Evolutionary Learning of Control and Strategies in Robot Soccer

Peter James Thomas, B.Eng(Hons)

A thesis submitted for the Degree of Doctor of Philosophy

Central Queensland University

James Goldston Faculty of Engineering and Physical Systems

School of Advanced Technology and Processes

25th July 2003

Abstract

Robot soccer provides a fertile environment for the development of artificial intelligence techniques. Robot controls require high speed lower level reactive layers as well as higher level deliberative functions.

This thesis focuses on a number of aspects in the robot soccer arena. Topics covered include boundary avoidance strategies, vision detection and the application of evolutionary learning to find fuzzy controllers for the control of mobile robot.

A three input, two output controller using two angles and a distance as the input and producing two wheel velocity outputs, was developed using evolutionary learning. Current wheel velocities were excluded from the input. The controller produced was a coarse control permitting only either forward or reverse facing impact with the ball. A five input controller was developed which expanded upon the three input model by including the current wheel velocities as inputs. The controller allowed both forward and reverse facing impacts with the ball.

A five input hierarchical three layer model was developed to reduce the number of rules to be learnt by an evolutionary algorithm. Its performance was the same as the five input model.

Fuzzy clustering of evolved paths was limited by the information available from the paths. The information was sparse in many areas and did not produce a controller that could be used to control the robots.

Research was also conducted on the derivation of simple obstacle avoidance strategies for robot soccer. A new decision region method for colour detection in the UV colour map to enable better detection of the robots using an overhead vision system. Experimental observations are given.

Publications

1. P. J. Thomas & R. J. Stonier, Evolutionary Learning of a 5I2O Fuzzy Controller Including Wheel Lift Constraint, *IEEE The Second International Conference on Computational Intelligence, Robotics and Autonomous Systems* (CIRAS2003), Singapore, 2003.

2. P. J. Thomas & R. J. Stonier, Hierarchical Fuzzy Control in Robot Soccer using Evolving Algorithms, *IEEE Congress on Evolutionary Computation* (CEC2003), Canberra, Australia, pp. 2434–2440, 2003.

3. P. J. Thomas & R. J. Stonier, Using Evolved Paths for Control in Robot-Soccer, *Computational Intelligence in Robotics and Automation* (CIRA-2003), Kobe, Japan, pp. 735–740, 2003.

4. P. J. Thomas & R. J. Stonier, Fuzzy Control in Robot-Soccer, Evolutionary Learning in the First Layer of Control, Journal of Systemics, Cybernetics and Informatics, Vol. 1, No. 1, 2003.

5. P. J. Thomas & R. J. Stonier, Evolutionary Learning of Fuzzy Control in Robot-Soccer, Book Chapter in M. Mohammadian (ed) *Computational Intelligence in Control*, Idea Publishing, USA, pp. 88-103, 2003.

6. P. J. Thomas & R. J. Stonier & P. J. Wolfs, Robustness of Colour Detection for Robot Soccer, *The Seventh International Conference on Control,*

Automation, Robotics and Vision (ICARCV'2002), Singapore, 2002.

7. P. J. Thomas & R. J. Stonier, Fuzzy Control in Robot-Soccer, Evolutionary Learning in the First Layer of Control, *Proceedings of the 6th World Multi-conference on Systemics, Cybernetics and Informatics* (SCI'2002), Florida, July, Vol. XII, pp. 181-186, 2002.

8. P. J. Thomas & R. J. Stonier, Evolutionary Learning of Fuzzy Control in Robot-Soccer, *Conference on Computational Intelligence for Modelling, Control and Automation* (CIMCA'2001), ISBN 0-85889847-0 on CD, M. Mohammadian (Ed), Las Vegas, pp 494-503, 2001.

9. X. Mao & R. J. Stonier & P. J. Thomas, An Effective Fuzzy Image Filter, *International Conference on Computational Intelligence for Modelling, Control and Automation* (CIMCA'2001), On CD, ISBN 0-85889847-0, M. Mohammadian (Ed), Las Vegas, pp 280-285, 2001.

10. P. J. Thomas & G. Springer & R. J. Stonier & P. J. Wolfs, Boundary-Avoidance and Attack Strategy in Robot-Soccer, *FIRA Robot World Cup France '98 Proceedings*, ISBN 89-7163-102-3, La Cité des Sciences et de l'Industrié Paris, France, pp 27-32, 1998.

Contents

1 Introduction **1**

1.1 Artificial Intelligence (AI) 2

1.2 Sensory Perception 3

1.3 Decision Making Techniques 4

1.4 Robot-Soccer Regulations 6

1.4.1 Communication 9

1.5 Software 9

1.6 Curvilinear Trajectory Control 9

1.7 Identification of Team Robot Colours 16

1.8 Towards More Intelligent Control Systems 19

1.9 Fuzzy Logic 21

1.9.1 Self-Organising Fuzzy Logic Controls 25

1.9.2 Hierarchical Fuzzy Logic Controllers 26

1.10 Evolutionary Learning of Fuzzy Control 28

1.10.1 Key Objectives 28

1.10.2 Evolutionary Algorithms 29

1.10.3 Learning the Knowledge Base 30

1.11 Scope 34

2 Fuzzy Logic **36**

2.1 Introduction 36

2.2 Review of Classical Set Theory 37

2.3 Fuzzy Sets 40

2.3.1 Zadeh Notation 43

2.3.2 Probability versus Fuzzy Sets 45

2.3.3 Some Basic Concepts 45

2.3.4 Basic Operations on Fuzzy Sets 47

2.3.5 Fuzzy Complement 49

2.3.6 Fuzzy Union: S-norm 50

2.3.7 Fuzzy Intersection: T-norm 52

2.4 Relations 53

2.5 Fuzzy relations 56

2.6 Projections and Cylindrical Extensions 58

2.7 Composition of Fuzzy Relations 64

2.8 Extension Principle 66

2.8.1 Linguistic Hedges 68

2.9 Fuzzy If-Then Rules 70

2.9.1 Fuzzy Propositions 73

2.9.2 Interpretation of Fuzzy If-Then Rules 77

2.9.3 Local and Global Implication 80

2.9.4 Mamdani Implications 81

2.10 Inference 83

2.10.1 Compositional Rule of Inference 85

2.11 Generalised Inference Rules 87

2.11.1 Generalised Modus Ponens 87

2.11.2 Generalised Modus Tollens 88

2.11.3 Generalised Hypothetical Syllogism 89

2.12 Fuzzy Rule Base and Fuzzy Inference Engine 91

2.13 Fuzzy Rule Base Structure 92

2.13.1 Properties of the Fuzzy Rule Base 93

2.14 Fuzzy Inference Engine 95

2.14.1 Compositional Based Inference 95

2.14.2 Individual Rule Based Inference 98

2.14.3 Some Inference Engines 99

2.15 Fuzzifiers and Defuzzifiers 104

2.16 Fuzzifiers 105

2.16.1 Singleton Fuzzifier 105

2.16.2 Gaussian Fuzzifier 107

2.16.3 Triangular Fuzzifier 111

2.17 Defuzzifiers 113

2.17.1 Centre Average Defuzzifier 114

2.17.2 Defuzzifier Comparison 117

3 Genetic/Evolutionary Algorithms 118

3.1 Introduction 118

3.2 Genetic Algorithms 121

3.2.1 Nomenclature 121

3.2.2 Data Structures 124

3.2.3 Selection 125

3.2.4 Crossover 126

3.2.5 Mutation 127

3.2.6 Elitism 128

3.2.7 Mapping Objective Functions to Fitness Coding 128

3.2.8 Genetic Algorithm Convergence 130

3.3 De Jong and Function Optimisation 133

3.4 Genetic Algorithms and Parallel Processors 136

3.5 Evolutionary Algorithms 140

3.6 Constraints 143

4 Design of Fuzzy Systems **146**

4.1 Introduction 146

4.2 Table Look-Up Scheme 146

4.3 Gradient Descent Training 147

4.4 Nearest Neighbourhood Clustering 151

4.5 Evolutionary Learning of Fuzzy Systems 152

4.5.1 Genetic Fuzzy Systems 152

4.6 Fuzzy Amalgamation 154

5 Robot Modelling 157

5.1 Introduction 157

5.2 Robot Parameters 158

5.3 Kinematics 162

5.4 Constraints 164

5.4.1 Time Constraint 165

5.4.2 Boundary Constraint 165

5.4.3 Robot to Ball Impact Constraint 166

5.4.4 Wheel Lift Constraint 169

5.5 Cartesian Coordinate System 172

5.5.1 Fuzzy Control using Cartesian Coordinates 172

5.6 Relative Coordinate System 174

6 Boundary Avoidance 178

6.1 Introduction 178

6.2 Basic Strategy Definitions 179

6.3 Boundary Avoidance 180

6.3.1 Stop and Turn 180

6.3.2 Artificial Potential Fields (APF) 181

6.3.3 Angle Limiting Functions 181

6.3.4 Proportional Cosine Control 184

6.4 Modified Attack Strategy 187

6.4.1 Strategy Implementation 189

6.5 Multi-Agent Strategies 190

6.6 Comments 191

7 Three Input Two Output (3I2O) Control 193

7.1 Introduction 193

7.2 Fuzzy Control System Design 194

7.3 Evolutionary Learning 197

7.4 Discussion 200

7.4.1 Results: Fuzzy Control at a Distance 201

7.4.2 Learning of Fuzzy Control Close to the Ball 204

7.5 Comments 206

8 Five Input Two Output (5I2O) 224

8.1 Introduction 224

8.2 Fuzzy Control System Design 226

8.3 Evolutionary Learning 229

8.4 Discussion 232

8.5 Comments 240

9 Hierarchical Modelling 249

9.1 Introduction 249

9.2 Fuzzy Control System Design 251

9.2.1 First Fuzzy System 251

9.2.2 Second Fuzzy System 253

9.2.3 Third Fuzzy System 255

9.2.4 System Design 256

9.3 Evolutionary Learning 258

9.4 Discussion 259

9.5 Comments 268

10 Evolving Velocity Profiles 277

10.1 Introduction 277

10.2 Evolutionary Learning of Paths 279

10.3 Discussion 284

10.4 Fuzzy Clustering 286

10.5 Comments 294

11 Colour Space **296**

11.1 Introduction 296

11.2 YUV 298

11.3 Colour Detection of the Ball 300

11.4 Pie Slice Decision Regions 301

11.5 Experimental Observations 305

11.6 Comments 305

12 Conclusions and future directions **307**

List of Figures

1.1 Fields of Artificial Intelligence [2] 3

1.2 Definition Categories of AI [2] 3

1.3 MIROSOT Configuration 1.3 7

1.4 MIROSOT Field Detail 1.3 8

1.5 Robot and Ball 8

1.6 Ball Attacking—Case 1 11

1.7 Position Error Caused by Vision Acquisition and Processing Delays 12

1.8 Robot Arc Ability 13

1.9 Ball Attack Strategy with Lead Transfer Function 14

1.10 Lead Transfer Function in Time Domain 14

1.11 Prediction Error caused by Position Errors 15

1.12 Boundary Avoidance by Piece-Wise Linear Profile 16

1.13 Basic Structure of a FLC 23

1.14 Basic Structure of a SOFLC 26

1.15 GA Fuzzy Rule Generator Architecture 26

2.1 Characteristic Functions for A', $A \cap B$ and $A \cup B$ given A and B 39

2.2 Membership Function for "Middle Age" 40

2.3 Possible Membership Functions "Close to or Equal to Zero" . . . 42

2.4 Centres of Fuzzy Sets A, B, C, and D 46

2.5 Fuzzy Membership of $A \cap B$, $A \cup B$ and A' 48

2.6 S-norms: (a) $s_{\max}(a, b) = \max(a, b)$, and (b) $s_{ab}(a, b) = a + b - ab$ 52

2.7 T-norms: $t_{\min}(a, b) = \min(a, b)$, and $t_{ap}(a, b) = ab$ 53

2.8 Unit Disc Projections and Extensions 59

2.9 Extension Principle: Ambiguity in Defining Membership $\mu_B(v)$ . . 68

2.10 Simulated Robot in Workspace 74

2.11 Truth Table for $p \wedge (p \rightarrow q) \rightarrow q$ 83

2.12 Truth Table for $\neg q \wedge (p \rightarrow q) \rightarrow \neg p$ 84

2.13 Truth Table for $(p \rightarrow q) \wedge (q \rightarrow r) \rightarrow (p \rightarrow r)$ 85

2.14 Components of a Pure Fuzzy System 91

2.15 A MIMO System Decomposed into MISO Systems 92

2.16 Membership Functions with $\mu_{A_p}(u_p^*) = 0.7$ 102

2.17 Membership Functions with $\mu_{A_p}(u_p^*) = 0.3$ 103

2.18 Components of Fuzzy System with Fuzzifier and Defuzzifier 104

2.19 Graphical Illustration of the Centre of Gravity Defuzzifier 114

2.20 Graphical Illustration of the Centre Average Defuzzifier 115

2.21 Illustration of Continuity Difficulties with the Maximum Defuzzifier116

2.22 Example Illustrating Problem with the Maximum Defuzzifier . . . 117

3.1 Basic Genetic Algorithm Structure 123

3.2 One-Point Crossover 126

3.3 Two-Point Crossover 127

3.4 GA with Full Replacement Policy 130

3.5 GA using Parallel Section for Fitness Evaluation 138

3.6 Parallel GA's Solving Velocity Profiles 139

5.1 Curvilinear Formulae Symbols 163

5.2 Ball Impact Symbols 166

5.3 Wheel Lift Diagram 169

5.4 Wheel Lift Diagram with $L = 68.5mm$ and $h = 37mm$ 171

5.5 Wheel Lift Diagram with $L = 75mm$ and $h = 12mm$ 171

5.6 Membership Sets for Cartesian Fuzzy System 173

5.7 Relative Coordinate Parameters 175

5.8 Robot with relative coordinate system at centre 175

6.1 Modules and dependencies 179

6.2 Piece-Wise Linear Profile 182

6.3 Boundary Avoidance 183

6.4 Cosine Control 185

6.5 Cosine Control Path 185

6.6 Ball Attack Strategy 188

6.7 Quadrant Division of Field 191

7.1 Three Input Two Output Model 194

7.2 Fuzzy Input Sets 196

7.3 All Medium Distance Paths 202

7.4 Evolutionary Statistics Average of Minimums over 11 runs 205

7.5 Evolutionary Statistics Average of Averages over 11 runs 206

7.6 All Short Distance Paths 207

7.7 Medium Distance Paths: c077–c104 210

7.8 Medium Distance Paths: c105–c125 211

7.9 Medium Distance Velocity Profiles: c077–c083 212

7.10 Medium Distance Velocity Profiles: c084–c090 213

7.11 Medium Distance Velocity Profiles: c091–c097 214

7.12 Medium Distance Velocity Profiles: c098–c104 215

7.13 Medium Distance Velocity Profiles: c105–c111 216

7.14 Medium Distance Velocity Profiles: c112–c118 217

7.15 Medium Distance Velocity Profiles: c119–c125 218

7.16 Individual short distance paths 219

7.17 Short Distance Velocity Profiles: c000–c006 220

7.18 Short Distance Velocity Profiles: c007–c013 221

7.19 Short Distance Velocity Profiles: c014–c020 222

7.20 Short Distance Velocity Profiles: c021–c027 223

8.1 Five Input Two Output Model 225

8.2 Fuzzy Input Sets 226

8.3 Average of averages over 11 runs for 50 000 generations 233

8.4 End point statistics of $\alpha_1 T_1$ term 234

8.5 End point statistics of $\alpha_2 T_2$ term 235

8.6 Long Distance Path from Left 236

8.7 Medium Distance Path from Above 237

8.8 Medium Distance Path from Bottom Right 237

8.9 Long Distance Path from Right 238

8.10 Table of Results c0056–c0083 242

8.11 Table of Results c0560–c0587 243

8.12 Table of Results c1148–c1175 244

8.13 Table of Results c2072–c2099 245

8.14 Table of Results c2324–c2351 246

8.15 Table of Results c2968–c2995 247

8.16 Table of Results c3864–c3891 248

9.1 Hierarchical Model 250

9.2 Membership Sets Defined for the First Hierarchical Layer 252

9.3 Membership Sets Defined for the Second Hierarchical Layer 254

9.4 Membership Sets Defined for the Third Hierarchical Layer 256

9.5 Average of averages over 11 runs for 50 000 generations 260

9.6 End point statistics of $\alpha_1 T_1$ term 261

9.7 End point statistics of $\alpha_2 T_2$ term 262

9.8 Long Distance Path from Left 263

9.9 Medium Distance Path from Above 264

9.10 Long Distance Path from Bottom Right 265

9.11 Long Distance Path from Right 265

9.12 Table of Results c0056–c0083 270

9.13 Table of Results c0560–c0587 271

9.14 Table of Results c1148–c1175 272

9.15 Table of Results c2072–c2099 273

9.16 Table of Results c2324–c2351 274

9.17 Table of Results c2968–c2995 275

9.18 Table of Results c3864–c3891 276

10.1 Path Plots from Configuration c08400(550, 550, 7π/9, 0, 0) 287

10.2 Path Plots from Configuration c08490(550, 650, 0, 54, 54) 287

10.3 Path Plots from Configuration c08900(550, 750, 4π/9, 27, 0) . . . 288

10.4 Path Plots from Configuration c09187(550, 850, 2π/9, -27, 0) . . . 289

10.5 Path Plots from Configuration c11587(650, 550, 16π/9, 78, 78) . . 290

10.6 Path Plots from Configuration c14587(750, 550, 4π/3, 103, -103) . 291

10.7 Path Plots from Configuration c14687(750, 750, 0, 56, 78) 292

10.8 Path Plots from Configuration c16887(850, 450, 2π/3, -78, 56) . . 293

10.9 Path Plots from Configuration c17087(850, 550, 0, -27, 54) 294

10.10 Path Plots from Configuration c17287(850, 550, 10π/9, 54, 0) . . 295

10.11 Path Plots from Configuration c18099(850, 850, $2\pi/9$, -54, -27) . . 295

11.1 RGB Colours Mapped on UV Coordinates 299

11.2 Averaged Pixels caused by Red-Yellow Boundary 300

11.3 Averaged Colours on UV Map 301

11.4 RGB Rectangle in YUV Space 302

11.5 UV Rectangle over YUV 303

11.6 Pie Segment 304

11.7 Screen Capture of Window using Pie Segment 306

12.1 Three Input Two Output Model 310

12.2 Five Input Two Output Model 311

12.3 Hierarchical Model 312

List of Tables

2.1 Fuzzy Identities [3] 49

2.2 Truth Table for $p \rightarrow q$ 77

2.3 Truth Table for $\neg p \wedge q$ and $(p \wedge q) \vee \neg p$ 78

2.4 Intuitive Reasoning in Generalised Modus Ponens 87

2.5 Intuitive Reasoning in Generalised Modus Tollens 89

2.6 Intuitive Reasoning in Generalised Hypothetical Syllogism 90

2.7 Comparing Defuzzifiers to their Criteria 117

3.1 Biological and GA Nomenclature Comparison [4] 122

3.2 Local Maxima of $f_1(x) = 1 - \sin(\pi x)\cos(11\pi x), x \in (0, 1)$ 124

5.1 Experimental Results of Robot Parameter Test 159

5.2 Angular robot error using $\overline{BG}$ and acceptable ball to goal trajectory angle 177

7.1 Medium Path Final Robot Statistics 208

7.2 Medium Path Final Robot Statistics (continued) 209

7.3 Short Paths Robot Statistics 209

8.1 Statistics of 5I2O fuzzy controller 234

9.1 Statistics of Hierarchal fuzzy controller 260

11.1 RGB Colour Mapping onto YUV Coordinates 299

List of Symbols

$\emptyset$	empty set
$\subset$	subset
$\supset$	superset
$\subseteq$	subset or equal
$\supseteq$	superset or equal
$\cap$	intersection
$\cup$	union
$\mid$	such that
$'$	compliment
$\forall$	for all
$\exists$	there exists
ℓ	length
$\in$	belongs to or is a member of
$\notin$	is **not** an element of
μ	degree of membership
$\overline{x}$	mean
σ	standard deviation
$\wedge$	and
$\vee$	or
$\neg$	not
Z	set of Natural numbers
$\rightarrow$	implies or mapping
$\leftrightarrow$	equivalent
b	bit
B	byte
ϕ	angle
x	x-coordinate or independent variable
y	y-coordinate
m	gradient
r	radius
P	population
t	time
U	universe

$\sum$	summation or union in Zadeh notation
$\int$	integral or continuous counterpart of $\sum$ in Zadeh notation
$\prod$	product
$\oplus$	bounded sum
$\ominus$	bounded difference
$\overline{A}$	not A
t	t-norm
s	s-norm
$\times$	cross product
$\circ$	composition
T	true
F	false
$\underset{\sim}{x}$	vector x
$\overline{x}$	average of x
∂	partial differential
i, j, k	index variables
v	linear or tangential velocity
ω	angular velocity
d	distance
m	mass
g	gravational constant
K_p	proportional gain
K_a	angular gain
Ω	angle
Δ	change in
F	fuzzifier
DF	defuzzifier

Acknowledgements

I would like to acknowledge the support of my supervisors: A/Professor P. J. Wolfs, School of Advanced Technology and Processes, and A/Professor R. J. Stonier, School of Mathematics and Decision Sciences; Central Queensland University. I would like to thank my wife, Sandra, for her support.

Declaration

This thesis contains no material that has been accepted for the award of another degree. Furthermore, to the best of my knowledge, this thesis does not contain any material previously published or written by another person, except where due reference is made in the text.

.....

P J Thomas

Chapter 1

Introduction

Robot soccer has been promoted as an environment that offers rich opportunities for the development of a broad spectrum of artificial intelligence and control system techniques. An international competition, based loosely on the concept of the Soccer World Cup, has been promoted for several years to encourage postgraduate study in these areas [1].

This thesis focuses on the author's experiences within this challenging field. While the thesis primarily focuses on evolutionary fuzzy controls, some useful contributions have also been made in robot vision. The author has also been responsible for the development of a competitive team, and fielding that team at international competitions. Some details relating the robot hardware systems will be included to give a holistic overview of the entire development.

The remainder of the chapter will briefly introduce artificial intelligence concepts, robot soccer, and outline the thesis. Chapter 2 gives a detailed overview of fuzzy logic systems. An overview of genetic and evolutionary algorithms are shown in Chapter 3. The use of genetic/evolutionary algorithms to learn a fuzzy rule base

is discussed in Chapter 4. Modelling of the soccer-robot characteristics is covered in Chapter 5. Chapters 6 to 10 are based on papers presented at conferences. A benchmark that describes the earliest work is presented in Chapter 6. Evolving a fuzzy rule base for a three input, two output (3I2O) robot path controller is shown in Chapter 7. A five input, two output (5I2O) fuzzy robot path controller is developed in Chapter 8. A five input, two output, three layer hierarchical fuzzy controller is developed in Chapter 9. The developing of a fuzzy rule base from evolutionary robot velocity profiles is presented in Chapter 10. The classification of robot colour patches from video composite input is shown in Chapter 11. Concluding remarks are given in Chapter 12.

The remainder of this chapter will give a compact overview of artificial intelligence, the issues pertinent to robot soccer and an overview of some of the concepts of fuzzy and evolutionary control.

1.1 Artificial Intelligence (AI)

Over the past 2000 years, philosophers have tried to understand and model truths or principles governing knowledge and being. Psychologists have provided tools to investigate the human mind and a scientific language to express the results. Mathematicians have provided formal theories of logic, probability, decision making and computation over the past 400 years. Linguists have provided theories of language structure and meaning. Computing technology has provided the means to apply AI concepts. Figure 1.1 shows some of the many sub-fields of AI.

AI [2] is an attempt to emulate human "intelligence", as illustrated in Figure 1.2, by thought processes or behaviour. These categories can also be broken into a

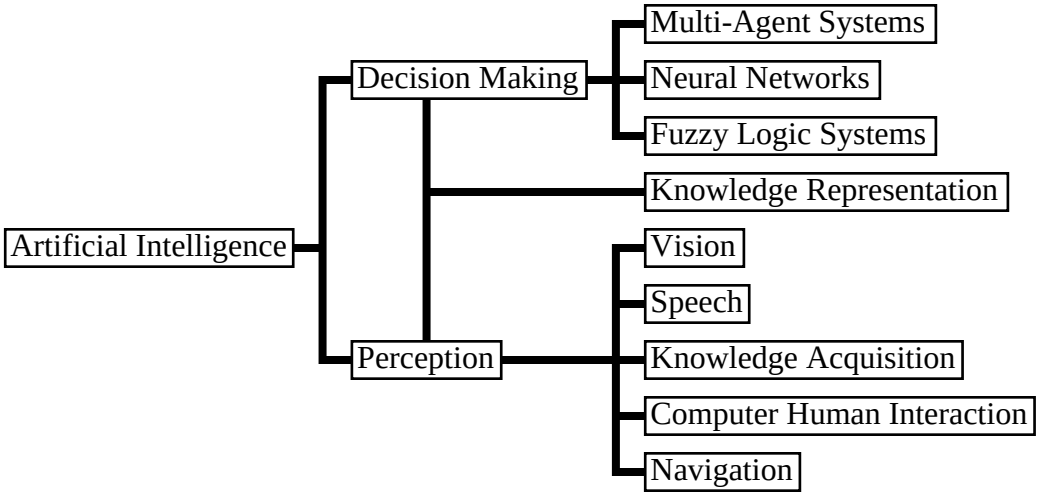


Figure 1.1: Fields of Artificial Intelligence [2]

measure of human performance and rationality. It is estimated that over 100 000 rules are required to simulate simple intelligent behaviour [5].

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

Figure 1.2: Definition Categories of AI [2]

AI often relies on perception of the environment through sensors. An overview of the available sensor technologies is now presented.

1.2 Sensory Perception

Lately, there is a growing trend of AI techniques used in real applications. In order for AI techniques to operate in real applications, measurements must be acquired from the environment. Some of the available sensors used to measure environmental quantities are:

- switches
- strain gauges
- optical sensors
- accelerometers
- gyroscopes
- proximity sensors
  - inductive devices
  - capacitive devices
  - resistance devices
  - magnetic sensors
- pressure sensors
  - piezoelectric
  - barometric devices
  - resistance devices
  - capacitive devices
- radar devices
- sonar devices
  - sound
  - light
- vision sensors.

A discussion of sensor devices is beyond the scope of this thesis and is covered extensively in a large number of books in AI, robotics, engineering texts and manufacturer data sheets.

1.3 Decision Making Techniques

Several AI programmes have been designed to solve specific problems. Game playing for example has received much attention because the rules are well defined and the environment of the game is well constrained. Simple AI [2] problems such as these can be solved using decision trees. Decision trees provide a hierarchy structure with all possible solutions from each node. Othello, tic-tac-toe and

draughts have been solved by decision trees. Another solution is to compute all possible outcomes from the current state of play.

Chess has a high combination of moves that prohibits the use of decision trees, but has simple calculations for moves. In 1997, Deep Blue became the first computer to beat a grand master level chess player, Kasparov. Chess playing programmes have been improving since then, and so have the grand master's abilities. Grand master's have developed methods that cause the search path for calculating the next move to be very long. This prevents the computer from being the outright winner to date. Computer chess programmes have their weakness and strengths as do human players. One of the methods used in developing intelligent systems such as playing chess is the expert system.

Expert systems [2] were developed to simulate an expert's knowledge of a problem. They consist of three major components: a knowledge base, a data base and a rule interpreter. The knowledge base contains the inference rules used during the reasoning process usually by if-then-else (Production Systems) coding. Reasoning is aided by propositional and predicate calculus. The data base contains the context or facts available at any given time, such as a mammal: has hair, gives milk. Hierarchical tree structures are commonly used to represent the problem context. The rule interpreter (inference engine) guides the reasoning process through the knowledge base by attempting to match the facts in the data base to the rule conditions, including resolving rule conflict.

Almost all real world problems are extremely non-linear. Linearisation has helped to simplify problems for modelling purposes. This technique simplifies the mathematics of the problem and in general allows only a small part of the problem to be solved.

Artificial Neural networks (ANNs) use structures that mimic the functioning of the brain [3, 6]. These are powerful tools, but issues of training are complex. Recent advancements in extracting rules from ANNs are shown in [7]. A similar and comparative non-linear system is fuzzy systems.

Fuzzy Logic Systems (FLSs) are based on the linguistic language that people use [5, 3, 8]. The spoken language is vague, but contains enough information for another person to interpret. Terms are used that cannot be given an exact number for a computer programme to use. An expression has different meaning in dissimilar environments.

This thesis will concentrate on the application of evolving a fuzzy system to control a soccer playing robot. The key features of fuzzy control will be developed in depth in subsequent chapters. An overview of the target hardware is now considered.

1.4 Robot-Soccer Regulations

A MIROSOT team was purchased from Micro Adventure to facilitate testing of AI techniques on a physical system in a competitive environment. The game is played by two opposing teams configured as Figure 1.3.

The CQU robot-soccer team has participated at competitions in Korea - joint team (1996), Korea - joint team (1997), Alaska (1998), Paris (1998) and Rockhampton (2000).

Details of up-to-date rules and regulations for the Mirosot competition may be

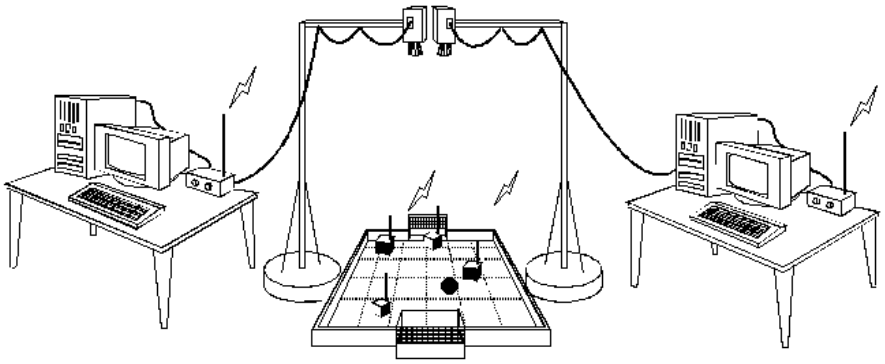


Figure 1.3: MIROSOT Configuration 1.3

found on the Federation of International Robot Soccer Association (FIRA) [1]. Some basic principles of the robot soccer game that were defined at the time the Central Queensland University team played in competition are now presented.

Robot soccer is played on a rectangular field with dimensions and marking as shown in Figure 1.4.

The basic control software purchased with the robots used simple strategies producing travel along straight line segments. The software would calculate a desired point to move to, rotate the robot to face that point, move off to that point in a straight line and stop. "Stop and Turn" algorithms such as this do not take momentum into consideration. As a consequence, the robot passes the stop position and often collides with another robots or the perimeter wall. Another disadvantage of "Stop and Turn" is loss of ball control by the robot. This motion control was replaced by a curvilinear trajectory control in the early development of the CQU team as described briefly in a following section.

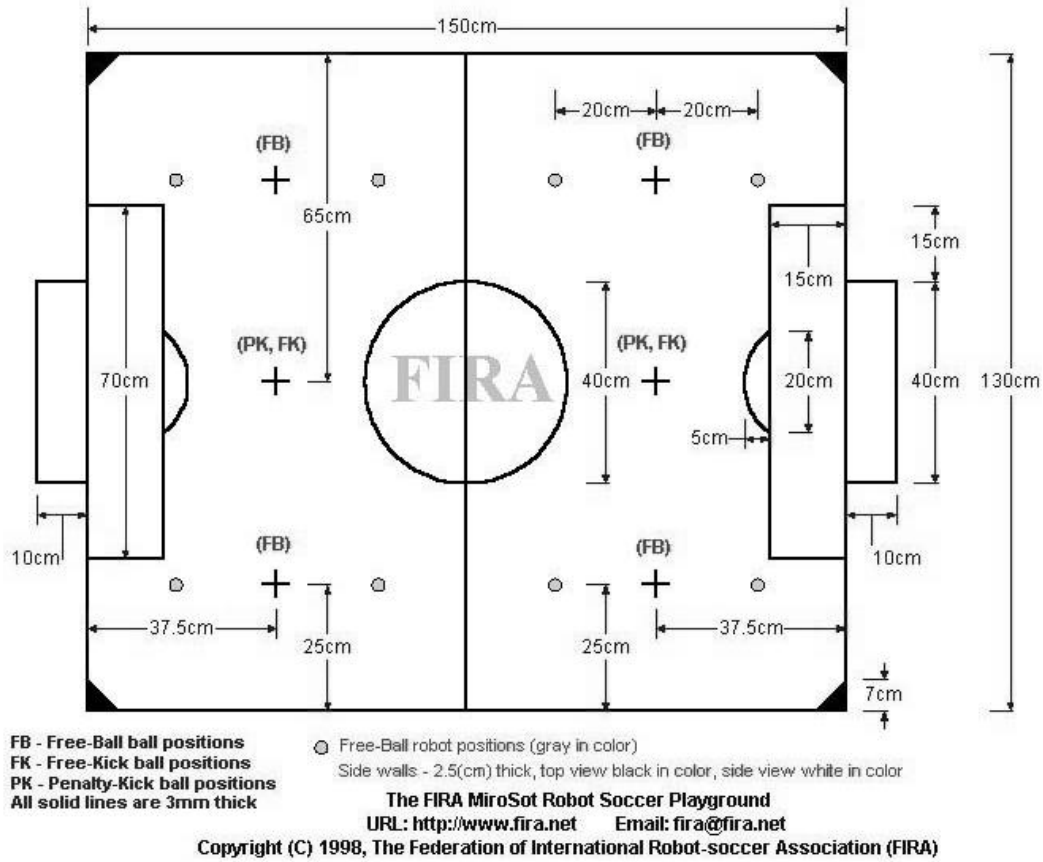


Figure 1.4: MIROSOT Field Detail 1.3

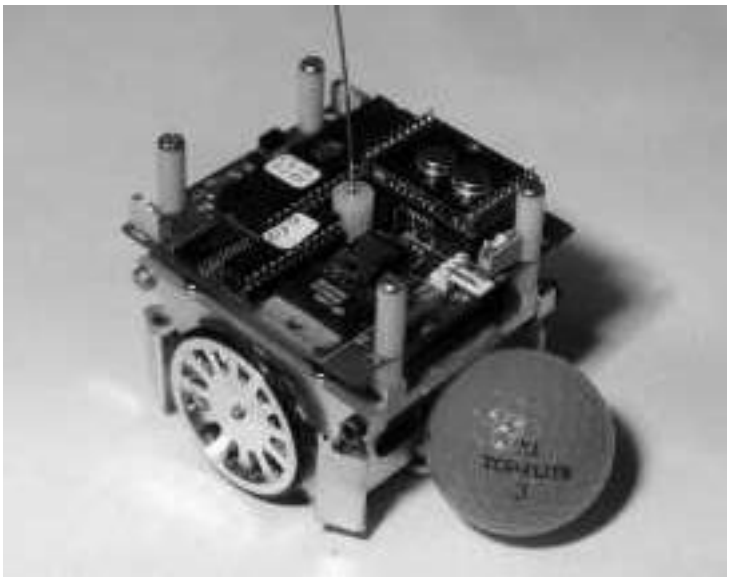


Figure 1.5: Robot and Ball

1.4.1 Communication

The current radio communications is by a commercially available Radiometrix module. The module is capable of transmitting 20kbps with Manchester encoding. Two bytes (velocity left and velocity right) are transmitted for each of three robots. Transmitting six bytes at 9600bps takes 6.25ms. This corresponds to the ball traversing 31mm at the peek speed of 5ms^{-1}.

To reduce positional error at 5ms^{-1}, the Radiometrix communication should be upgraded to a module capable of 60kbps.

1.5 Software

Control of the robots is performed by a centralised programme hosted on a computer. The basic control software purchased with the robots was written in Microsoft Visual C++ version 1.52. The programme was written using the Microsoft Foundation Class library (MFC) [15]. A full description is found in the Microsoft MFC library reference manuals [16, 17, 18, 19]. The use of the MFC library assumes a background knowledge in Application Programming Interface (API) programming [20].

1.6 Curvilinear Trajectory Control

At the Robot Soccer Competition at KAIST in Korea, 1996, [21], the Central Queensland University (CQU) team presented a paper which described a circular

path planning algorithm to attack the ball. The algorithm modified the actual direction vector of the robot to a desired heading vector by simple geometric calculations. Figure 1.6 illustrates this simple method of generating a circular trajectory. Except in the parallel case, every pair of tangent lines on a circle intersect at some point H. An isosceles triangle is formed with the tangent lines BH and RH. An important property of an isosceles triangle is that two angles $\angle HBR$ and $\angle HRB$ are equal. Let G represent the goal and B represent the ball. A line passing through GB is the desired direction for the robot R to move the ball towards the goal. The angle of the GB line (ϕ_1) is calculated by the arctangent of ($\Delta y_{BG}/\Delta x_{BG}$). The angle of the BR line (ϕ_2) is calculated by the arctangent of ($\Delta y_{RB}/\Delta x_{RB}$). Using the property of an isosceles triangle, the angle $\angle BRH$ is $\phi_1 + \phi_2$. The line BR is at angle ϕ_2 to the horizontal. Equation 1.1 gives the desired angle for the robot to follow a curved path to the ball and be lined up to run the ball straight into the goal. The centre of the curve can be calculated by the intersection of the perpendicular of the GB line at B and the perpendicular of the HR line at R. The centre of curvature is not required in the curvilinear calculation.

$$\phi_3 = 180^\circ - \phi_1 - 2\phi_2 \tag{1.1}$$

For the case of the robot being on the goal side of the ball, Equation 1.1 directs the robot on a curve to push the ball away from the goal. Equation 1.2 prevents this by adding a constant angle to the line connecting the ball and robot until Equation 1.1 takes effect. Equation 1.2 always brings the robot around the right side of the ball before Equation 1.1 directs the robot into a curved path.

$$\phi_3 = \phi_2 + 180^\circ + \alpha \tag{1.2}$$

This scheme worked well in simulation [21], and in real application provided the

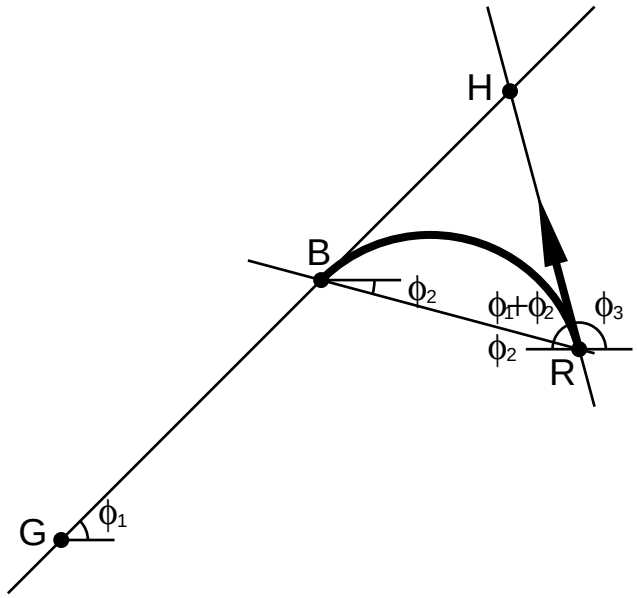


Figure 1.6: Ball Attacking—Case 1

speed of the robot was low. In the simulation small time steps were used to calculate the path of the robots. The robot soccer environment has large delays caused by vision acquisition and processing so when the robot is moving slowly, the step between vision acquisition and processing is small and the robot traverses as predicted. However, high robot speed results in a long travel distances and large positional errors. The robot either hits the ball on the corner of the robot, propelling the ball away, or the robot misses the desired position behind the ball and hunts around trying to get back to the required position. Figure 1.7 shows the effect of the robot travelling large line segments due to time delay caused by vision acquisition and processing.

The robot wheels are placed in a “wheel chair” arrangement. The “wheel chair” configuration gives the robot the ability to scribe any arc. Figure 1.8 shows the robot arcing capability including the capacity to pirouette. This inherent behaviour is best taken advantage of by developing equations that allow the robot to use the arcing ability.

