred to as heterogeneous robots [16]. The complexity of the design is increased dramatically compared with the design process with homogeneous robots. Once again the designer has to be able to understand the problem in order to decide what kind of strategy will be implemented for given task allocations. Despite the fact that multiple robot architecture design is very complex using heterogeneous robot teams, work has been reported by [28, 1, 29, 30, 31].

3.3 Co-operation with-or-without Communication

Communication between multiple robots may make the team able to perform some co-operative tasks more efficiently. Communication is expressed as a form of interaction in which the dynamic relationship between agents is expressed through the intermediary of mediators, signals, which once interpreted, will affect these agents [32]. However, achieving co-operation within the robot team with communication, produces several advantages i.e. achieving very complex tasks and also disadvantages i.e. waiting time, and transmission error. The above benefits and restrictions of communication led the research community to distinguish between explicit (with) or implicit (without) communication. The two main forms of communication are explained below.

If the robots (agents) communicate with each other directly, or if there is broadcast communication between them, then, this form of communication is called explicit communication. Ichikawa *et al.* [5] proposed the Hello-Call communication protocol, which was utilised low-level intelligence multi-robot system, in order to investigate the message transmission ability among the robots. Asama *et al.* [33], developed a communication system between multiple robotic agents able to exchange messages by radio communication. Yoshida *et al.* [22], presents an optimal design for local communication between multiple mobile robots. Their aim was to minimise the information transmission time by creating an optimal communication area. In other work, Yoshida *et al.* [34], classify mobile robot communication into two categories. *Global* communication (≥ 10 robots) with wide-area media and *local* communication (< 10 robots) with limited capacity. Parnichkun and Ozono [35], support that efficient co-operation depends

upon two main factors, communication, and movement of the robots. They propose a communication method for a co-operative robot system using CDCSMA-CD (Code Division Carrier Sensing Multiple Accesses with Collision Detection), which can be used for both point-to-point and broadcast communication.

If the robots do not communicate with each other, or there is no broadcast communication between them, but there is communication through the world environment, then this form of communication is called implicit communication. Arkin [36], suggests that communication bottlenecks between agents pose potentially serious drawbacks in co-ordinated behaviour. He proposes the use of multiagent schema-based robot navigation involving co-operation without communication. Deneubourg J. *et al.* [11], presents work in which robots move randomly, do not communicate, have no hierarchical organisation, have no global representation, can only perceive objects just in front of them, but can distinguish between objects of two or more types with a certain degree of error.

3.4 Making Agents that Work as a Team

The key issue in team working is how well the modelling between agents and environment has been developed [37]. This key issue has a strong relationship with the communication key issue (modelling of purely communicating agents, or not). However, there are several options available of how modelling could be achieved, for and between agents. Ferber [32], proposes a technique of modelling agents and their environment. Other works and techniques can be found in the literature such as Friedrich *et al.* [38], who presents (EOs) Elementary Operations, a method of combining the agent specific nature of skills with the requirements for a general action knowledge representation, inherent to multi-agent systems.

3.5 Multiple Mobile Robots Path Planning

Multiple robot path planning differs from single robot path planning in several ways. A mobile robot has to avoid obstacles and also other robots. To address this particular problem is not an easy task. Significant work has been reported on multiple robot path planning, by Hashimoto and Oba [39], who proposes a dynamic control method to transfer a common heavy object by several wheeled mobile robots. Ota *et al.* [40], present a motion planning method for robot groups. They classify the robots into two groups (P-group/cluster) and implement a motion-planning method using the virtual impedance method. Khatib [41], described a unique real-time obstacle avoidance approach for mobile robots based on the artificial potential field concept. Sasaki *et al.* [42], propose an algorithm for grasping and handling a large object by co-operative multiple robots. Their

work is based on the difference (whether the mass centre of the object is recognised or not) between two optimisation problems (decision of initial robot arrangement/decision of final robot arrangement). Azarm and Schmidt [43], present a novel approach to decentralised motion planning and conflict-resolution for multiple mobile robots.

3.6 Learning

The final key issue in architecture design is learning in multiple robot systems. Changes within the robot's environment can result in poor robot performance and decreases the ability to achieve a given task. One solution to this problem is to introduce a learning method. The chosen method will increase the robot's performance and its ability to respond correctly to environment changes. This desirable result will be achieved by learning, so the robot system will be able to optimise and set its own control features. The main problem is that compared to single robot learning, co-operative learning adds the challenge of a much larger search space, awareness of other team members, and also the synthesis of the individual behaviours with respect to the task given to the group. So far the main learning approach used in autonomous robotics is reinforcement learning. More recently, lazy learning has become the leading bias. Today the full integration of those two techniques lead to what is called lazy Q-learning. Work in this field have been done by [44, 15].

4. Forms of Co-operation

The word 'co-operation' has been applied to behaviour between humans, animals and robots, although co-operation may take various forms. One form of co-operation is by exchanging information or signals. In this kind of co-operation an entity synchronises its action with respect to the other entity (replace the word) or co-ordination (entity co-ordinates its activity with another) takes place instead. Co-ordination is based on a system of signals by which an entity influences the behaviour of another entity. Examples of such co-ordination can be found both in animal and human society. People have to learn specific rules to be able to interact with others. In the animal society a signal is produced by, for example, a movement of the animal itself (a dance) or a modification of its aspect (colour of its skin changes). Work in this area, which also examines the social behaviour of animals, has been carried out by Tinbergen [45] and Wilson [46].

Another form of co-operation in the robotics field is where different robots provided with different skills work toward a common goal. This kind of co-operation appears commonly in human society and can be described through the classic example of the experts metaphor. This metaphor is already applied in several artificial intelligent systems [47], where

experts from different fields are trying to solve a large problem together. Experts decompose the problem into sub-problems, each sub-problem being handled by a specific expert. They then define such issues as how they will work together and what kind of information they have to exchange. Ferber [32] describes in detail forms and methods of co-operation in multi-agent systems.

5. Robot Football at UWCN

Recently, robot football is gaining much interest in the field of robotics because it provides a suitable platform for experimentation and investigation of co-operative multi-robot systems. Robot football is an excellent testbed for studying co-operative multi-robot systems because it involves real-time vision processing, obstacle avoidance, sensing local information, and position correction.

In the following a brief description of the robot football at UWCN is presented. Figure 1 shows the overall system structure comprising robots, communication and vision system.

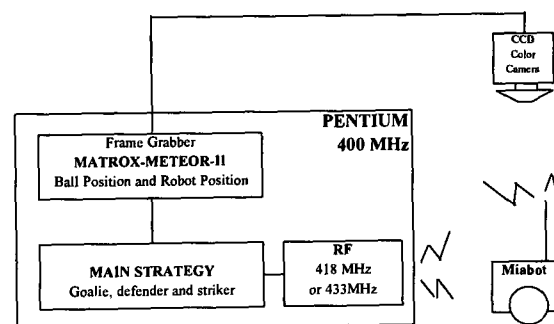


Figure 1: Overall System Structure

5.1 Robots

The design philosophy is to realise autonomous micro robots with various functions so that they can be used not only for robot football but in other applications as well. The robots illustrated in Figure 2 should be small in size (7.5cm^3) to comply with the MIROSOT rules [48], their body must be very robust in order to protect the internal hardware from damage caused by collisions with other robots. The robots carry a control board with the CPU, RX module, H-Bridge, extension ports, and other functional modules.

The DC motors driving the two wheels can be controlled separately so that the robot can respond to a large set of flexible commands. Table 1 summarises the specification of the robots.

5.2 Communication System

According to MIROSOT rules, the method of communication to the robots must be by RF communication with the host-PC using the serial port. The frequency can be selected by exchanging

between 418 and 433 MHz modules. An optional TX module can be added for two-way remote communications.

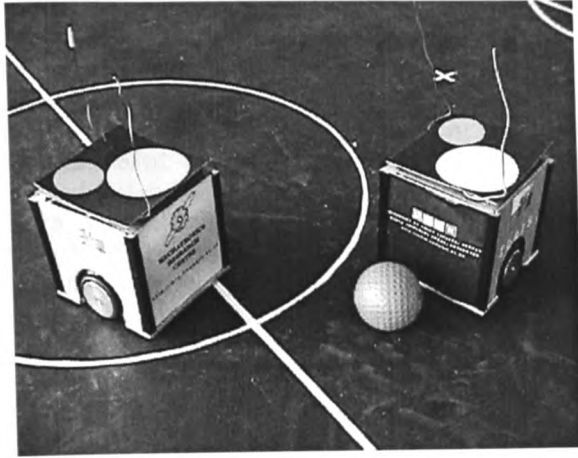


Figure 2: MIABOT V2 Robots at UWCN

Table 1. Specification of MIABOT V2 Robots

Frame	Metal case (7.5cm x 7.5cm x 7.5cm)
CPU	AT90S8515
	4 MIPS oper, 8KB RAM, 512B SRAM
Motor	2 DC Motors
Drive Train	Speed 1m/s, opto-feed for each wheel
Encoder	78 Pulse per revolution
Com. Module (TX/RX)	RF receive com mod (418 or 433Mhz)
Battery	2 x Re-chargeable NIMH (2 x 4.8V)

5.3 Vision System

In order that the vision system can identify the robots and their IDs, uniforms of blue or yellow colour are placed on top of the robots. The uniforms must contain a team colour and must not contain the ball colour (orange). To find the direction of the robot and its ID two circles are used, one 5cm in diameter (team colour) and another 3cm in diameter (robot ID colour). A CCD colour camera takes the pictures and the frame grabber grabs the images at a rate of about 30 frame/sec. The algorithm running on the host computer calculates the centre of the robot and works out its direction.