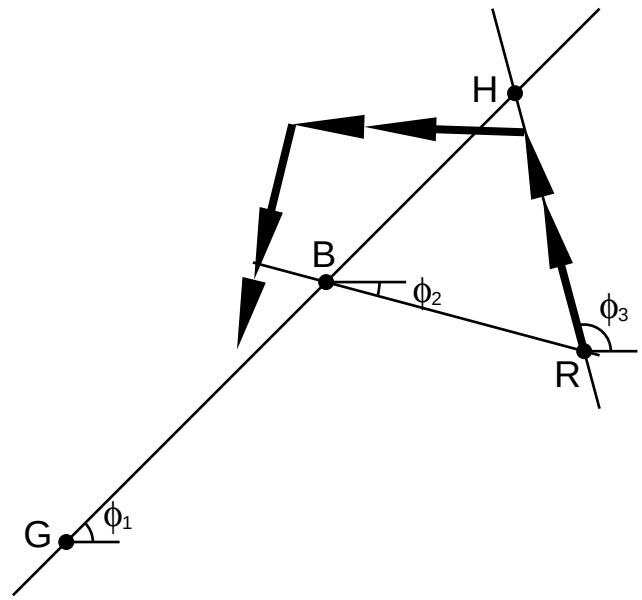


Figure 1.7: Position Error Caused by Vision Acquisition and Processing Delays

Circular paths around the ball are at times very large. Equation 1.1 makes a curved path to the ball depending on the tangent circle. Figure 1.9 shows some of the circular paths produced by Equation 1.1.

The attack the ball strategy was further modified for the Paris competition in 1998. The details of this research are more fully discussed in Chapter 6. A term coined as "lead transfer function" [23] (LTF) was used to reduce the arc that the robot traversed when attacking the ball. This term was used as the function resembled a lead transfer function, but is actuality is not.

Figure 1.9 shows the effect of applying the LTF to the ball attack strategy. The transfer function is used to add a proportion of the angle α to the heading angle to guide the robot to transverse a least distance such that the robot curves around behind the ball. To make the robot have a better chance of getting behind the ball, the point P_1 is placed a constant distance behind the ball. The fixed point P_1 caused a problem with different speeds of the approaching robot. The point

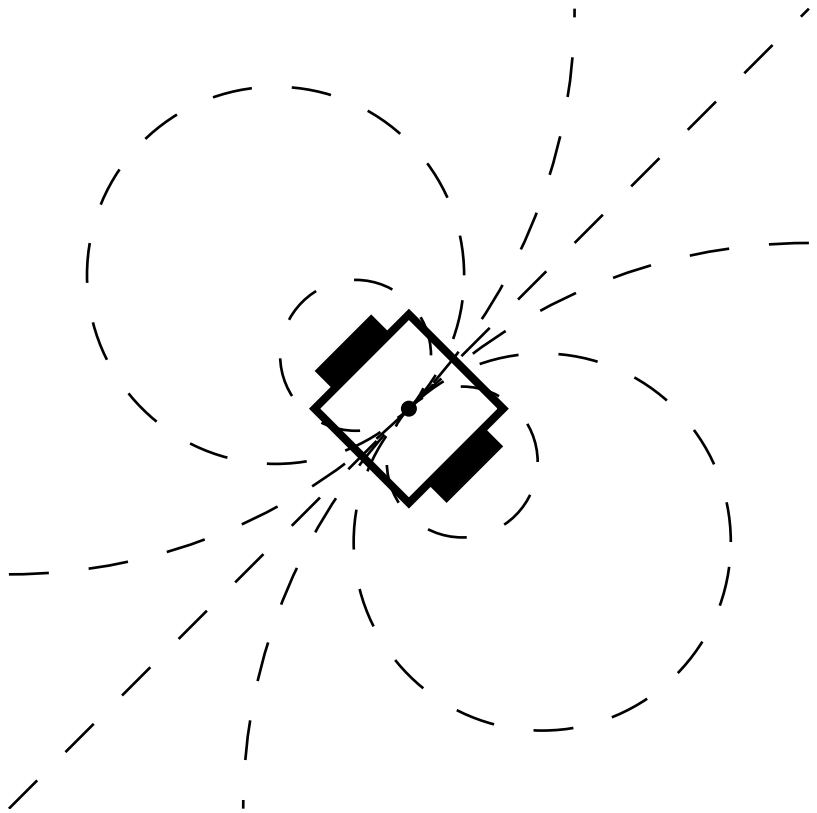


Figure 1.8: Robot Arc Ability

needed to be at the ball for low robot speed and a large distance away for high speed.

The lead transfer function is a high pass filter. A high pass filter in the s-domain is: $sT/(1 + sT)$. Figure 1.10 shows the function in the spatial rather than the usual time domain.

One possible solution to large errors caused by long time delays in the image capture and control would be to use prediction to find the position of the robot after a finite time delay. Prediction was used in the CQU team's robot soccer programme intended for competition in Paris. The problem with prediction is that it is highly sensitive to position error. If the object is moving fast, the position error causes a very large prediction error. If the object is moving slow,

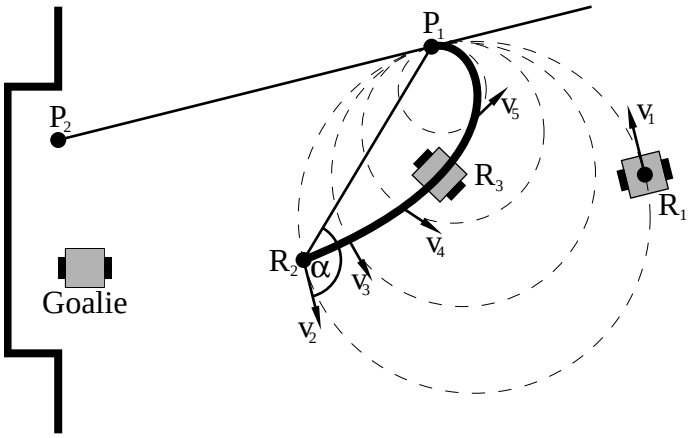


Figure 1.9: Ball Attack Strategy with Lead Transfer Function

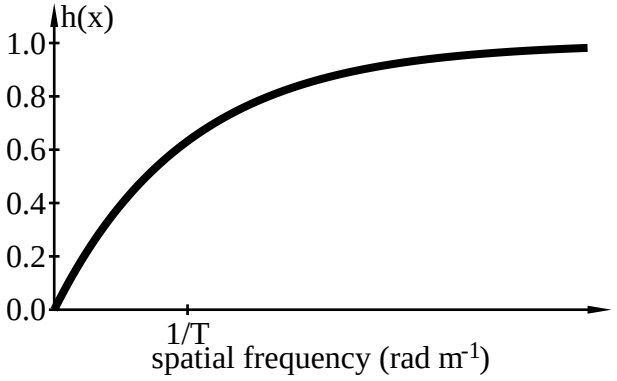


Figure 1.10: Lead Transfer Function in Time Domain

the prediction is more accurate.

However, other problems occur at high speed. Figure 1.11 shows the prediction area and range of prediction angles caused by position error. Prediction is almost impossible without precise position data.

The current status of robot control in multi-agent terms is in its infancy. All teams observed in top level MIROSOT competitions use a centralised agent playing several roles. Observations also indicate that all robot soccer teams in competition level are a reactive agent. A reactive agent produces reflex actions to environmental changes—such as follow the ball. Even the winner of the Paris

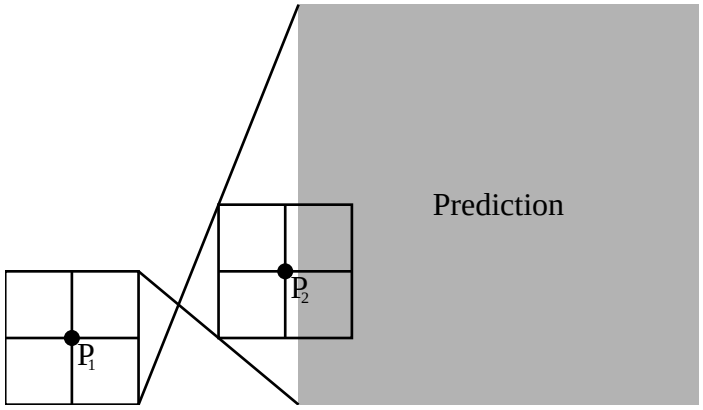


Figure 1.11: Prediction Error caused by Position Errors

competition was observed to be purely reactive. No teams showed deliberative control of their robots at the FIRA Robot World Cup in Paris.

The second strategy implemented for the Paris competition was boundary avoidance strategy. The robots had a habit of colliding with the wall due to the attack the ball strategy. To overcome this problem a potential field was placed near the boundary. The potential field theory was tested and found unsatisfactory due to the robot tending to meander down the wall. The meandering was caused by interactions between the potential field forcing the robot away from the wall and the robot controls directing the robot towards the wall. A piece-wise linear profile was used to limit the robot meandering. Figure 1.12 shows the piece-wise linear profile designed for the Paris competition.

The discussion above showed that there was a need for research into intelligent control algorithms which would deliver more precise control of the robot, especially in regions close to the ball. It has led to the fuzzy control research presented in this thesis. A related area is robot vision. Competition experience has shown this area has been a source of difficulty. Some identification issues are now considered.

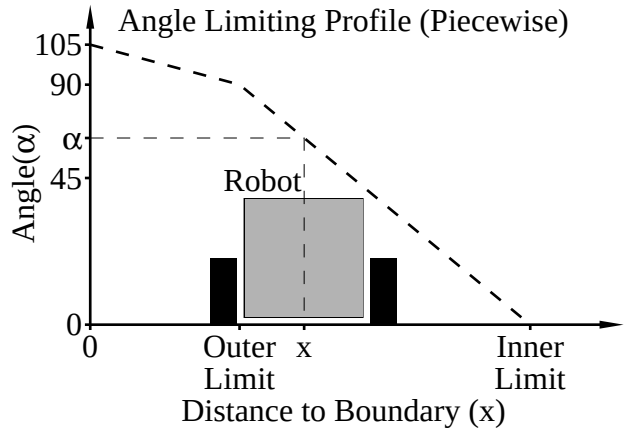


Figure 1.12: Boundary Avoidance by Piece-Wise Linear Profile

1.7 Identification of Team Robot Colours

The main loop for the robot system requires the following actions:

- Image capture,
- Image processing,
- Robot control, and
- RF Transmission.

A commercial video camera is generally used to capture the image. While these cameras are low cost, they are designed for television and lack resolution and image transfer speed. NTSC cameras are generally used as they have a higher frame rate of 30fps compared to the PAL 25fps.

There are 525 lines/frame at a rate of 29.97 frames/s for NTSC. Of the 63.5μs/line, $\approx$ 10.5μs is required for horizontal blanking, leaving 53.33μs/line of displayable

image. With NTSC video composite bandwidth of 6MHz, 53.33μs/line corresponds to 320 black and white cycles, giving $2 \times 320 = 640$ distinguishable horizontal black/white picture elements. Vertical blanking time is around 1.3ms, leaving approximately 480 lines/frame for the image. The DOOIN image capture card used provides YUV 4:1:1 video composite format, giving 160×240 UV chroma elements per field. Projecting the camera resolution onto the playing area equates to each chroma element being 11×5.5mm in size. YUV 4:2:2 can be obtained from a Philips video processing chip [24], giving chroma resolution 320×240 per field.

Using the chroma information at a rate of 60fps gives an effective resolution of 320×240 for odd and even fields. Projecting this resolution on the field requires pixel size to be 5.5mm giving image area of 1760×1320mm, provided that the system is capable of classifying all pixels covering a coloured object. A minimum sized robot team identification patch positioned square to pixels will contain a possible 36 chroma elements. The same colour patch rotated 45° will contain at most a possible $1+3+5+7+7+5+3+1 = 32$ chroma elements. Thirty-seven chroma elements cover the ball. The practical limit to identifying an object is a measure of how well the system can classify pixels correctly.

Image transfer speed of an NTSC commercial video camera is sixty fields per second (16.6ms). Assuming classification of objects, calculation of control and transmission to the robot takes 6.3ms, the ball moving at 5ms^{-1} traverses 115mm in 23ms. In camera terms, a ball moving in the horizontal frame of view, will traverse 15 chroma elements. Some video cameras allow integration times of 1ms. The ball will traverse less than one chroma element for a video camera designed for capturing high speed sports mode. Otherwise, the object in the image is a translucent streak which is very hard to classify when travelling at high speed.

Depending on the design of the video camera, the integration time could be as high as 1/30 seconds. Within a translucent streak of 30 chroma elements is not only hard to classify an object of length 6 chroma elements, but also difficult to determine position.

Delay times of 20ms make it impossible to control a robot travelling at 2.6ms^{-1}. A camera available from Dalsa (*htp://www.dalsa.com*), model DALSTAR 1M75-SA, is suitable for playing robot-soccer. This video camera transfers full frames at 75fps of 1 024 $\times$ 1 024 pixels (transfer data rate 2 $\times$ 40MHz). It also is capable of transferring sub-fields at 100 000fps. The data sheet does not specify the smallest sub-field, but an estimation is $\sqrt{80MHz/100000fps} \approx 28$; rounding to 2^n gives 32 $\times$ 32 pixels. Pixel size projected onto field is 1.5mm giving coverage of 1 536 $\times$ 1 536mm. Assuming integration time of 1ms , image transfer of seven sub-fields taking 1ms, transmission to the robots taking 6.25ms, a 5ms^{-1} object traverses 41.25mm. If the camera has a separate transfer area, the integration time and transfer time is 1ms. Improvement of transmission to the robots will give better system performance with this camera. If the capturing images to transferring to robot time is reduced to 2ms, there will be no need to estimate the position of the robot.

A good reference on image analysis is Gonzalez [25]. An excellent reference on the storage of images in computers is found in Levine [26].

CQU's team first actual competition was in the North American Championship, Anchorage Alaska during the World Automation Congress (WAC) in May 1998. This first competition showed some key problem areas to be solved. The most significant problem related to the data communication link which was prone to failure. This problem was rectified easily and was related to a coding issue.

The second problem was loss of identity of the robots. Each robot is identified by a team colour patch and a personal colour patch. For discussion sake, let the team colour be yellow, the personal colour of robot 1 be red, the personal colour of robot 2 be green and the personal colour of robot 3 be blue. The software constructs regions of colour blobs and calculates the geometric centroid of each colour. Robot centre is calculated by averaging the coordinates of the centres of the two colours. Loss of identification is caused by the software connecting a team colour, say yellow of robot 1, with the personal green colour of robot 2. The heading is calculated by adding 45° to the line joining the centres of the two colours. As a result, the position and heading of the robot is incorrectly identified. Attempts to rectify loss of identification by scanning methods consumed too much computation time and were discarded. The vision issues lead to the development of improved methods discussed in Chapter 11.

1.8 Towards More Intelligent Control Systems

In this thesis we are interested in the application of such modern artificial intelligence techniques to multi-robot systems, path planning and collision-avoidance problems.

Most teams now entering in the FIRA world cup and in the RoboCup competitions are using intelligent systems techniques ranging from neural networks [6], fuzzy logic [8], evolutionary learning [27], Q-learning [30], to multi-agent strategies [31].

The following papers show attempt to design more highly intelligent control systems [32], with: problem solving, planning, adaptation, self-organisation, cog-

nition, and other human operator characteristics. They begin with the more classical approaches of adaptive control and advance to control by modern artificial intelligence techniques which include neural networks, fuzzy logic control (FLC) and genetic or evolutionary algorithms (GA/EA):

- hierarchical coordination of dynamic control has been used for both problems [33, 34]
- artificial potential fields for avoidance in [35]
- potential functions for attraction to targets and avoidance of obstacles [36]
- growing polyhedral obstacles for planning collision-free paths [37]
- differential game theory (precursor to multi-agent strategy) applications in [38]
- Liapunov techniques for point mass and articulated robot arms in [39, 40]
- neural network optimisation [41]
- genetic algorithm application [42]
- evolutionary algorithms [43, 44]
- integration of fuzzy logic and genetic algorithm for adaptive learning in [45, 46, 47, 48, 49, 50, 51]
- and fuzzy logic controllers for motion planning of mobile robots [53].

The usual definition that an adaptive system is one that is capable of accommodating unpredictable environmental changes or disturbances (whether these arise within the system or external to it). The concept of adaptation is a fundamental

characteristic of all living organisms as they attempt to maintain physiological equilibrium in the midst of changing environmental conditions. Indeed, numerous definitions of adaptive control systems exist, (see Ogata [54] for a formal definition in control systems). Vagueness surrounds most definitions and classifications of adaptive systems due to the large variety of mechanisms by which adaptation may be achieved. Definitions also fail to differentiate between external manifestations of adaptive behaviour and the internal mechanisms used to achieve it. However, inclusion of adaptability into systems has great appeal to designers for versatility by accommodating environmental changes, to compensate for minor engineering errors and show robustness against minor system failures. The ultimate goal of high system reliability with minimum design appears achievable with the new control techniques.

Intelligent control systems self-organise, learn and adapt control laws or control rules to environmental changes. As such, these systems are said to be less deterministic than their classical control counterparts which are modelled by exact algebraic and differential equations. Complex systems do not have an exact modelling process, input and output data have a degree of uncertainty. This thesis will focus on fuzzy logic systems to the soccer robot control problem.

1.9 Fuzzy Logic

The fuzzy logic based approach to solving problems in control has been found to excel in those systems which are very complex, highly non-linear and with parameter uncertainty. This is certainly the case for the game of robot soccer. We will now present a very compact overview of fuzzy logic systems and the evolutionary methods that may be used to drive their adaptation to the robot

problem. This very compact overview will be significantly expanded in Chapters 2 and 3. A reader who finds this section rather too compact may refer to the following chapters.

We may view a fuzzy logic controller as a real-time expert system that employs fuzzy logic to analyse input to output performance. Fuzzy logic provides a method of conversion from/to crisp values to fuzzy values by fuzzification/defuzzification. This allows the main reasoning unit to use linguistic control strategy derived directly from expert knowledge. Inherent in the fuzzy system is a simple method to interrogate the system to answer the "why?" question in linguistic quantities. This is the main advantage over potential theory approaches and the application of neural networks. It is impossible to obtain a knowledge of what is happening in the system of the alternate methods without making inferences between all possible input and output combinations.

A Fuzzy logic controller (FLC), see Figure 1.13, usually consists of:

(i) A fuzzification unit which maps measured inputs of crisp value into fuzzy linguistic values to be used by a fuzzy reasoning mechanism.

(ii) A knowledge base (KB) which is the collection of expert control knowledge required to achieve the control objective.

(iii) A fuzzy reasoning mechanism that performs various fuzzy logic operations to infer the control action for the given fuzzy inputs.

(iv) A defuzzification unit which converts the inferred fuzzy control action into the required crisp control values to be entered into the system process.

Knowledge in fuzzy reasoning (the decision making unit), is usually expressed as

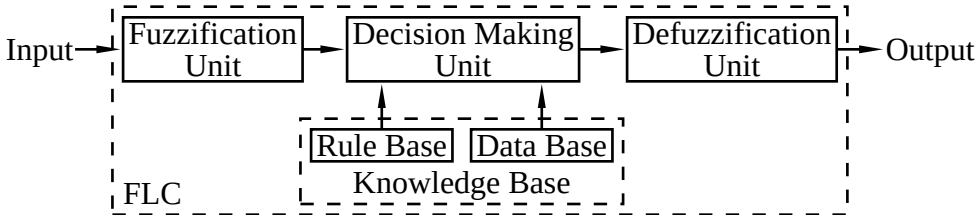


Figure 1.13: Basic Structure of a FLC

rules with sentence conjunctives AND, OR and ALSO, for example:

> **If** Distance x to second player is **SMALL** OR Distance y to obstacle is **CLOSE** AND speed v is **HIGH Then** Perform **LARGE** Correction to steering angle θ ALSO make **MEDIUM** reduction in speed v.

Three antecedents in this example give rise to two outputs. For Multi-Input Single-Output (MISO) system, letting the *linguistic* variable **MEDIUM** be represented by B_k for example, in general can be written:

> **If** x_{k1} is A_{k1} AND x_{k2} is A_{k2} AND $\cdots$ AND x_{kn} is A_{kn} **Then** y is B_k.

In control, language statements such as this can be expressed easily as fuzzy control rules stored in set called the fuzzy KB. Reasoning is usually based on the general modus ponens or direct reasoning law. Each rule in the KB corresponds to a fuzzy relation. For the kth rule defined above, the fuzzy relation can be expressed as $R_k = A_k \rightarrow B_k$. For N rules in the KB, the overall fuzzy relation $R = \cup_{k=1}^{N} R_k$, where R may take either point valued fuzzy implication form or inter-valued fuzzy implication form. Point Value Composition Rule of Inference (PVCRI) involves the following operators:

- Identify 'AND' and 'OR' in the antecedent by intersection and union operators respectively (Intersection is always performed first)
- Find the firing strength α_k of the kth rule,
- Apply the chosen compositional operator to infer control actions in the consequent part of the rule.

A number of compositional operators are in the literature. We illustrate PVCRI; using the sup-min operator. Assume the KB has two rules.

If x is A_1 AND y is B_1 **Then** z is C_1 **If** x is A_2 AND y is B_2 **Then** z is C_2

The membership of the inferred consequence C is point-wise given by $\mu_C(z) = (\alpha_1 \wedge \mu_{C1}(z)) \vee (\alpha_2 \wedge \mu_{C2}(z))$.

A defuzzification strategy produces a non fuzzy control action that best represents the possibility distribution of the inferred control action. The *mean of maximum defuzzification* yields the mean value of all local control actions whose membership functions reach the maximum. The crisp value of z is given in Equation 1.3.

$$z = \sum_{k=1}^{2} \alpha_k H_k W_k / \sum_{k=1}^{2} \alpha_k H_k, \tag{1.3}$$

with W_k being the crisp support value at which the membership function reaches maximum H_k (most usually 1 for normalised membership functions). A symmetrical function is required for each rules consequent fuzzy set.

In general a 'rule of inference' in fuzzy logic reasoning is a mathematical statement describing how linguistic variables are to be manipulated and employed to control

the problem environment. The first step in any application is to determine which variables will be important in choosing an effective control action. Any number of decision variables may appear, but the more that are used, clearly the larger the rule set that must be written. Several approaches to FLC development are possible.

1.9.1 Self-Organising Fuzzy Logic Controls

Typically the KB is generated by an expert but a fundamental weakness with the static fuzzy logic is the acquisition of the rule base which is frequently incomplete and its control strategies are conservative. One approach to overcome this is the construction of *self-organising fuzzy logic controllers* [55] (SOFLC) which have been found useful in developing controllers for systems subject to time varying changes and unknown environmental disturbances. Figure 1.14 outlines the typical structure of a SOFLC. These controllers consist of a two-level hierarchical rule based controller in which the fuzzy control rule base is created and monitored by a self-organising learning algorithm in the upper level and used by a fundamental controller in the lower level. The learning module may contain a performance index table and a rule generation and modification algorithm for creating new rules or modifying existing ones. This module can be developed indirectly through on-line identification of a control system model which can then be used to formulate the control strategies or directly from actual knowledge of the control system itself.

Such self-organising fuzzy logic controllers are used mainly for the creation and modification of the rule base. An adaptive FLC is characterised by a self-tuning ability to generate new rules and modify existing rules. Tuning is also done by

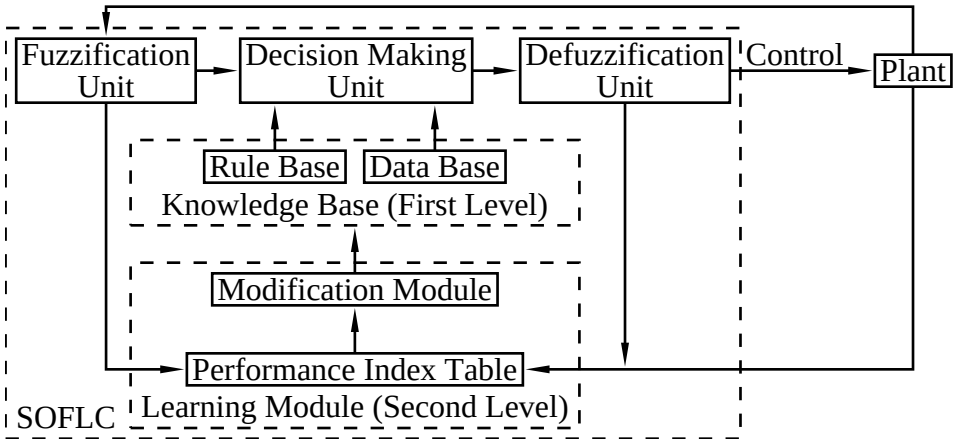


Figure 1.14: Basic Structure of a SOFLC

adjusting their membership functions, the universes of discourse, scaling factors or control resolutions. Of interest is the question of how this self-organisation and adaptation can be carried out in an automated fashion.

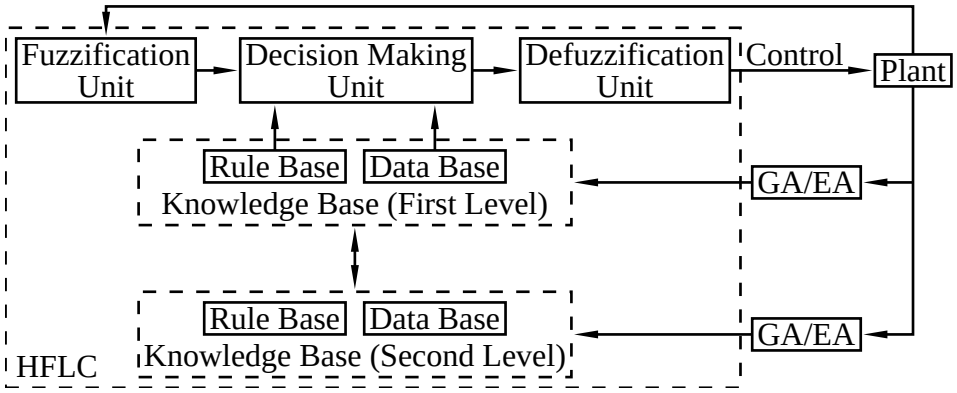


Figure 1.15: GA Fuzzy Rule Generator Architecture

1.9.2 Hierarchical Fuzzy Logic Controllers

As mentioned previously the first step is to determine which variables will be important in choosing an effective control action. Any number of these decision variables may appear, but the more that are used, the larger the rule set that

must be written. It is known [56], that the total number of rules in a system is an exponential function of the number of system variables. This is referred to as the "curse of dimensionality". For a multi-dimensional system, it may become unproductive to realise the fuzzy rule-based controller. To overcome this problem, a hierarchical fuzzy control structure (HFLC) can be used where the most influential parameters are chosen as the system variables in the first level, the next most important parameters are chosen as the system variables in the second level, and so on, [57]. In this hierarchy, the first level gives an approximate output which is then modified by the second level rule set. This procedure can be repeated in succeeding levels of hierarchy. The number of rules in a complete rule set is reduced towards a linear function of the number of variables by the hierarchy. Methods for transforming human knowledge or experience into the rule base and database of a fuzzy inference system seek to generate the rule-base of a system by learning all parameters of the system at once. Their success is limited and if there is a need to add or remove one of the parameters of the rule-base the whole rule-base must be rebuilt.

Problems encountered in this application area of fuzzy logic particularly in multi-robot systems such as the robot-soccer test bed are:

- How to develop reactive and deliberative rules in the knowledge, this involves an overlay of multi-agent strategy theory.
- How to reduce the number of rules for effective fuzzy control—the curse of dimensionality. This involves understanding of how to design fuzzy knowledge based systems using hierarchical and multi-layered fuzzy systems.
- How to build the fuzzy knowledge base for each robot so that it is adaptive to a dynamic changing environment. This involves incorporating of a self-

organising structure that changes/modifies, adds to, or deletes rules from the knowledge base.

- How to evolve the knowledge base directly without expert knowledge, directly from input-output data via simulation or from real-time operation within a robot-soccer game.

1.10 Evolutionary Learning of Fuzzy Control

1.10.1 Key Objectives

Our key objective is to examine techniques for evolving fuzzy logic controllers for each robot in a team, that incorporates multi-agent strategies between team robots. We believe this can be achieved in two ways:

1. Evolve fuzzy controllers via simulation for the robots-soccer scenario similar to that done by Mohammadian and Stonier, see [45, 46, 52, 67, 68, 69, 70, 71]. This will require the use of a simulation package with sufficient kinematic/dynamic structure of each robot's behaviour such that the end result will have acceptable performance in real hardware. Learning of the KB could be done using input output data from actual real-time robot soccer games. FIRA is now to implement codes within international competition such all teams are required to keep logged data from competition matches to be used by all world teams as simulation data input-output.

2. Evolve/modify/adapt a fuzzy control knowledge base for each robot during a real-time robot soccer game between two teams.

To undertake these objectives, research is required in:

- The design of fuzzy logic control systems and their application to control systems.
- The application of evolutionary algorithms to evolve fuzzy logic systems and evolve fuzzy logic controllers in mobile robot systems.

1.10.2 Evolutionary Algorithms

Since their appearance genetic algorithms and variants thereof now commonly lumped under the heading of evolutionary algorithms, have been used in a variety of applications to solve many real world problems in such areas as: ecology, machine learning, engineering and management. As one of a number of evolutionary computation techniques, including evolution strategies, genetic programming and evolutionary programming, it is based on principles of evolution associated with inheritance and the fight for survival of the fittest. It is a heuristic search technique that transforms a population of individuals $P(t) = \{x_1^t, \ldots x_n^t\}$ at iteration t to the next $t + 1$. Each individual can be considered to represent a potential solution to a given problem. Although in their original formulation the structure of each individual was binary encoded [4, 59] modern formulation of structure includes for example, integer and real encoding [27]. To each individual is an associated measure of its fitness for survival. The new population $P(t+1)$ is obtained from the old by the use of genetic operators such as crossover—creating children by recombining parts of usually more than one parent through a process of selection for mating of the parents, and mutation—creating new individuals by perturbing the individual's structure. The form of the algorithm may be written as:

```
begin
   t = 0
   Create random P(0)
   Evaluate Fitness of P(0)
   while (not Terminated) do
   begin
      Create P(t+1) from P(t)
      Evaluate Fitness of P(t+1)
      t = t+1
   end
end
```

In creating the new population, there are many different variants of operators that may be found in the literature. These include proportional selection where the probability of selection for mating is proportional the individual's fitness, ranking methods and tournament selection, one point and many point crossover as well as arithmetic crossover. A full coverage of evolutionary computational algorithms and the applications can be found in the following two surveys [60, 61] and papers [62, 63].

1.10.3 Learning the Knowledge Base

An important aspect of fuzzy logic application is the determination of a fuzzy logic knowledge base to satisfactorily control the specified system. This may be derivable from an appropriate mathematical model or just from system input-output data. Inherent in this are two main problems. The first is to obtain an adequate knowledge base structure for the controller, usually obtained from

expert knowledge, and second is that of selection of key parameters defined in the method.

The KB is typically generated by an expert but a fundamental weakness is that it is frequently incomplete, and its control strategies are conservative. To overcome this one approach is to construct *self-organising fuzzy logic controllers* [55]. These self-organising fuzzy logic controllers are used mainly for the creation and modification of the rule base.

Of interest is the question of how this self-organisation and adaptation can be carried out in an automated fashion. One way is to incorporate genetic/evolutionary algorithms and a substantial amount of literature is now available on these so called *genetic fuzzy systems*. Initial research appears to have commenced with Karr [64] who applied genetic algorithms for learning efficient membership functions and also Thrift [65] who used genetic algorithms for learning the control rules for the cart pole balancing problem. Details of the many interesting approaches can be found in the survey by Cordón and Herrera [66].

As an illustration, consider the following collision-avoidance problem in a two robot system. A three-level hierarchical, fuzzy logic system is proposed to solve the problem, full details can be found in [67]. In the first layer, two knowledge bases, one for each robot, are developed to find the steering angle to control each robot to its target. In the second layer two new knowledge bases are developed using the knowledge in the first layer to control the speed of each robot so that each robot approaches its target with near zero speed. Finally in the third layer, a single knowledge base is developed to modify the controls of each robot to avoid collision in a restricted common workspace.

The important issue is that of learning knowledge in a given layer sufficient for use in higher layers. Consider the knowledge base of a single robot in layer one. It is not sufficient to learn a fuzzy knowledge base from an initial configuration and use this knowledge base for information on the steering angle of the robot to learn fuzzy controllers in the second layer. Quite clearly this knowledge base is only guaranteed to be effective from this initial configuration as not all the fuzzy rules will have fired in taking the robot to its target. We have to find a KB that is effective to some acceptable measure, in controlling the robot to its target from 'any' initial configuration. One way is to first learn a set of local fuzzy controllers, each KB learnt by an evolutionary algorithm from a given initial configuration within a set of initial configurations spread uniformly over the configuration space. These KBs can then be *fused* through a *fuzzy amalgamation* process [49] into the global (final), fuzzy control knowledge base. An alternative approach [68, 69], is to develop an evolutionary algorithm to learn directly the 'final' KB by itself over the region of initial configurations. This can be achieved by appropriately evaluating the fitness of a string at different configurations within the chosen set of initial configurations in each generation and the use of appropriate operators which transfers this local knowledge from generation to generation.

This work differs from other learning methods of evolving rules in fuzzy systems in that it examines how evolutionary learning of the fuzzy knowledge base for a given system can be achieved directly over a broad region of initial configurations for single layered structures with many rules, to multi-layered and hierarchical structures. It is particularly important for the multi-layered and hierarchical structures as pointed out in the previous discussion. This learning evolves an initial *arbitrary* knowledge base using evolutionary algorithms, tuning of membership functions can be easily incorporated but has not been presented

in this report. This work has been reported and fully analysed in the paper by Mohammadian and Stonier [70] in 1998.

Fuzzy logic controllers have been successfully used to replace conventional controllers. However, a subjective choice of the FLC's parameters has a great influence on the overall performance of the system. One of the main issues in recent research has been the improvement of the system performance through automatic design of the FLC without human intervention. Evolution is an effective method for forming FLC's since it does not suffer from the local optimum problem with the use of arbitrary cost function that can not be expressed in exact mathematical formulas, such as the convergence time, the number of rules used, etc. This is why the evolutionary fuzzy approach is recently drawing significant attention as an alternative to the neuro-fuzzy approach.

Evolutionary fuzzy systems, EA's and especially GA's in the majority of cases are used to tune the membership functions and/or select the optimal rule set. Two main approaches are used to develop these systems:

1. FLC-level evolution (Pittsburgh approach)

   Each individual encodes the whole rule set of an FLC. Thus an individual by itself represents a possible solution.

2. Rule-level evolution (Michigan approach)

   The individual is a single rule and an FLC is constructed by collecting some individuals. Thus an individual is only a partial solution and a special credit assignment scheme must be introduced for the appropriate evaluation of each rule.

While in FLC-level evolution each individual evolves independently of the other ones, in Rule-level evolution, the individuals must coevolve by evaluating how well they perform when combined with other individuals. This is essentially the essence to strategy tasking amongst agents (mobile robots) in a multi-agent system. In general, this coevolutionary approach decomposes a high-level composite into sub-components, which get specialised in interaction with the other ones. As a result of collaboration of these sub-components, the high-level goal is effectively achieved. According to a study on the high-dimensional fuzzy classification application, the Michigan approach is reported to be superior to the Pittsburgh approach.

Mohammadian and Stonier have used the Pittsburgh approach in evolving a full knowledge base for each robot in their robot simulation system [71].

1.11 Scope

Present robot soccer competitions appear to ignore the basics of control and concentrate of the more elaborate higher level controls. However, observations at competitions have highlighted a lack or control over the robots. The higher level AI techniques, such as multi-agent, cannot compensate for poor robot control in the lower levels.

Most teams seem to be having difficulty with misalignment errors and do not account for robot dynamics. The "Stop-Turn-Run" algorithms introduce large momentum forces as average robot speed increases. Robots have been increasing in speed every competition and are continuing to show loss of control caused by limitations on: image capture rates, image resolution, soft real time operating

systems (processing) and smoothness of robot control. As robot speed increases, the need for smooth control becomes dominant. A good foundation of mobile robot control is needed for higher level AI techniques to be effective.

Our early work in mobile robot control highlighted the difficulty of using linear techniques in a highly non-linear, time varying, multi-dimensional, system—robot soccer. A decision to use fuzzy systems seemed appropriate to control this system, for they deal with problems that are non-linear and consume little execution time for small systems. The high dimensionality of the system leads to development of the fuzzy knowledge base by using evolutionary algorithms.

The next chapter introduces fuzzy logic concepts used in this thesis.

Chapter 2

Fuzzy Logic

2.1 Introduction

Fuzzy systems have become an attractive alternative to Neural Network systems. Both apply to non-linear applications, use transfer curves and are trainable. The distinction of fuzzy over neural is that fuzzy can be interpreted through the fuzzy if-then rule base structure as opposed to a hidden non-interpretable weight matrix in the neural structure.

This chapter reviews the fundamentals of fuzzy logic systems to establish the terminology used in this thesis. It is intended to act as a primer for readers that are not familiar with fuzzy logic. The chapter is based on notes taken from a fuzzy logic course authored by Russel Stonier.

2.2 Review of Classical Set Theory

A basic review of some elements of classical set theory is now made.

DEFINITION 2.2.1 *A **set** is a collection or group of objects. The objects of a set are **distinguishable** and **definite**. The objects are called **elements** or **members**.*

Any pair of objects of a set are distinguishable if they can be qualified as being the same or different. An object is definite if it is possible to determine whether the object **is**, or **is not**, a member of a set. A set is completely determined by its members.

Examples of sets are:

$$\begin{array}{rcl} V & = & \{a, e, i, o, u\} \\ Z & = & \{1, 2, 3, \cdots\} \\ B & = & \{x | x \in Z\} \\ C & = & \{x | 0 \leq x \leq 1\} \end{array}$$

Sets V and Z are shown using the *list method*. Sets B and C are shown using the *rule method*. The general form of the rule method is:

$$A = \{x \in U | x \textit{ meets some condition.}\}$$

A third method called the *membership method* is also used. It uses a characteristic or discriminant function (zero-one membership function), for A, such that

$$\mu_A = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$$

The set A is mathematically equivalent to its membership function μ_A in the sense that knowing μ_A is the same as knowing A itself. In classical set theory an object in either in a set or it is not.

Let U be defined as the *universal* set containing all elements of the universe of discourse. The empty set $\emptyset$ is that set with no elements.

Three basic operations on sets are the unary compliment operation on a set A, and the binary operations of intersection and union of two sets A and B, defined by:

$$\begin{array}{lcl} A' & = & \{x|x \notin A\} \\ A \cap B & = & \{x|x \in A \text{ and } x \in B\} \\ A \cup B & = & \{x|x \in A \text{ or } x \in B\} \end{array}$$

EXAMPLE 2.2.1 *Given sets $A = \{x|0 \leq x \leq 1\}$ and $B = \{x|0.5 \leq x \leq 1.5\}$, and universe of discourse $U = \{x \mid 0 \leq x \leq 2\}$, the Characteristic functions of A, B, A', $A \cap B$ and $A \cup B$, are shown in Figure 2.1.*

Characteristic functions for A', $A \cap B$ and $A \cup B$, can be defined as follows:

$$\mu_{A'}(x) = 1 - \mu_A(x) \tag{2.1}$$

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \tag{2.2}$$

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \tag{2.3}$$

The characteristic function μ_A is an alternative way of defining the set A and the max, min and $1 - \mu_A$ operations are alternative ways of defining union, intersection and compliment.