A problem that arises when the vision system has to identify the robots is the changes of the lighting conditions. As the lighting conditions change camera re-calibration is needed. To overcome this problem ongoing research on auto-calibration is being carried out. However the illumination is rarely constant in intensity or colour through a scene. Experiments have shown that the colour information varies a little compared to intensity, which varies widely [49]. In this situation the HSI (Hue Saturation Intensity) colour model is the best solution because the intensity is decoupled from the colour information in HSI model. The problem in this case is the transformation or conversion from the RGB

(Red Green Blue) values to HSI, which is time consuming for the computer. The equations 1, 2 and 3 below show how to convert the RGB values into HSI [50].

$$H = \cos^{-1} \left\{ \frac{\frac{1}{2}[(R-G) + (R-B)]}{\left[(R-G)^2 + (R-B)(G-B) \right]^{\frac{1}{2}}} \right\} \quad (1)$$

$$S = 1 - \frac{3}{R+G+B} \min(R, G, B) \quad (2)$$

$$I = \frac{1}{3}(R+G+B) \quad (3)$$

6. Future Developments and Improvements

The field of co-operative mobile robotics is still an open research area with a significant application domain. Future developments and improvements are vital in order that the field of co-operative robotics can become more for applications in the real world. So far the research in this area has to show more theory and scepticism rather than integrated solutions of how to solve problems, which involves multiple mobile robots. As stated in section 2 several researchers have addressed and solved partial the problem of co-operation between multiple robots with various solutions and techniques.

In this section the paper proposes possible developments and improvements in the field of co-operative robotics, which are still very attractive for the near future. Firstly the theory and study of social insects, animals by biologists should be applied to multiple robot systems more extensively for the development and improvement of control strategies. What is still missing on the current research is the mathematics on which to base models, of both robots and the tasks, which they are designed to accomplish. Secondly in case of group of robots (swarm intelligence) the interaction period affects the effectiveness. For instance if there is no interaction between each robot, their working ability is in proportion to a number of robots. Therefore research is necessary to attack the problem of interaction period for multiple mobile robots. Third, achieving co-operation within competitive situations is difficult a task. In the robot football players have to co-operate efficiently in order to win the game. A challenging area of research is the improvement and development of more sophisticated solutions to improve system robustness. Those solutions could be adding passing capabilities and communication to improve the ability of players to co-operate, better sensing elements to increase awareness and may be construction of players which are able to learn through their experience. Fourthly, problems concerning a co-operation of robots with different

features, is an attractive subject to be surveyed in the future.

7. Conclusions

This paper has considered the problem of co-operation within multiple mobile robots. Recent research activities within co-operative robotics, the architecture design for multiple mobile co-operative robot systems and examples of applications have also been discussed. This paper has discussed the six key issues for developing such co-operative systems and has highlighted related work. Forms of co-operation have been presented and analysed with appropriate examples. Robot football at UWCN has been presented as an application of co-operative robotics with a brief description of its subsystems. Possible developments and improvements in the field of co-operative robotics, which can lead to more efficient and effective systems, were also discussed.

References

- [1] D. Jung, G. Cheng, and A. Zelinsky, "Experiments in Realising Cooperation between Autonomous Mobile robots," *International Symposium on Experimental Robotics*, 1997.
- [2] R. C. Arkin, and T. Balch, Cooperative Multiagent Robotic. In: *Artificial Intelligence and Mobile Robots. Case Studies of Successful Robot system*, eds. D. KORTENKAMP, R. P. BONASSO, and R. MURPHY. USA: AAAI Press/The MIT Press, 1998. pp. 277-296.
- [3] L. E. Parker, "Distributed Control of Multi-Robot Teams: Cooperative Baton-Passing Task," *Proc. IEEE Int. Conf. on Information Systems Analysis & Synthesis*, pp. 89-94, 1998.
- [4] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, Cooperative Mobile Robots: Antecedents and Directions *Autonomous Robots*, vol. 4, pp. 7-27, 1997.
- [5] S. Ichicawa, F. Hara, and H. Hosokai, "Cooperative Route-Searching Behavior of Multi-Robot System Using Hello-Call Communication," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, Yokohama, Japan, pp. 1149-1156, 1993.
- [6] T. Fukuda, and S. Nakagawa, "A Dynamically Reconfigurable Robotic System (Concept of a system and Optimal Configurations)," *Int. Conf. on Industrial Electronics, Control, and Instrumentation*, pp. 588-595, 1987.
- [7] G. Beni, "The Concept of cellular Robotic System," *Proc. IEEE Int. Symposium on Intelligent Control*, pp. 57-62, 1988.
- [8] H. Asama, A. Matsumoto, Y. Ishida, "Design of an Autonomous and Distributed Robot System: ACTRESS," *In IEEE/RSJ*, pp. 283-290, 1989.
- [9] P. Caloud, W. Choi, J. C. Latombe, C. L. Pape, and M. Yim, "Indoor Automation With Many Mobile Robots," *IEEE Int. Workshop on Intelligent Robots and Systems*, pp. 67-72, 1990.
- [10] L. Steels, Cooperation Between Distributed Agents Through Self-Organization *Decentralized A. I.*, vol. pp. 175-196, 1990.
- [11] J. L. Deneubourg, S. Goss, N. Franks, A. S. Franks, C. Detrain, and L. Chretien, "The Dynamics of Collective Sorting Robot - Like Ants and Ant - Like Robots," *1st Int. Conf. on Simulation of Adaptive Behaviour*, pp. 356-363, 1990.
- [12] H. Asama, M. K. Habib, I. Endo, K. Ozaki, A. Matsumoto, and Y. Ishida, "Functional Distribution among Multiple Mobile Robots in An Autonomous and Decentralised Robot System," *Proc. IEEE Int. Conf. on Robotics and Automation*, Sacramento, California, pp. 1921-1926, 1991.
- [13] P. K. C. Wang, Navigation Strategies for Multiple Autonomous Mobile Robots Moving in Formation *Robotic Systems*, vol. 8, pp. 177-195, 1991.
- [14] R. C. Arkin, Cooperation Without Communication: Multiagent Schema-Based Robot Navigation. *Robotic Systems*, vol. 9, pp. 351-364, 1992.
- [15] M. J. Mataric, Integration of Representation Into Goal-Driven Behavior-Based Robots *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 304-312, 1992.
- [16] F. R. Noreils, Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment *The Int. Journal of Robotics Research*, vol. 12, pp. 79-98, Feb, 1993.
- [17] L. E. Parker, "Designing control Laws for Cooperative Agent Teams," *Proc. IEEE Int. Conf. on Robotics and Automation*, Atlanta, GA, pp. 582-587, 1993.
- [18] C. R. Kube and H. Zhang, Collective Robotics: From Social Insects to Robos *Adaptive Behavior*, vol. 2, pp. 189-219, 1993.
- [19] T. Laengle and T. C. Lueth, "Decentralized Control of Distributed Intelligent Robots and Subsystems," *Proc. of the IFAC Symposium on Artificial Intelligence in Real Time Control (ARTC)*, Valencia, Spain, 1994.
- [20] D. P. Barnes, A Behavior Synthesis Architecture for Co-operant Mobile Robots. In: *Advanced Robotics & Intelligent Machines*, eds. J. O. Gray and D. G. Caldwell. Institution of Electrical Engineers, 1996. pp. 295-314.
- [21] H. S. Shim, H. S. Kim, M. J. Jung, I. H. Choi, J. H. Kim, and J. O. Kim, Designing Distributed Control Architecture for Cooperative Multi-Agent System and Its Real-Time Application to