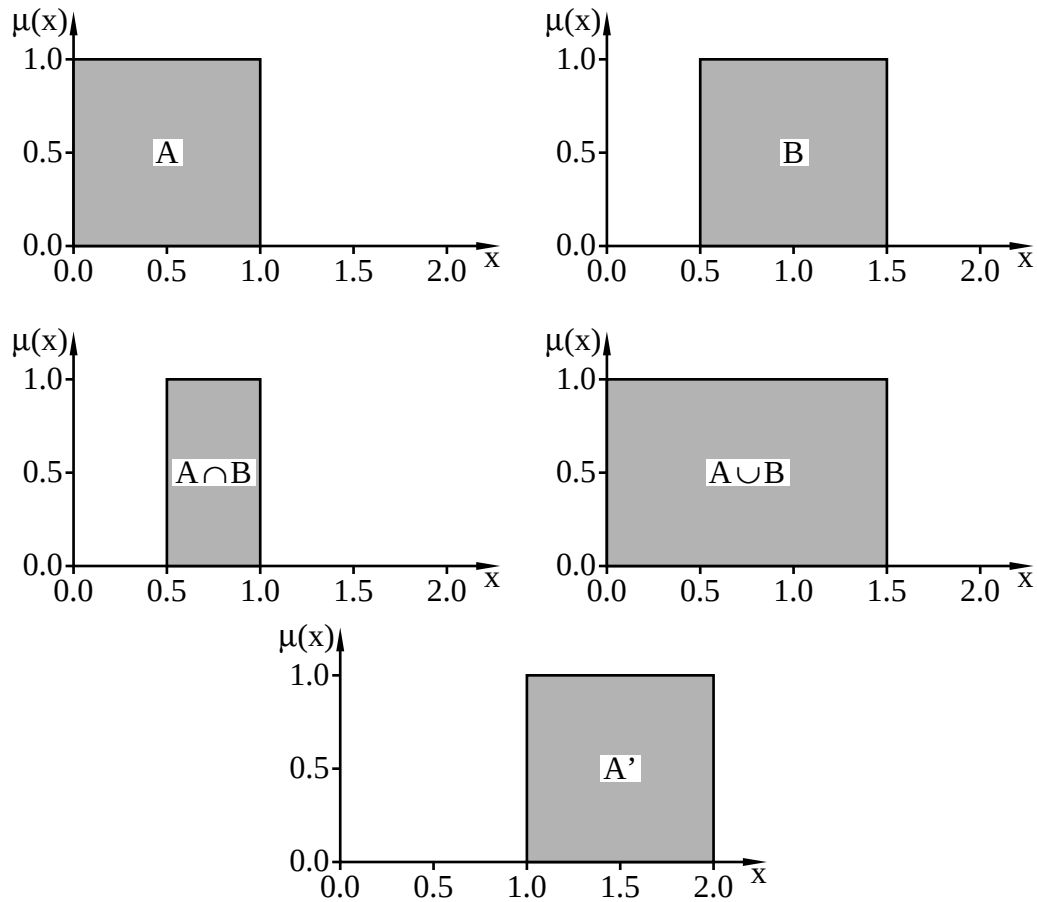


Figure 2.1: Characteristic Functions for A', $A \cap B$ and $A \cup B$ given A and B

Equations 2.1, 2.2 and 2.3 are not unique and belong to families of Compliment, Union and Intersection respectively. For example, intersection can be defined as:

$$\begin{aligned} \mu_{A\cap B}(x) &= \mu_A(x)\mu_B(x) \\ \mu_{A\cap B}(x) &= \max(0, \mu_A(x) + \mu_B(x) - 1) \end{aligned}$$

Conflict arises with objects that are not precisely in a set or not in a set. An object may be defined as being partially in set A by some degree and partially in set B by some degree. Fuzzy sets are useful in cases where objects cannot be completely classified into crisp sets.

2.3 Fuzzy Sets

The theory of fuzzy sets deals with sets in a universe of discourse U as does classical sets. A fuzzy set F in U is a generalisation of the concept of an ordinary or classical set. It is typically defined by membership function $\mu_F : U \rightarrow [0, 1]$, rather than the discrete values of 0 and 1 as with classical sets.

Characteristics of fuzzy sets are:

- If $\mu_F(x_0) = 1$, then element x_0 is definitely a member of the fuzzy set F. If $\mu_F(x_0) = 0$, then x_0 is definitely not a member of the fuzzy set F.
- All intermediate values for $\mu_F(x_0)$ indicate that x_0 has a degree of membership in [0,1].
- Classical or crisp sets are special cases of fuzzy sets.

EXAMPLE 2.3.1 *Age does not have a definitive transition between young, middle age and old. A fuzzy set describing "middle age" may be defined as shown in Figure 2.2.*

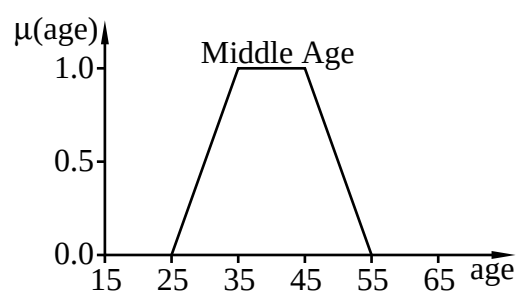


Figure 2.2: Membership Function for "Middle Age"

The fuzzy set "middle age " could have been labelled as K, but "middle age"

is more descriptive and less ambiguous. At age $= 30$ *the degree of membership* $\mu(age) = 0.5$*; that is getting to middle age.*

For a discrete discourse U, F is usually represented as a set of ordered pairs with element u and grade of membership value:

$$F = \{(u, \mu_F(u)) | u \in U\}.$$

EXAMPLE 2.3.2 *Consider the discourse* Z_4 *be the set of integers* $z_4 = \{z| -3 \leq z \leq 3\}$*. The following set describes the set of integers "close to zero".*

$$CZ = \{(-3, 0.1), (-2, 0.3), (-1, 0.6), (0, 1.0), (1, 0.6), (2, 0.3), (3, 0.1)\}.$$

It is assumed that if elements in Z_4 *are not listed then the membership of the element is zero.*

For this example it should also be noted that this is only one interpretation of what is meant by the fuzzy set "close to zero". The fuzzy set CZ could have been chosen as:

$$CZ = \{(-2, 0.1), (-1, 0.5), (0, 1.0), (1, 0.5), (2, 0.1)\}$$

It is just as appropriate. Just what is appropriate will be dependent on the application under consideration. Fuzzy sets can mathematically describe objects that are vague in natural language. As in all natural languages, the interpretation of objects can differ.

EXAMPLE 2.3.3 *Consider the continuous discourse R being the set of real numbers. Define the fuzzy set CE of real numbers "close to or equal to zero". Figure 2.3(a) and (b) illustrate two different membership functions for the set CE.*

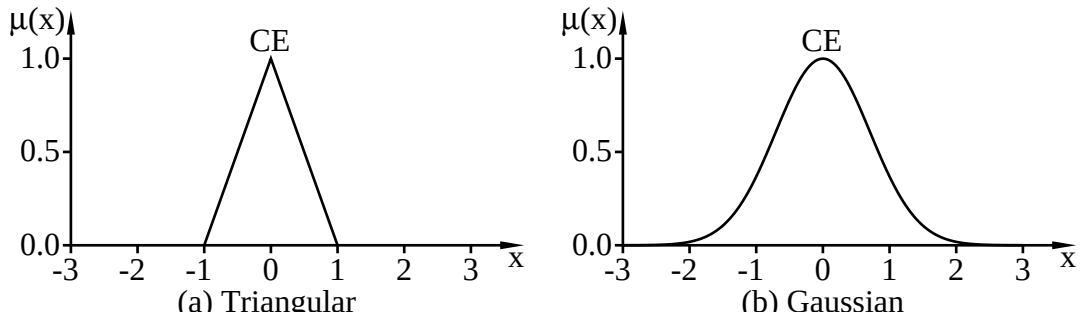


Figure 2.3: Possible Membership Functions "Close to or Equal to Zero"

Figure 2.3(a) is defined as a triangular membership function of CE:

$$\mu_{CE} = \begin{cases} 0 & \text{if } x < 1, \\ x+1 & \text{if } -1 \leq x \leq 0, \\ 1-x & \text{if } 0 \leq x \leq 1, \\ 0 & \text{if } x > 1. \end{cases}$$

Figure 2.3(b) is defined as a Gaussian membership function of CE:

$$\mu_{CE}(x) = \exp(-x^2) \qquad \forall x \in R$$

Note that this function satisfies the condition $0 \leq \mu_{CE} \leq 1$. These functions are typical of those used in defining such a fuzzy set, but there can be many others. For example, the membership of set CE could be defined as:

$$\mu_{CE}(x) = \frac{1}{1+x^2} \qquad \forall x \in R$$

The general form of the Gaussian is dependent upon two parameters, the mean

$\overline{x}$ of the distribution and the variance σ:

$$\mu_A(x; \overline{x}, \sigma) = \exp\left(-\left(\frac{x - \overline{x}}{\sigma}\right)^2\right) \qquad \forall x \in R \tag{2.4}$$

Triangular membership functions are succinctly written in the form:

$$\mu_A(x; a, b, c) = \begin{cases} 0 & \text{if } x < a, \\ (x-a)/(b-a) & \text{if } a \leq x < b, \\ (c-x)/(c-b) & \text{if } b \leq x < c, \\ 0 & \text{if } x \geq c. \end{cases} \tag{2.5}$$

where the base of the triangle $[a, c]$ is termed the support of the fuzzy set, and apex b is called the centre of the fuzzy set.

Another commonly used form of fuzzy membership is the trapezoid function shown in Example 2.3.1. The general form of trapezoid membership function is:

$$\mu_A(x; a, b, c, d) = \begin{cases} 0 & \text{if } x < a, \\ (x-a)/(b-a) & \text{if } a \leq x < b, \\ 1 & \text{if } b \leq x < c, \\ (d-x)/(d-c) & \text{if } c \leq x < d, \\ 0 & \text{if } x \geq d. \end{cases} \tag{2.6}$$

2.3.1 Zadeh Notation

Zadeh [72] proposed an alternative notation for fuzzy sets. For the discrete classical set with a finite or infinitely countable set of elements.

$$A = \{a_1, a_2, a_3, \cdots, a_n\},$$

Zadeh proposed the notation:

$$A = \{a_1 + a_2 + a_3 + \cdots + a_n\},$$

where it should be noted that the operator '+' does not mean the elements are added, but refers to set union.

The familiar notation of defining a set A as ordered pairs is:

$$A = \{(a_1, \mu_A(a_1)), (a_2, \mu_A(a_2)), \cdots, (a_n, \mu_A(a_n))\},$$

but in Zadeh's notation, set A is written:

$$\begin{aligned} A &= \mu_A(a_1)/a_1 + \mu_A(a_2)/a_2 + \cdots + \mu_A(a_n)/a_n \\ &= \sum_{i=1}^{n} \mu_A(a_i)/a_i. \end{aligned}$$

The notation $\sum$ used refers to set union rather than to arithmetic summation and '/' is simply used to connect an element and it's membership value and has no connection with arithmetic division. An infinite membership written in Zadeh's notation is:

$$A = \sum_{a \in U} \mu_A(a)/a.$$

The membership function CZ defined in Example 2.3.2 is written in Zadeh notation as:

$$\begin{aligned} C &= \{(-3, 0.1), (-2, 0.3), (-1, 0.6), (0, 1.0), (1, 0.6), (2, 0.3), (3, 0.1)\} \\ &= 0.0/-4 + 0.1/-3 + 0.3/-2 + 0.6/-1 + 1.0/0 + 0.6/1 + 0.3/2 + \\ &\quad 0.1/3 + 0.0/4. \end{aligned}$$

Zadeh's notation is further extended to a fuzzy set F with continuous membership as:

$$F = \int_{a \in U} \mu_A(a)/a,$$

where the integral symbol $\int$ is the continuous counterpart of the summation symbol $\sum$.

2.3.2 Probability versus Fuzzy Sets

Probability describes the uncertainty of a given event occurring. The events themselves are described by crisp sets. Fuzzy sets generalise the notion of an event, they say nothing about the probability of the occurrence of events. Note that the degrees of membership for a fuzzy set do not have to sum or integrate to one. The "degree of membership" value is sometimes called a "possibility" value and indicates the extent to which a given element being a member of the set can be considered true.

2.3.3 Some Basic Concepts

Some basic concepts of classical crisp sets and concepts unique to fuzzy sets are considered here as preparation for the mathematical analysis appearing later in this chapter.

The *support* of a fuzzy set A in a universe of discourse U is a crisp set that contains all the elements of U that have non-zero membership values, namely:

$$supp(A) = \{x \in U | \mu_A(x) > 0\}, \tag{2.7}$$

where $supp(A)$ denoted the support of the fuzzy set A.

For example, the support of the fuzzy set "close to zero" in Example 2.3.2 is the set of integers $\{-3, -2, -1, 0, 1, 2, 3\}$. If the support of a fuzzy set is empty, it is called an *empty fuzzy set*. Similarly, a *fuzzy singleton* is a fuzzy set whose support is a single point in U.

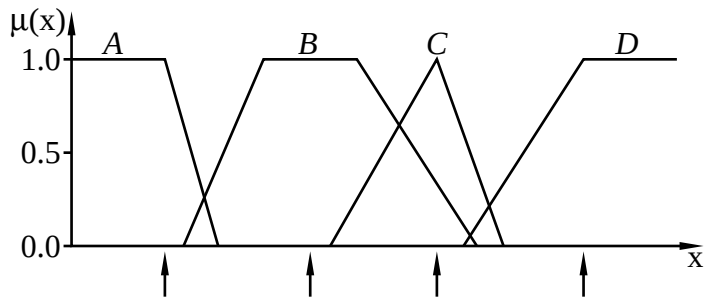


Figure 2.4: Centres of Fuzzy Sets A, B, C, and D

The *centre* of a fuzzy set is defined as follows:

- If the mean of all points at which the membership function of the fuzzy set achieves its maximum value is finite, then the centre is this mean value.
- If the mean value is positive infinite, then the centre is smallest of all points that achieve the maximum membership.
- If the mean value is negative infinite, then the centre is largest of all points that achieve the maximum membership.

This definition typically relates to fuzzy sets in one dimension. Some examples are shown in Figure 2.4.

The *height* of a fuzzy set is the largest membership value attained. For all the examples so far given the heights have all been equal to one, but not necessarily so. A fuzzy set is called *normal* if its height equals one. It is typical to normalise fuzzy sets so that their height equals one.

An α-*cut* of a fuzzy set is the crisp set

$$A_\alpha = \{x \in U | \mu_A(x) \geq \alpha.\} \tag{2.8}$$

For example, in Figure 2.3(a), $CE_{0.5} = [-0.5, 0.5]$. Note that an α-cut discards the points in the support, whose membership values are lower than α.

2.3.4 Basic Operations on Fuzzy Sets

Let $A, B \in U$, be two point-valued sets with associated membership functions μ_A and μ_B respectively. Then based upon notions described for classical sets above, the following basic operators can be defined:

- Two sets A and B are **equal** if and only if $\mu_A(u) = \mu_B(u) \quad \forall u \in U$.
- The set A is **contained** in B if and only if $\mu_A(x) \leq \mu_B(x) \quad \forall x \in U$.
- The **union** of A and B is that set with membership function defined by:

$$\mu_{A \cup B}(u) = \max(\mu_A(u), \mu_B(u)) = \mu_A(u) \vee \mu_B(u) \quad \forall u \in U. \tag{2.9}$$

- The **intersection** of A and B is that set with membership function defined by:

$$\mu_{A \cap B}(u) = \min(\mu_A(u), \mu_B(u)) = \mu_A(u) \wedge \mu_B(u) \quad \forall u \in U. \tag{2.10}$$

- The **complement** of A, A', is the set with membership function defined by:

$$\mu_{A'}(u) = 1 - \mu_A(u) \quad \forall u \in U. \tag{2.11}$$

These basic definitions maintain the basic premises associated with the classical definitions, for example:

- The union $A \cup B$ is the smallest fuzzy set containing both A and B and if C is any fuzzy set containing both A and B then it also contains the union $A \cup B$.

- The intersection $A \cap B$ is the largest set contained in both A and B, and if C is any fuzzy set contained in both A and B then it is contained in the intersection $A \cap B$.

Figure 2.5 illustrates the concepts of fuzzy set intersection, union and compliment.

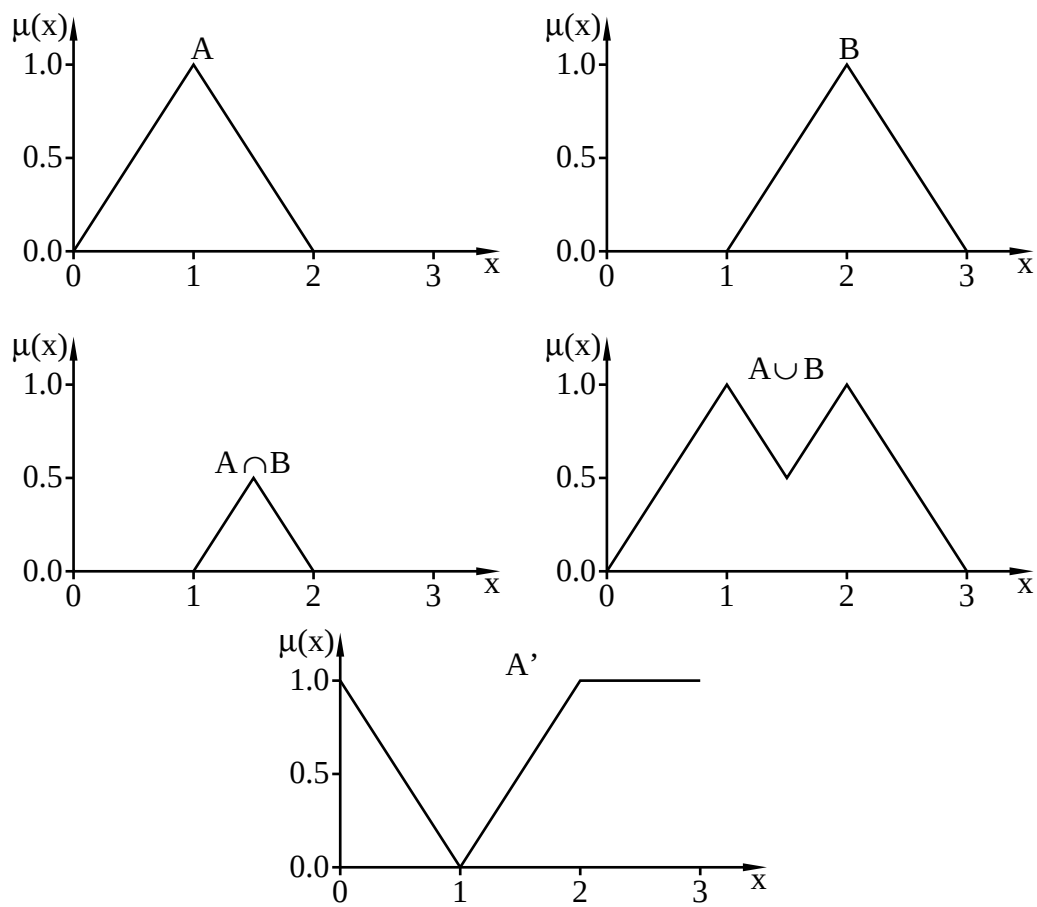


Figure 2.5: Fuzzy Membership of $A \cap B$, $A \cup B$ and A'

Most of the identities used in classical sets can be verified for fuzzy sets. Table 2.1 shows the identities applicable for fuzzy sets.

Table 2.1: Fuzzy Identities [3]

Complement	$\mu_{\overline{A}}(x) = 1 - \mu_A(x), \forall x \in U, \mu_A(x) \in [0,1]$
Intersection	$\mu_{A \cap B}(x) = \min[\mu_A, \mu_B], \forall x \in U$
Union	Equality $\mu_A(x) = \mu_B(x), \forall x \in U$
Subset	$\mu_A(x) \leq \mu_B(x), \forall x \in U$
Double negation	$\overline{\overline{A}} = A$
DeMorgan's Law	$\overline{A \cup B} = \overline{A} \cap \overline{B}$
	$\overline{A \cap B} = \overline{A} \cup \overline{B}$
Idempotence	$A \cup A = A, A \cap A = A$
Distributive	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
Commutative	$A \cup B = B \cup A, A \cap B = B \cap A$
Associative	$(A \cup B) \cup C = A \cup (B \cup C)$
	$(A \cap B) \cap C = A \cap (B \cap C)$
Absorption	$A \cup (A \cap B) = A, A \cap (A \cup B) = A$
Law of zero	$A \cup U = U, A \cap \emptyset = \emptyset$
Law of identity	$A \cup \emptyset = A, A \cap U = A$
Cartesian product	$\mu_{A_1 \times A_2 \times \ldots A_n}(x_1, x_2, \ldots, x_n)$
	$= \min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_n}(x_n)]$
Algebraic sum	$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$
Algebraic product	$\mu_{a \cdot B}(x) = \mu_A(x) \cdot \mu_B(x)$
Bounded sum	$\mu_{A \oplus B}(x) = \min[1, \mu_A(x), \mu_B(x)]$
Bounded difference	$\mu_{A \ominus B}(x) = \max[0, \mu_A(x) - \mu_B(x)]$

2.3.5 Fuzzy Complement

DEFINITION 2.3.1 *A complement operator* $c : [0,1] \rightarrow [0,1]$, *is a class of unary operators that can be used to construct the complement of a fuzzy set. For a fuzzy set* A *in a universe of discourse* U*:*

$$\mu_{A'}(u) = c[\mu_A(u)] \quad \forall u \in U.$$

The unary operator must satisfy the following axioms:

C_1: $c[0] = 1$ *(boundary condition)*
C_2: $a < b \Rightarrow c[a] \geq c[b]$ *(non-increasing condition)*
C_3: $c[c[a]] = a$

where $a, b \in [0, 1]$.

Axiom C_1 shows that if an element belongs to a fuzzy set with degree 0 then it belongs to the complement of the fuzzy set with membership 1. Axiom C_2 states that c must be a non-increasing function of its argument, so that an increase in the membership of a fuzzy set must result in a decrease or no change in membership value for the complement. Axiom C_3 states that for the membership value of an element is the same as the membership value in the complement of the complement of the fuzzy set. These rules can be reduced to two by removing C_3 and amending the boundary condition C_1 to:

$$C_1^1\text{:} \quad c[0] = 1 \text{ and } c[1] = 0.$$

The standard operator, Equation 2.1, was shown in Figure 2.5 and satisfies the complement axioms. Other classes of fuzzy complement, such as the Sugeno and Yager class are found in detail in [8, 3].

2.3.6 Fuzzy Union: S-norm

DEFINITION 2.3.2 *A triangular co-norm or S-norm is a class of binary operators denoted by $\vee$ that can be used to construct the fuzzy union of two sets. An S-norm for fuzzy sets A and B in a universe of discourse U is written as:*

$$\mu_{A \cup B}(u) = \mu_A(u) \vee \mu_B(u) \quad \forall u \in U.$$

The binary operator must satisfy the following axioms:

S_1: $a \vee b = b \vee a$ *(commutative)*
S_2: $(a \vee b) \vee c = a \vee (b \vee c)$ *(associative)*
S_3: $a \leq c$ *and* $b \leq d \Rightarrow a \vee b \leq c \vee d$ *(non-decreasing condition)*
S_4: $a \vee 1 = a$ *(boundary condition)*

where $a, b \in [0, 1]$.

It is common notation to also write

$$s[\mu_A(u), \mu_B(u)] = \mu_A(u) \vee \mu_B(u) = \mu_{A \cup B}(u) \quad \forall u \in U.$$

where $s : [0, 1] \rightarrow [0, 1]$, *is the transformation that takes the membership functions of the fuzzy sets* A *and* B *into the membership function of* $A \cup B$.

Axiom S_1 describes the union function at extreme cases when an element has full or null membership in a fuzzy set. Axiom S_2 is the commutative axiom to ensure the commutative of the union of fuzzy sets. Axiom S_3 describes the requirement that an increase in the membership values in the two fuzzy sets should result in an increase in membership value in the union of two fuzzy sets. Axiom S_4 clearly extends the associative axiom of classical set theory.

The standard fuzzy union, Equation 2.3, satisfies all of the S-norm axioms. Wang [8] and Lin [3] list many other classes of S-norms such as: the Dombi class, Dubois-Prade, class, Yager class, Drastic sum, Einstein sum, Algebraic sum and the Minimum sum. Figure 2.6 shows two S-norms; the standard S-norm and the Algebraic sum s_{as} S-norm.

These union operators are all extensions of non-fuzzy set union. Note that some S-norms may be more meaningful than others in some applications.

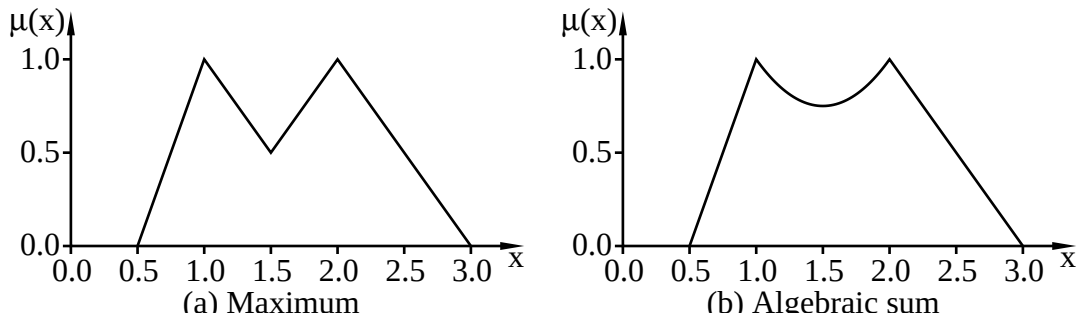


Figure 2.6: S-norms: (a) $s_{\max}(a,b) = \max(a,b)$, and (b) $s_{ab}(a,b) = a + b - ab$

2.3.7 Fuzzy Intersection: T-norm

DEFINITION 2.3.3 *A triangular norm or T-norm is a class of binary operators denoted by $\wedge$ that can be used to construct the fuzzy intersection of two sets. A T-norm for fuzzy sets A and B in a universe of discourse U is written as:*

$$\mu_{A\cap B}(u) = \mu_A(u) \wedge \mu_B(u) \quad \forall u \in U.$$

The binary operator must satisfy these axioms:

T_1 $a \wedge b = b \wedge a$ *(commutative)*
T_2 $(a \wedge b) \wedge c = a \wedge (b \wedge c)$ *(associative)*
T_3 $a \leq c$ *and* $b \leq d \Rightarrow a \wedge b \leq c \wedge d$ *(non-decreasing condition)*
T_4 $a \wedge 1 = a$ *(boundary condition)*

where $a, b \in [0,1]$.

It is common notation to also write:

$$t[\mu_A(u), \mu_B(u)] = \mu_A(u) \wedge \mu_B(u) = \mu_{A\cap B}(u) \quad \forall u \in U.$$

where $t : [0,1] \to [0,1]$, is the transformation that takes the membership functions of the fuzzy sets A and B into the membership function of $A \cap B$.

It is easy to verify that Equation 2.2 satisfies all of the axioms for the T-norms. Further, it should be clear from the above discussion, that one would expect these conditions to specify a T-norm to define intersection membership. As with the union operators, there are many other classes of T-norms in the literature such as: the Dombi class, Dubois-Prade class, Yager class, Drastic product, Einstein product, Algebraic product, and the Minimum product. Figure 2.7 shows two T-norms; the standard min T-norm and the Algebraic product t_{ap} T-norm.

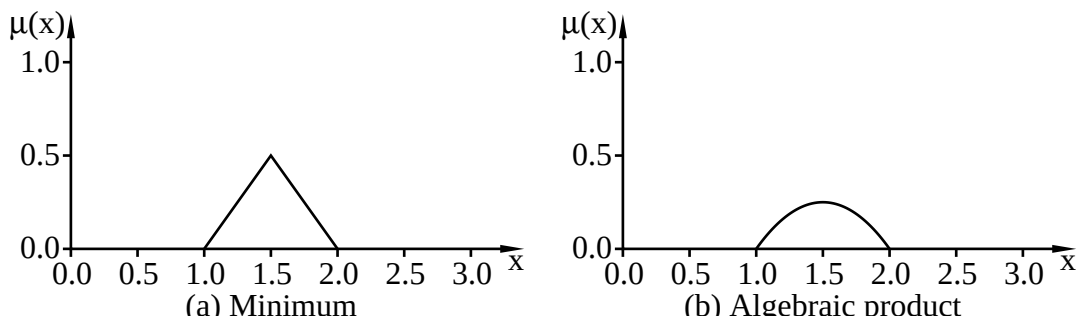


Figure 2.7: T-norms: $t_{\min}(a, b) = \min(a, b)$, and $t_{ap}(a, b) = ab$

2.4 Relations

Let U and V be two non-fuzzy sets. The Cartesian product of U and V, $U \times V$ is defined by the set of ordered pairs:

$$U \times V = \{(u, v) |\ u \in U \text{ and } v \in V\}. \tag{2.12}$$

DEFINITION 2.4.1

A binary relation R on U and V is a subset of $U \times V$.

This is a simple definition but it is the foundation point for the development of functions which are relations satisfying particular properties. Some simple

examples of relations are given in the following example.

EXAMPLE 2.4.1

Let $U = \{1, 2, 3\}$ *and* $V = \{1, 2, 3, 4, 5, 6\}$. *The following sets define relations on* U *and* V.

(a) $R_1 = \{(1, 1), (2, 2), (3, 3)\}$.

(b) $R_2 = \{(1, 1), (2, 1), (3, 3), (3, 6), (1, 2), (3, 4), (3, 5)\}$.

(c) $R_3 = \{(1, 2), (2, 4), (3, 6)\}$.

(d) $R_4 = \{(2, 1), (2, 2), (3, 3)\}$.

(e) $R_5 = \{(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6)\}$.

Each relation has a different property.

Relation R_1 *defines the association that* $u \in U$ *is associated with the identical element in* V. *It defines the identity relation in* U. *The domain of* R_1 *is* U, *being the set of elements* u *in the first position of the ordered pairs in* R_1. *The range of* R_1 *is* $\{1, 2, 3\}$, *this set being the set of elements in* V *that occur in the ordered pairs of* R_1. R_1 *has also the property that each* u *its domain is associated with one* $v \in V$. *This property describes* R_1 *as a function. It is also said to be* $1 - 1$ *(read as one to one), as one and only one element in* V *is associated with one element in* U.

Relation R_2 *clearly has domain equal to* U *and range* V. *Since* R_2 *fills up all of* V, *the relation is said to be* ***onto***. *It however is not a function.*

Relation R_3 is again a function with domain U and range $\{2, 4, 6\}$. It describes the association that each element in the range is twice the value of the element in U, written as: $v = f(u) = 2u$ and $R_3 = \{(u, f(u))|u \in U\}$.

Relation R_4 is clearly not a function. Its domain is $\{2, 3\}$, not all of U, and range $\{1, 2, 3\}$. The relation is not onto and it is also not $1 - 1$.

Relation R_5 describes the association that the first element of the order pair is strictly less than the second element in the ordered pair. The domain of R_5 is U and its range is $\{2, 3, 4, 5, 6\}$.

It should be clear that each relation is intimately related to its domain in U and its range in V. This dependence can be more clearly written, for example: $R_1(U, V) = R_1$ and $R_1 : U \to V$.

The Cartesian product definition is easily extended to the Cartesian product of n non-fuzzy sets $\{U_i, i = 1, \cdots, n\}$, written as:

$$U_1 \times U_2 \times \cdots \times U_n = \{(u_1, u_2, \cdots, u_n)|u_i \in U_i, i = 1, \cdots n\}. \tag{2.13}$$

This concept can extend the definition of binary relation to a relation on sets $\{U_i, i = 1, \cdots, n\}$.

DEFINITION 2.4.2

A relation B on $U_1 \times U_2 \times \cdots U_n$ is a subset of $U_1 \times U_2 \times \cdots U_n$:

$$B = B(U_1, \cdots, U_n) = \{(u_1, \cdots, u_n)|P(u_1, \cdots, u_n)\},$$

where $P(u_1, \cdots, u_n)$ is a condition to be satisfied by n-tuples in $U_1 \times U_2 \times \cdots U_n$.

2.5 Fuzzy relations

The classical notation as defined in the notation above represents a crisp association among set elements, that is, there is either a relationship or there is not. The entries in the relational matrix are either 1 or 0. For a certain associations, it is difficult to describe such 1 or 0 assessment. For example, consider the sets $U = V = \{1, 2, 3\}$, and the binary relation R defined on U and V by the correspondence "approximately equal".

To define a fuzzy relation based on crisp sets allows the characteristic or membership function μ_b to take all values in $[0, 1]$, not just 0 or 1. A binary relation is defined as:

DEFINITION 2.5.1

Let U and V be two universes of discourse and

$$\mu_R : U \times V \to [0, 1],$$

then

(a) *A binary fuzzy relation defined on continuous U and V is*

$$R = \int_{U \times V} \mu_R(u, v)/(u, v) = \int_U \int_V \mu_R(u, v)/(u, v).$$

(b) *A binary fuzzy relation defined on countable or finite U and V is*

$$R = \sum_{U \times V} \mu_R(u, v)/(u, v) = \sum_U \sum_V \mu_R(u, v)/(u, v).$$

EXAMPLE 2.5.1

Consider the sets $U_1 = U_2 = \{-2, -1, 0, 1, 2\}$ and the fuzzy relation R, defined

by the relational matrix:

$R =$

		U_2				
		-2.00	*-1.00*	*0.00*	*1.00*	*2.00*
U_1	*-2.00*	*0.00*	*0.01*	*0.02*	*0.01*	*0.00*
	-1.00	*0.01*	*0.14*	*0.37*	*0.14*	*0.01*
	0.00	*0.02*	*0.37*	*1.00*	*0.37*	*0.02*
	1.00	*0.01*	*0.14*	*0.37*	*0.14*	*0.01*
	2.00	*0.00*	*0.01*	*0.02*	*0.01*	*0.00*

This relation/set can be said to define the set of ordered pairs $(u_1, u_2) \in U_1 \times U_2$, *such that* u_1 *is "close to zero" and* u_2 *is "close to zero".*

The notation above can be extended to the Cartesian product $U_1 \times U_2 \times \cdots \times U_n$ of crisp sets as in the next definition.

DEFINITION 2.5.2

Let $U_k, k = 1, \cdots, n$ *be* n *universes of discourse with membership function*

$$\mu_R : U_1 \times U_2 \times \cdots \times U_n \to [0, 1],$$

then

(a) *A fuzzy relation defined on continuous* $U_1 \times U_2 \times \cdots \times U_n$ *is:*

$$\begin{aligned} R &= \int_{U_1 \times U_2 \times \cdots \times U_n} \mu_R(u_1, \cdots, u_n)/(u_1, \cdots, u_n) \\ &= \int_{U_1} \cdots \int_{U_n} \mu_R(u_1, \cdots, u_n)/(u_1, \cdots, u_n). \end{aligned} \tag{2.14}$$

(b) *A fuzzy relation defined on countable or finite* U_k *is:*

$$\begin{aligned} R &= \sum_{U_1 \times U_2 \times \cdots \times U_n} \mu_R(u_1, \cdots, u_n)/(u_1, \cdots, u_n) \\ &= \{((u_1, \cdots, u_n), \mu_R(u_1, \cdots, u_n))|\ (u_1, \cdots, u_n) \in U_1 \times U_2 \times \cdots \times U_n\} \end{aligned} \tag{2.15}$$

2.6 Projections and Cylindrical Extensions

Consider a simple illustration. Let $U_1, U_2 = R$, and consider the set $D \subset U_1 \times U_2 = R^2$, defined by:

$$D = \{(u_1, u_2)|\ u_1^2 + u_2^2 \leq 1\},$$

that describes all coordinate pairs within the unit disc on the plane. This set has characteristic function:

$$\mu_D(u_1, u_2) = \begin{cases} 1 & \text{if } u_1^2 + u_2^2 \leq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.16}$$

The *projection* of this set D on the u_1-axis, D_1, is the set of all u_1 occurring as the first element in the ordered pairs constituting the set D. The projection of this set on the u_2-axis, D_2, is the set of all u_2 occurring as the second element in the ordered pairs constituting the set D. These sets are clearly

$$D_1 = [-1, 1] \subset U_1 \text{ and } D_2 = [-1, 1] \subset U_2.$$

It is easy to see that the characteristic function describing D_1 is defined by:

$$\mu_{D_1}(u_1) = \begin{cases} 1 & \text{if } \exists\, u_2 \in U_2 \text{ such that } u_1^2 + u_2^2 \leq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

The set D is a relation according to the definition given above. Note that a unit circle

$$C = \{(u_1, u_2)|\ u_1^2 + u_2^2 = 1\},$$

has projections on the axes C_1 and C_2 equal the interval $[-1, 1]$.

Cylindrical extensions are defined as $DE_k \subset U_1 \times U_2 = R^2$ of these projections D_k "back into" the two dimensional space $U_1 \times U_2 = R^2$ by extending their

definitions based upon their evaluation in D_1 and D_2, as follows:

$$DE_1 = \{(u_1, u_2)|\ u_1 \in D_1, u_2 \in U_2 = R\} = [-1, 1] \times U_2,$$

$$DE_2 = \{(u_1, u_2)|\ u_1 \in U_1 = R, u_2 \in D_2\} = U_1 \times [-1, 1].$$

These definitions are illustrated in Figure 2.8 on a restricted domain [-2,2].

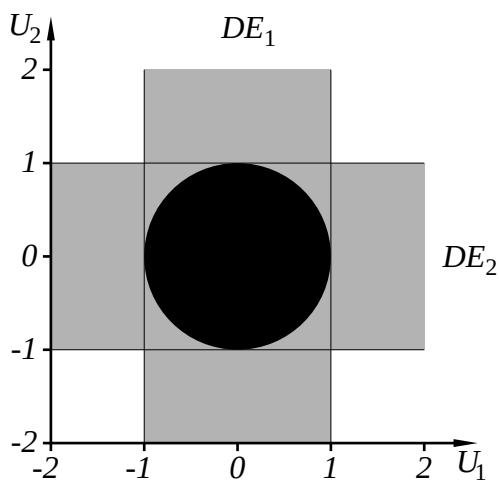


Figure 2.8: Unit Disc Projections and Extensions

These concepts are readily extended to sets in higher dimensions. The concepts shown above can be extended to fuzzy relations, whilst maintaining all the characteristics of those defined in classical set and function theory.

DEFINITION 2.6.1

Let Q be a fuzzy relation in $U_1 \times U_2 \times \cdots \times U_n$ and $\{i_1, \cdots, i_k\}$ be a subsequence of the indices $\{1, 2, \cdots, n\}$, then the projection of Q on $U_{i_1} \times \cdots \times U_{i_k}$ is a fuzzy relation Q_P in $U_{i_1} \times \cdots \times U_{i_k}$, defined by the membership function:

$$\mu_{Q_P}(u_{i_1}, \cdots, u_{i_k}) = \max_{u_{j1} \in U_{j1}, \cdots, u_{j(n-k)} \in U_{j(n-k)}} \mu_Q(u_1, \cdots, u_n), \tag{2.18}$$

where $\{u_{j1}, \cdots, u_{j(n-k)}\}$ is the complement of $\{u_{i_1}, \cdots, u_{i_k}\}$ with respect to $\{u_1, \cdots, u_n\}$.

If Q is a binary relation in $U_1 \times U_2$, then the projection of Q on U_1, denoted by Q_1, is a fuzzy set in U_1 with membership:

$$\mu_{Q_1} = \max_{u_2 \in U_2} \mu_Q(u_1, u_2). \tag{2.19}$$

Similarly, the projection of Q on U_2, denoted by Q_2, is a fuzzy set in U_2 with membership:

$$\mu_{Q_2} = \max_{u_1 \in U_1} \mu_Q(u_1, u_2). \tag{2.20}$$

This definition is valid if Q is a crisp relation, as can be verified for the relation D and its projection D_1 defined by Equations 2.16 and 2.17. It is a natural extension of the projection for a crisp relation. An illustration of a fuzzy binary relation follows.

ILLUSTRATION 2.6.1

Referring to Example 2.5.1 and using Equations 2.19 and 2.20, the two projections R_1 and R_2 using Zadeh's notation are:

$$R_1 = 0.02/-2.0 + 0.37/-1.0 + 1.00/0.0 + 0.37/1.0 + 0.02/2.0,$$

and

$$R_2 = 0.02/-2.0 + 0.37/-1.0 + 1.00/0.0 + 0.37/1.0 + 0.02/2.0.$$

Definition 2.6.1 was given for the case when sets $U_k, k = 1, \cdots n$ were finite or countable. In the case when these sets are continuous universes of discourse, Equation 2.18 is written in the form:

$$\mu_{Q_P}(u_{i_1}, \cdots, u_{i_k}) = \int_{U_{i_1} \times \cdots \times U_{i_k}} \sup_{(u_{j1} \in U_{j1}, \cdots, u_{j(n-k)} \in U_{j(n-k)})} \mu_Q(u_{i_1}, \cdots, u_{i_n})/(u_{i_1}, \cdots, u_{i_k}), \tag{2.21}$$

where $\{u_{j1}, \cdots, u_{j(n-k)}\}$ is the complement of $\{u_{i_1}, \cdots, u_{i_k}\}$ with respect to $\{u_1, \cdots, u_n\}$.

For a binary relation Q on $U_1 \times U_2$, and its projections on U_1 and U_2, this definition reduces to the formulae:

$$\mu_{Q_1}(u_1) = \int_{U_1} \sup_{u_2} \mu_Q(u_1, u_2)/u_1,$$

and

$$\mu_{Q_2}(u_2) = \int_{U_2} \sup_{u_1} \mu_Q(u_1, u_2)/u_2.$$

The projection of a fuzzy relation constrains a fuzzy relation to a subspace in its domain of definition. The concept of cylindrical extension extends a fuzzy relation (or fuzzy set) from a base subspace to space of higher dimension. This concept of cylindrical extension is given in the following definition.

DEFINITION 2.6.2

Let Q be a fuzzy relation in $U_{i_1} \times \cdots \times U_{i_k}$ and $\{i_1, \cdots, i_k\}$ be a subsequence of the indices $\{1, 2, \cdots, n\}$, then the cylindrical extension of Q on $U_1 \times \cdots \times U_n$ is a fuzzy relation QE in $U_1 \times \cdots \times U_n$, defined by the membership function:

$$\mu_{QE}(u_1, \cdots, u_n) = \mu_Q(u_{i_1}, \cdots, u_{i_k}), \tag{2.22}$$

where $\{u_{j1}, \cdots, u_{j(n-k)}\}$ is the complement of $\{u_{i_1}, \cdots, u_{i_k}\}$ with respect to $\{u_1, \cdots, u_n\}$. Using Zadeh's notation, for continuous U_j:

$$QE = \int_{U_1 \times \cdots \times U_n} \mu_Q(u_{i_1}, \cdots, u_{i_k})/(u_1, \cdots, u_n), \tag{2.23}$$

and for finite or countable U_j:

$$QE = \sum_{U_1 \times \cdots \times U_n} \mu_Q(u_{i_1}, \cdots, u_{i_k})/(u_1, \cdots, u_n). \tag{2.24}$$

In the special case of a fuzzy relation A defined on the universe of discourse U_1, its cylindrical extension to the fuzzy relation AE defined on $U_1 \times U_2$ is defined by:

- For continuous U_1 and U_2:

$$AE = \int_{U_1 \times U_2} \mu_A(u_1)/(u_1, u_2). \tag{2.25}$$

- For countable or finite U_1 and U_2:

$$AE = \sum_{U_1 \times U_2} \mu_A(u_1)/(u_1, u_2). \tag{2.26}$$

This definition is also valid for crisp sets, and it is illustrated in the following example.

EXAMPLE 2.6.1

Referring to Example 2.5.1, and Illustration 2.6.1, the cylindrical extensions RE_1 of R_1 on $U_1 \times U_2$ and RE_2 of R_2 on $U_1 \times U_2$ are:

$RE_1 =$

		U_2				
		-2.00	-1.00	0.00	1.00	2.00
U_1	-2.00	0.02	0.02	0.02	0.02	0.02
	-1.00	0.37	0.37	0.37	0.37	0.37
	0.00	1.00	1.00	1.00	1.00	1.00
	1.00	0.37	0.37	0.37	0.37	0.37
	2.00	0.02	0.02	0.02	0.02	0.02

$RE_2 =$

		U_2				
		-2.00	-1.00	0.00	1.00	2.00
U_1	-2.00	0.02	0.37	1.00	0.37	0.02
	-1.00	0.02	0.37	1.00	0.37	0.02
	0.00	0.02	0.37	1.00	0.37	0.02
	1.00	0.02	0.37	1.00	0.37	0.02
	2.00	0.02	0.37	1.00	0.37	0.02

Let $A_1, \cdots, A_n$ be fuzzy sets in $U_1, \cdots, U_n$, respectively. The *Cartesian product* of the $A_k, k = 1, \cdots n$, written as $A_1 \times \cdots \times A_n$, is a fuzzy relation in $U_1 \times \cdots \times U_n$. It has membership defined by:

$$\mu_{A_1 \times \cdots \times A_n}(u_1, \cdots, u_n) = \mu_{A_1}(u_1) \wedge \cdots \wedge \mu_{A_n}(u_n), \tag{2.27}$$

where $\wedge$ represents any T-norm.

Using the min T-norm, the reconstructed relation R^* is:

$R^* =$

		U_2				
		-2.00	-1.00	0.00	1.00	2.00
U_1	-2.00	0.02	0.02	0.02	0.02	0.02
	-1.00	0.02	0.37	0.37	0.37	0.02
	0.00	0.02	0.37	1.00	0.37	0.02
	1.00	0.02	0.37	0.37	0.37	0.02
	2.00	0.02	0.02	0.02	0.03	0.02

The cylindrical extension principle results in larger R^* than the original R fuzzy relations. The following Lemma's are given without proof.

LEMMA 2.6.1

*let A and B be fuzzy sets in a universe of discourse U. Let fuzzy set $Q \subset U$ be a subset of each of the fuzzy sets A and B, that is $Q \subset A$ and $Q \subset B$. Then provided "*min*" is used for the intersection norm, $Q \subset A \cap B$.*

LEMMA 2.6.2

If Q is a fuzzy relation in $U_1 \times \cdots \times U_n$ and $Q_1, \cdots, Q_n$ are its projections on $U_1, \cdots, U_n$ respectively, then:

$$Q \subset Q_1 \times \cdots \times Q_n, \tag{2.28}$$

*where "*min*" is the T-norm used for the membership function of $Q_1 \times \cdots \times Q_n$ in Equation 2.27.*

2.7 Composition of Fuzzy Relations

The definitions of fuzzy relations can be extended to the composition of fuzzy relations as is true in the theory of crisp sets and relations/functions. Indeed this can be done at the same time as maintaining the mathematics for crisp sets and relations. The combinations of fuzzy sets and fuzzy relations with the aid of cylindrical extensions and projections is called *composition*.

First recall that if $P : U \to V$ and if $Q : V \to W$ are two crisp binary relations, in which the range of P in V contains the domain of Q in V, then the composition of the two relations, written $P \circ Q$ can be defined. Using set notation, $P \subset U \times V$ and $Q \subset V \times W$, then the definition of the composition $P \circ Q$ is:

$$P \circ Q = \{(u, w)| \text{ if } \exists\, v \in V \text{ such that } (u, v) \in P \text{ and } (v, w) \in Q\} \subset U \times W. \tag{2.29}$$

The following lemma is proven for crisp sets and relations using the characteristic membership function. This result identifies the composition of two relations through its membership function. It is given for countable or finite sets only.

LEMMA 2.7.1

Given crisp sets U, V, W and relations $P : U \to V$ and $Q : V \to W$ then $P \circ Q$ is the composition of P and Q as defined by Equation 2.29 if and only if

$$\mu_{P \circ Q}(u, w) = \max_{v \in V} t[\mu_P(u, v), \mu_Q(v, w)] \;\; \forall (u, w) \in U \times W. \tag{2.30}$$

The result holds for crisp sets and relations, using an arbitrary T-norm, it can be extended to the concept of composition through the membership function defined in Equation 2.30.

DEFINITION 2.7.1

Given sets U, V, W in appropriate universes of discourse, and fuzzy relations $P : U \to V$ and $Q : V \to W$ then $P \circ Q$ is the composition of fuzzy relations P and Q and its membership function is defined by Equation 2.30.

This definition incorporates an arbitrary T-norm, and so for each one selected a different composition of the binary relations is obtained. With the notation given above, the two most commonly used compositions in the literature are:

- The *max-min composition* of fuzzy relations P and Q is the fuzzy relation $P \circ Q$, defined by the membership function

$$\mu_{P \circ Q}(u, w) = \max_{v \in V} \min(\mu_P(u, v), \mu_Q(v, w)) \ \ \forall (u, w) \in U \times W. \tag{2.31}$$

- The *max-product composition* of fuzzy relations P and Q is the fuzzy relation $P \circ Q$, defined by the membership function

$$\mu_{P \circ Q}(u, w) = \max_{v \in V} \mu_P(u, v) \mu_Q(v, w) \ \ \forall (u, w) \in U \times W. \tag{2.32}$$

These definitions use the minimum and algebraic product for the T-norm.

LEMMA 2.7.2

Let A be a fuzzy set defined on U_1 and consider a fuzzy relation $Q : U_1 \to U_2$. Then the composition $B = A \circ Q$ is the fuzzy set B defined on U_2 given by:

$$B = A \circ Q = (AE \cap Q)_2.$$

That is, B is the projection of the relation obtained by forming the intersection of the cylindrical extension of A with the relation (fuzzy set) Q, onto U_2.

Proof. This result is verified by establishing equality of membership functions using the formulae above. Let t be any T-norm, then by Equation 2.30,

$$\mu_{A\circ Q}(u_2) = \max_{u_1 \in U_1} t[\mu_A(u_1), \mu_Q(u_1, u_2)] \ \ \forall u_2 \in U_2.$$

By definition of the cylindrical extension of A to $U_1 \times U_2$,

$$\mu_{AE}(u_1, u_2) = \mu_A(u_1) \ \ \forall (u_1, u_2) \in U_1 \times U_2.$$

Using the T-norm the membership of the relation $AE \cap Q$ is just

$$\mu_{AE\cap Q}(u_1, u_2) = t[\mu_A(u_1), \mu_Q(u_1, u_2)] \ \ \forall (u_1, u_2) \in U_1 \times U_2.$$

Now using the definition for membership of the projection of $AE \cap Q$ on U_2

$$\mu_{(AE\cap Q)_2}(u_2) = \max_{u_1 \in U_1} t[\mu_A(u_1), \mu_Q(u_1, u_2)] \ \ \forall u_2 \in U_2.$$

This establishes that $B = A \circ Q$ equals $(AE \cap Q)_2$, the projection of the fuzzy relation $AE \cap Q$ onto the set U_2.

□

This result now establishes the connection between projection, cylindrical extension and composition.

2.8 Extension Principle

The *extension principle* is a basic identity that allows the domain of a function to be extended from crisp points in U to fuzzy sets in U. Let U, V be crisp sets and consider an onto function $f : U \to V$ whose domain is U.

Suppose there is a fuzzy set $A \subset U$. It is possible to induce a fuzzy set $B \subset V$ such that $B = f(A)$ in the following manner. If f is a $1-1$ function, then the fuzzy set B can be defined as:

$$\mu_B(v) = \mu_A\left[f^{-1}(v)\right], \ \forall v \in V,$$

where $f^{-1} : V \to U$ is the unique inverse of f, that is $f^{-1}(f(u)) = u$ for all $u \in U$.

Consider a universe $U = \{u_1, \cdots, u_n\}$ and fuzzy set A defined as:

$$A = \mu_1/u_1 + \mu_2/u_2 + \cdots + \mu_n/u_n.$$

The above definition implies:

$$\begin{aligned} f(A) &= f(\mu_1/u_1 + \mu_2/u_2 + \cdots + \mu_n/u_n) \\ &= \mu_1/f(u_1) + \mu_2/f(u_2) + \cdots + \mu_n/f(u_n). \end{aligned} \tag{2.33}$$

This describes the fuzzy set $B = f(A) \subset V$ under the mapping f.

If f is not $1-1$, then it is possible that for a given $v \in V$ there are more than one $u \in U$ such that $f(u) = v$. In this case an ambiguity arises in how to determine a unique value for $\mu_B(v)$, see Figure 2.9.

This figure illustrates an example for which sets U and V have continuous definition. It can be seen that for the indicated $v \in V = [0, 1]$, that there exists three values $u_1, u_2, u_3 \in U = [0, 3]$ such that $f(u_k) = v, k = 1, 2, 3$. This can be resolved by using the following more general formula for the membership function of B:

$$\mu_B(v) = \max_{u \in f^{-1}(v)} \mu_A\left[f^{-1}(v)\right], \ \forall v \in V. \tag{2.34}$$

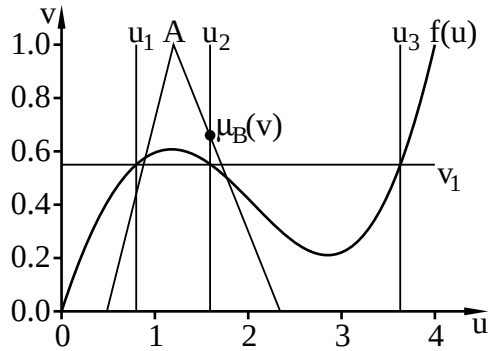


Figure 2.9: Extension Principle: Ambiguity in Defining Membership $\mu_B(v)$

The notation used here is:

$$f^{-1}(v) = \{u|\ f(u) = v \text{ and } u \in U\}.$$

Hence in Figure 2.9, the value u_2 is used to determine the value for $\mu_B(v)$.

The identity in Equation 2.34 is called the *extension principle*, and shows how a fuzzy set in V using a function $f : U \to V$ with fuzzy set $A \subset U$ as input can be induced. It was first introduced by Zadeh (1978).

More importantly the principle allows the generalisation of crisp mathematical concepts to the fuzzy set framework and extends point to point functions or mappings to mappings for fuzzy sets.

2.8.1 Linguistic Hedges

It is common in our description of things and actions to use more than one word to describe a variable. For example, "the water is not very hot", "the water is very very hot", "the wind is blowing from around (more or less) the south west", "the car was travelling only slightly fast".

In general the value of a linguistic variable is a concatenation of *atomic* terms, which may be classified into the groups

- *Primary terms* which are labels for fuzzy sets, eg "slow", "hot".
- Conjunctives such as complement "not", and connections "and" and "or".
- *Hedges* such as "very", "slightly", "more or less" etc.

The interpretation of how to characterise the term "hedge" in everyday life varies from one person to another. Primary terms are difficult to define in a strict mathematical environment. To give some meaning to them, a few commonly used definitions follow.

DEFINITION 2.8.1

Given a universe of discourse U and fuzzy subset $A \subset U$, then

- very A *is that subset, commonly referred to as* $A^2 \subset U$*, defined by the membership function:*

$$\mu_{very A}(u) = \mu_{A^2}(u) \ = [\mu_A(x)]^2 \quad \forall u \in U, \tag{2.35}$$

- more or less A *is that subset, commonly referred to as* $A^{1/2} \subset U$*, defined by the membership function:*

$$\mu_{moreorless A}(u) = \mu_{A^{1/2}}(u) \ = [\mu_A(x)]^{1/2} \quad \forall u \in U, \tag{2.36}$$

These hedges *very* and *more or less* are also known as *concentration* and *dilation*, respectively. Some texts use the alternative notation: $A^2 = CON(A)$ and $A^{1/2} =$

$DIL(A)$ for concentration and dilation respectively. Also, the hedge *more or less* is sometimes called *fairly*. Other hedges are in use and the reader is directed to [3] for a more extensive list.

2.9 Fuzzy If-Then Rules

Linguistic variables have thus far been introduced, it is now time to turn our discussion to "Fuzzy if-then rules".

Our description of our environment and our actions in moving in this environment, is most usually characterised by words and descriptive language rules. For example, looking from my window, I see that the "the sky is very clear", there is a "moderate breeze blowing", and can say the today's "temperature is cool". I certainly do not know the exact temperature unless I check a thermometer, and for that matter I am really not concerned with the exact temperature.

Temperature is described by various phrases, such as "today's temperature is hot", "today's temperature is very hot", "today's temperature is cold" and "today's temperature is very cold". Actions to these statements can be stated in simple rules such as: "**if** the temperature is hot **then** turn on the air-conditioning", "**if** the temperature is hot **then** change the air-conditioning to very cold".

Consider driving a car so that it stays in the middle of a lane. Typically, a driver directs a car along the centre of the carriageway. Therefore, this controller only needs to minimise the distance between the centre of the car and the centre of the carriageway. The term *distance from centre* is expressed in terms such as:

- "centre", if the car is in the centre of the lane.
- "small left", if it is a little to the left of centre.
- "small right", if it is a little to the right of centre.
- etc.

Distance serves to define positioning of the car with respect to the centre of the carriageway. The term *action* is expressed as:

- "straight", steer straight ahead.
- "small left", steer a little to the left.
- "large left", steer a large amount to the left.
- etc.

Action serves to define the change in steering at the steering wheel to bring the car to the centre of the carriageway.

Typical *rules* to control the car are:

1. **If** distance is "zero" **then** action is "straight".
2. **If** distance is "small left" **then** action is "small right".
3. etc.

These rules are typical of those that a human would use to control the car and further more use in describing our control of the vehicle to someone else. Indeed

humans control the car without precise measurements of location and action of a knowledge of a mathematical model. Similarly, a human changes the temperature of an air-conditioning unit using linguistic terms.

A driver observing the speedometer is measuring in terms of; is the speed below, at or above the speed limit? A driver cannot safely stare at an analogue instrument to determine that the exact speed is: 48 ± 5km/h. The exact speed is inconsequential, but maintaining speed that is below the speed limit and fast enough as not to infuriate other drivers is important. A driver is able to maintain speed in a car with a few linguistic variables as input.

The rules above are of the form "**If** A **Then** B", where A is called the *antecedent* and B is called the *consequent*. A set of these rules is called a *knowledge base*, or *fuzzy knowledge base* for the control of this problem.

Location is called a linguistic variable because it takes values such as "centre", "small left", etc., which are linguistic and not numeric. "Centre" and "small left" for example are called linguistic values.

These concepts can be defined as:

DEFINITION 2.9.1

If a variable takes words in a natural language as its values, it is called a linguistic *variable.*

A linguistic variable is characterised by

- its name. For example, the speed of the car.
- its set of linguistic values. For example { slow, medium, fast }.
- its domain of definition. For example the interval $U = [0, 100]$.
- a semantic rule. This rule relates for example what "fast" means in terms of a membership function.

Fuzzy **if-then** rules encapsulate human linguistic knowledge and can be expressed generally in the form:

$$\textbf{if} \ < \textit{fuzzy proposition} > \ \textbf{then} < \textit{fuzzy proposition} > \tag{2.37}$$

Relating fuzzy rules to fuzzy membership sets is made through fuzzy propositions.

2.9.1 Fuzzy Propositions

There are two types of fuzzy propositions, namely *atomic* and *compound* propositions. An *atomic* fuzzy proposition is just a single statement such as

$$u \text{ is } A \qquad \text{or} \qquad u \in A.$$

Atomic propositions are commonly written as "u is A" as a synonym of "$u \in A$". Another common usage of notation synonymous with "$u \in A$" is "$u = A$".

A *compound* fuzzy proposition is a composition of atomic fuzzy propositions using the connectives "and", "or", and "not", which represent fuzzy intersection, fuzzy union and fuzzy complement, respectively.

Consider the following problem to control a simulated mobile robot R in a workspace $X \times Y \in [0, 100] \times [0, 100]$ to reach a target T at $(50, 100)$ with heading angle $\phi = \pi/2$ *rad*. The control variable is a steering angle correction $\theta \in [-\pi/6, \pi/6]$ *rad*. The domain for ϕ is in the interval $[-\pi, \pi)$ *rad*. The initial configuration of the robot is: $R_x = 20, R_y = 20$, and, $R_\phi = \pi/4$ as shown in Figure 2.10. Recursive point mass kinematic equations for the simulated robot with constant velocity v are:

$$\begin{aligned} R_\phi(t+1) &= R_\phi(t) + \theta(t) \\ R_x(t+1) &= R_x(t) + v\cos(R_\phi(t+1)) \\ R_y(t+1) &= R_y(t) + v\sin(R_\phi(t+1)) \end{aligned}$$

where time t is updated by $t_k = t_0 + k\Delta t$ for $k = 1, \cdots, T$ and T is the total time of the simulation.

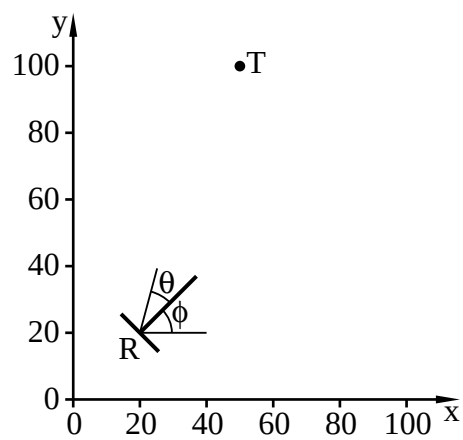


Figure 2.10: Simulated Robot in Workspace

Let R_x be divided into five triangular fuzzy sets: LE = *Left*, LC = *Left Centre*, C = *Centre*, CR = *Centre Right* and RI = *Right*. Five triangular fuzzy sets for R_y: FB = *Far Below*, BC = Below Centre, C = *Centre*, CA = *Centre Above* and FA = *Far Above*. Five triangular fuzzy sets for R_ϕ defined as: R = *Right*, CR = *Centre Right*, C = *Centre*, LC = *Left Centre*, L = *Left*. Also, seven triangular

fuzzy sets for steering angle correction θ defined as: NB = *Negative Big*, NM = *Negative Medium*, NS = *Negative Small*, ZE = *Zero*, PS = *Positive Small*, PM = *Positive Medium*, and PB = *Positive Big*.

A typical fuzzy rule to control the mobile robot,

$$\textbf{if } (R_x \text{ is LC and } R_y \text{ is BC and } R_\phi \text{ is CR}) \textbf{ then } \theta \text{ is PB},$$

says that if the robot is positioned at R_x = *Left Centre* and R_y = *Below Centre*, and has heading angle R_ϕ = *Centre Right*, then apply a steering correction to the vehicle θ= *Positive Big*. Angles are taken to be measured positive anti-clockwise.

This fuzzy if-then rule includes the compound fuzzy proposition:

$$(R_x \text{ is LC and } R_y \text{ is BC and } R_\phi \text{ is CR}).$$

Other compound fuzzy propositions could be:

$$((R_x \text{ is LC and } R_y \text{ is not BC}) \text{ and } R_\phi \text{ is CR}),$$

or,

$$((R_x \text{ is LC or } R_y \text{ is BC}) \text{ and } R_\phi \text{ is not CR}).$$

Both atomic and compound fuzzy propositions are fuzzy relations, and therefore, have associated membership functions. The membership function of a compound proposition is determined by the use of:

- fuzzy intersection for the connective "and" with an appropriate T-norm,
- fuzzy union for the connective "or" with an appropriate S-norm, and
- fuzzy complement for the connective "not" with an appropriate complement norm.

ILLUSTRATION 2.9.1

Consider the two fuzzy propositions:

- FP_1 = *(R_x is LC and R_y is BC and R_ϕ is CR), and*
- FP_2 = *((R_x is not LC and R_y is BC) or R_ϕ is CR).*

Using the above outline the membership functions for these two fuzzy relations are:

$$\begin{aligned}
\mu_{FP_1}(x, y, \phi) &= t[t[\mu_{LC}(x), \mu_{BC}(y)], \mu_{CR}(\phi)] \\
&\quad \forall\ (x, y, \phi) \in [0, 100] \times [0, 100] \times [-\pi, \pi), \\
\mu_{FP_2}(x, y, \phi) &= s[t[c(\mu_{LC}(x)), \mu_{BC}(y)], \mu_{CR}(\phi)] \\
&\quad \forall\ (x, y, \phi) \in [0, 100] \times [0, 100] \times [-\pi, \pi).
\end{aligned}$$

Fuzzy propositions can now be interpreted as fuzzy relations/fuzzy sets with appropriately defined membership functions, but still have a measure of choice in the appropriate selection of T-norm, S-norm and complement.

2.9.2 Interpretation of Fuzzy If-Then Rules

Interpretation of the fuzzy **if-then** rule requires a background in propositional calculus of classical set theory. Fuzzy propositions can be re-written in propositional calculus nomenclature by assigning the connectives: "and" = $\wedge$, "or" = $\vee$, and "not" = $\neg$. This section will outline how the antecedent relates to the consequent of fuzzy if-then rules.

Propositional calculus uses implications of the form $p \rightarrow q$, read as p "implies" q. This implication is also written as "**if** p **then** q". For example, such an implication may be:

If today is fine **then** I will be going to town.

with propositions p = "today is fine" and q = "I will be going to town". These propositional statements or variables have values which are only "true" (T) or "false" (F).

The truth table for $p \rightarrow q$ is shown in Table 2.2. It is easy to show that the

Table 2.2: Truth Table for $p \rightarrow q$

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

following three classical propositions: $p \rightarrow q$, $\neg p \vee q$, and $(p \wedge q) \vee \neg q$ are equivalent. These are demonstrated by using truth tables as shown in Table 2.3.

Table 2.3: Truth Table for $\neg p \wedge q$ and $(p \wedge q) \vee \neg p$

p	q	$\neg p$	$p \wedge q$	$\neg p \vee q$	$(p \wedge q) \vee \neg p$
T	T	F	T	T	T
T	F	F	F	F	F
F	T	T	F	T	T
F	F	T	F	T	T

Two propositions p_1 and p_2 are said to be equivalent, written as $p_1 \leftrightarrow p_2$, if the truth table of $p_1 \leftrightarrow p_2$ is a tautology. This means that for the resulting truth value for $p_1 \leftrightarrow p_2$ for any value of p_1 and p_2 is always "True". The double implication $\leftrightarrow$ means that:

$$p_1 \leftrightarrow p_2 = (p_1 \leftarrow p_2) \wedge (p_1 \rightarrow p_2).$$

A generalisation can now be made using the above discussion. Given fuzzy propositions FP_1 and FP_2 in the implication:

$$\textbf{if } FP_1 \textbf{ then } FP_2, \tag{2.38}$$

can now be replaced by the equivalence of:

$$\neg FP_1 \vee FP_2, \qquad \text{or} \tag{2.39}$$

$$(FP_1 \wedge FP_2) \vee \neg FP_1. \tag{2.40}$$

where $\neg$ is interpreted as fuzzy complement, $\vee$ as fuzzy union, and $\wedge$ as fuzzy intersection.

The fuzzy proposition FP_1 is called the *rule antecedent* and the fuzzy proposition FP_2 is called the *rule consequent*.

Using this generalisation allows many interpretations of the implication from the wide variety of fuzzy compliment, fuzzy union, and fuzzy intersection operators.

Also, compound fuzzy propositions such as FP_1 being a relation in $U_1 \times U_2 \times \cdots \times U_n$ and FP_2 being a relation in $V_1 \times V_2 \times \cdots \times V_M$ are well defined in the implication. It is customary to use linguistic terms when defining $u \in U$ and $v \in V$.

Some commonly used implications are:

- **Dienes-Rescher Implication:** This implication is obtained by replacing the $\neg$ and $\vee$ in Equation 2.39 with the basic fuzzy complement and the max operator of the fuzzy union, respectively. The fuzzy implication Equation 2.38 is interpreted as the fuzzy relation Q_D, it being a fuzzy subset of $U \times V$ with membership function:

$$\mu_{Q_D}(u, v) = \max(1 - \mu_{FP_1}(u), \mu_{FP_2}(v)) \quad \forall \ (u, v) \in U \times V. \tag{2.41}$$

- **Lukasiewicz Implication:** This implication is obtained by replacing the $\neg$ and $\vee$ in Equation 2.39 with the basic fuzzy complement and the Yager S-norm with $\omega = 1$ for the fuzzy union, respectively. The fuzzy implication Equation 2.38 is interpreted as the fuzzy relation Q_L, it being a fuzzy subset of $U \times V$ with membership function:

$$\mu_{Q_L}(u, v) = \min(1, 1 - \mu_{FP_1}(u) + \mu_{FP_2}(v)) \quad \forall \ (u, v) \in U \times V. \tag{2.42}$$

- **Zadeh Implication:** This implication is obtained by replacing the $\neg$, $\vee$ and $\wedge$ in Equation 2.40 with the basic fuzzy complement, the max operator for the fuzzy union, and the min operator for fuzzy intersection, respectively. The fuzzy implication Equation 2.38 is interpreted as the fuzzy relation Q_Z, it being a fuzzy subset of $U \times V$ with membership function:

$$\mu_{Q_Z}(u, v) = \max(\min(\mu_{FP_1}(u), \mu_{FP_2}(v)), 1 - \mu_{FP_1}(u)) \quad \forall \ (u, v) \in U \times V. \tag{2.43}$$

As mentioned above these are just a few commonly used fuzzy implications, there are many others that can be formed.

Generalising the use of classical implication from crisp to fuzzy sets has introduced the problem of "local" and "global" interpretations.

2.9.3 Local and Global Implication

One major issue arises from the previous generalisation. Are the propositions $p \rightarrow q$, $\neg p \vee q$, and $(p \wedge q) \vee \neg q$ still "equivalent" when considering fuzzy propositions?

The crisp set analysis of the propositions are *global* because every combination of the arguments can be examined in a truth table. In moving from crisp to fuzzy analysis, the dichotomy of "true" and "false" is replace by a continuous function. It is no longer possible to examine the equivalence of: $p \rightarrow q$, $\neg p \vee q$, and $(p \wedge q) \vee \neg q$ using truth tables. The fuzzy implication becomes *local* in the sense that the implication has large truth *only* when both FP_1 and FP_2 have large truth values. For example, when making a statement such as:

"**if** robot speed is *high* and distance from obstacle is *close*
then set control steering angle *high*"

only implies local information about the situation. There is no information for when robot speed is **not** *high* or distance from obstacle is **not** *close*. Therefore, such a fuzzy **if-then** rule Equation 2.38,

$$\textbf{if } FP_1 \textbf{ then } FP_2, \tag{2.44}$$

should be interpreted as:

$$\textbf{if } FP_1 \textbf{ then } FP_2 \textbf{ else } NOTHING, \tag{2.45}$$

where $NOTHING$ means that this rule does not exist. Nothing else is known. In classic logic terms this means:

$$p \rightarrow q = p \wedge q. \tag{2.46}$$

One may not agree with this argument, if one considers the implication:

"**if** speed of the rocket is *high* **then** air resistance is *high*".

This statement implicitly implies or infers that:

"**if** speed of the rocket is *low* **then** air resistance is *low*."

In this sense one could argue that fuzzy implications are global operators. In such a case, Equations 2.39 and 2.40 are used to obtain various forms of fuzzy implications.

2.9.4 Mamdani Implications

Equation 2.46 can now be used in defining local implications as:

$$\textbf{if } FP_1 \textbf{ then } FP_2 = FP_1 \wedge FP_2 \tag{2.47}$$

Using min or *algebraic product* for the $\wedge$ operator gives the *Mamdani* implications.

Mamdani Implications are obtained by replacing the $\wedge$ in Equation 2.47 with min or *algebraic product* for the operator $\wedge$. The fuzzy implication Equation 2.47 is interpreted as fuzzy relations Q_{MM} (min) and Q_{MP} (product), being fuzzy subsets of $U \times V$ with membership functions:

$$\mu_{Q_{MM}}(u,v) = \min(\mu_{FP_1}(u), \mu_{FP_2}(v)) \quad \forall \ (u,v) \in U \times V, \tag{2.48}$$

or

$$\mu_{Q_{MP}}(u,v) = \mu_{FP_1}(u)\mu_{FP_2}(v) \quad \forall \ (u,v) \in U \times V. \tag{2.49}$$

The Mamdani implications are widely used in fuzzy systems and fuzzy control as they are local implications and are easily calculated.

EXAMPLE 2.9.1

Consider the fuzzy rule defined earlier in discussing the fuzzy control of the mobile robot,

if *(R_x is LC and R_y is BC and R_ϕ is CR)* **then** *θ is PB.*

Let the fuzzy sets, LC, BC, CR and PB be defined as follows:

$$\begin{array}{rcll}
\mu_{LC}(x) & = & \exp(-((x-25)/15)^2 & \forall x \in [0,100], \\
\mu_{BC}(y) & = & \exp(-((y-25)/15)^2 & \forall y \in [0,100], \\
\mu_{CR}(\phi) & = & \exp(-(3(\phi+\pi/2)/\pi)^2 & \forall \phi \in [-\pi,\pi), \\
\mu_{PB}(\theta) & = & \exp(-(30(\theta-\pi/6)/\pi)^2 & \forall \theta \in [-\pi/6,\pi/6].
\end{array}$$

The relation Q_{MP} defining the Mamdani implication using the algebraic product, Equation 2.49, for $\wedge$ is defined by:

$$\begin{aligned}
\mu_{Q_{MP}}(x,y,\phi,\theta) &= e^{(-((x-25)/15)^2} e^{(-((y-25)/15)^2} e^{(-(3(\phi+\pi/2)/\pi)^2} e^{(-(30(\theta-\pi/6)/\pi)^2}, \\
&\qquad \forall (x,y,\phi,\theta) \in [0,100] \times [0,100] \times [-\pi,\pi) \times [-\pi/6,\pi/6].
\end{aligned}$$

2.10 Inference

There are three commonly used rules of *inference* in classical logic, namely Modus Ponens, Modus Tollens and Hypothetical Syllogism. Their basic description follows:

- **Modus Ponens:** This inference rule states that given the truth of two propositions p and $p \rightarrow q$, the truth of the proposition q should be inferred. Symbolically, the inference is represented by

$$p \wedge (p \rightarrow q) \rightarrow q. \tag{2.50}$$

The propositions p and $p \rightarrow q$ are called the *premises* of the inference rule and q is called the *conclusion*. Figure 2.11 describes the truth table for this inference rule. Observe that this inference rule is always "True", no matter

p	q	$p \rightarrow q$	$p \wedge (p \rightarrow q)$	$p \wedge (p \rightarrow q) \rightarrow q$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

Figure 2.11: Truth Table for $p \wedge (p \rightarrow q) \rightarrow q$

what the truth values of the propositions p and q. It is a tautology. The truth of q can be inferred from the truth of the proposition $p \wedge (p \rightarrow q)$. The modus ponens is closely related to forward data driven inference and can be written as:

Premise 1: p is True.
Premise 2: **if** $p \rightarrow q$ is True.
Consequence: q is True.

- *Modus Tollens:* This inference rule states that given the truth of two propositions $\neg q$ and $p \rightarrow q$, the truth of the proposition $\neg p$ should be inferred.

Symbolically, the inference is represented by:

$$\neg q \wedge (p \rightarrow q) \rightarrow \neg p. \tag{2.51}$$

Figure 2.12 describes the truth table for this inference rule. This inference

p	q	$\neg p$	$\neg q$	$p \rightarrow q$	$\neg q \wedge (p \rightarrow q)$	$\neg q \wedge (p \rightarrow q) \rightarrow \neg p$
T	T	F	F	T	F	T
T	F	F	T	F	F	T
F	T	T	F	T	F	T
F	F	T	T	T	F	T

Figure 2.12: Truth Table for $\neg q \wedge (p \rightarrow q) \rightarrow \neg p$

rule is also a tautology. The truth of p can be inferred from the truth of the proposition $\neg q \wedge (p \rightarrow q)$. The modus tollens is closely related to backward data driven inference. The inference of modus tollens can be described by:

Premise 1: $\neg q$ is True.
Premise 2: **if** $p \rightarrow q$ is True.
Consequence: $\neg p$ is True.

- *Hypothetical Syllogism* This inference rule states that given the truth of two propositions $p \rightarrow q$ and $q \rightarrow r$, the truth of the proposition $p \rightarrow r$ should be inferred. Symbolically, the inference is represented by:

$$(p \rightarrow q) \wedge (q \rightarrow r) \rightarrow (p \rightarrow r). \tag{2.52}$$

Figure 2.13 describes the truth table for this inference rule. where $p_1 = p \rightarrow q$, $p_2 = q \rightarrow r$, and $p_3 = p \rightarrow r$. This inference rule is also a tautology. The inference hypothetical syllogism can be described by:

Premise 1: **if** $p \rightarrow q$ is True.
Premise 2: **if** $q \rightarrow r$ is True.
Consequence: $p \rightarrow r$ is True.

In classical logic propositions such as "u is A" have only values "True" or "False". But the notation fits well with the fuzzy set notation already developed. The next step is to provide a foundation for approximate reasoning through an extension of these propositions using fuzzy set theory.

p	q	r	p_1	p_2	p_3	$p_1 \wedge p_2$	$p_1 \wedge p_2 \rightarrow p_3$
T	T	T	T	T	T	T	T
T	T	F	T	F	F	F	T
T	F	T	F	T	T	F	T
T	F	F	F	T	F	F	T
F	T	T	T	T	T	T	T
F	T	F	T	F	T	F	T
F	F	T	T	T	T	T	T
F	F	F	T	T	T	T	T

Figure 2.13: Truth Table for $(p \rightarrow q) \wedge (q \rightarrow r) \rightarrow (p \rightarrow r)$

2.10.1 Compositional Rule of Inference

To develop the compositional rule of inference, Lemma 2.7.2 is restated here with a slightly modified notation.

LEMMA 2.10.1

Let A be a fuzzy set defined on U and consider a fuzzy relation $Q : U \rightarrow V$. Then the composition $B = A \circ Q$ is the fuzzy set B defined on V given by:

$$B = A \circ Q = (AE \cap Q)_2. \tag{2.53}$$

That is, B is the projection of the relation obtained by forming the intersection of the cylindrical extension of A with the relation (fuzzy set) Q, onto V.

For any t be any T-norm calculation of the membership of B yields:

$$\mu_B(v) = \mu_{A \circ Q}(v) = \mu_{(AE \cap Q)_2}(v) = \max_{u \in U} t[\mu_A(u), \mu_Q(u, v)] \ \ \forall v \in V. \tag{2.54}$$

For continuous universes, sup replaces max giving:

$$\mu_B(v) = \mu_{A \circ Q}(v) = \mu_{(AE \cap Q)_2}(v) = \sup_{u \in U} t[\mu_A(u), \mu_Q(u, v)] \ \ \forall v \in V. \tag{2.55}$$

Equation 2.55 is called the *compositional rule of inference*. So given a fuzzy set A in U, the fuzzy relation Q infers a membership for fuzzy set B in V.

A fuzzy **if-then** rule such as:

$$\textbf{if } u \text{ is } A \textbf{ then } v \text{ is } B, \tag{2.56}$$

can be interpreted as a fuzzy relation Q in the Cartesian product $U \times V$.

The premise and conclusion of Modus Ponens in the fuzzy interpretation satisfy the above Lemma, recognising the relation Q as the interpretation of the implication in Premise 2. It is a simple matter to extend the results in this Lemma to handle the inference of Hypothetical Syllogism.

Given Modus Ponens 2.10, described by:

Modus Ponens:

Premise 1:	u is A.
Premise 2:	**if** u is A **then** v is B.
Consequence:	v is B.

it is clear what is being inferred from the two premises. However even though this be the case, our reasoning in daily situations is not so clearly defined. Intuitive reasoning would state: if u is *very high*, that is u is A^2, then given Premise 2, it would be normal to expect that v is *very high* (v is B^2), or at least v is B. Similarly if u is *more or less high*, then given Premise 2, one would reason that v is *more or less high*.

The direct translation of Modus Ponens to fuzzy sets and relations does not give approximate reasoning. The inference rules need to be generalised to incorporate intuitive reasoning.

2.11 Generalised Inference Rules

The following *generalised Modus Ponens*, *generalised Modus Tollens* and *generalised Hypothetical Syllogism* have been proposed.

2.11.1 Generalised Modus Ponens

Using the notation introduced above the generalisation of Modus Ponens is undertaken with the new inference rule which states that:

Premise 1:	u is A'.
Premise 2:	**if** u is A **then** v is B.
Consequence:	v is B'.

Generalised Modus Ponens is a fuzzy proposition inferred from two propositions u is A' and **if** u is A **then** v is B, such that the closer A' is to A, the closer B' is to B. Note that A, A' are fuzzy sets in U and B, B' are fuzzy sets in V, A' is NOT the complement A, nor B' the complement of B.

Table 2.4: Intuitive Reasoning in Generalised Modus Ponens

	u is A'	v is B'
criterion P_1	u is A	v is B
criterion P_2	u is *very* A	v is *very* B
criterion P_3	u is *very* A	v is B
criterion P_4	u is *more or less* A	v is *more or less* B
criterion P_5	u is *more or less* A	v is B
criterion P_6	u is not A	v is unknown
criterion P_7	u is not A	v is not B

Table 2.4 shows the intuitive criteria expected by relating Premise 1 to the conclusion in generalised modus ponens. For example, if the relation in Premise P_2 between u is A and v is B, is not strong, then the satisfaction of criteria P_3 and P_5 is allowed, otherwise one would expect the satisfaction of criteria P_2 and P_4. Criteria P_7 has interpretation: **if** u is A **then** v is B, **else** v is not B. This relation is not valid in classical logic but has meaning in the fuzzy context and close to interpretation in everyday reasoning.

Hence given universes of discourse U and V, fuzzy sets A, A' in U, fuzzy sets B in V, and a fuzzy relation $Q = A \rightarrow B$ in $U \times V$, a fuzzy set B' in V is inferred with membership function

$$\mu_{B'}(v) = \sup_{u \in U} t[\mu_{A'}(u), \mu_{A \rightarrow B}(u, v)] \ \ \forall v \in V \tag{2.57}$$

(Replace sup by min for discrete universes of discourse).

2.11.2 Generalised Modus Tollens

With the notation introduced above the generalisation of Modus Tollens is undertaken with the new inference rule which states that:

Premise 1:	v is B'.
Premise 2:	**if** u is A **then** v is B.
Consequence:	u is A'.

Generalised Modus Tollens is a fuzzy proposition inferred from two propositions v is B' and **if** u is A **then** v is B, such that the closer B' is to B, the closer A' is to A.