- Soccer Robot *Robotics and Autonomous Systems*, vol. 21, pp. 149-165, 1997.
- [22] E. Yoshida, T. Arai and J. Ota, Local Communication of Multiple Mobile Robots: Design of Group Behavior for Efficient Communication *Advanced Robotics*, vol. 11, pp. 759-779, 1998.
- [23] B. B. Werger, Cooperation Without Deliberation: A Minimal Behavior-Based Approach to Multi-Robot Teams *Artificial Intelligent*, pp. 293-320, 1999.
- [24] R. A. Brooks, A Robust Layered Control System for a Mobile Robot *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. 14-23, 1986.
- [25] W. M. Shen, J. Adibi, R. Adobbati, B. Cho, A. Erdem, H. Moradi, B. Salemi, and S. Tejada, Towards Integrated Soccer Robots *AI Magazine*, vol. 19, pp. 79-85, 1998.
- [26] B. Hayes-Roth, An Architecture for Adaptive Intelligent Systems *Artificial Intelligence*, vol. 72, No. 1-2, pp. 329-365, 1995.
- [27] L. E. Parker, ALLIANCE: An Architecture for fault Tolerant Multirobot Cooperation *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 220-240, 1998.
- [28] L. E. Parker, Adaptive Heterogeneous Multi-Robot Teams *Neurocomputing*, vol. 28, pp. 75-95, 1999.
- [29] K. Singh and K. Fujimura, "Map Making by Cooperating Mobile Robots," *Proc. IEEE Int. Conf. on Robotics & Automation*, pp. 254-259, 1993.
- [30] D. Jung, and A. Zelinsky, An Architecture for Distributed Cooperative Planning in a Behavior-Based Multi-Robot System *Robotics and Automation Systems*, vol. 26, pp. 149-174, 1999.
- [31] R. S. Aylett, A. M. Coddington, G. R. A. Hercok, and D. P. Barnes, Heterogeneous Agents for Multi-Robot Cooperation *Design and Development of Autonomous Agents*, vol. 211, pp. 3-7, 1995.
- [32] J. Ferber, Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence Anonymous 1999. Addison Wesley Longman Inc. New York, USA. 0-201-36048-9
- [33] H. Asama, K. Ozaki, A. Matsumoto, Y. Ishida, M. K. Habib, H. Kaetsu, and I. Endo, "A Communication System Between Multiple Robotic Agents," *In Proc. Symposium on Flexible Automation*, pp. 647-654, 1992.
- [34] E. Yoshida, T. Arai, M. Yamamoto and J. Ota, Local Communication of Multiple Mobile Robots: Design of Optimal Communication Area for Cooperative Tasks *Journal of Robotic Systems*, vol. 15, pp. 407-419, 1998.
- [35] M. Parnichkun and S. Ozono, CDCSMA-CD Communication method for Cooperative Robot Systems *Advanced Systems*, vol. 11, pp. 669-694, 1998.
- [36] R. C. Arkin, Motor Schema-Based Mobile Robot Navigation *International Journal of Robotics Research*, vol. 8, pp. 92-112, 1989.
- [37] C. R. Kube and H. Zhang, Task Modelling in Collective Robotics *Autonomous Robots*, vol. 4, pp. 53-72, 1997.
- [38] H. Friedrich, O. Rogalla, and R. Dillmann, Integrating Skills Into Multi-Agent Systems *Intelligent Manufacturing*, vol. 9, pp. 119-128, 1998.
- [39] M. Hashimoto, and F. Oba, "Dynamic Control Approach for Motion Coordination of Multiple Wheeled Mobile Robots Transporting a Single Object," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, Yokohama, Japan, pp. 1944-1951, 1993.
- [40] J. Ota, T. Arai, and D. Kurabayashi, "Dynamic Grouping in Multiple Robots System," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, Yokohama, Japan, pp. 1963-1970, 1993.
- [41] O. Khatib, Real-Time Obstacle Avoidance for Manipulators and Mobile Robots *The International Journal of Robotics Research*, vol. 5, pp. 90-98, 1996.
- [42] J. Sasaki, J. Ota, E. Yoshida, D. Kurabayashi, and T. Arai, "Cooperating Grasping of a Large Object by Multiple Mobile Robots," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1205-1210, 1995.
- [43] K. Azarm, and G. Schmidt, A Decentralised Approach for the Conflict-Free Motion of Multiple Mobile Robots *Advanced Robotics*, vol. 11, pp. 323-340, 1997.
- [44] P. Maes and R. A. Brooks, "Learning to Coordinate Behaviors," *AAAI*, Boston, pp. 796-802, 1990.
- [45] N. Tinbergen, *Social Behavior in animals*. London, England: Methuen and Co, 1953.
- [46] O. E. Wilson, *The Insect Societies*. Cambridge, MA: Harvard University Press, 1971.
- [47] C. Hewitt, Viewing Control Structures as patterns of Passing Messages *Artificial Intelligence*, Vol. 8, pp. 323-335, 1977.
- [48] FIRA MiroSot Game Rules, <http://www.fira.net/games/mirosot.html>.
- [49] S. H. Kim, J. S. Choi, J. K. Kim, and B. K. Kim, A Cooperative Micro-robot System Playing Soccer: Design and Implementation *Robotics and Autonomous Systems*, vol. 21, pp. 177-189, 1997.
- [50] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 1992. Addison-Wesley, Reading, MA.