Table 2.5: Intuitive Reasoning in Generalised Modus Tollens

	v is B'	u is A'
criterion T_1	v is not B	u is not A
criterion T_2	v is not *very* B	u is not *very* A
criterion T_3	v is not *more or less* B	u is not *more or less* A
criterion T_4	v is B	u is unknown
criterion T_5	v is B	u is A

Similar to the generalised modus ponens, the table given in Table 2.5 shows the intuitive criteria which relates Premise 1 to the conclusion in generalised modus tollens. Hence given universes of discourse U and V, fuzzy set A in U, fuzzy sets B, B' in V, and a fuzzy relation $Q = A \to B$ in $U \times V$, a fuzzy set A' in U is inferred with membership function

$$\mu_{A'}(u) = \sup_{v \in V} t[\mu_{B'}(v), \mu_{A \to B}(u, v)] \ \ \forall u \in U. \tag{2.58}$$

(Replace sup by min for discrete universes of discourse).

2.11.3 Generalised Hypothetical Syllogism

With the notation introduced above the generalisation of modus ponens and tollens is undertaken with the new inference rule which states that:

Premise 1: **if** u is A **then** v is B.
Premise 2: **if** v is B' **then** w is C.
Consequence: **if** u is A **then** w is C'.

Given two fuzzy propositions **if** u is A **then** v is B and **if** v is B' **then** w is C, a new fuzzy proposition **if** u is A **then** w is C' is intuitively inferred. Table 2.6 shows the expected intuitive criteria for hypothetical syllogism.

Table 2.6: Intuitive Reasoning in Generalised Hypothetical Syllogism

	v is B'	v is C'
criterion S_1	v is B	w is C
criterion S_2	v is *very* B	w is *more or less* C
criterion S_3	v is *very* B	w is C
criterion S_4	v is *more or less* B	w is *very* C
criterion S_5	v is *more or less* B	w is C
criterion S_6	v is not B	w is unknown
criterion S_7	v is not B	w is not C

To test criteria S_2, with hedge $B' = veryB$, Premise 1 can be modified to **if** u is A^2 **then** v is B^2 and the consequence becomes **if** u is A^2 **then** w is C. Applying the hedge very on the consequent results in: **if** u is A **then** w is $C^{1/2}$ (*more or less* C).

Hence given universes of discourse U, V, and W, fuzzy set A in U, fuzzy sets B, B' in V, fuzzy sets C, C' in W, a fuzzy relation $Q = A \rightarrow B$ in $U \times V$, a fuzzy relation $P = B' \rightarrow C$ in $V \times W$, a fuzzy relation $R = A \rightarrow C'$ in $U \times W$, is inferred with membership function:

$$\mu_R(u, w) = \sup_{v \in V} t[\mu_Q(u, v), \mu_P(v, w)] \ \ \forall (u, w) \in U \times W. \tag{2.59}$$

(For discrete universes of discourse replace sup by max.)

A diverse collection of inference rules is obtained by using different T-norms and implication rules. Not all of these combinations give rise to the intuitive criteria of approximate reasoning upon which the generalisation has been made from: modus ponens, modus tollens, and hypothetical syllogism. It is important therefore to recognise that an appropriate selection if necessary to ensure acceptable approximate reasoning is made in any application.

2.12 Fuzzy Rule Base and Fuzzy Inference Engine

There are three types of fuzzy systems commonly used in the literature:

(i) pure fuzzy systems,

(ii) Takagi-Sugeno-Kang (TSK) fuzzy systems, and

(iii) fuzzy systems with fuzzifier and defuzzifier.

The material presented thus far has laid the ground work in developing fuzzy systems. In particular, the previous section has led to the pure fuzzy system shown in Figure 2.14.

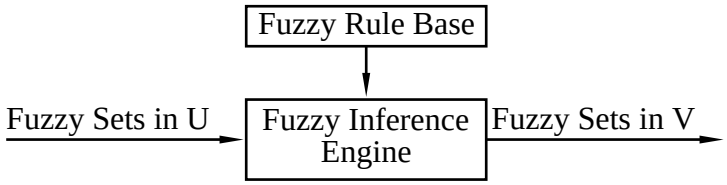


Figure 2.14: Components of a Pure Fuzzy System

The previous discussion developed a fuzzy inference engine consisting of a single rule by approximate reasoning to associate a mapping from fuzzy sets in the input space U to fuzzy sets in the output space V. In reality, the fuzzy rule base consists of many rules, not just a single rule. This section will develop a fuzzy inference engine to handle multiple fuzzy rules.

The discussion that follow will only consider the multi-input-single-output (MISO) systems, as multi-input-multi-output (MIMO) systems can be decomposed into

a collection of MISO systems as shown in Figure 2.15.

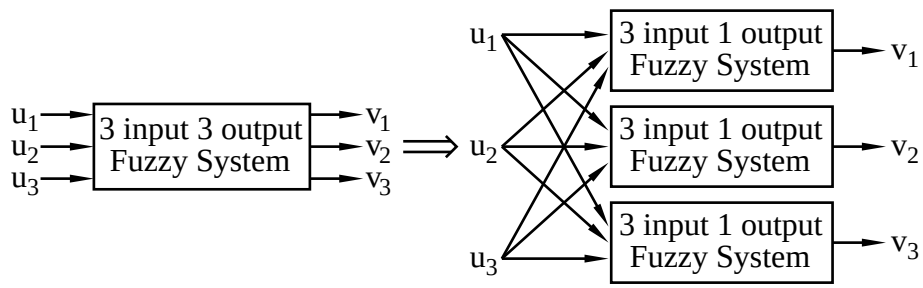


Figure 2.15: A MIMO System Decomposed into MISO Systems

As illustrated, a 3I3O system can be decomposed to three 3I1O systems. Each of these systems can now be treated as separate units.

2.13 Fuzzy Rule Base Structure

The key component of the fuzzy system is the *fuzzy rule base* (FRB) which is just a set of fuzzy **if-then** rules. All other components of the fuzzy system are used to interpret and implement these rules in a reasonable and efficient manner.

The FRB is the set of rules:

$$\{R^k : R^k = \textbf{ if } (u_1 \text{ is } A_1^k) \text{ and } \cdots \text{ and } (u_n \text{ is } A_n^k) \textbf{ then } v \text{ is } B^k,\ k = 1, \cdots, M\}, \tag{2.60}$$

where R^k is a fuzzy implication/relation in $U \times V = (U_1 \times U_2 \times \cdots \times U_n) \times V$, A^i are fuzzy sets in U_i, B^i are fuzzy sets in V, and $u = [u_1 \ \ u_2 \ \ \cdots \ \ u_n]^T \in U$ and $v \in V$ are the input and output linguistic variables of the fuzzy system, respectively. There are M rules in this fuzzy rule base, the rules are called *canonical fuzzy rules* as they include many other types of fuzzy rules and fuzzy propositions as special

cases. A FRB may be written as $FRB : U \rightarrow V$ to represent the mapping from U to V associated with these set of rules.

Canonical fuzzy rules in the form of Equation 2.60, include the following special cases:

- **Partial Rules:**

$$PR^k = \textbf{if } (u_1 \text{ is } A_1^k) \text{ and } \cdots \text{ and } (u_m \text{ is } A_m^k) \textbf{ then } v \text{ is } B^k, \quad (2.61)$$

where $m < n$.

- **Or Rules:**

$$OR^k = \textbf{if } (u_1 \text{ is } A_1^k) \text{ and } \cdots \text{ and } (u_m \text{ is } A_m^k) \text{ or } (u_{m+1} \text{ is } A_{m+1}^k) \text{ and } \cdots \text{ and } (u_n \text{ is } A_n^k) \textbf{ then } v \text{ is } B^k. \quad (2.62)$$

- **Single fuzzy statement**

$$v \text{ is } B^k. \quad (2.63)$$

- **Gradual Rules:**

$$\text{The smaller the } u_i\text{, the larger the } v. \quad (2.64)$$

- **Non-fuzzy Rules:** Conventional standard fuzzy logic implications.

2.13.1 Properties of the Fuzzy Rule Base

Collecting human knowledge in the form of a set of rules, raises some interesting questions. For example, "Do any of the rules conflict?" that is, "Are there rules

with the same antecedent part that have different consequents?". Is this knowledge complete in the sense that all possible rules governing the implication from input to output of the system are covered?

To make these ideas more precise the following concepts are introduced.

DEFINITION 2.13.1

A fuzzy rule base $FRB : U \to V$ is complete if for any $u \in U$, there exists at least one rule in FRB, say rule R^ℓ such that $\mu_{A_i^\ell}(u_i) \neq 0$, for all $i = 1, \cdots, n$.

This definition basically says that the rule base is complete if at any point u of the input space U at least one rule "fires", in the the sense that the membership value of the antecedent part of the rule at this point is non-zero.

DEFINITION 2.13.2

A fuzzy rule base is consistent if there are no rules with the same antecedent part but different consequents.

In classical logic, consistency is an important requirement for conflicting rules means that an reasoning process cannot be progressed. This is not critical in fuzzy logic as the "defuzzifier" averages the conflict to produce a compromised output. Defuzzifiers are discussed in Section 2.17.

DEFINITION 2.13.3

A fuzzy rule base is said to be continuous if there do not exist neighbouring rules in the rule base for whose fuzzy sets in the consequent parts have empty intersection.

Intuitively, for a fuzzy rule base to be continuous, its output must be "smooth".

2.14 Fuzzy Inference Engine

A single fuzzy rule R has been considered that could be interpreted as a fuzzy relation $Q : U \to V$, where U and V are the input and output spaces respectively. A number of reasoning methods such as generalised modus ponens that specified a mapping from a given fuzzy input set A' in U to a fuzzy output set B' in V were examined. The next concept is to make inference and approximate reasoning with a set of many rules. There are two methods discussed in the literature: compositional based inference and individual rule based inference.

2.14.1 Compositional Based Inference

The first method is called *compositional based inference*. All rules are combined into a single fuzzy relation in $U \times V$, which is then viewed as a single fuzzy *if-then* rule.

If the rules are considered as independent conditional implications, then the rules should be combined using the union operator. Otherwise, the rules are considered strongly coupled and all of the rules should be satisfied by using the intersection operator.

Consider the fuzzy rule base FRB consisting of the rules:

$$\{R^k : R^k = \textbf{ if } (u_1 \text{ is } A_1^k) \text{ and } \cdots \text{ and } (u_n \text{ is } A_n^k) \textbf{ then } v \text{ is } B^k,\ k = 1, \cdots, M\}, \tag{2.65}$$

where R^k is a fuzzy implication/relation in $U \times V = (U_1 \times U_2 \times \cdots \times U_n) \times V$, A^i are fuzzy sets in U_i, B^i are fuzzy sets in V. Each rule is a fuzzy implication and can be written in the notation:

$$R^k = A_1^k \times \cdots \times A_n^k \to B^k.$$

The Cartesian product $A_1^k \times \cdots \times A_n^k$ is itself a fuzzy relation in $U_1 \times \cdots \times U_n$ with membership function:

$$\mu_{A_1^k \times \cdots \times A_n^k}(u) = \mu_{A_1^k}(u_1) \wedge \cdots \wedge \mu_{A_n^k}(u_n), \tag{2.66}$$

where $\wedge$ represents any T-norm operator. Any of the implication forms discussed can be used, including the Mamdani implications.

Mamdani combination

Considering the rules as independent, M rules can be interpreted as a single fuzzy relation Q_M in $U \times V$, defined by a union operator:

$$Q_M = \bigcup_{k=1}^{M} R^k. \tag{2.67}$$

This combination is called the *Mamdani combination*. The fuzzy relation Q_M has membership:

$$\mu_{Q_M}(u, v) = \mu_{R^1}(u, v) \vee \cdots \vee \mu_{R^M}(u, v), \tag{2.68}$$

where $\vee$ is the general S-norm (union) operator.

Gödel combination

The second point of view requires that all rules must hold together, so it is therefore appropriate to connect them by "and", and use intersection to obtain

the fuzzy relation Q_G in $U \times V$, defined by:

$$Q_G = \bigcap_{k=1}^{M} R^k. \tag{2.69}$$

This combination is called the *Gödel combination*. The fuzzy relation Q_G has membership:

$$\mu_{Q_G}(u, v) = \mu_{R^1}(u, v) \wedge \cdots \wedge \mu_{R^M}(u, v), \tag{2.70}$$

where $\wedge$ is the general T-norm (intersection) operator.

Fuzzy inference

Let A' be a fuzzy set in U as input to the fuzzy inference engine. Considering Q_M and Q_G as a single fuzzy **if-then** rule. Then using generalised modus ponens, the output of the fuzzy inference engine $B' \in V$ is determined by:

$$\mu_{B'}(v) = \sup_{x \in U} t[\mu_{A'}(u), \mu_{Q_M}(u, v)], \tag{2.71}$$

using the Mamdani combination, or

$$\mu_{B'}(v) = \sup_{x \in U} t[\mu_{A'}(u), \mu_{Q_G}(u, v)], \tag{2.72}$$

using the Gödel combination.

The steps to evaluate these inference engines are:

(i) Determine $\mu_{A_1^k \times \cdots \times A_n^k}(u)$ for $u \in U$, according to Equation 2.66 for the M fuzzy rules R^k in Equation 2.60.

(ii) Determine the membership of each fuzzy implication R^k in the FRB, namely $\mu_{R^k}(u, v) = \mu_{A_1^k \times \cdots \times A_n^k \to B^k}(u, v)$ for $(u, v) \in U \times V$, $k = 1, \cdots M$, according to any one of the implication interpretations.

(iii) Determine μ_{Q_M} or μ_{Q_G} according to Equation 2.68 or 2.70.

(iv) Determine for a given fuzzy set A' in U, the membership of the output fuzzy set B' in V, namely $\mu_{B'}(v)$ for $v \in V$, according to Equation 2.71 or 2.72.

2.14.2 Individual Rule Based Inference

In this form of inference, called *individual rule based inference* each rule in the FRB is considered separately. Given the fuzzy set A' in U, for each rule R^k in the FRB an output fuzzy set $(B')^k$ is calculated in V. The output of the fuzzy inference engine is then formed as a combination of the fuzzy sets $(B')^k, k = 1, \cdots M$. This combination can be either formed by using union or intersection. The computational procedure is identified in the following steps.

(i) Determine $\mu_{A_1^k \times \cdots \times A_n^k}(u)$ for $u \in U$, according to Equation 2.66 for the M fuzzy rules R^k in Equation 2.60.

(ii) Determine the membership of each fuzzy implication R^k in the FRB, namely $\mu_{R^k}(u, v) = \mu_{A_1^k \times \cdots \times A_n^k \to B^k}(u, v)$ for $(u, v) \in U \times V$, $k = 1, \cdots M$, according to any one of the implication interpretations.

(iii) Determine for a given fuzzy set A' in U, the membership of the output fuzzy set $(B')^k$ in V, namely $\mu_{(B')^k}(v)$ for $v \in V$, for each rule $R^k, k = 1, \cdots M$ in the FRB using generalised modus ponens, that is,

$$\mu_{(B')^k}(v) = \sup_{u \in U} t[\mu_{A'}(u), \mu_{R^k}(u, v)] \quad v \in V. \tag{2.73}$$

(iv) Determine the output of the fuzzy inference engine B' through calculation of its membership by either by union or intersection, that is,

$$\mu_{B'}(v) = \mu_{(B')^1}(v) \vee \cdots \vee \mu_{(B')^M}(v) \quad v \in V. \tag{2.74}$$

or

$$\mu_{B'}(v) = \mu_{(B')^1}(v) \wedge \cdots \wedge \mu_{(B')^M}(v) \quad v \in V. \tag{2.75}$$

2.14.3 Some Inference Engines

Many different inference engines can be constructed from a plethora of choices:

- compositional and individual-rule based reference, and within compositional based inference Mamdani and Gödel inference,
- different implication interpretations, for example, Mamdani min and product implications, and
- different operations for the T and S-norms within the various formula calculation.

Selection of inference engine is based on three criteria:

(i) *Intuitive appeal:* The choice of inference engine should be make sense intuitively. If a human expert gives a set of rules and considers them to be independent then the rules should be combined using the union operation,

(ii) *Computational efficiency:* The choice should be computationally efficient in that it is easy to compute the $\mu_{B'}$ given the input fuzzy set A',

(iii) *Special properties:* The choice should reflect any special properties that an application requires.

Two inference engines commonly used in fuzzy systems and fuzzy control are the *Product inference engine* and *Minimum inference engine*.

Product inference engine

The product inference engine is constructed from:

- individual-rule based inference with union combination Equation 2.74,
- Mamdani's product implication, and
- algebraic product for all the T-norm operators and max for all the S-norm operators.

Given a fuzzy set A' in U, the product inference engine gives the membership of the output fuzzy set B' in V as:

$$\mu_{B'}(v) = \max_{k=1,\cdots,M} \left[\sup_{u \in U} (\mu_{A'}(u) \prod_{i=1}^{n} \mu_{A_i^k}(u_i) \mu_{(B)^k}(v)) \right] \quad v \in V. \tag{2.76}$$

Minimum inference engine

This inference engine is constructed from: (i) individual-rule based inference with union combination Equation 2.74, (ii) Mamdani's min implication, and (iii) min for all the T-norm operators and max for all the S-norm operators. Given a fuzzy set A' in U, the minimum inference engine gives the membership of the output fuzzy set B' in V as:

$$\mu_{B'}(v) = \max_{k=1,\cdots,M} \left[\sup_{u \in U} \min(\mu_{A'}(u), \mu_{A_1^k}(u_1), \cdots \mu_{A_n^k}(u_n), \mu_{(B)^k}(v)) \right] \quad v \in V. \tag{2.77}$$

Considerable simplification to the calculation of the output membership function for both the inference engines is achieved when the input is just a singleton fuzzy set.

THEOREM 2.14.1

Given input space $U = U_1 \times \cdots \times U_n$ and output space V with a fuzzy rule base $FRB : U \to V$, consider the input singleton fuzzy set A' in U with membership

$$\mu_{A'}(u) = \begin{cases} 1 & \text{if } u = u^* \in U, \\ 0 & \text{otherwise.} \end{cases}$$

Show that

(a) *For the product inference engine the membership of the output fuzzy set B' in V is given by*

$$\mu_{B'}(v) = \max_{k=1,\cdots,M} \left[\prod_{i=1}^{n} \mu_{A_i^k}(u_i^*) \mu_{(B)^k}(v) \right] \qquad v \in V. \tag{2.78}$$

(b) *For the minimum inference engine the membership of the output fuzzy set B' in V is given by*

$$\mu_{B'}(v) = \max_{k=1,\cdots,M} \left[\min(\mu_{A_1^k}(u_1^*), \cdots, \mu_{A_n^k}(u_n^*), \mu_{(B)^k}(v) \right] \qquad v \in V. \tag{2.79}$$

Even though there is simplicity in both the reasoning and the computation of both these inference engines, they have a disadvantage in that if at some $u \in U$ the membership values $\mu A_i^k(x_i)$'s are very small, then the membership $\mu_{B'}(v)$ obtained will be very small also. The following fuzzy inference engines overcome this disadvantage.

For the product inference engine the resulting fuzzy set B'_P has triangular membership function simply scaled by the value $\mu_A(u^*)$. Indeed $B' \subset B$.

For the minimum inference engine:

$$\mu_{B'}(v) = \begin{cases} \mu_B(v) & \text{if } \mu_B(v) \leq mu_{A_p}(u_p^*), \\ \mu_{A_p}(u_p^*) & \text{otherwise,} \end{cases}$$

which shows that the membership function for B' is just that for B but clipped at the value of $\mu_{A_p}(u_p^*)$. Again $B' \subset B$. These two cases are illustrated in Figure 2.16 with $\mu_{A_p}(u_p^*) = 0.7$.

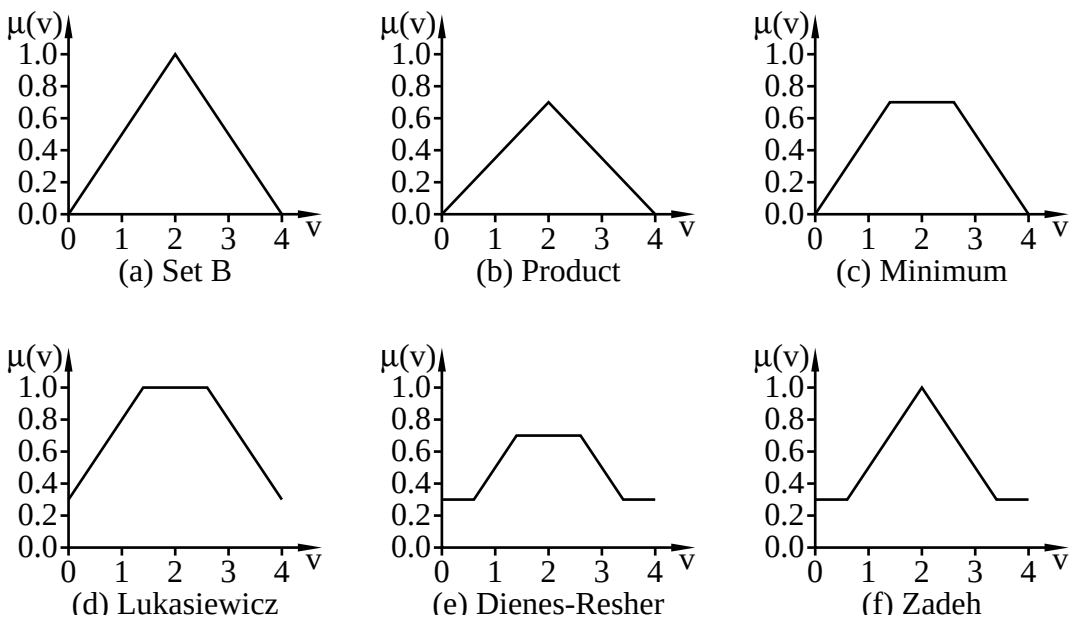


Figure 2.16: Membership Functions with $\mu_{A_p}(u_p^*) = 0.7$

The membership for the output functions are again shown in Figures 2.16 and 2.17 for a smaller value $\mu_{A_p}(u_p^*) = 0.3$.

Observe that if the value $\mu_{A_p}(u_p^*)$ is small then the product and minimum inference engines result in very small membership functions, as previously mentioned.

A major problem with the pure fuzzy system in application is that its inputs and outputs are fuzzy sets, that is, words in natural languages. In most physical and engineering systems, the inputs and outputs are real variables.

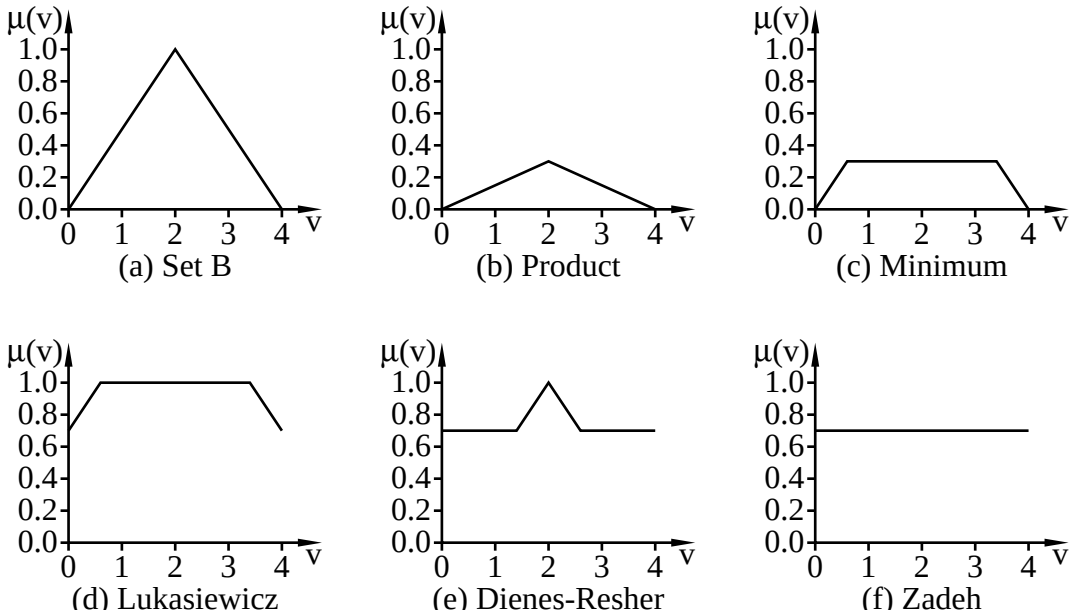


Figure 2.17: Membership Functions with $\mu_{A_p}(u_p^*) = 0.3$

To solve this problem, Takagi, Sugeno, and Kang (1985, 1988) proposed a fuzzy system whose inputs and outputs are real-valued variables. The basic rule in the TSK fuzzy system has the form:

$$\textbf{if} \text{ speed } v \text{ of the rocket is } \textit{high} \textbf{ then} \text{ air resistance is } R = 100v^2, \tag{2.80}$$

The consequent part of the rule has changed from using words in natural languages to a simple mathematical formula. Because of this such a system may not provide a natural framework to represent human knowledge.

The third fuzzy system (iii) with fuzzifier and defuzzifier was consequently introduced. Its basic configuration is shown in Figure 2.18. These systems take the form of the basic pure fuzzy system and include a fuzzifier that transforms a real variable into a fuzzy sets, and a defuzzifier that transforms the output fuzzy sets into a real output variable.

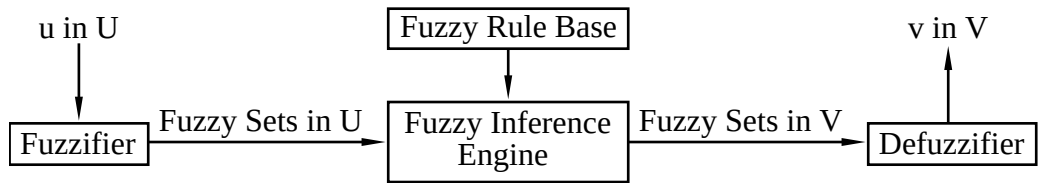


Figure 2.18: Components of Fuzzy System with Fuzzifier and Defuzzifier

2.15 Fuzzifiers and Defuzzifiers

The basic fuzzy system of Figure 2.14 is improved by including a fuzzifier and defuzzifier to interface to real variables as shown in Figure 2.18.

A fuzzy rule base $FRB : U_1 \times \cdots \times U_n \rightarrow V$ is represented as the set of M canonical rules:

$$\{R^k : R^k = \textbf{ if } (u_1 \text{ is } A_1^k) \text{ and } \cdots \text{ and } (u_n \text{ is } A_n^k) \textbf{ then } v \text{ is } B^k,\ k = 1, \cdots, M\}, \tag{2.81}$$

where each rule R^k is a fuzzy implication/relation in $U \times V = (U_1 \times U_2 \times \cdots \times U_n) \times V$, A^i are fuzzy sets in U_i, B^i are fuzzy sets in V, and $u = [u_1 \ \ u_2 \ \ \cdots \ \ u_n]^T \in U$ and $v \in V$ are the input and output linguistic variables of the fuzzy system.

The pure fuzzy system with a fuzzy inference engine gives an output fuzzy set B' in V for a given input fuzzy set A' in U. In practical applications, such as controlling the steering of a vehicle, the inputs and outputs are real values. The inference engine needs to work between an input fuzzifier component and an output defuzzifier component. The singleton fuzzifier analysed is the simplest case.

2.16 Fuzzifiers

A *fuzzifier* is a mapping $\mathcal{F}$ that takes a real valued point in the input space U to the set of fuzzy sets in U. This mapping can be written as: $\mathcal{F} : U \to \mathcal{FS}(U)$, where $\mathcal{FS}(U)$ defines the set of all fuzzy sets in U. The defining criteria for $\mathcal{F}$ should include:

- Given $u^* \in U$ then $\mu_{\mathcal{F}}(u^*)$ should have a large value reflecting the strength at this input value.
- The fuzzifier $\mathcal{F}$ should suppress noise in input evaluation.
- The fuzzifier $\mathcal{F}$ should maintain simplicity of computation in the inference engine, for example, simplification of the evaluation of $\sup_{u \in U}$ in such calculations.

The following discussion identifies three fuzzifier that meet these criteria.

2.16.1 Singleton Fuzzifier

The *singleton* fuzzifier $\mathcal{F}_S : U \to \mathcal{FS}(U)$ is defined by by $\mathcal{F}_S(u^*) = A'$ where

$$\mu_{A'}(u) = \begin{cases} 1 & \text{if } u = u^*, \\ 0 & \text{otherwise.} \end{cases} \tag{2.82}$$

The singleton fuzzifier is very straight forward, very simple. If at $u = u^*$ the membership of A' is one, otherwise zero, establishing a crisp evaluation at this point in U. The singleton fuzzifier has been used in examples that illustrate its use greatly simplifies calculations in the inference engine. To illustrate, some of the previous results are reiterated here.

LEMMA 2.16.1

Assuming that the input space is $U = U_1 \times \cdots \times U_n$, an output space of V and a fuzzy rule base $FRB : U \to V$. Using a singleton fuzzifier for $u^* \in U$.

(a) *For the product inference engine the membership of the output fuzzy set B' in V is given by:*

$$\mu_{B'}(v) = \max_{k=1,\cdots,M} \left[\prod_{i=1}^{n} \mu_{A_i^k}(u_i^*) \mu_{(B)^k}(v) \right] \qquad v \in V. \tag{2.83}$$

(b) *For the minimum inference engine the membership of the output fuzzy set B' in V is given by:*

$$\mu_{B'}(v) = \max_{k=1,\cdots,M} \left[\min(\mu_{A_1^k}(u_1^*), \cdots, \mu_{A_n^k}(u_n^*), \mu_{(B)^k}(v) \right] \qquad v \in V. \tag{2.84}$$

It should be noted that each of these inference engines are based on the method of individual-based inference, treating each rule separately in the knowledge base and combining the resulting consequents through union or intersection to determine the output fuzzy set B' in V.

It has also been shown that triangular membership functions for the consequent B in a fuzzy rule, the following simple calculations are obtained through the use of a singleton fuzzifier.

EXAMPLE 2.16.1 *Consider universes of discourse $U = U_1 \times \cdots \times U_n$ and V with the $FRB : U \to V$ consisting of the single rule*

$$R = \textbf{ if } (u_1 \text{ is } A_1^k) \text{ and } \cdots \text{ and } (u_n \text{ is } A_n^k) \textbf{ then } v \text{ is } B, \tag{2.85}$$

where A_i are fuzzy sets in U_i and B is a fuzzy set in V and the membership of the set B in V has the form

$$\mu_B(v) = \begin{cases} 1 - |v| & \text{if } y \in [-1, 1], \\ 0 & \text{otherwise.} \end{cases}$$

For the singleton fuzzifier associated with $u^* \in U$, the membership of the output fuzzy set B' in V for the five defined fuzzy inference engines are:

$$\mu_{B'_P}(v) = \mu_A(u^*)\mu_B(v) \quad \forall v \in V. \tag{2.86}$$

$$\mu_{B'_M}(v) = \min[\mu_{A_p}(u_p^*), \mu_B(v)] \quad \forall v \in V. \tag{2.87}$$

Here the notation B'_P and B'_M have been used to define the fuzzy set B' obtained from the product and minimum inference engines respectively.

Observe this simplification occurs regardless of the membership evaluation in the consequent section of each rule. It has also been commented that:

- For the product inference engine the result B' is just the fuzzy set B clipped at a height $\mu_A(u^*) = \prod_{i=1}^n \mu_{A_i}(u_i^*)$, and
- For the minimum inference engine, the fuzzy set B' is just B clipped at a height $\mu_{A_p}(u_p^*) = \min[\mu_{A_1}(u_1^*), \cdots, \mu_{A_n}(u_n^*)]$.

2.16.2 Gaussian Fuzzifier

The *Gaussian* fuzzifier $\mathcal{F}_G : U \to \mathcal{FS}(U)$ is defined by by $\mathcal{F}_G(u^*) = A'$ where

$$\mu_{A'}(u) = \exp\left(-\frac{(u_1 - u_1^*)^2}{\alpha_1^2}\right) \wedge \cdots \wedge \exp\left(-\frac{(u_n - u_n^*)^2}{\alpha_n^2}\right). \tag{2.88}$$

where $\alpha_k, k = 1, \cdots, n$ are positive constants.

This fuzzifier establishes a Gaussian membership function around the point u^* with maximum membership of one at u^*. Choosing the suitable constants for α_k that define the spread of the Gaussian membership function. If the fuzzy sets in

the consequent part of the rules in the FRB are also Gaussian then the following simplification can be identified.

LEMMA 2.16.2

Assume a FRB with M rules of the form Equation 2.81 and that the membership function for each A_i^k has the form:

$$\mu_{A^k}(u_i) = \exp\left(-\frac{(u_i - \bar{u}_i^k)^2}{(\sigma_i^k)^2}\right) \quad i = 1, \cdots, n \text{ and } k = 1, \cdots, M, \tag{2.89}$$

where $\bar{u}_i^k$ and σ_i^k are Gaussian mean and variance parameters. Applying the Gaussian fuzzifier associated with input $u^* \in U$:

(a) *Given the algebraic product for the T-norm $\wedge$ in Equation 2.88, and the product inference engine, the membership function of the output fuzzy set B' in V is given by:*

$$\mu_{B'}(v) = \max_{k=1,\cdots,M}\left[\prod_{i=1}^{M}\exp\left(-\frac{(u_{iP}^k - \bar{u}_i^k)^2}{(\sigma_i^k)^2}\right)\exp\left(-\frac{(u_{iP}^k - u_i^*)^2}{(\alpha_i)^2}\right)\mu_{B^k}(v)\right], \tag{2.90}$$

for all $v \in V$, where

$$u_{iP}^k = \frac{\alpha_i^2\bar{u}_i^k + (\sigma_i^k)^2 u_i^*}{\alpha_i^2 + (\sigma_i^k)^2}. \tag{2.91}$$

(b) *Given "min" for the T-norm $\wedge$ in Equation 2.88, and the minimum inference engine, the membership function of the output fuzzy set B' in V is given by:*

$$\mu_{B'}(v) = \max_{k=1,\cdots,M}\left[\min\left(e^{-\frac{(u_{iM}^1 - \bar{u}_1^k)^2}{(\sigma_1^k)^2}}, \cdots, e^{-\frac{(u_{nM}^k - \bar{u}_n^k)^2}{(\alpha_n)^2}}, \mu_{B^k}(v)\right)\right], \tag{2.92}$$

for all $v \in V$, where

$$u_{iM}^k = \frac{\alpha_i\bar{u}_i^k + \sigma_i^k u_i^*}{\alpha_i + \sigma_i^k}. \tag{2.93}$$

Proof. These results are easy to obtain.

(a) First recall the formula for the product inference engine as:

$$\mu_{B'}(v) = \max_{k=1,\cdots,M}\left[\sup_{u\in U}(\mu_{A'}(u)\prod_{i=1}^{n}\mu_{A_i^k}(u_i)\mu_{(B)^k}(v))\right] \quad v \in V. \tag{2.94}$$

For the calculation of $\mu_{A'}(u)$, using Equation 2.88 and Equation 2.89 for each $\mu_{A_i^k}(u_i)$, Equation 2.94 becomes:

$$\begin{aligned}\mu_{B'}(v) &= \max_{k=1,\cdots,M}\left[\sup_{u\in U}(\prod_{i=1}^{n} e^{-\frac{(u_i-u_i^*)^2}{(\alpha_i)^2}} e^{-\frac{(u_i-\bar{u}_i^k)^2}{(\sigma_i^k)^2}}\mu_{(B)^k}(v))\right] \quad v \in V,\\ &= \max_{k=1,\cdots,M}\left[\prod_{i=1}^{n}\sup_{u\in U}(e^{-\frac{(u_i-u_i^*)^2}{(\alpha_i)^2}-\frac{(u_i-\bar{u}_i^k)^2}{(\sigma_i^k)^2}}\mu_{(B)^k}(v))\right] \quad v \in V.\end{aligned}$$

This last interchange of supremum and product operators is validated as all terms are positive and the product is over a finite number of terms. Now completing the square in the exponential numerator

$$\begin{aligned}-\frac{(u_i-u_i^*)^2}{(\alpha_i)^2}-\frac{(u_i-\bar{u}_i^k)^2}{(\sigma_i^k)^2} &= -k_1\left(u_i-\frac{\alpha_i^2\bar{u}_i^k+(\sigma_i^k)^2u_i^*}{\alpha_i^2+(\sigma_i^k)^2}\right)^2+k_2,\\ &= -k_1(u_i-u_{iP}^k)^2+k_2.\end{aligned}$$

where k_1 and k_2 are not functions of the u_i variables. Hence the supremum in the formula for B' is achieved at $u_P^k = (u_{1P}^k, \cdots, u_{nP}^k) \in U$. This completes the result for part (a).

(b) For part (b) first recall for the minimum inference engine:

$$\mu_{B'}(v) = \max_{k=1,\cdots,M}\left[\sup_{u\in U}\min(\mu_{A'}(u), \mu_{A_1^k}(u_1), \cdots \mu_{A_n^k}(u_n), \mu_{(B)^k}(v))\right] \quad v \in V. \tag{2.95}$$

For the calculation of $\mu_{A'}(u)$, using Equation 2.88 and Equation 2.89 for each $\mu_{A_i^k}(u_i)$, Equation 2.95 becomes:

$$\mu_{B'}(v) = \max_{k=1,\cdots,M}[\sup_{u\in U}\min(\min\left(e^{-\frac{(u_1-u_1^*)^2}{\alpha_1^2}}, \cdots, e^{-\frac{(u_n-u_n^*)^2}{\alpha_n^2}}\right),$$

$$\mu_{A_1^k}(u_1), \cdots \mu_{A_n^k}(u_n), \mu_{(B)^k}(v))] \quad \forall v \in V.$$

$$= \max_{k=1,\cdots,M}[\sup_{u\in U}\min(\min\left(e^{-\frac{(u_1-u_1^*)^2}{\alpha_1^2}}, \mu_{A_1^k}(u_1)\right), \cdots, \left(e^{-\frac{(u_n-u_n^*)^2}{\alpha_n^2}}, \mu_{A_n^k}(u_n)\right),$$

$$\mu_{(B)^k}(v))] \quad \forall v \in V$$

$$= \max_{k=1,\cdots,M}[\min(\sup_{u\in U}\min\left(e^{-\frac{(u_1-u_1^*)^2}{\alpha_1^2}}, e^{-\frac{(u_1-\bar{u}_1^k)^2}{(\sigma_1^k)^2}}\right), \cdots,$$

$$\sup_{u\in U}\min\left(e^{-\frac{(u_n-u_n^*)^2}{\alpha_n^2}}, e^{-\frac{(u_n-\bar{u}_n^k)^2}{(\sigma_n^k)^2}}\right), \mu_{(B)^k}(v))] \quad \forall v \in V.$$

Here again the "min" and supremum operators have been changed and use has been made of the fact that for the "min" operator

$$\min(\min(a_1, a_2), A_1, A_2) = \min(\min(a_1, A_1), \min(a_2, A_2)),$$

for constants $a_\ell, A_\ell, \ell = 1, 2$ which clearly can be extended to a more general case of $\ell = n$ values. Now clearly the $\sup_{u\in U}\min$ operation is achieved when

$$\exp\left(-\frac{(u_i - u_i^*)^2}{\alpha_i^2}\right) = \exp\left(-\frac{(u_i - \bar{u}_i^k)^2}{(\sigma_i^k)^2}\right).$$

This occurs at $u_M^k = (u_{1M}^k, \cdots, u_{nM}^k) \in U$ where

$$u_{iM}^k = \frac{\alpha_i \bar{u}_i^k + \sigma_i^k u_i^*}{\alpha_i + \sigma_i^k}.$$

Substituting these values into the right hand side of the above equation:

$$\mu_{B'}(v) = \max_{k=1,\cdots,M}\left[\min\left(e^{-\frac{(u_{iM}^1-\bar{u}_1^k)^2}{(\sigma_1^k)^2}}, \cdots, e^{-\frac{(u_{nM}^k-\bar{u}_n^*)^2}{(\alpha_n)^2}}, \mu_{B^k}(v)\right)\right], \tag{2.96}$$

gives the required result. This completes the result for part (b).

□

REMARK 2.16.1 *Observe if the Gaussian constants $\alpha_i = 0$ for $i = 1, \cdots, n$, then $u^* = u_P^k = u_M^k, k = 1, \cdots M$, and the formulae in this Lemma for the membership of the output fuzzy set B' revert to the case of the singleton fuzzifier.*

Recall that the Gaussian constants α_i for $i = 1, \cdots, n$ are at the choice of the user. Observe that in Equations 2.16.2 and 2.93 if α_i is much greater that σ_i, then u_P^k and u_M^k will be close to the value $\bar{u}^k = (\bar{u}_1^k, \cdots, \bar{u}_n^k) \in U$. Hence these values u_P^k and u_M^k will be insensitive to any noise in the evaluation of u^*. One therefore would expect that by choosing larger α_i, the Gaussian fuzzifier will suppress any noise in the input u^*. In more mathematical terms, let the input be

$$u^* = u_0^* + \eta^*,$$

where η^* is a vector representing noise in the evaluation of the base variable input u_0^*.

Substituting this value into Equation 2.16.2, for each k and for $i = 1, \cdots, n$:

$$u_{iP}^k = \frac{\alpha_i^2 \bar{u}_i^k + (\sigma_i^k)^2 u_{0i}^*}{\alpha_i^2 + (\sigma_i^k)^2} + \frac{(\sigma_i^k)^2}{\alpha_i^2 + (\sigma_i^k)^2} \eta_i^*.$$

This equation shows that with the Gaussian fuzzifier, the noise in the evaluation of u_{iP}^k can be suppressed by decreasing the last term in this equation which obviously can be obtained by making α_i much larger that $(\sigma_i^k)^2$ so that the factor in front of η_i^* is substantially reduced.

2.16.3 Triangular Fuzzifier

The *triangular* fuzzifier $\mathcal{F}_T : U \to \mathcal{FS}(U)$ is defined by by $\mathcal{F}_T(u^*) = A'$ where:

$$\mu_{A'}(u) = \begin{cases} (1 - \frac{|u_1 - u_1^*|}{\beta_1}) \wedge \cdots \wedge (1 - \frac{|u_n - u_n^*|}{\beta_n}) & \text{if } |u_i - u_i^*| \leq \beta_i,\ i = 1, \cdots, n, \\ 0 \text{ otherwise.} \end{cases} \tag{2.97}$$

Choosing constants $\beta_k, k = 1, \cdots, n$ define the width of the base of each triangular membership function. The maximum membership of $\mathcal{F}_T$ occurs with a value

of one when $u = u_*$. As expected considerable simplification occurs when the antecedent part of each rule is defined by a triangular membership function. This is given in the following lemma for which the proof is left to the reader.

LEMMA 2.16.3

Assume a FRB with M rules of the form Equation 2.81 and that the membership function for each A_i^k has the form:

$$\mu_{A^k}(u_i) = \left(1 - \frac{|u_i - \bar{u}_i^k|}{\delta_i^k}\right) \quad i = 1, \cdots, n \text{ and } k = 1, \cdots, M, \tag{2.98}$$

where $\bar{u}_i^k$ and δ_i^k are centre and base triangular parameters. Applying the triangular fuzzifier associated with input $u^* \in U$:

(a) *Given the algebraic product for the T-norm $\wedge$ in Equation 2.88, and the product inference engine, the membership function of the output fuzzy set B' in V is given by:*

$$\mu_{B'}(v) = \max_{k=1,\cdots,M} \left[\prod_{i=1}^{M} \left(1 - \frac{|u_{iP}^k - \bar{u}_i^k|}{\delta_i^k}\right)\left(1 - \frac{|u_{iP}^k - \bar{u}_i^*|}{\beta_i^k}\right)\mu_{B^k}(v)\right], \tag{2.99}$$

for all $v \in V$, where

$$u_{iP}^k = \frac{\delta_i u_i^* + \beta_i^k \bar{u}_i^k}{\delta_i + \beta_i^k}. \tag{2.100}$$

(b) *Given "min" for the T-norm $\wedge$ in Equation 2.88, and the minimum inference engine, the membership function of the output fuzzy set B' in V is given by:*

$$\mu_{B'}(v) = \max_{k=1,\cdots,M} \left[\min\left(e^{-\frac{(u_{iM}^1 - \bar{u}_1^k)^2}{(\sigma_1^k)^2}}, \cdots, e^{-\frac{(u_{nM}^k - \bar{u}_n^*)^2}{(\alpha_n)^2}}, \mu_{B^k}(v)\right)\right], \tag{2.101}$$

for all $v \in V$, where

$$u_{iM}^k = \frac{\alpha_i \bar{u}_i^k + \sigma_i^k u_i^*}{\alpha_i + \sigma_i^k}. \tag{2.102}$$

2.17 Defuzzifiers

The *defuzzifier* is a mapping $\mathcal{DF}$ which maps from the set of fuzzy sets in V, $\mathcal{FS}(V)$ to real valued crisp value in V. This can be written as $\mathcal{DF} : \mathcal{FS}(V) \rightarrow V$. If $B \in \mathcal{FS}(V)$ and $v^* = \mathcal{DF}(B)$, the defining criteria for $\mathcal{DF}$ should take into account:

Plausibility The crisp value $v^* \in V$ should reflect a high membership in in the fuzzy set B, intuitively it should lie in the middle of the support of B.

Computational simplicity The defuzzifier $\mathcal{DF}$ should maintain simplicity of computation in the inference engine.

Continuity A small change in B should not result in a large change in v^*.

Three widely used defuzzifiers are now examined.

Centre of gravity defuzzifier

The *centre of gravity* defuzzifier $\mathcal{DF}_g : \mathcal{FS}(V) \rightarrow V$ is defined by:

$$v^* = \mathcal{DF}_g(B) = \frac{\int_V v\mu_B(v)dv}{\int_V \mu_B(v)dv}, \tag{2.103}$$

where $B \in \mathcal{FS}(V)$.

This defuzzifier specifies v^* as its name suggests, the centre of area covered by the membership function of B. Here integration is the conventional integral with area interpretation. The defuzzifier definition is illustrated in Figure 2.19

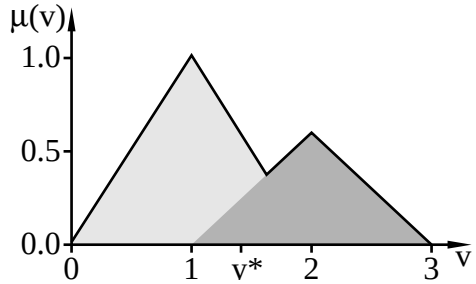


Figure 2.19: Graphical Illustration of the Centre of Gravity Defuzzifier

The centre of gravity defuzzifier is intuitively plausible but is computationally expensive for many instances due to the mathematical nature of the membership functions. As a point, consider integrating Gaussian membership functions.

2.17.1 Centre Average Defuzzifier

Recall that the output fuzzy set B of the inference engine is either a union or intersection of M fuzzy sets, depending on the method chosen to define the reasoning of the collection of M rules in the knowledge base. A good approximation to the integral definition of the centre of gravity fuzzifier is the weighted average of the centres of the M fuzzy sets, with the weights equal to the heights of the corresponding fuzzy sets. Then the *centre of average* defuzzifier $\mathcal{DF}_a : \mathcal{FS}(V) \to V$ is defined by:

$$v^* = \mathcal{DF}_a(B) = \sum_{k=1}^{k=M} v^k \omega_k / \sum_{k=1}^{k=M} \omega_k, \tag{2.104}$$

where $B \in \mathcal{FS}(V)$, v^k is the centre of the Mth fuzzy output set from each rule and ω_k is its height.

This is the most used defuzzifier for it is both easy to compute and it is intuitively plausible. Observe also that from the mathematical point of view, small changes in v^k and ω_k results in small changes in the value of v^*, yielding a measure of

continuity.

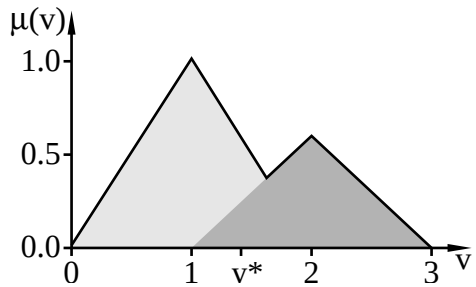


Figure 2.20: Graphical Illustration of the Centre Average Defuzzifier

Maximum defuzzifier

Let the set of point defined in a fuzzy set B for which the membership μB of B attains its maximum value:

$$\mathcal{H}(B) = \{\bar{v} \in V|\ \mu_B(\bar{v}) = \sup_{v \in V} \mu_B(v)\}. \tag{2.105}$$

The *maximum* defuzzifier, $\mathcal{DF}_m : \mathcal{FS}(V) \to V$, is now defined as:

$$v^* = \mathcal{DF}_m(B) \text{ if } v^* \in \mathcal{H}(B). \tag{2.106}$$

The definition of the maximum defuzzifier is clearly non-unique. It is unique if $\mathcal{H}(B)$ consists of a single point. This definition lends itself now to three obvious definitions: the smallest, largest and mean of such values lying in $\mathcal{H}(B)$.

smallest maximum defuzzifier:

$$v^* = \mathcal{DF}_{sm}(B) = \inf\{v \in \mathcal{H}(B)\}. \tag{2.107}$$

largest maximum defuzzifier:

$$v^* = \mathcal{DF}_{\ell m}(B) = \sup\{v \in \mathcal{H}(B)\}. \tag{2.108}$$

least maximum defuzzifier:

$$v^* = \mathcal{DF}_{mm}(B) = \frac{\int_{\mathcal{H}}(B)vdv}{\int_{\mathcal{H}}(B)dv} \text{ if continuous,} \tag{2.109}$$

In the last definition, integration reverts to summation if the variable distribution is discrete.

The maximum defuzzifier is computationally simple if the membership function is triangular or trapezoidal. But small changes in the fuzzy set B may result in large changes in v^*, See figure 2.21.

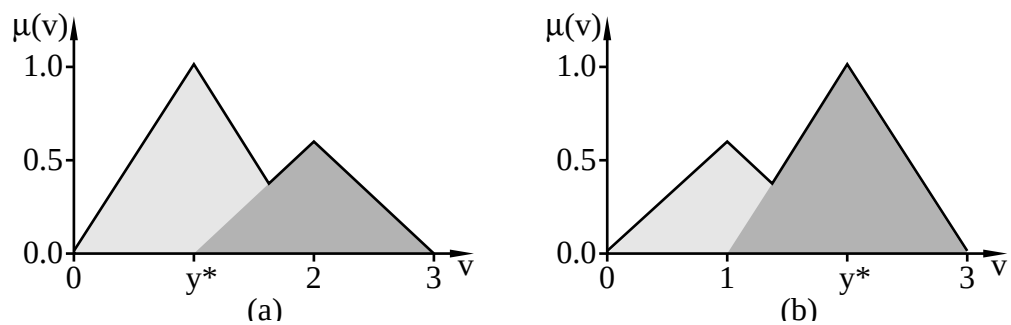


Figure 2.21: Illustration of Continuity Difficulties with the Maximum Defuzzifier

Furthermore the idea of a maximum may loose its intuitive meaning when considering the resultant fuzzy set B, being the union of two fuzzy sets with triangular memberships as shown in Figure 2.22.

Note that the mean definition of the maximum fuzzifier does have a value of zero and indeed a zero membership in the resulting definition of B.

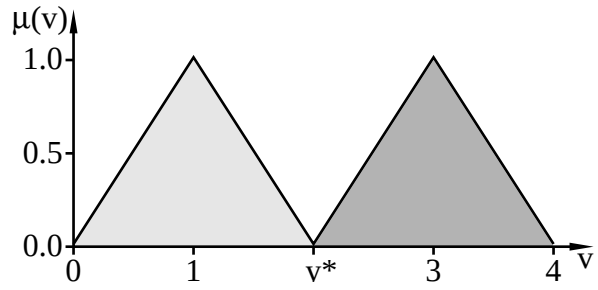


Figure 2.22: Example Illustrating Problem with the Maximum Defuzzifier

2.17.2 Defuzzifier Comparison

Table 2.7 summarises the discussed defuzzifiers compared to the three criteria: plausibility, computational simplicity, and continuity.

Table 2.7: Comparing Defuzzifiers to their Criteria

	centre of gravity	centre average	maximum
plausibility	yes	yes	yes
computational simplicity	no	yes	yes
continuity	yes	yes	no

Table 2.7 clearly shows that the average defuzzifier meets all criteria and consequently is the most used of the three defuzzifiers.

Fuzzy systems can be built from a priori knowledge. However, when the system becomes large or there is an absence of a priori knowledge, the fuzzy knowledge base is built by using a heuristic approach.

Chapter 3

Genetic/Evolutionary Algorithms

3.1 Introduction

The goal of optimisation is to minimise or maximise a function or a set of functions with the main objective being to determine the global optima (lowest minima or highest maxima). Typically the unconstrained minimisation problem has the mathematical form:

Minimise

$$f(\underset{\sim}{x}) \tag{3.1}$$

with $\underset{\sim}{x} \in R^n$ and where f is the function to be optimised. The vector space R^n is called the search space and the value $f(\underset{\sim}{x})$ is called the fitness of the function f at the point $\underset{\sim}{x}$.

A point $x^* \in R^n$ is said to be a **local** optimum if

$$f(\underset{\sim}{x}^*) \leq f(\underset{\sim}{x}) \tag{3.2}$$

for all $\underset{\sim}{x}$ in some open neighbourhood of $\underset{\sim}{x}^*$.

The point $x^* \in R^n$ is said to be a **global** optimum if

$$f(\underset{\sim}{x}^*) \leq f(\underset{\sim}{x}) \tag{3.3}$$

for ALL $\underset{\sim}{x} \in R^n$.

Often, a local optimum that gives the required solution to a "close enough" measure is adequate and the search can be terminated. Unnecessarily high precision can produce unreasonably long algorithm execution times.

Three key traditional search methods [10] are: gradient descent [11], simulated annealing [12], and random search.

Gradient descent methods are designed to move a solution estimate of the optimisation problem in the direction of the local gradient towards the optimum. The gradient is determined explicitly or by a local numerical method. Step size is an important issue in these descent methods, for a large step size may result in missing a minima or maxima of small feature size, while a small step size results in long solution times. It is noted that gradient descent methods also become easily trapped in local minima or maxima. Further it is necessary to start the search multiple times from different starting points to show confidence that an optimum point is global or an acceptable local minima or maxima.

Simulated Annealing (SA) is an optimisation search method [13, 14] based on the statistical mechanics of annealing in solids. The method was developed in 1983 to deal with highly non-linear problems.

It is well known that metals held at elevated temperatures are able to repair crystal dislocations. Metal atoms are in a thermally excited state and the random vibrations due to their thermal energy allows atoms to migrate to locations of

lower energy (the annealing process).

For the global minimisation problem the SA approach is similar to using a bouncing ball that can bounce over mountains from valley to valley. Starting with a high "temperature" the ball is able to make very high bounces, which enables it to bounce over any mountain to access any valley, that is, given enough bounces. By decreasing the temperature the ball cannot bounce so high, and it can therefore settle down and become trapped in relatively small ranges of valleys.

In such an optimisation problem disturbing the solution by injecting a random input to can allow the solution to escape a local optima and find a better solution. The annealing process may monotonically decrease the random disturbance or temperature. Alternate schemes where temperature is cyclically varying, but gradually falling are also popular.

The method consists of four components:

- A representation of possible solutions $\underset{\sim}{x}$,
- A generator of random changes in solutions,
- A means of evaluating the fitness functions (if not given by a function), and
- An annealing schedule with an initial temperature and rules for lowering it as the search progresses.

Random Search uses a brute force approach for difficult functions. Random locations $\underset{\sim}{x}$ are tested for fitness, if the fitness is less than the currently held minimum, it is replaced by the new fitness. At it's worst it performs no better

than an enumerative scheme of checking every possible point. The method is an unintelligent strategy, and it is rarely used in solving optimisation problems by itself.

3.2 Genetic Algorithms

Of recent note, the simple genetic algorithm has been used very effectively in solving the optimisation problem, see [4].

The genetic algorithm offers a powerful alternative to the more traditional optimisation techniques.

These algorithms are modelled on the idea of natural selection as seen in the animal kingdom. The normal usage of genetic algorithms is however, a simplified version of natural selection suitable for solving the optimisation problem.

3.2.1 Nomenclature

As with any technology, there is a nomenclature used in the field. Many terms used in this thesis interchange Natural Selection nomenclature and that used in Genetic Algorithms. Table 3.1 shows the relationship among the terms used throughout this thesis.

Since their appearance genetic algorithms and variants thereof, have been used in a variety of applications to solve many real world problems in such areas as: ecology, machine learning, engineering and management. They are one of a

Table 3.1: Biological and GA Nomenclature Comparison [4]

Biological	Genetic Algorithm
chromosome	string
gene	feature, character or detector
allele	feature value
locus	string position
genotype	structure
phenotype	parameter set, alternative solution, a decoded structure
epistasis	non-linearity

number of evolutionary computation techniques, including: evolution strategies, genetic programming and evolutionary programming.

Based on principles of evolution associated with inheritance and the fight for survival of the fittest the GA is a heuristic search technique that maintains a population of individuals $P(t) = \{\underset{\sim}{x}_1, \ldots \underset{\sim}{x}_N\}$ at iteration t to the next $t + 1$.

Each individual can be considered to represent a potential solution to a given problem and needs to be properly encoded. Although in their original formulation the structure of each individual was binary encoded [59, 4], modern formulation of structure for what is now termed evolutionary algorithms includes for example, integer and real encoding [27].

To each individual $\underset{\sim}{x}_k$, is an associated measure of its fitness for survival, typically for the optimisation problem stated this is $f_k = f(\underset{\sim}{x}_k)$. Figure 3.1 shows the general form of a population used in genetic algorithms.

The new population $P(t + 1)$ is obtained from the old by the use of genetic operators such as selection and crossover—creating children by recombining parts

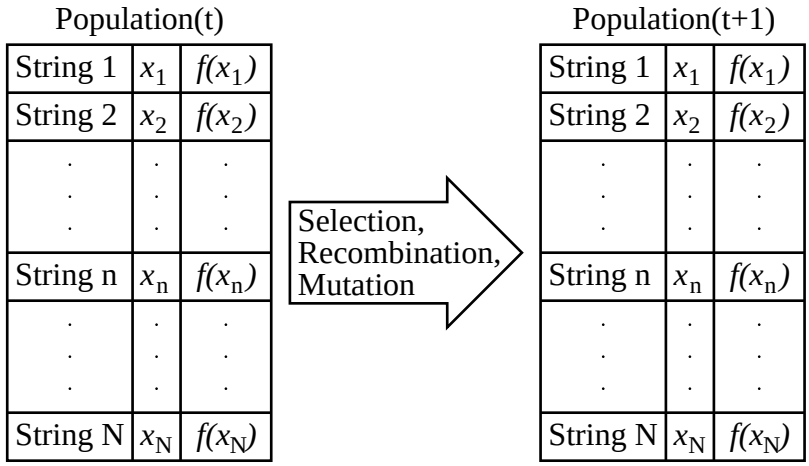


Figure 3.1: Basic Genetic Algorithm Structure

of usually more than one parent through a process of selection for mating of the parents, and mutation—creating new individuals by perturbing the individual's structure. The form of the algorithm may be written as:

```
begin
   t = 0
   Create random P(0)
   Evaluate Fitness of P(0)
   while (not Terminated) do
   begin
      Create P(t+1) from P(t)
      Evaluate Fitness of P(t+1)
      t = t+1
   end
end
```

3.2.2 Data Structures

The encoding of a potential solution $\underset{\sim}{x}$ in the genetic algorithm is a string of binary bits. Most introductory programmes encode a "bit" as a word sized true (1) or false (0) indication. For real components of the vector, this necessarily brings with it approximation to the exact real value that must be recognised as well as to the functions value associated with this point. Further this encoding requires functions to translate from the problem itself in real terms to the data structure used.

Sampling of the search space can be done using equi-spaced samples or non-linear spaced increments. Equi-spaced samples are normally chosen for encoding problems that have an unknown function form.

Sampling a smooth continuous function with discrete steps has some limitations. Sampling the continuous function slower than twice the highest frequency (Nyquist rate) causes aliasing. Aliasing is a term given for having insufficient data to reconstruct the original waveform.

Say we wish to maximise $f_1(x) = 1 - \sin(\pi x)\cos(11\pi x),\ x \in (0,1)$. The function has local maximums listed in Table 3.2. The global maxima is at $x = 0.454921$.

Table 3.2: Local Maxima of $f_1(x) = 1 - \sin(\pi x)\cos(11\pi x), x \in (0,1)$

x	$f_1(x)$
0.0988933	1.294145527
0.27497	1.758062304
0.454921	1.989905289
0.635175	1.910408307
0.814223	1.545912568
0.975123	1.050956808

Rearranging function $f_1(x) = 1 - 0.5(sin(12\pi x) - sin(10\pi x))$ over the range $x \in (0, 1)$, the function cycles through six cycles. Nyquist [73] sampling requires twelve samples to capture enough data for the signal to be reconstructed. Effectively, the Nyquist rate causes samples to be taken half way between zero crossings of the highest frequency. Therefore, the sample points are: $x = (2n+1)/24,\ n \in (0, 1, \cdots, 11)$ Encoding the domain into binary string for genetic optimisation will require four bits $2^4 = 16$ with sample points $x = (2n + 1)/32,\ n \in (0, 1, \cdots, 15)$. The optimum value for this encoding is located at $x = 15/32$ giving $f_1(15/32) = 1.877674572$.

Taking samples at discrete time steps has introduced a sampling error of $\pm 1/32$. Therefore, we are confident that the maximum of the function $f_1(x)$ is within $x = 15/32 \pm 1/32$. Giving an error of less than 5%.

3.2.3 Selection

In the selection process typically a small number of parents are chosen for breeding. These parents may be, or may not be, successful in producing one or more children. In computer simulation, the propagation success is determined by pseudo-random number generation. As with natural selection, mates are chosen from the better individuals in the population. Computer simulation of mate selection is by weighted random selection. The fitter individuals are assigned heavier weights. Choosing the better individual more often than weaker individuals is drawn by a weighted random selection process. Two common selection mechanisms are:

- Proportional Selection, and

- Tournament Selection.

In Proportional Selection a simulated "roulette wheel" [4] with weighted divisions towards the fitter individuals is 'spun' to determine a suitable parent. This weighting is of the form $f_k/\Sigma_{k=1}^{n} f_k$. A second spin of the roulette wheel selects the partner for recombination.

In Tournament Selection the victor from a tournament among prospective individuals is used to select a fitter parent. The fittest individual, the one with the least fitness value, from a sub-group of randomly selected string-values is chosen as a parent. The partner for mating is also chosen in a similar manner.

3.2.4 Crossover

Crossover operators are used on the selected parents to form children in the next generation. In Natural selection, this is achieved by the splitting and recombining of DNA. In the computer model, a crossover method is used.

In one point crossover, with given probability, the tails of two parents are exchanged at a random position in the string to form two children. If there are a large number of genes in the chromosome, the crossing can be performed on a gene level.

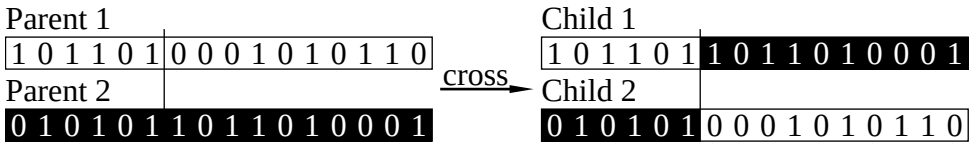


Figure 3.2: One-Point Crossover

In a full replacement strategy the two children are then placed in the next generations' population and the process of selection and crossover repeated until the population is full.

In multi-point crossover exchange locations are chosen in the chromosome in a similar manner. A cross occurs at each exchange position. Figure 3.3 shows an example of two-point crossover.

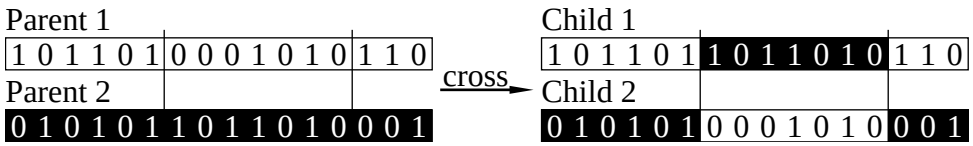


Figure 3.3: Two-Point Crossover

3.2.5 Mutation

In natural selection, an anomaly in the recombination of the DNA may cause a radical shift in the species. The anomaly (mutation) may or may not be successful. The unsuccessful usually die or are infertile and this prevents a continuation of unsuccessful changes.

This is implemented in the basic genetic algorithm in the following way. Each bit in an individual string is considered and with a given probability, the bit value is mutated from 0 to 1 or 1 to 0.

This mutation operator is applied to all children passed to the next generation population. The intuitive purpose behind the mutation operator is that it introduces some extra viability into the population.

3.2.6 Elitism

The above discussion assumed a full replacement policy, that is, the population size remains the same from one generation to the next.

Instead of completing the reproduction process of selection, crossover and mutation to produce children to fill the next population, an elitist strategy is sometimes used. In this strategy the fittest individual or a group of the fittest individuals is multiply copied into the new generation. The remainder of the next population is filled through the normal reproduction process described.

This strategy enhances the reproduction of fit individuals in the problem but may result in premature convergence of the genetic algorithm to a poor solution of the optimisation problem.

3.2.7 Mapping Objective Functions to Fitness Coding

Genetic algorithms that use proportional selection require all fitness evaluations to be positive [4]. If the problem is of a minimisation type, it must be converted to a maximisation problem for solving. Conversion from minimisation to maximisation is simply $g(x) = K - f(x)$, where $K = \max(f(x))$. If the maximum of the problem is unknown or is not easily derivable, then a large positive constant will suffice. For maximisation problems passing through negative values, the function must be shifted to positive only values: $g(x) = f(x) + C_{shift}$. Normalisation is calculated by dividing by the range of function: $g(x) = f(x)/Range$, where $Range = \max(f(x)) - \min(f(x))$. Although normalisation is appealing, it tends to cause crowding of fitness values and thus premature convergence.

Using proportional selection and small populations, a few extraordinary individuals dominate the selection process and cause premature convergence early in a run (high selection pressure). Proportional selection suffers from low selection pressure late in the run because best fitness and average fitness are very close and a parent is chosen from this set of individuals. This tends to slow convergence towards the end of a run.

A way of overcoming these problems is to re-scale the fitness values. Fitness scaling reduces premature convergence by two notable traits:

(i) shrinking the distance among fitness values early in the run, and

(ii) spreading the distance among fitness values later in the run.

By shrinking the distance among fitness values, the selection process keeps diversity by choosing among several individuals instead of choosing a few outstanding individuals very often. The converse occurs for spreading the fitness values. The usual method of fitness scaling is to use the average as a fulcrum [8], and helps to prevent stagnation.

Fitness scaling is not needed when using tournament selection as the selection pressure is controlled by the number of contestants used. For this and other reasons, the tournament selection appears more widely used in the literature. Another reasons for using tournament selection are: simple calculation, and can be used directly for minimisation and maximisation.

3.2.8 Genetic Algorithm Convergence

Figure 3.4 shows a genetic algorithm in more detail using full replacement policy.

```
begin
  t = 0
  for (i=0; i<popsize; i=i+1)
  begin
    Initialise random P(t,i)
    Evaluate Fitness of P(t,i)
  end
  do
  begin
    Scale Fitness Values of P(t)
    for (i=0; i<popsize; i=i+2)
    begin
      Select Parent(i) and Parent(i+1) from P(t)
      Combine to form Child(t,i) and Child(t,i+1)
      Mutate Child(t,i) and Child(t,i+1)
      Evaluate Fitness of Child(t,i) and Child(t,i+1)
    end
    Copy Child(t) to P(t+1)
    t = t+1
  until (t >= maxTime)
end
```

Figure 3.4: GA with Full Replacement Policy

The robustness characteristic of genetic algorithms is generated by "a population" of solutions. Having multiple possible solutions means that if the best individual is bred out by a bad recombination with another individual, parts or all of the good coding still remain in other individuals. In gradient descent, a minima/maxima can be missed and may never be found again. However, with the robustness of genetic algorithms, a minima/maxima can be recovered from other individuals in the population.

Selection mechanism tends to choose more highly fit individuals for recombination. Each new population will contain on average individuals that are better than the current population. As the algorithm converges, the best individual will dominate the population with multiple copies of itself and allele loss is high. Allele loss occurs when a count of alleles (feature values) at one string position (chromosome position) is either zero or the population size.

In super convergence, allele loss is high early in the run and a non-optimum individual dominates the population. Super convergence can be caused by high selection pressure. Tournament selection suffers from high selection pressure as increasing the number of competitors gives a much greater chance that the best individual will be chosen multiple times. Proportional selection has a similar effect when a few poor individuals are in a population of many highly fit individuals. Spreading of the fitness values with scaling helps to prevent super convergence using proportional selection.

Proof of convergence for genetic algorithms is normally shown using schemata.

Genetic algorithms use a binary alphabet $A \in \{0, 1\}$. To discuss schemata, we introduce a third element in the alphabet called the "don't care" condition, symbol '*'. The '*' indicates that this bit position can be either a '0' or a '1', we don't care which. The schemata H alphabet is $A' \in \{0, 1, *\}$. The *order* $o(H)$ is the count of A in the string. The *defining length* $\delta(H)$ is the distance between the first occurring A and the last occurring A in the string. There are 3^ℓ schemata templates available to compare with n strings in the population, where ℓ is the length of the genetic string. At time t there will be m strings which contain a particular schemata H. In proportional selection, a parent is chosen with probability $p_i = f_i/\Sigma f_j$. Using non-overlapping and full replacement,

$m(H, t+1) = m(H, t) f_i(H)/\Sigma f_j$. Comparing the average fitness of individuals to the average fitness of the population:

$$m(H, t+1) = m(H, t)\frac{f(H)}{\overline{f}} \tag{3.4}$$

where, $f(H)$ is the average fitness matching the schemata H and $\overline{f}$ is the average fitness of the population.

Equation 3.4 shows that schemata fitness greater than $\overline{f}$ will increase, and schemata less than $\overline{f}$ will decrease in the next population.

The probability of schemata surviving crossover is $p_s = 1 - \delta(H)/(\ell - 1)$. Applying crossover with probability p_c gives:

$$p_s \geq 1 - p_c \frac{\delta(H)}{\ell - 1} \tag{3.5}$$

Assuming selection and crossover operators are independent, there will be:

$$m(H, t+1) \geq m(H, t)\frac{f(H)}{\overline{f}}\left[1 - p_c \frac{\delta(H)}{\ell - 1}\right] \tag{3.6}$$

schemata surviving to the next population. The survival of a schemata depends on two criteria; fitness above average and short defining length.

Mutation alters A with probability p_m. Therefore, survival of A is $1 - p_m$. Since mutations are statistically independent of each other, selection and crossover operators, a particular schemata survives with probability: $p_s = (1 - p_m)^{o(H)}$. If $p_m << 1$, survival probability is approximately: $p_s = 1 - o(H)p_m$. Therefore, the expected survival of a schemata during selection, crossover and mutation is:

$$m(H, t+1) \geq m(H, t)\frac{f(H)}{\overline{f}}\left[1 - p_c \frac{\delta(H)}{\ell - 1} - o(H)p_m\right] \tag{3.7}$$

Equation 3.7 is a geometric progression that increases for schemata with fitness greater than the population average fitness and decreases for schemata with fitness less than the population average. Therefore, the genetic algorithm can optimise either minimisation problems or maximisation problems. However, using proportional selection limits the genetic algorithm to optimise maximisation only.

The reader is referred to [4] for a more complete analysis of GA convergence using similarity templates.

3.3 De Jong and Function Optimisation

Kenneth A. De Jong [74] performed well known research on genetic algorithm parameter tuning. This work was conducted in 1975, and the computers used were extremely small in comparison to modern machines. De Jong's work involved well formed mathematical equations.

De Jong compared five functions (3.8), (3.9), (3.10), (3.11), (3.12) using six genetic algorithm plans $R1$ to $R6$. Investigations on the effect on convergence while varying: the population size, crossover probability, mutation probability and generation gap were undertaken. Comparisons were reported by "lost allele", on-line and off-line performance indicators.

$$\begin{aligned} f_1(x_i) &= \sum_{i=1}^{3} x_i^2,\ x_i \in [-5.12, -5.11, \cdots, 5.10, 5.11] \qquad (3.8) \\ f_2(x_i) &= 100(x_1^2 - x_2)^2 + (1 + x_1)^2, \end{aligned}$$

$$x_i \in [-2.048, -2.047, \cdots, 2.046, 2.047] \tag{3.9}$$

$$f_3(x_i) = \sum_{i=1}^{5} \text{integer}(x_i),\ x_i \in [-5.12, -5.11, \cdots, 5.10, 5.11] \tag{3.10}$$

$$f_4(x_i) = \sum_{i=1}^{30} i x_i^4 + \text{Gauss}(0,1),\ x_i \in [-1.28, -1.27, \cdots, 1.26, 1.27] \tag{3.11}$$

$$f_5(x_i) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6},$$
$$x_i \in [-65.536, -65.535, \cdots, 65.534, 65.535] \tag{3.12}$$

where

$$[a_{ij}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \cdots & 32 & 32 & 32 \end{bmatrix}$$

De Jong used six optimisation plans:

1. Reproductive plan $R1$ used a full replacement policy, non-overlapping, proportional selection, one point crossover and binary mutation.

2. Reproductive plan $R2$, elitist model, used plan $R1$ with added elitism. If the next population did not contain the elite individual from current population, the least fit individual was replaced by this elite individual.

3. Reproductive plan $R3$, expected value model, assigned an expected selection count to each record. The selection count was calculated by $f_i/\overline{f}$. Every time an individual was selected, its count was reduced by 0.5 or 1.0 until the count became negative. This was an attempt the reduce the stochastic errors caused by proportional selection.

4. Reproductive plan $R4$, elitist expected value model, combined plans $R2$ and $R3$.

5. Reproductive plan $R5$, crowding factor model, newly made strings replaced older similar parents in the hope of keeping diversity with overlapping populations.

6. Reproductive plan $R6$, generalised crossover model, used multi-point crossover.

De Jong investigated the effects of genetic parameters in a methodical scientific way. The genetic parameters investigated were: population size, selection probability, mutation probability and generation gap. Population sizes used were: 50, 100, 200. Mutation probability rates used were: 0.001, 0.005, 0.01, 0.02, 0.1. To show the results, De Jong, developed two performance indicators:, on-line, Equation 3.13, and off-line, Equation 3.14, performance.

$$x_e(s) = \frac{1}{T}\sum_{1}^{T} f_e(t) \tag{3.13}$$

where: $f_e(t)$ is the objective function value for environment e on trial t, being the running average of all fitness evaluations up to and including the current trail.

$$x_e^*(s) = \frac{1}{T}\sum_{1}^{T} f_e^*(t) \tag{3.14}$$

where: $f_e^*(t) = \text{best}\{f_e(1), f_e(2), \cdots, f_e(t)\}$, being the running average of of the best performance values to a particular time.

He found that the elitist model performed well on unimodal surfaces, but both performance indices degraded on multi-modal surfaces. Larger populations give better off-line performance due to diversity. Inertia of larger populations causes poorer initial on-line performance, while small population have better initial on-line performance because they change more rapidly. Examining allele loss, showed that the mutation probability should be as high as possible, promoting good diversity. However, if the mutation is too high, it degrades the convergence of the algorithm.

The work of De Jong showed that mutation is an important operator in genetic algorithms, and this was contrary to popular belief at that time.

3.4 Genetic Algorithms and Parallel Processors

Genetic algorithms are suitable for parallel evaluation. Each string represents an independent solution to all other strings in the population. Having solution independence means that it is possible to spread the genetic algorithm across a number of parallel processors up to the population size.

Having a programme execute on several processors means that each process (or task) must communicate with another at some time or be sequenced in some way. Literature shows that there are three basic methods of controlling a concurrent programme [75, 76].

(i) Semaphores and Monitors,

(ii) Message Based Interaction, and

(iii) Operation Oriented Programmes.

Method (ii) is used in this thesis by use of the Portable Message-Passing Interface (MPI) standard from MPICH [77]. Tasks can be allocated to processing by three types of parallel techniques:

(i) Blocking

(ii) Data Mining (Interleaving), and

(iii) Processor farming.

The task is broken down into several blocks of code that can be computed sep-

arately from other block where possible in the blocking technique. Usually the number of sub-tasks is less than or equal to the number of processors in blocking. Each sub-task should execute in approximately the same time to prevent one child process delaying the entire process.

Data mining or interleaving is used by breaking the task into many components and are issued to processors in an interleaving manner. The number of sub-tasks is greater than the number of processors in interleaving. Each sub-task should execute in approximately the same time to prevent one child process delaying the entire process.

Processor farming is suitable for many sub-tasks that have different execution times. The two methods above have the parent process dispensing the sub-tasks. In processor farming, the child asks for a new task after completing the current task. Processor farming is the method used in this thesis.

If a programme can be broken into independent sections, then it is suitable for concurrent programming. While there are methods to help decompose a sequential programme into multiple independent programmes, they are not required for genetic algorithms. Genetic algorithms search by using a population of independent individuals. Each individual can be evaluated individually and the objective result sent to the parent process for genetic operations to be performed. Making a genetic programme into a parallel programme requires a little more thought then just described.

A simple method of programming is to split the main loop in Figure 3.4 to that shown in Figure 3.5.

```
begin
  t = 0
  for (i=0; i<popsize; i=i+1)
  begin
    Initialise random P(0,i)
    Evaluate Fitness of P(0,i)
  end
  do
  begin
    for (i=0; i<popsize; i=i+2)
    begin
      Select Parent(i) and Parent(i+1) from P(t)
      Combine to form Child(t,i) and Child(t,i+1)
      Mutate Child(t,i) and Child(t,i+1)
    end

    // This loop is calculated using multiple processes
    for (i=0; i<popsize; i=i+1)
    begin
      Evaluate Fitness of Child(t,i)
    end

    Copy Child(t) to P(t+1)
    t = t+1
  until (t >= maxTime)
end
```

Figure 3.5: GA using Parallel Section for Fitness Evaluation

If the population is maintained in the parent process and transferred to child processes for objective calculation, the traffic between parent and child processes can be very high. For example: transmitting a population of size 10Mbytes over 100Mbit/s TCP/IP LAN takes more than one second of time. If the objective of an individual takes less than one second to compute, then the programme is better run in serial mode on a single processor.

The objective of concurrent processing is to minimise the communications between processes and maximise the amount of work in each process. The scenario

above can be solved by maintaining the population in each process with a small amount of communications. However, this is not necessary in this thesis as paths are evaluated from multiple starting configurations. Figure 3.6 shows the pseudo code implemented in this thesis.

```
begin
  for (InitConfig = 0; InitConfig < TotConfig; InitConfig++)
  begin

    // This section is calculated using multiple processes
    t = 0
    for (i=0; i<popsize; i=i+1)
    begin
      Initialise random P(0,i)
      Evaluate Fitness of P(0,i)
    end
    do
    begin
      for (i=0; i<popsize; i=i+2)
      begin
        Select Parent(i) and Parent(i+1) from P(t)
        Combine to form Child(t,i) and Child(t,i+1)
        Mutate Child(t,i) and Child(t,i+1)
        Evaluate Fitness of Child(t,i)
      end
      Copy Child(t) to P(t+1)
      t = t+1
    until (t >= maxTime)
    Send individual with best fitness to parent process

    Write individual with best fitness to file
  end
end
```

Figure 3.6: Parallel GA's Solving Velocity Profiles

3.5 Evolutionary Algorithms

As noted above, binary encoding of a possible solution string for a given real optimisation problem can lead to a loss of precision. The basic issue here is that we have to code the real problem to fit the GA structure. Michalewicz [27] comments:

> "It seems that GAs are too domain independent to useful in many applications.
>
> ...
>
> we can transform the problem into a form appropriate for the genetic algorithm, or we can transform the genetic algorithm to suit the problem."

The transformation of the genetic algorithm is termed by Michalewicz, "Evolution program". In this thesis we shall use the name "Evolutionary algorithm" that remains consistent with current terminology.

The departure here is from a classical bit string approach with a basis in evolutionary theory, to more appropriate data structures for more complex problems together with modified operators that have little or no theoretical basis in evolution.

The mathematical algorithm "Evolutionary algorithm" becomes a mathematical tool that has been found extremely useful in solving complex optimisation problems. This too has been the path for "artificial neural networks", [6].

Convergence theory for these algorithms and analysis of performance is still an open topic of research. Yet their application and usefulness to solve real world problems is evident from the vast research literature that now exists.

The basic structure of the evolutionary algorithms is same as given for the genetic algorithm, yet for each different data structure used to represent the individual as a solution to the given optimisation problem, one has to develop suitable range reproduction operators in order that the algorithm

```
begin
   n = 0
   Create random P(0)
   Evaluate Fitness of P(0)
   while (not Terminated) do
   begin
      Create P(n+1) from P(n)
      Evaluate Fitness of P(n+1)
      n = n+1
   end
end
```

Let us assume that in a given optimisation problem each string in the population has the vector form

$$\underset{\sim}{x}_k = [\ x_1\ \cdots\ x_j\ \cdots\ x_n\ ]^T \tag{3.15}$$

in which each component variable x_j is real, may lie within a constrained range, and the n components constituent a solution to the optimisation problem.

In the reproduction process, proportional selection and tournament selection can still be used to determine mating parents in the population.

One point crossover is still applicable in the generation of children for the next population, with the crossover point lying between one of the n component variables. Other variants of this crossover exist for real encoded algorithms.

A popular crossover operator is the arithmetic crossover operator defined as follows.

Given two parents $\underset{\sim}{x}_1$ and $\underset{\sim}{x}_2$ selected by tournament then two children $\underset{\sim}{c}_1$ and $\underset{\sim}{c}_2$ are determined by the equations:

$$\underset{\sim}{c}_1 = \alpha_c \underset{\sim}{p}_1 + (1 - \alpha_c)\underset{\sim}{p}_2.$$

$$\underset{\sim}{c}_2 = (1 - \alpha_c)\alpha_c \underset{\sim}{p}_1 + \alpha_c \underset{\sim}{p}_2.$$

with constant parameter $\alpha_c \in [0, 1]$. If the search space is convex, then the two offspring will again lie within the search space. Many variants of this arithmetic crossover also exist, [27].

Clearly binary bit mutation is no longer applicable. Indeed mutation operators are very different as each gene is now a real variable and is mutated within a dynamic range, [27].

Let us assume that each component of $\underset{\sim}{x}_k$ is bounded simply by

$$x_{lb} \leq x_j \leq x_{ub}.$$

Here we have assumed the same lower bound and upper bound for each x_j but the argument can be extended for varying upper and lower bounds for each component variable.

The **random** mutation operator selects a random component x_m and produces the new component x'_m which is a random value in the range $[x_{lb}, x_{ub}]$ using a uniform probability distribution.

The **non-uniform** mutation operator selects a random component x_m and produces the new component x'_m defined by

$$x'_m = \begin{cases} x_m + \Delta(n, x_{ub} - x_m) & \text{if a random binary digit is 0} \\ x_m - \Delta(n, x_m - x_{lb}) & \text{if a random binary digit is 1} \end{cases}$$

The function Δ is typically defined by

$$\Delta(n, y) = ry\,(1 - n/maxgen)^b \in [0, y],$$

where r is a random number from $[0, 1]$, n is the current generation number, $maxgen$ is the maximum number of generations and b is a system parameter determining the degree of non-uniformity. This operator covers the search space uniformly initially (when the generation number is small) and very locally at the later stages of evolution.

The reader is referred to Michalewicz [27] for further examples of the mutation operator.

3.6 Constraints

Most real problems involve the optimisation of non-linear functions of many variables and are subject to non-linear equality and inequality constraints.

These problems are described as general non-linear programming problems and typically have the mathematical form

Minimise

$$f(\underset{\sim}{x}) \tag{3.16}$$

subject to p *equality* constraints

$$c_i(\underset{\sim}{x}) = 0, \quad i = 1, \cdots, p \ ,$$

and q *inequality* constraints

$$c_i(\underset{\sim}{x}) \leq 0, \quad i = p+1, \cdots, p+q = m \ .$$

There is no known method for determining the global minimum to such a general non-linear problem except in the specific circumstances when the objective function f and the constraints c_i satisfy certain properties. However many classical optimisation search techniques exist, typically based on multi-variable calculus that use the gradient function and require the functions to be continuously differentiable, [29]. The most widely developed techniques use the concept of penalty functions to convert the problem to an unconstrained optimisation function, and penalise infeasible solutions, that is, those $\underset{\sim}{x}$ which do not satisfy the constraints.

We define:

$$F(\underset{\sim}{x}) = \begin{cases} f(\underset{\sim}{x}) & \text{if } \underset{\sim}{x} \text{ is feasible,} \\ f(\underset{\sim}{x}) + P(\underset{\sim}{x}) & \text{otherwise.} \end{cases}$$

Here the penalty $P(\underset{\sim}{x}) = 0$ if no violation of constraint occurs, and is positive otherwise. The penalty is usually made up of a sum of functions f_k which measure the violation of the kth constraint as follows:

$$f_k(\underset{\sim}{x}) = \begin{cases} max\{0, c_j\}(\underset{\sim}{x}) & \text{if } 1 \leq j \leq p, \\ |c_j(\underset{\sim}{x})| & \text{if } p+1 \leq j \leq m. \end{cases}$$

Many of the current methods use a penalty term of the form

$$P(\underset{\sim}{x}) = \sum_{k=1}^{m} \lambda_k f_k^{\beta}(\underset{\sim}{x}),$$

where the coefficients λ_k may be constant or varying with generation number and the parameter $\beta = 2$. By increasing the coefficients, pressure on infeasible solutions is increased and are evolved out of the population of possible solutions to the optimisation problem.

In our development of a fuzzy knowledge base for the control of a mobile robot in later chapters we have need to introduce various penalty terms into the optimisation process to secure various physical and time constraints. More details will therefore be given in these chapters. The reader is referred to [27] for further discussion and implementation of a number of recent techniques in handling constraints.

Chapter 4

Design of Fuzzy Systems

4.1 Introduction

The design of a fuzzy system can be done using a variety of methods. Wang [8] discusses the following methods: Table Look-Up Scheme, Gradient Descent Training, Recursive Least Squares, and Nearest Neighbourhood Clustering. The rules of a fuzzy system can also be learnt using Evolutionary Algorithms.

Recursive Least Squares has not been used in this thesis and is thus beyond the scope of this thesis. A brief discussion on the other methods follows.

4.2 Table Look-Up Scheme

Given N input-output pairs $(\underset{\sim}{x}^{\ell}, \underset{\sim}{y}^{\ell})$, $\ell = 1, \cdots, N$, the procedure for using the tabling method is as follows:

Step 1: Define fuzzy sets to cover the input and output spaces that are complete ($\mu_{A_i^j}(x_i) \neq 0$).

Step 2: Generate one rule from one input-output pair.

Step 3: Assign a degree to each rule generated in Step 2.

Step 4: Create the fuzzy rule base.

4.3 Gradient Descent Training

In order to keep the number of tuning parameters small and use a fuzzy system that is easily differentiable, a fuzzy system using: Mamdani product inference, Gaussian fuzzifier and centre average defuzzification can be developed. This type of fuzzy system is described by Equation 4.1.

$$f(x) = \frac{\sum_{l=1}^{M} \overline{y}^{\ell} \prod_{i=1}^{N} \exp\left(\frac{-(x_i - \overline{x}_i^{\ell})^2}{(\sigma_i^{\ell})^2}\right)}{\sum_{\ell=1}^{M} \prod_{i=1}^{N} \exp\left(\frac{-(x_i - \overline{x}_i^{\ell})^2}{(\sigma_i^{\ell})^2}\right)} \tag{4.1}$$

where: M is fixed and $\overline{y}^{\ell}$, $\overline{x}_i^{\ell}$ and σ_i^{ℓ} are to be determined.

In the Gradient Descent Training Method the objective is to tune three parameters: $\overline{y}^{\ell}$, $\overline{x}_i^{\ell}$ and σ_i^{ℓ} by taking a step down the direction of the greatest gradient. To ease the calculation of the partial derivative, Equation 4.1 can be broken into smaller parts that abide by the chain rule of differentiation. Equation 4.1 can be separated to:

$$f = a/b \tag{4.2}$$

$$a = \sum_{\ell=1}^{M} \overline{y}^{\ell} z^{\ell} \tag{4.3}$$

$$b = \sum_{\ell=1}^{M} z^{\ell} \tag{4.4}$$

$$z^\ell = \prod_{i=1}^{N} \exp\left(\frac{-(x_i - \overline{x}_i^\ell)^2}{(\sigma_i^\ell)^2}\right) \tag{4.5}$$

Tuning each parameter requires a step down the gradient that reduces the error of the function compared to the objective value:

$$e^p = \frac{1}{2}[f(x_0^p) - y_0^p]^2 \tag{4.6}$$

The next value of $\overline{y}^\ell$ is a step down the gradient that reduces the error 4.6:

$$\begin{aligned}
\overline{y}^\ell(q+1) &= \overline{y}^\ell(q) - \alpha \left.\frac{\partial e^p}{\partial \overline{y}^\ell}\right|_q \\
&= \overline{y}^\ell(q) - \alpha(f(x_0^p) - y_0^p)\frac{\partial f}{\partial a}\left.\frac{\partial a}{\partial \overline{y}^\ell}\right|_q \\
&= \overline{y}^\ell(q) - \frac{\alpha z^\ell(f(x_0^p) - y_0^p)}{b}
\end{aligned} \tag{4.7}$$

The next value of $\overline{x}_i^\ell$ is a step down the gradient that reduces the error:

$$\begin{aligned}
\overline{x}_i^\ell(q+1) &= \overline{x}_i^\ell(q) - \alpha \left.\frac{\partial e^p}{\partial \overline{x}_i^\ell}\right|_q \\
&= \overline{x}_i^\ell(q) - \alpha(f(x_0^p) - y_0^p)\frac{\partial f}{\partial z^\ell}\left.\frac{\partial z_\ell}{\partial \overline{x}_i^\ell}\right|_q
\end{aligned}$$

$$\begin{aligned}
\frac{\partial f}{\partial z^\ell} &= \frac{b\frac{\partial a}{\partial z^\ell} - a\frac{\partial b}{\partial z^\ell}}{b^2} \\
&= \frac{b\overline{y}^\ell - a}{b^2} \\
&= \frac{\overline{y}^\ell}{b} - \frac{a}{b^2} \\
&= \frac{\overline{y}^\ell - f}{b}
\end{aligned}$$

$$\frac{\partial z^\ell}{\partial \overline{x}_i^\ell} = \exp\left(-\left(\frac{x_i - \overline{x}_i^\ell}{\sigma_i^\ell}\right)^2\right) \cdot \frac{\partial}{\partial \overline{x}_i^\ell}\left[\prod_{k=1}^{N} \exp\left(-\left(\frac{x_k - \overline{x}_k^\ell}{\sigma_k^\ell}\right)^2\right)\right]_{k \neq i}$$

$$
\begin{aligned}
&+ \quad \frac{2(x_i - \overline{x}_i^\ell)}{(\sigma_i^\ell)^2} \exp\left(-\left(\frac{x_i - \overline{x}_i^\ell}{\sigma_i^\ell}\right)^2\right) \left[\prod_{k=1}^{N} \exp\left(-\left(\frac{x_k - \overline{x}_k^\ell}{\sigma_k^\ell}\right)^2\right)\right]_{k \neq i} \\
&= \quad \exp\left(-\left(\frac{x_i - \overline{x}_i^\ell}{\sigma_i^\ell}\right)^2\right) \cdot \frac{\partial}{\partial \overline{x}_i^\ell} \left[\prod_{k=1}^{N} \exp\left(-\left(\frac{x_k - \overline{x}_k^\ell}{\sigma_k^\ell}\right)^2\right)\right]_{k \neq i} \\
&+ \quad z^\ell \cdot 2\left(\frac{x_i - \overline{x}_i^\ell}{(\sigma_i^\ell)^2}\right) \\
&= \quad 2z^\ell \left(\frac{x_i - \overline{x}_i^\ell}{(\sigma_i^\ell)^2}\right)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\overline{x}_i^\ell(q+1) &= \overline{x}_i^\ell(q) - \alpha \left.\frac{\partial e^p}{\partial \overline{x}_i^\ell}\right|_q \\
&= \overline{x}_i^\ell(q) - \frac{2\alpha z^\ell (f(x_{0i}^p) - y_0^p)(\overline{y}^\ell(q) - f(x_{0i}^p))(x_{0i}^p - \overline{x}_i^\ell(q))}{b(\sigma_i^\ell(q))^2} \qquad (4.8)
\end{aligned}
$$

$$
\begin{aligned}
\sigma_i^\ell(q+1) &= \sigma_i^\ell(q) - \alpha \left.\frac{\partial e^p}{\partial \sigma_i^\ell}\right|_q \\
&= \sigma_i^\ell(q) - \alpha (f(x_0^p - y_0^p) \frac{\partial f}{\partial z_\ell} \left.\frac{\partial z_\ell}{\partial \sigma_i^\ell}\right|_q
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial z^\ell}{\partial \sigma_i^\ell} &= \exp\left(-\left(\frac{x_i - \overline{x}_i^\ell}{\sigma_i^\ell}\right)^2\right) \cdot \frac{\partial}{\partial \sigma_i^\ell} \left[\prod_{k=1}^{N} \exp\left(-\left(\frac{x_k - \overline{x}_k^\ell}{\sigma_k^\ell}\right)^2\right)\right]_{k \neq i} \\
&+ \quad \frac{2(x_i - \overline{x}_i^\ell)^2}{(\sigma_i^\ell)^3} \exp\left(-\left(\frac{x_i - \overline{x}_i^\ell}{\sigma_i^\ell}\right)^2\right) \left[\prod_{k=1}^{N} \exp\left(-\left(\frac{x_k - \overline{x}_k^\ell}{\sigma_k^\ell}\right)^2\right)\right]_{k \neq i} \\
&= \quad 2z^\ell \frac{(x_i - \overline{x}_i^\ell)^2}{(\sigma_i^\ell)^3}
\end{aligned}
$$

Therefore,

$$
\sigma_i^\ell(q+1) = \sigma_i^\ell(q) - \alpha \left.\frac{\partial e^p}{\partial \overline{x}_i^\ell}\right|_q
$$

$$= \sigma_i^\ell(q) - \frac{2\alpha z^\ell (f(x_{0i}^p) - y_0^p)(\overline{y}^\ell(q) - f(x_{0i}^p))(x_{0i}^p - \overline{x}_i^\ell(q))^2}{b(\sigma_i^\ell(q))^3} \quad (4.9)$$

The procedure for gradient decent is:

Step 1: Set M, $\overline{y}^\ell(0)$, $\overline{x}_i^\ell(0)$, and $\overline{\sigma}_i^\ell(0)$

Step 2: For a given input-output pair $(x_0^p; y_0^p)$, $p = 1, 2, \cdots$, and at the qth step in training, $q = 1, 2, \cdots$, present x_0^p to:

$$\begin{aligned} z^\ell &= \prod_{i=1}^{N} \exp\left(\frac{-(x_{0i}^p - \overline{x}_i^\ell(q))^2}{(\sigma_i^\ell(q))^2}\right) \\ b &= \sum_{\ell=1}^{M} z^\ell \\ a &= \sum_{\ell=1}^{M} \overline{y}^\ell(q) z^\ell \\ f &= a/b \end{aligned}$$

Step 3: Update the parameters $\overline{y}^\ell(q+1)$, $\overline{x}^\ell(q+1)$ and $\sigma_i^\ell(q+1)$ using Equations 4.7, 4.8, and 4.9.

Step 4: $q = q + 1$, goto Step 2 until $|f(x_{0i}^p) - y_0^p| < \epsilon$ or $q > Q$, where Q is the predetermined maximum number of iterations.

Step 5: $p = p + 1$, goto Step 2 until $p > P$, no more input-output data.

This method is usually tested from at least a few initial parameters in order to demonstrate that an acceptable local optimum has been found.

4.4 Nearest Neighbourhood Clustering

The key objective here is to develop a universe of fuzzy sets using Mamdani product inference, Gaussian fuzzifier and centre average defuzzification. This type of fuzzy system is described by Equation 4.10.

$$f(x) = \frac{\sum_{l=1}^{M} y_0^l \exp\left(\frac{-(x-x_0^l)^2}{\sigma^2}\right)}{\sum_{l=1}^{M} \exp\left(\frac{-(x-x_0^l)^2}{\sigma^2}\right)} \tag{4.10}$$

Assuming there are N input-output pairs: $(\underset{\sim}{x}^\ell, \underset{\sim}{y}^\ell)$, $\ell = 1, \cdots, N$. These input-output pairs are to be clustered to $\underset{\sim}{x}_C^k$, $k = 1, \cdots, M$ using neighbourhood clustering.

The process for nearest neighbourhood clustering is as follows:

Step 1: $\ell = 1$, $M = 1$, and Select radius r.

Step 2: $\underset{\sim}{x}_C^M = \underset{\sim}{x}^\ell$, $A^M = \underset{\sim}{y}^\ell$, $B^M = 1$.

Step 3: $\ell = \ell + 1$.

Step 4: Find $\min_{j=1}^{M} |\underset{\sim}{x}^\ell - \underset{\sim}{x}_C^j|$.

Step 5: If $|\underset{\sim}{x}^\ell - \underset{\sim}{x}_C^j| > r$, Then $M = M + 1$, $\underset{\sim}{x}_C^M = \underset{\sim}{x}^\ell$, $A^M = \underset{\sim}{y}^\ell$, $B^M = 1$.
Else $A^j = A^j + \underset{\sim}{y}^\ell$, $B^j = B^j + 1$.

Step 6: While $\ell < N$, Goto Step 3

The complete fuzzy controller using clustering is described by Equation 4.11.

$$f(x) = \frac{\sum_{m=1}^{M} A^m \exp\left(\frac{-(x-x_C^m)^2}{\sigma^2}\right)}{\sum_{m=1}^{M} B^m \exp\left(\frac{-(x-x_C^m)^2}{\sigma^2}\right)} \tag{4.11}$$

Obtaining a fuzzy controller with the required number of membership sets is determined by the radius r. A large r results in a small number of membership sets with coarse control. A small r results in a large number of membership sets with fine control. The best radius gives a reasonable number of memberships that provide adequate control. The smoothness of the transition between fuzzy membership sets is controlled by σ. A large σ gives smooth transition between fuzzy memberships and reduces the amount of $NULL$ fuzzy sets for some untrained input. Fine control is given with small σ, however, may allow $NULL$ fuzzy memberships to appear for untrained inputs.

4.5 Evolutionary Learning of Fuzzy Systems

4.5.1 Genetic Fuzzy Systems

The above methods of finding a suitable fuzzy system have particular quirks. They all require input output pairs as inputs to training a fuzzy knowledge base. The Tabling Method also requires the fuzzy memberships determined before training. The Gradient Descent method tunes each input-output pair individually and thus does not necessarily result in the global knowledge base for the entire system. Fuzzy Clustering does produce a global knowledge base in the form of M rules of one membership each, resulting in large computation requirements by not utilising the extension principle.

Another method of finding the fuzzy knowledge base is to evolve chromosomes containing the output set centres $\overline{y}^{\ell}$, $\ell = 1, \cdots, M$ and providing membership sets for the fuzzy inputs prior to training. Using the very powerful search method of evolutionary algorithms avoids error accumulation in back-propagation as in

Gradient Descent and permits the use of the extension principle. However, determining the membership sets prior to running the algorithm can cause the fuzzy system to be inadequate or over specified. An in depth discussion of this genetic learning of fuzzy systems can be found in [51]. The process of evolutionary learning of fuzzy rules will be given in full detail for each problem discussed in later Chapters.

Fuzzy membership parameters may be learnt using an evolutionary algorithm by appending them to the end of the chromosome, [52]. Care must be exercised when using this method as different evolutionary operators may need to be used in the $\overline{y}^{\ell}$, $\ell = 1, \cdots, M$ part of the chromosome and the fuzzy membership parameter section.

Encoding a hierarchical fuzzy system is done by catenation of the output centres of each fuzzy knowledge base $\overline{y}^{\ell}$, $\ell = 1, \cdots, M$. Let a fuzzy rule base be represented by a string of output centres $\underset{\sim}{s}_k = \{s^1, s^2, \cdots, s^k, \cdots, s^M\}$, where s^k represents each $\overline{y}^{\ell}$. Then, the catenated string appears as: $\underset{\sim}{s} = \{\underset{\sim}{s}_1, \underset{\sim}{s}_2, \cdots, \underset{\sim}{s}_N\}$ for an N layer system.

Calculation of the objective function of each fuzzy system uses an extracted subset of the chromosome. The evolution of an initial random population proceeds as described for the basic evolutionary algorithm incorporating possible modifications to the essential operators that are used to ensure convergence of the algorithm.

In the Pittsburgh approach each individual evolves independently of the other individuals. In the Michigan approach the individuals must coevolve by evaluating how well they perform when combined with other individuals. This is essentially

the essence to strategy tasking amongst agents (mobile robots) in a multi-agent system. In general, this coevolutionary approach decomposes a high-level composite into sub-components, which become specialised in interaction with the other ones. As a result of collaboration of these sub-components, the high-level goal is effectively achieved. According to a study on the high-dimensional fuzzy classification application, the Michigan approach is reported to be superior to the Pittsburgh approach but this is yet to be proven.

Mohammadian and Stonier have used the Pittsburgh approach in evolving a full knowledge base for each robot in their robot simulation system [71]. It is this approach that will be developed and used for application in this thesis.

4.6 Fuzzy Amalgamation

Generally an evolutionary algorithm works best for the optimisation of functions that use every allele in a chromosome. This can be seen in the experiments by De Jong [74]. Each function $F1, \cdots, F5$ used every allele in a chromosome to calculate the fitness of the function.

In developing fuzzy control for mobile robots, a subset of alleles from a chromosome is used in the fitness evaluation. The effectiveness of crossover and mutation can be diminished when using a subset of a chromosome as crossover point may or may not have bearing on the result of subset of chromosome $p_C = p_C \times \delta(H)/\ell$, where ℓ is the chromosome length, used for a single robot path and mutation probability is reduced by: $p_m = p_m \times \text{subset count}/\ell$ for a particular path.

Evolutionary learning of fuzzy control rules is essentially a straight forward pro-

cess when evaluating the fitness for a system which evolves from a single given initial state or configuration. This can be seen from the work of Mohammadian and Stonier (see References in the Introduction), and in [85].

The process of learning a set of fuzzy control recognised in the sense of a "feedback" or "closed" loop controller, requires that the fuzzy controller, whether in single of multi-layered, be learnt across a grid of initial states or configurations. This is very clearly the case that one has to consider in the development of a multi-layered hierarchical fuzzy controller as has already been mentioned. The Mamdani product inference engine ensures that this controller has a measure of continuity across the space of all initial states or configurations, provided that the grid is sufficiently fine.

Fuzzy amalgamation, as defined earlier, can be used in two ways: as a post process or "on-the-fly" process in an evolutionary algorithm.

Post process amalgamation begins with an evolutionary algorithm to find a fuzzy knowledge base that yields a "good" path described from an initial state or configuration. Each fuzzy knowledge base is stored in an associated separate file. Let N files contain a chromosome consisting of M alleles. There are a_i^ℓ, $i = 1, \cdots, M$, $\ell = 1, \cdots, N$ alleles to be considered. Let unused alleles be denoted by $NULL$. There will be m_j used alleles for each a_i^ℓ, $i = 1, \cdots, M$, $j = 1, \cdots, N$. Then amalgamation will produce a global chromosome b, built by:

$$b^\ell = \frac{\sum_{i=1}^{M} a_i^\ell}{m^\ell}, \quad \ell = 1, \cdots, N \tag{4.12}$$

where: m_i is the count of non-$NULL$ alleles.

Amalgamation "on-the-fly" is performed at the end each generation. A subpopulation of chromosomes are amalgamated by using Equation 4.12. Each ini-

tial configuration is run for a few generations to exercise various parts of the chromosome. It works in a similar way to building new neural pathways in the brain. The more heavily a path way is traversed, the more likely it will survive, see [70, 71].

Chapter 5

Robot Modelling

5.1 Introduction

The basic robot soccer system considered is that defined for the Federation of Robot-soccer Association, Robot World Cup [1]. All calculations for vision data processing, strategies and position control of the robots are performed on a centralised host computer. Full specifications of hardware, software and basic robot strategies that are employed in this type of micro-robot soccer system can be found in [78].

Field sizing has changed over the years. The old field playable region had dimensions of $1300 \times 900mm$. The current playable region in use has dimensions of $1500 \times 1300mm$.

5.2 Robot Parameters

The robot parameters were obtained by experimentation using a Stroboscope flashing at thirty Hertz and an analogue film camera set to bulb (B) setting. Both motors were connected in parallel to the robot battery. A thin white line was painted on top of the robot. The wheels were held to lock the motor rotors and the camera shutter opened. The robot was released and the camera shutter closed when either the robot reached the opposite end of the field or became unstable ending in a pirouette. The motor torque was found to be exceptionally high giving one reasonably straight path in eighteen photographs. The motor torque was reduced by placing a series resistance of 2.7Ω between the paralleled motors and the battery. The value of the resistor was determined through experimentation. Small valued resistors were tried until one was found that gave maximum torque without causing wheel slippage. Table 5.1 shows the data obtained from this experiment.

The distance between the goalkeeper boxes in the photograph was 119mm, which is a scale of 119/1200mm. It was assumed that the slope of the lines between successive points could be used to satisfactorily determine the acceleration. The change in distance values were plotted using Microsoft Excel and fitted using linear regression. The line of best fit was:

$$y = 5.35x + 4.89 \tag{5.1}$$

giving an acceleration of: $5.35 \times 30 \times 30/1000 = 4.8\text{ms}^{-2}$.

To verify the crude approximation to acceleration, a quadratic function is now fitted to the position plot of the results from the robot acceleration experiment.

Table 5.1: Experimental Results of Robot Parameter Test

Position in Photo	Distance from Datum (mm)	ΔDistance (mm)	Scaled to field (mm)
0	0		
1	0.91	0.91	9.17
2	2.42	1.51	15.23
3	4.40	1.98	19.97
4	6.84	2.44	24.61
5	10.23	3.39	34.18
6	14.09	3.86	38.92
7	18.26	4.17	42.05
8	22.94	4.68	47.19
9	28.32	5.38	54.25
10	34.17	5.85	58.99
11	40.72	6.55	66.05
12	47.38	6.66	67.16
13	54.70	7.32	73.82
14	62.46	7.76	78.25

The system of equations are:

$$\begin{aligned} y_1 &= ax_1^2 + bx_1 + c \\ y_2 &= ax_2^2 + bx_2 + c \\ &\vdots \\ y_n &= ax_n^2 + bx_n + c \end{aligned} \tag{5.2}$$

The system of equations from Equation 5.2 can be expressed in matrix form as:

$$\mathbf{Y} = \mathbf{AX} \text{ where } \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \mathbf{A} = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{5.3}$$

This is an over-determined system and can be determined using the least squared

method. Least squares requires a function to minimise such as:

$$E = \sum_{i=1}^{n} [y_i - ax_i^2 - bx_i - c]^2 \tag{5.4}$$

Now taking the partial derivatives of the three quantities being minimised:

$$\begin{aligned} \frac{\partial E}{\partial a} &= -2\sum_{i=1}^{n} x_i^2[y_i - ax_i^2 - bx_i - c] = 0 \\ \frac{\partial E}{\partial b} &= -2\sum_{i=1}^{n} x_i[y_i - ax_i^2 - bx_i - c] = 0 \\ \frac{\partial E}{\partial c} &= -2\sum_{i=1}^{n} [y_i - ax_i^2 - bx_i - c] = 0 \end{aligned} \tag{5.5}$$

Expanding the sums of Equation 5.5 and using $\sum_{i=1}^{n} c = nc$, the system of equations becomes:

$$\begin{aligned} \left(\sum_{i=1}^{n} x_i^4\right) a + \left(\sum_{i}^{n} x_i^3\right) b + \left(\sum_{i=1}^{n} x_i^2\right) c &= \sum_{i=1}^{n} x_i^2 y_i \\ \left(\sum_{i=1}^{n} x_i^3\right) a + \left(\sum_{i}^{n} x_i^2\right) b + \left(\sum_{i=1}^{n} x_i\right) c &= \sum_{i=1}^{n} x_i y_i \\ \left(\sum_{i=1}^{n} x_i^2\right) a + \left(\sum_{i}^{n} x_i\right) b + \qquad nc &= \sum_{i=1}^{n} y_i \end{aligned} \tag{5.6}$$

The equivalent matrices [80] to Equation 5.6 are:

$$\mathbf{A}^T\mathbf{A}\mathbf{X} = \mathbf{A}^T\mathbf{Y} \tag{5.7}$$

The x_i quantities are from Table 5.1, "Position in Photo" times $\Delta t = 1/30$ seconds. The y_i values are from Table 5.1, "Distance from Datum" times $1200/(119\times 1000)$ to get "Scaled to field values".

Provided that the data points do not lie on the same vertical line, the system of equations can be solved using:

$$\mathbf{X} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y} \tag{5.8}$$

where:

$$\mathbf{A} = \frac{1}{900}\begin{bmatrix} 0 & 0 & 900 \\ 1 & 30 & 900 \\ 4 & 60 & 900 \\ 9 & 90 & 900 \\ 16 & 120 & 900 \\ 25 & 150 & 900 \\ 36 & 180 & 900 \\ 49 & 210 & 900 \\ 64 & 240 & 900 \\ 81 & 270 & 900 \\ 100 & 300 & 900 \\ 121 & 330 & 900 \\ 144 & 360 & 900 \\ 169 & 390 & 900 \\ 196 & 420 & 900 \end{bmatrix} \quad \text{and} \quad \mathbf{Y} = \frac{1200}{119 \times 1000}\begin{bmatrix} 0 \\ 0.91 \\ 2.42 \\ 4.40 \\ 6.84 \\ 10.23 \\ 14.09 \\ 18.26 \\ 22.94 \\ 28.32 \\ 34.17 \\ 40.72 \\ 47.38 \\ 54.70 \\ 62.46 \end{bmatrix}$$

Substitution of $\mathbf{A}$ and $\mathbf{Y}$ in Equation 5.8 gives:

$$\mathbf{X} = \begin{bmatrix} 2.42 \\ 0.228 \\ -1.56 \times 10^{-3} \end{bmatrix} \tag{5.9}$$

Acceleration is obtained by the second derivative of the quadratic equation giving: $y'' = 2a = 2 \times 2.42 = 4.8\text{ms}^{-2}$. This verifies the approximation of acceleration made earlier.

The unloaded maximum speed of the robot was calculated by using the Stroboscope and a white painted dot on the robot wheel as a tachometer. Rotational speed to make the white dot appear stationary was: 1120rpm. The robot wheel diameter of 44.6mm was measured by using Vernier callipers. The unloaded maximum linear speed of the robot is: $44.6 \times 10^{-3}/2 \times 1120 \times 2\pi/60 = 2.6\text{ms}^{-1}$. The maximum change in distance in the photograph was 7.78mm, which translates to a loaded speed of $7.78/1000 \times 1200/119 \times 30 = 2.4\text{ms}^{-1}$.

The acceleration of the robot with the series resistance had approximately the

same acceleration as without the resistor. However, the final speed was slightly lower because of the voltage drop across the resistor. This experiment suggests that the motor should be current limited to bring the torque to a reasonable level for traction.

5.3 Kinematics

The kinematics an omni-directional robot is given by Equation 5.10 from [81]. This equation is equally valid for multi-directional wheel chair style robots.

$$\begin{bmatrix} v_C \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} v_L \\ v_R \end{bmatrix} \tag{5.10}$$

where:

- v_L in ms^{-1} is the instantaneous tangential speed for the left wheel of the robot,
- v_R in ms^{-1} is the instantaneous tangential speed for the right wheel of the robot,
- L in m is the wheel base length ($L = 68.5 \times 10^{-3}$m for our robots),
- v_C in ms^{-1} is the instantaneous speed of the robot centre, and
- ω in rad s^{-1} is the instantaneous angular speed about the instantaneous point of rotation (x_0, y_0) (refer to Figure 5.1),

The distance between (x_0, y_0) and v_C is the instantaneous radius of curvature r and is determined by $v_C = r\omega$. Substituting in Equation 5.10 gives:

$$r = \frac{L}{2}\left(\frac{v_L + v_R}{v_R - v_L}\right) \tag{5.11}$$

Let the next robot position be approximated by a small time interval Δt. Assume v_L and v_R are constant over this interval. If $\omega = 0$, the robot is moving in a straight line. Equation 5.12 gives the next robot position using linear displacement $\Delta s = v_C \Delta t$.

$$\begin{bmatrix} x'_R \\ y'_R \\ \phi'_R \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \\ 0 \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\phi_R) \\ \Delta s \sin(\phi_R) \\ \phi_R \end{bmatrix} \tag{5.12}$$

When $\omega \neq 0$, the robot scribes an arc. Curvilinear robot paths are calculated using translation, rotation and translation Equation 5.13. Angular position is determined by $\Delta\theta = \omega \Delta t$. Refer to Figure 5.1 for the following derivation:

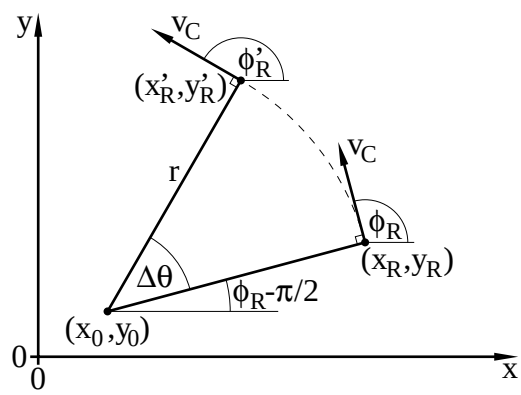


Figure 5.1: Curvilinear Formulae Symbols

The point of rotation (x_0, y_0) is determined by:

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \end{bmatrix} - r \begin{bmatrix} \cos(\phi_R - \pi/2) \\ \sin(\phi_R - \pi/2) \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \end{bmatrix} + r \begin{bmatrix} -\sin(\phi_R) \\ \cos(\phi_R) \end{bmatrix}$$

This is translated to the origin:

$$\begin{bmatrix} x_R^1 \\ y_R^1 \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = r \begin{bmatrix} \sin(\phi_R) \\ -\cos(\phi_R) \end{bmatrix}$$

The effects of rotation around the z-axis are included (counter-clockwise positive):

$$\begin{bmatrix} x_R^2 \\ y_R^2 \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & \sin(\Delta\theta) \\ -\sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} x_R^1 \\ y_R^1 \end{bmatrix}$$

$$
\begin{aligned}
&= r \begin{bmatrix} \cos(\Delta\theta) & \sin(\Delta\theta) \\ -\sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} \sin(\phi_R) \\ -\cos(\phi_R) \end{bmatrix} \\
&= r \begin{bmatrix} \sin(\phi_R)\cos(\Delta\theta) - \cos(\phi_R)\sin(\Delta\theta) \\ -\cos(\phi_R)\cos(\Delta\theta) + \sin(\phi_R)\sin(\Delta\theta) \end{bmatrix} \\
&= r \begin{bmatrix} \sin(\phi_R + \Delta\theta) \\ -\cos(\phi_R + \Delta\theta) \end{bmatrix}
\end{aligned}
$$

The effect to translate back to (x_0, y_0) is included:

$$
\begin{aligned}
\begin{bmatrix} x'_R \\ y'_R \end{bmatrix} &= r \begin{bmatrix} \sin(\phi_R + \Delta\theta) \\ -\cos(\phi_R + \Delta\theta) \end{bmatrix} + \begin{bmatrix} x_R \\ y_R \end{bmatrix} + r \begin{bmatrix} -\sin(\phi_R) \\ \cos(\phi_R) \end{bmatrix} \\
&= \begin{bmatrix} x_R \\ y_R \end{bmatrix} + r \begin{bmatrix} \sin(\phi_R + \Delta\theta) & -\sin(\phi_R) \\ -\cos(\phi_R + \Delta\theta) & \cos(\phi_R) \end{bmatrix}
\end{aligned}
$$

Conclude by including the robot angle correction:

$$
\begin{bmatrix} x'_R \\ y'_R \\ \phi'_R \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \\ \phi_R \end{bmatrix} + \begin{bmatrix} \sin(\phi_R + \Delta\theta) & -\sin(\phi_R) & 0 \\ -\cos(\phi_R + \Delta\theta) & \cos(\phi_R) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r \\ r \\ \Delta\theta \end{bmatrix} \tag{5.13}
$$

where $\phi'_R \in [0, 2\pi)$ is necessarily constrained for input into the fuzzy system.

A video camera provides 60 interlaced fields per second. Assuming position can be determined from each frame, the simulation of the robot used time steps of $\Delta t = 1/60$ seconds. The next robot position is calculated by using Equation 5.11 and Equation 5.12, or Equation 5.11 and Equation 5.13.

One constraint of particular concern is the problem caused by a wheel lifting off the ground while turning. This constraint is evident in most teams playing robot-soccer.

5.4 Constraints

Nearly all physical systems involve one or more constraints. To simulate a real system properly, one must also consider handling these constraints. Several con-

straints are used in the robot simulation:

1. Time,
2. Boundaries,
3. Robot to ball impact, and
4. Wheel lift

5.4.1 Time Constraint

Time constraints are included to assist in the determination of fitness for various control strategies.

5.4.2 Boundary Constraint

The Cartesian coordinate model used the boundaries of the field as set out in [78]. This model was an early one and used the 1300×900 field. The path was terminated if any part of the robot exited the field rectangle. A large penalty was applied if the robot-ball impact angle was outside an arbitrary range: $5\pi/6 \leq \theta < 7\pi/6$. If this range was found to be experimentally erroneous, the range would be adjusted and a new fuzzy controller developed.

The relative coordinate model used a predefined maximum distance between the robot and the ball. Scribing the path was stopped and no penalty was applied when the robot exceeded this bound.

5.4.3 Robot to Ball Impact Constraint

Care needs to be taken recognising the finite size of the robot. The robot is a square with a side of 80mm and the ball has a diameter of 42.7mm. Using a relative coordinate system with the origin translated to the robot centre and rotated to match the robot angle ϕ_R. The robot-ball collision is found by checking if the position of the ball centre is within three testing regions.

Refer to Figure 5.2 for the following derivation of the distance between the ball and the robot using the relative coordinates along the line (AL) and normal to the line (NL) axes.

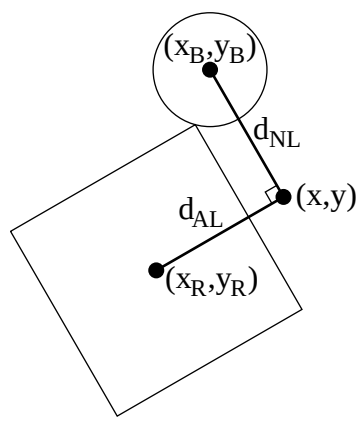


Figure 5.2: Ball Impact Symbols

The gradient of the line passing through the centre of the robot, in the direction of the robot is:

$$m_R = \tan(\phi_R) = \frac{\sin(\phi_R)}{\cos(\phi_R)}, \quad \text{when } \cos(\phi_R) \neq 0$$

Using the equation of a line passing through point (x_R, y_R):

$$y - y_R = m_R(x - x_R) \tag{5.14}$$

and the equation of a perpendicular line passing through (x_B, y_B):

$$\begin{aligned} y - y_B &= \frac{-1}{m_R}(x - x_B) \\ y &= y_B - \frac{x - x_B}{m_R} \end{aligned} \tag{5.15}$$

To find the intersection of these lines, substitute Equation 5.15 into Equation 5.14:

$$\begin{aligned} m_R(x - x_R) &= \left(y_B - \frac{x - x_B}{m_R}\right) - y_R \\ y_B - y_R &= m_R(x - x_R) + \frac{x - x_B}{m_R} \\ m_R(y_B - y_R) &= m_R^2 x - m_R^2 x_R + x - x_B \\ (m_R^2 + 1)x &= m_R(y_B - y_R) + m_R^2 x_R + x_B \\ x &= \frac{m_R(y_B - y_R) + m_R^2 x_R + x_B}{m_R^2 + 1} \end{aligned}$$

Substitute x in Equation 5.14:

$$\begin{aligned} y &= y_R + \frac{m_R}{m_R^2 + 1}(m_R(y_B - y_R) + m_R^2 x_R + x_B - x_R(m_R^2 + 1)) \\ &= y_R + \frac{m_R}{m_R^2 + 1}(m_R(y_B - y_R) + (x_B - x_R)) \end{aligned}$$

The distance of the ball along the line passing through the robot centre (x_R, y_R) and the intersection point (x, y) is:

$$\begin{aligned} d_{AL}^2 &= (x - x_R)^2 + (y - y_R)^2 \\ &= \left(\frac{m_R(y_B - y_R) + m_R^2 x_R + x_B - x_R(m_R^2 + 1)}{m_R^2 + 1}\right)^2 \\ &+ \left(y_R + \frac{m_R}{m_R^2 + 1}[m_R(y_B - y_R) + (x_B - x_R)] - y_R\right)^2 \\ d_{AL}^2(m_R^2 + 1)^2 &= [m_R(y_B - y_R) + (x_B - x_R)]^2 + (m_R[m_R(y_B - y_R) \\ &+ (x_B - x_R)])^2 \\ &= (m_R^2 + 1)[m_R(y_B - y_R) + (x_B - x_R)]^2 \\ d_{AL} &= \frac{|(x_B - x_R) + m_R(y_B - y_R)|}{\sqrt{m_R^2 + 1}} \end{aligned} \tag{5.16}$$

A similar derivation for the distance of the ball perpendicular to the line passing through the robot centre is calculated using:

$$\begin{aligned} d_{NL}^2 &= (x - x_B)^2 + (y - y_B)^2 \\ d_{NL} &= \frac{|(y_B - y_R) - m_R(x_B - x_R)|}{\sqrt{m_R^2 + 1}} \end{aligned} \tag{5.17}$$

In the case when $\cos(\phi_R) = 0$, the distance along the line is $d_{AL} = |y_B - y_R|$ and the distance perpendicular to the line is $d_{NL} = |x_R - x_B|$.

Equation 5.16 and Equation 5.17 are used to detect a collision between the robot and the ball. Three testing regions are used:

1. a rectangle of size $(d_{AL}, d_{NL}) = (40, 61.35)$,
2. a rectangle of size $(d_{AL}, d_{NL}) = (61.35, 40)$, and
3. a circle at $r_{corner}^2 = (d_{AL} - 40)^2 + (d_{NL} - 40)^2$.

The path is terminated when a flag "HitBall" is raised with the following condition:

IF $(((d_{NL} < 40)$ AND $(d_{AL} < 61.35))$ OR $((d_{NL} < 61.35)$ AND $(d_{AL} < 40))$ OR $(r_{corner}^2 < 61.35^2))$ THEN (HitBall = TRUE).

Simulation in the relative coordinate system allowed the ball to be fixed at a stationary point $(x_B, y_B) = (750, 650)$. The simulation can use either the current robot position or the next robot position(x'_R, y'_R), depending on order in programming the iteration loop.

5.4.4 Wheel Lift Constraint

Wheel lift occurs when the centrifugal force exceeds the weight about the outer wheel of the robot while scribing an arc. If a wheel lifts off the ground, the robot is no longer in control and behaves chaotically. In order to control the robot fully, both wheels should remain in contact with the ground. Refer to Figure 5.3 for the following derivation of the wheel lift constraint presented in Equation 5.19.

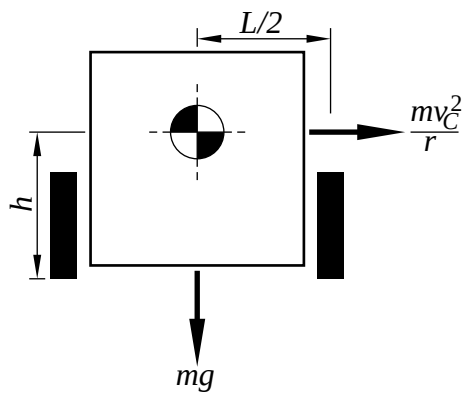


Figure 5.3: Wheel Lift Diagram

Assuming that friction is sufficient to prevent wheel slip, wheel lift occurs when the torque about the outer wheel caused by centrifugal force exceeds the torque caused by gravity.

Two cases must be considered. One for positive angular velocity ($v_R > v_L$) and the other for negative angular velocity ($v_R < v_L$). Equating torques about the outer wheel for positive angular velocity:

$$\frac{mgL}{2} - \frac{mv_C^2 h}{r} = 0 \tag{5.18}$$

Substituting $v_C = (v_L + v_R)/2$ from Equation 5.10 and Equation 5.11 into Equa-

tion 5.18 gives:

$$v_R^2 - v_L^2 = \frac{gL^2}{h}$$

For the case of negative angular velocity, the equation becomes:

$$v_L^2 - v_R^2 = \frac{gL^2}{h}$$

Combining both equations, wheel lift occurs when:

$$|v_L^2 - v_R^2| < \frac{gL^2}{h} \tag{5.19}$$

Near to the boundary of Equation 5.19, the inner wheel has a very low down force. The torque of the motor will cause wheel spin. To keep the equation small and to allow short computation time in simulation, Equation 5.19 is degraded to:

$$|v_L^2 - v_R^2| \leq \frac{gL^2}{h}$$

The robots used, have a mass centroid height of thirty-seven millimetres. Figure 5.4 shows greyed regions which cause wheel lift from Equation 5.19. The figure shows that 37% of the selectable velocities are unconstrained for $-2.6 \leq v_L \leq 2.6ms^{-1}$ and $-2.6 \leq v_R \leq 2.6ms^{-1}$. The allowable velocities contain a squared region of velocities free from wheel lift constraint between $-1.115 \leq v_L \leq 1.115ms^{-1}$ and $-1.115 \leq v_R \leq 1.115ms^{-1}$ giving 18.4% of the velocity range. Many teams have encountered this constraint on robot control and have developed strategies to avoid wheel lift, such as the "KEYS" [79] competitors.

The two parameters mass centroid height and wheel base width determine the wheel lift constraint. Lowering the mass centroid height to $12mm$ and extending

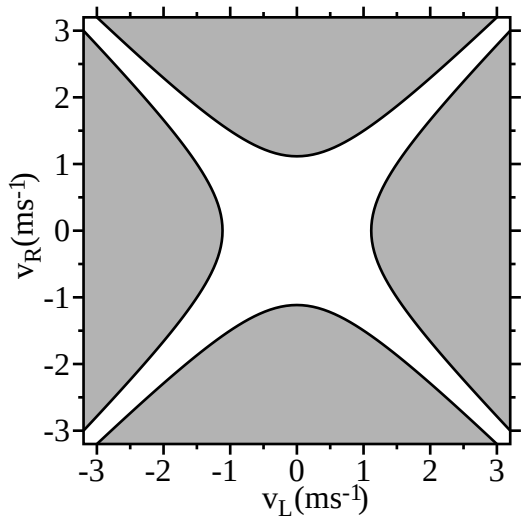


Figure 5.4: Wheel Lift Diagram with $L = 68.5mm$ and $h = 37mm$

the wheel base width to $75mm$ gives the possible maximums that are physically realisable. Figure 5.5 show the wheel lift constraint with these parameters. The unconstrained velocity selection is 87% of the possible velocity selections. A squared region free from wheel lift constraint between $-2.14 \leq v_L \leq 2.14\text{ms}^{-1}$ and $-2.14 \leq v_R \leq 2.14\text{ms}^{-1}$ gives 68% of the velocity range.

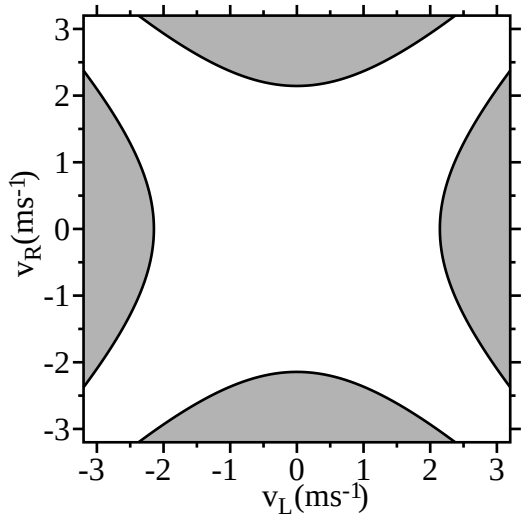


Figure 5.5: Wheel Lift Diagram with $L = 75mm$ and $h = 12mm$

5.5 Cartesian Coordinate System

Initial testing of the robot fuzzy control used a Cartesian coordinate system. The x-axis and y-axis directions were normalised to reduce the error in calculations by the floating point unit (FPU) of the computer. The ball was positioned at $(0.5, 0.8)$. The fuzzy knowledge base was learnt using a genetic algorithm over a grid of initial configurations.

5.5.1 Fuzzy Control using Cartesian Coordinates

Three fuzzy inputs were: $x_1 = x_R$, $x_2 = y_R$ and $x_3 = \phi_R$. The five fuzzy sets defined for x_R were: **L** is Left as $\mu_L(x_1; 0, 0, 0.15)$, **LM** is Left Middle as $\mu_{LM}(x_1; 0.1, 0.25, 0.4)$, **M** is Middle as $\mu_M(x_1; 0.35, 0.5, 0.65)$, **MR** is Middle Right as $\mu_{MR}(x_1; 0.6, 0.75, 0.9)$, and **R** is Right as $\mu_R(x_1; 0.85, 1, 1)$. Five fuzzy sets defined for y_B were: **B** is Bottom as $\mu_B(x_2; 0, 0, 0.15)$, **BM** is Bottom Middle as $\mu_{BM}(x_2; 0.1, 0.25, 0.4)$, **M** is Middle as $\mu_M(x_2; 0.35, 0.5, 0.65)$, **MT** is Middle Top as $\mu_{MT}(x_2; 0.6, 0.75, 0.9)$, and **T** is Top as $\mu_T(x_2; 0.85, 1, 1)$. Eight fuzzy sets defined for ϕ_R were: **EN** is Extremely Negative as $\mu_{EN}(x_3; -\pi, -\pi, -\frac{5\pi}{7})$, **LN** is Large Negative as $\mu_{LN}(x_3; -\pi, -\frac{5\pi}{7}, -\frac{3\pi}{7})$, **MN** is Medium Negative as $\mu_{MN}(x_3; -\frac{5\pi}{7}, -\frac{3\pi}{7}, -\frac{\pi}{7})$, **SN** is Small Negative as $\mu_{SN}(x_3; -\frac{3\pi}{7}, -\frac{\pi}{7}, \frac{\pi}{7})$, **SP** is Small Positive as $\mu_{SP}(x_3; -\frac{\pi}{7}, \frac{\pi}{7}, \frac{3\pi}{7})$, **MP** is Medium Positive as $\mu_{MP}(x_3; \frac{\pi}{7}, \frac{3\pi}{7}, \frac{5\pi}{7})$, **LP** is Large Positive as $\mu_{LP}(x_3; \frac{3\pi}{7}, \frac{5\pi}{7}, \pi)$, and **EP** is Extremely Positive as $\mu_{EP}(x_3; \frac{5\pi}{7}, \pi, \pi)$. Shown in Figure 5.6.

A single fuzzy output variable $y = \Delta\theta$ with centre locations $\overline{y}_k = -0.4 + 0.1k,\ k = 0, \cdots, 7$ was used to update the robot position by: $\phi'_R = \phi_R + \Delta\theta$, $x'_R = x_R + v_R \cos(\phi'_R)$ and $y'_R = y_R + v_R \sin(\phi'_R)$. The robot velocity v_R was a constant

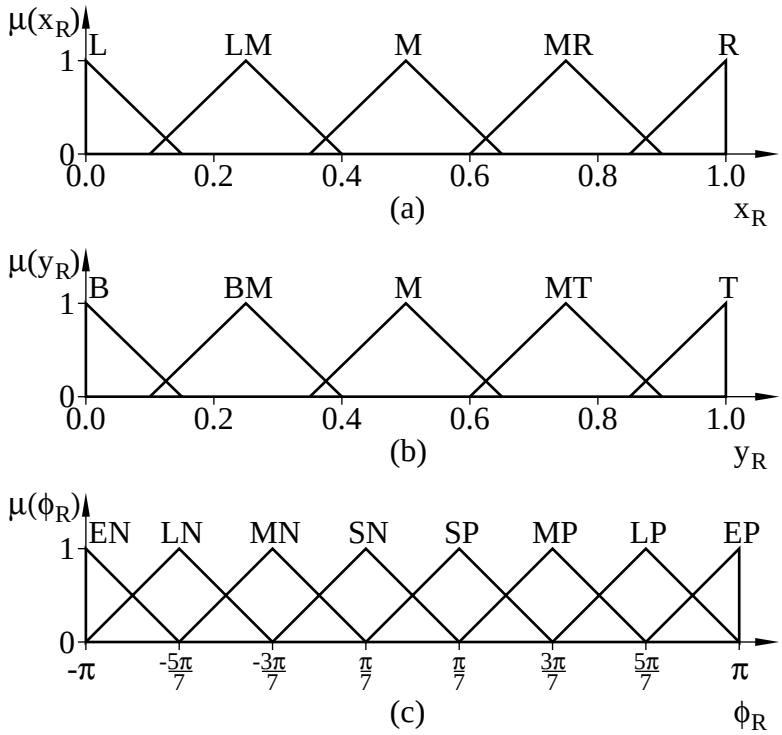


Figure 5.6: Membership Sets for Cartesian Fuzzy System

0.005 units in normalised space. The time step was set at one unit. Adjustment caused by normalising the rectangular field was not considered as this was a crude simulation to test the effectiveness of the evolutionary fuzzy controller for this application.

There were $5 \times 5 \times 8 = 200$ rules in a complete fuzzy knowledge base for this system. In general the ℓth fuzzy rule had the form:

$$\text{If } (x_1 \text{ is } A_1^\ell \text{ and } x_2 \text{ is } A_2^\ell \text{ and } x_3 \text{ is } A_3^\ell) \text{ Then } (y \text{ is } B^\ell)$$

where $A_k^\ell, \quad k = 1, 2, 3$ were normalised fuzzy sets for input variables $x_k, \quad k = 1, 2, 3$, and B^ℓ are normalised fuzzy sets for output variable y.

Given a fuzzy rule base of M rules, a fuzzy controller using a singleton fuzzifier,

Mamdani product inference engine and centre average defuzzifier was used to determine the output variable as given in 7.1. These values, 200 of them, are typically unknown and require determination in establishing valid output for controlling the robot. With the absence of a priori knowledge about this system, a genetic algorithm was used to search for an acceptable solution.

5.6 Relative Coordinate System

Using the Cartesian coordinate system places the ball at one grid position. However, to learn a suitable control system using this coordinate system would require additional inputs into the fuzzy system namely: the x-coordinate of the ball x_B and the y-coordinate of the ball y_B. The inputs to the fuzzy controller would be $x_1 = x_R$, $x_2 = y_R$, $x_3 = \phi_R$, $x_4 = x_B$ and $x_5 = y_B$. This system would require $5 \times 5 \times 8 \times 5 \times 5 = 5000$ rules in the fuzzy knowledge base. This rule base appears excessively large for merely conveying positional information (the curse of dimensionality).

A better method of defining positional information is to use the ball as a point of reference. A relative coordinate system was developed to reduce the number of rules in the fuzzy knowledge base and decrease the computational load in learning the rules. Figure 5.7 illustrates the relative coordinate system chosen.

The angle of the robot relative to the ball goal line was used instead of the ball robot line because of positional error arising from image capture pixel size in determining the position of each object. The vision system has an inherent $\pm 4.5mm$ error caused by pixel size. The pixel size error causes the angle of the $\overline{BG}$ line error to be inversely proportional to the distance between the points

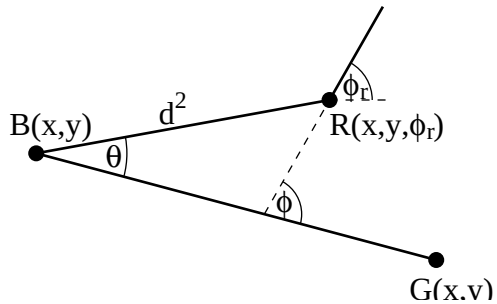


Figure 5.7: Relative Coordinate Parameters

used to calculate the line. However, one of the points used to calculate the $\overline{BG}$ line is at the centre of the goal line. The allowable angle range when the ball is close to the goal offsets the error caused in determining the line. The vision system error has negligible effect on placing the ball into the goal when using $\overline{BG}$ as a reference.

The robot position and angle is derived from blob analysis of colour patches on the robot. The angle is calculated from the centres of the patches. Figure 5.8 shows a robot with relative coordinates at centre. The error in calculating the

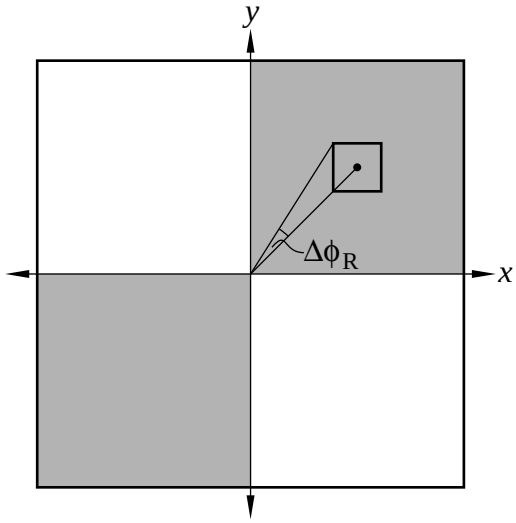


Figure 5.8: Robot with relative coordinate system at centre

robot angle is given by:

$$\Delta\phi_R = 45^\circ - \tan^{-1}(15.5/24.5) = 13^\circ \tag{5.20}$$

The calculated coordinates of the robot are: $(x_R \pm 4.5\text{mm}, y_R \pm 4.5\text{mm}, \phi_R \pm 13^\circ)$. Similarly, the coordinates of the ball are: $(x_B \pm 4.5\text{mm}, y_B \pm 4.5\text{mm})$.

Calculating the angle of the robot to the ball-robot line increases the angular error as both robot and the ball have position errors. The calculation using a square caused by error in the x-axis and y-axis is cumbersome, and can be approximated by a circle of the same area:

$$r = \ell/\sqrt{\pi} \tag{5.21}$$

where r is the radius of the circle and ℓ is the length of one side of a square.

Using this circle to approximate position errors, the robot-ball line $\overline{RB}$ angular error is:

$$\Delta\overline{BR} \approx \tan^{-1}\left(\frac{2r}{|x_R - x_B|}\right) \tag{5.22}$$

When a stationary ball is hit in the centre of the robot face, the angular error of the $\overline{BR}$ is: 9°. The maximum total error in calculating the robot relative to the robot-ball line is: $13 + 9 = 22^\circ$. The minimum error in calculating the robot-ball line is: 3°.

Conversely, the centre of goal point is a static position and is not derived from the image. The angular error of the $\overline{BG}$ is:

$$\Delta\overline{BG} \approx \tan^{-1}\left(\frac{r}{|x_G - x_B|}\right) \tag{5.23}$$

The minimum angular error using the ball-goal line is $\pm 0.2^\circ$ and maximum approaches 90°. Table 5.2 compares the total robot angular error and the acceptable angle of the ball trajectory to reach the goal.

Table 5.2: Angular robot error using $\overline{BG}$ and acceptable ball to goal trajectory angle

$\lvert x_G - x_B \rvert$	$\Delta\phi_R + \Delta\overline{BG}$	$\tan^{-1}(200/\lvert x_G - x_B \rvert)$
1500	13	8
800	13	14
200	14	45
100	16	63
50	18	76
10	40	87
2	81	89
0	103	90

Chapter 6

Boundary Avoidance

Substantial parts of this chapter were published [23] in proceedings of the Paris 1989 FIRA Robot World Cup Competition. Two aspects of the game of robot-soccer were discussed.

6.1 Introduction

The first involves avoidance of the defined boundaries and restricted regions of the playing field. Observation of games played shows that all teams have inherent difficulties in control near the boundary walls, and in particular with the goal-keeper in the goal area either hitting the end wall or being trapped in the behind goal area. We show how to use a linear, piece-wise linear and continuous limiting functions to adjust both direction and speed of the robot near a field boundary.

The second problem we may define as the "attack the ball" strategy. This is related to basic positioning of a robot to move to a position behind the ball, given any position of the ball on the playing field, and any position of the robot

on the field, so that it can dribble the ball or drive it to the goal. We show how to modify the algorithm shown in [21] to incorporate speed control as well as direction control.

6.2 Basic Strategy Definitions

The implementation of the robot motor control strategies is made through a multi-layered structure. Each layer has an input vector and a corresponding output vector. The output of a given layer becomes a possible input of the next layer. A hierarchical approach is to place low priority modules in the lower layers proceeding to high priority modules in the higher numbered layers. The high priority modules perform the most critical calculations involving direction and speed. Modules and dependencies from the lowest priority up are shown in Figure 6.1.

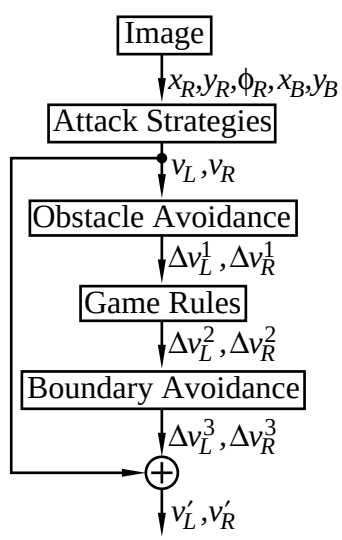


Figure 6.1: Modules and dependencies

The desired heading layer is the first layer and is the lowest priority module.

A new desired heading is calculated from the image information. If the robot is stationary, the module gives an initial heading. If the robot is moving, the module gives a modification to the current heading to allow the robot to traverse a smooth continuous path. This could result in a direction that is required to be modified in a higher priority layer. It may be modified to comply with a game rule, or to comply with boundary avoidance criteria for example.

6.3 Boundary Avoidance

Boundary avoidance is positioned in the programming model as the highest priority layer. Three basic boundary avoidance schemes were considered. These are: "Stop and Turn" as demonstrated in the Micro-Adventure software [82, 78], Artificial Potential Fields (APF), and Angle Limiting Functions (ALF).

6.3.1 Stop and Turn

The code shipped with the robots contained a "Stop and Turn" algorithm for boundary avoidance. This algorithm stops a robot that is on a collision course with the boundary wall and turns the robot to face parallel to the boundary. The robot then proceeds towards its target by following the boundary wall.

The "Stop and Turn" algorithm, as it is used in this Micro-Adventure code, is discontinuous and does not consider the robots velocity component in the direction of the boundary. Consequently, the algorithm often fails to stop a fast moving robot before it collides with the boundary, due to the robots high momentum.

6.3.2 Artificial Potential Fields (APF)

The use of Artificial Potential Fields, see [35, 36, 21], for boundary avoidance generally results in oscillatory robot movement along the boundary. This is caused by the APF directing the desired heading vector sharply away from the boundary and back into the play area. The robots' desired heading redirects the robot into the boundary and the process repeats.

6.3.3 Angle Limiting Functions

Angle Limiting, limits a robots desired heading angle to ensure that the robot does not make contact with the boundary. The degree of limiting that is imposed on a desired heading angle is determined by an Angle Limiting Function (ALF) and the robots velocity toward the boundary. Angle Limiting Functions can be of the type: stepped, linear, piece-wise linear, curvilinear or piece-wise curvilinear.

Stepped (Discontinuous Functions)

Stepped Angle Limiting sets discrete heading angles as a robot moves closer to the boundary. The desired heading angles are sequenced to steer the robot along the boundary wall.

Linear functions

Linear functions provide proportional limiting of the robots desired heading angle and may or may not be velocity dependent. ALF's are easy to implement

and are suitable for on robot processing. However, as with Artificial Potential Fields, Linear functions generally result in oscillatory robot movement along the boundary. As a robot moves close to the boundary, the heading angle is limited to an angle greater than 90° which forces the robot away from the boundary. When ALF's are used with a low speed vision system, the corrective action is generally applied too late and for too long.

Piece-wise linear functions

Piece-wise linear functions provide superior control of the rate of vector change as the robot approaches the boundary. One linear function controls angle limiting up to 90° and another controls angle limiting for angles greater than 90°. Careful adjustment of function parameters can result in an Angle Limiting Profile that permits the robot to travel at high speed along the boundary with minimum oscillatory movement. Figure 6.2 shows the piece-wise linear function used to modify the desired heading vector of the robot implemented in our software.

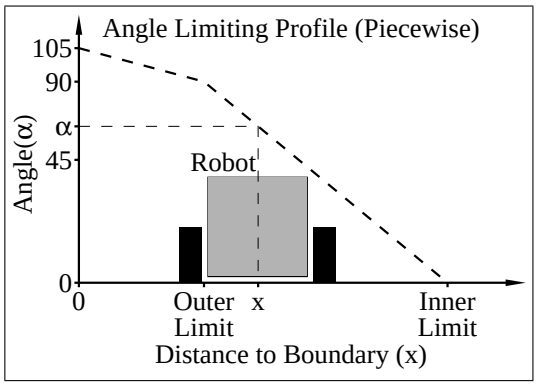


Figure 6.2: Piece-Wise Linear Profile

Continuous Functions

Curvilinear functions, such as cosine or sigmoid functions permit smooth and

well defined non-linear vector control. Curvilinear functions may be used either as continuous functions or piece-wise curvilinear functions.

We elected to use Piece-wise Linear, Angle Limiting Functions for Boundary Avoidance in Mirosot'98. The piece-wise linear functions are defined by three variables; the inner limit, outer limit and a maximum limiting angle. Desired heading angle is modified according to the limiting profile in Figure 6.2 as a function of distance x from a boundary.

The boundary avoidance module does not alter any heading in the range $(180^o - \alpha,\ -180^o + \alpha)$. Angle α is modified by the piece-wise linear profile shown in Figure 6.2. Figure 6.3 shows the effect of the piece-wise linear profile as the robot approaches the boundary. As the robot nears the boundary, the angle α approaches 90°, forcing the robot to travel parallel with the boundary. If the robot moves closer to the boundary, angle limiting by the second piece-wise function coerces the robot away from the boundary.

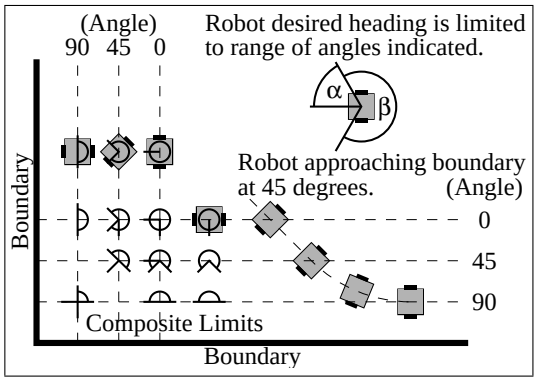


Figure 6.3: Boundary Avoidance

It should be noted that the behaviour of a robot near the boundary depends to a great extent on the robots ability to align its heading vector with its desired heading vector.

Figure 6.3 also illustrates corrective action being applied to a robot approaching the boundary at 45°. No corrective action is applied to the robots desired heading angle until the robot is sufficiently close to the boundary, at which stage the desired heading angle is altered according to the piece-wise function until the robot is moving parallel to the boundary. Figure 6.3 also shows the Angle Limiting Profiles for a corner section as a composite of the profiles used on the two adjacent boundaries. The composite profiles are constructed by logically anding the adjacent functions so robots can only travel in the directions that satisfy both sets of boundary functions.

6.3.4 Proportional Cosine Control

The software accompanying the Micro-Adventure robots employs proportional control for both linear and angular velocities. Equation 6.1 and Equation 6.2 show the original controls packaged with the Micro-Adventure software.

$$v_L = K_p e + K_a \phi, \tag{6.1}$$

$$v_R = K_p e - K_a \phi, \tag{6.2}$$

where:

v_L	is left wheel velocity,
v_R	is right wheel velocity,
K_p	is the proportional gain,
K_a	is the angular gain,
e	is the distance error, and
ϕ	is the angular error.

The control equations propel the robot in the forward direction with a velocity proportional to the distance error e while providing a turning velocity proportional to the angular error ϕ. See Figure 6.4 and Figure 6.5. While these control

laws work adequately under normal circumstances, they must be modified to work more efficiently in more extreme cases.

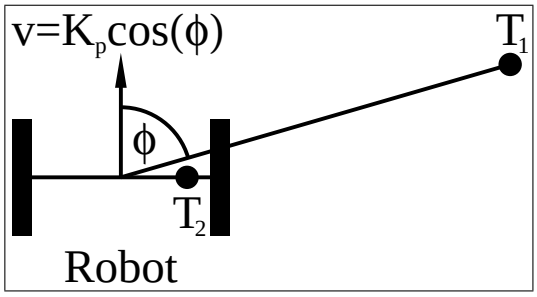


Figure 6.4: Cosine Control

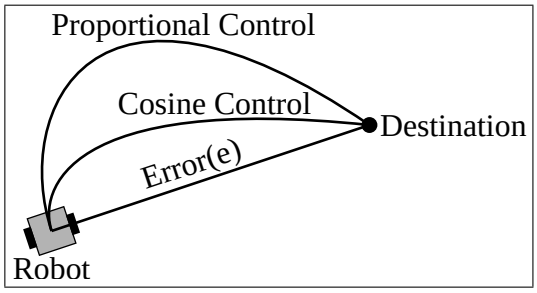


Figure 6.5: Cosine Control Path

To help understand the problems associated with the stated control equations, consider when a robot is orientated at 90° to the target point, and the distance error e is large. The control equations yield large forward velocities that drive the robot around in a large arc toward the destination point as shown by the Proportional Control curve in Figure 6.5. In addition, when the target point is close to the centre of the robot and the angular error ϕ is large, as illustrated by point T_2 in Figure 6.4, the robot has a tendency to move away from that point due to the distance error producing forward motion. To counteract this effect, the proportional gain is reduced, which can hinder the robots performance during other manoeuvres.

A more efficient approach is for the robot to turn and face the target before moving in a forward direction. However, this approach leads to a discontinuous algorithm and creates uncertainty concerning the transition point between turning and moving, particularly when the robot is trying to align with a moving target. The transition decision is usually aided by considering the robot to be sufficiently aligned with the target when the angular error is within a given threshold.

Our solution to this problem was to consider the robots forward velocity component v as a function of the angular error ϕ as shown in Figure 6.4. The outcome is that a robot at 90° degrees to the target has no forward velocity component and a maximum angular velocity. As the robot turns, its forward velocity increases as the angular velocity decreases. When the robot is facing the target, it has maximum forward velocity and no angular velocity.

A control mechanism called a Proportional Cosine Controller was developed in an effort to achieve this type of control. The basic Cosine Controller determines a forward velocity that is proportional to the cosine of the angular error ϕ as shown in Figure 6.4. The resulting controller allows the robot to turn toward a target T_1 much faster, effectively allowing the robots to turn on the spot and move toward the target in one continuous motion. This feature makes Cosine control more suitable for use in positioning algorithms that guide the robot to points close to the robots centre. External speed control or speed limiting, and the exclusion of distance error, can produce a control algorithm that is both continuous and capable of rapid turning.

Equation 6.3 and Equation 6.4 show the basic cosine control equations for left and right wheel velocities:

$$v_L = K_p \cos(\phi) + K_a \phi, \tag{6.3}$$

$$v_R = K_p \cos(\phi) - K_a \phi. \tag{6.4}$$

The decision to include or exclude the use of distance error in Cosine control depends on, the type of behaviour that is desired in the robot, and the overall control philosophy being implemented.

Equation 6.5 and Equation 6.6 show the Proportional Cosine Control for left and right wheel velocities including distance error:

$$v_L = K_p e \cos(\phi) + K_a \phi, \tag{6.5}$$

$$v_R = K_p e \cos(\phi) - K_a \phi. \tag{6.6}$$

Standard P, PI or PID control laws may be used in the linear and angular components of the Cosine control equations. Our preferred choice is to use PID (or incremental PID) control equations to reduce dynamic error.

Figure 6.5 illustrates the path taken by a robot using the original Proportional control and Cosine control.

6.4 Modified Attack Strategy

The algorithm for the "attack strategy" used in Mirosot'98 was developed from the algorithm given in [21] which was designed to guide the robot on a circular

approach path to primary point P_1, bringing the robot onto the tangent alignment with the second point P_2 as illustrated in Figure 6.6.

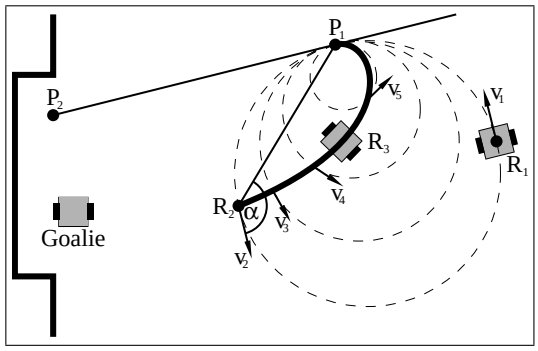


Figure 6.6: Ball Attack Strategy

The refinements result in a more efficient approach path for the robot. The basic algorithm determines a *direction vector* for the robot to follow that causes the robot to traverse a circular path, see Figure 6.6, which forms a tangent with a line through the primary and secondary points P_1 and P_2 respectively. The algorithm successfully guides the robot R_1 on a curved path toward the primary point P_1 where the robot is positioned behind the target. However, a robot located in front of primary point P_1, as is illustrated by R_2, is initially guided away.

To improve the efficiency of the algorithm, the direction vector guiding the robot is modified to provide a shorter path to the primary point. The direction vector produced by the basic algorithm for a robot in position R_2 is shown as v_2. The *modified algorithm* determines the *difference angle* α between the direction vector v_2 and the line (R_2, P_1). A portion of the *difference angle* is added to the angular component of direction vector v_2, to determine a desired heading angle for the robot, that is more in the direction of P_1.

As the robot approaches point P_1, it is desirable to have the robot follow the circular path more closely, to ensure the robot is aligned with the secondary

point P_2, when point P_1 is reached.

A suitable function to control the percentage of α that is added to the vector v_2, is one that is close to 100% when the distance from P_1 is large and close to 0% when the distance from P_1 is small. Function 6.7 is a suitable function for controlling α and has the appearance of a "lead transfer function".

$$H(x) = xT/(1 + xT), \tag{6.7}$$

where T is the *distance constant* in distance units (mm^{-1} or $pixels^{-1}$) and x is the distance.

The distance constant T determines the range at which the robot adheres to the curved path and hence, how quickly the robot approaches point P_1.

6.4.1 Strategy Implementation

The soccer robots used the modified algorithm to perform two basic manoeuvres: attacking the ball and tackling an opponent.

Attacking the Ball

To attack the ball, it is assumed the ball is free from contact with any robot and is either stationary or moving. The robot usually defines primary point P_1 as the ball and defines the secondary point P_2 as either another robot or a target point in the goal area. The robot then proceeds to approach the ball along the $\overline{P_1P_2}$ tangent. The robot can dribble the ball by colliding with the ball at low speed or it can kick the ball by ensuring its speed is high when contact is made with

the ball. When P_2 lies in the goal area, the robot is executing a goal kick and when P_2 is defined as another robot, the robot is passing the ball.

Tackling an Opponent

The Mirosot rules [1] prohibit a robot from intentionally colliding with an opponent. Therefore, to tackle an opponent that has control of the ball, it is essential that a robot approach the opponent from the front and along the line through the ball and the centre of the opponent. Here, the modified approach algorithm is used with primary point P_1 defined as the ball, and secondary point P_2 defined as the centre of the opponent robot. The modified algorithm takes the robot to meet the ball and block the path of the opponent. A turning manoeuvre quickly dislodges the ball, which is then free to be gathered up by either team.

6.5 Multi-Agent Strategies

Currently, the system is a centralised agent. Three roles are assigned in the control programme for: attack, defender and goalkeeper. Roles are assigned to remote robot slaves to execute control strategies. The goal keeper role is assigned to a unique robot slave which remains in the goal area. The attack and defend roles each control one robot, but may exchange robots so that the closest to the ball is controlled by the attacking role. The agent perceives common knowledge of all robots and the ball by video position information. Actuator output is enabled through a radio communication link. Our next step is to begin implementation in a heterogeneous non-communicating domain for multi-agent development.

The playing field is segregated into four equi-sized quadrants to determine the defending position. Figure 6.7 shows the quadrant segregation of the field. If the attacking role robot is in the first quadrant, the defending role robot is sent to quadrant four defence coordinates. If the attacking role robot is in the fourth quadrant, the defending role robot is sent to quadrant one defence coordinates. The mirror image of this technique is used for quadrants two and three.

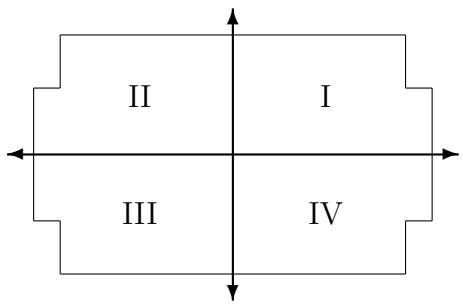


Figure 6.7: Quadrant Division of Field

6.6 Comments

In this research we have given two modifications to existing algorithms in the area of boundary-avoidance and "attack the ball" strategy. In the first we found that the piece-wise linear, angle limiting function used was quite effective in control of the robots near the boundaries. However it required careful evaluation of the defining variables in association with maximum speed control of the robots.

The cosine control algorithm for determining the path to "attack the ball" was found to be superior to the previous algorithm. It too depended on appropriate parameter settings.

Accurate determination of positioning of the robots and ball from information

obtained by the vision system was seen to be critical to both these algorithms. Further, slippage on the surface at high speeds caused inaccurate positioning of the robots when high turning rates were required from various configurations of the ball and robots. To overcome these later problems our team is now building an efficient digital signal processing (DSP) board and a new colour detection system to enable more accurate determination of the position and orientation of all robots on the soccer field, as well as redesigning new robots to allow more accurate control on the playing surface.

The building of the DSP vision processor was a side issue to this thesis. It was designed to process the position of objects on the field at a rate of sixty fields per second. Another team member was building the board until leaving the team. To date the board has not been finished.

Chapter 7

Three Input Two Output (3I2O) Control

Substantial parts of this chapter were published [22] as a chapter in a book: Computational Intelligence in Control. Development of a three input, two output fuzzy controller is considered. The results presented in this chapter are developed through simulation as the resulting controller was incomplete.

7.1 Introduction

This chapter discusses evolving a three input, two output fuzzy knowledge base to control a soccer micro-robot from a grid of initial configurations to hit the ball towards the centre of the goal. The relative coordinate system detailed in Chapter 5 is used and robot front or robot rear contact with the ball is permitted. Physical robot dimensions are used in all simulations as detailed in Chapter 1. Control of the robot includes the physical parameters as detailed in Chapter 5. Consideration is given for far and medium distance initial configuration and spe-

cial consideration is given for initial configurations touching the ball.

A relative coordinate system is used and terms are introduced into the fitness evaluations which allow both forward and reverse motion of the soccer robot. This chapter defines the robot soccer system, the design of the fuzzy controller, the design of the evolutionary algorithm and a discussion of results. Two scenarios are considered: initial configurations at a distance from the ball and a special cases of the robot touching the ball. The kinematics equations used are: Equation 5.10, Equation 5.12, and Equation 5.13.

7.2 Fuzzy Control System Design

The discussion on kinematics shows, excluding momentum and friction, that only two variables, the velocity of the left and right wheels, $y_1 = v_L$ and $y_2 = v_R$ control the motion of the robot. These two variables are taken as output of the fuzzy control system now described below. Figure 7.1 shows the Three Input Two Output model. Input variables for the fuzzy control system are taken to be the position of the robot relative to the ball, described by $n = 3$ variables $x_1 = d^2$, $x_2 = \theta$ and $x_3 = \phi$ as shown in Figure 5.7.

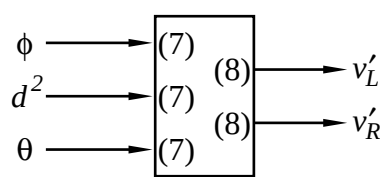


Figure 7.1: Three Input Two Output Model

The robot position relative to the destination point behind the ball is completely defined by using two angles and one distance in this system. As the ball moves,

the destination point appears to be statically bound to a point behind the ball as it lies on the robot-goal line.

These relative coordinates were used in preference to Cartesian coordinate variables for a number of reasons, one being that it reduced the number of rules in the fuzzy KB. Distance squared was used to reduce the calculation cost by not using a square root function. It's effect is to apply a "more or less" hedge. The angle of the robot relative to the ball goal line was used instead of the ball robot line because of positional error arising from image capture pixel size in determining the position of each object. The vision system has an inherent $\pm 4.5mm$ error caused by pixel size. The pixel size error causes the angle of the line error to be inversely proportional to the distance between the points used to calculate the line. However, one of the points used to calculate the $\overline{BG}$ line is at the centre of the goal line. The allowable angle range when close to the goal offsets the error caused in determining the line. The vision system error has negligible effect on placing the ball into the goal when using $\overline{BG}$ as a reference.

Figure 7.2 shows the fuzzy input sets used to define the "attack ball strategy". For all rules seven sets are defined for both angles θ and ϕ: **VS** is Very Small, **S** is Small, **SM** is Small Medium, **M** is Medium, **ML** is Medium Large, **L** is Large and **VL** is Very Large. Five sets are defined for distance squared: **VC** is Very Close, **C** is Close, **N** is Near, **F** is Far and **VF** is Very Far.

The values of y_1 and y_2 are taken to be integers lying in the interval $[-128, 127]$. There are 256 B_k output fuzzy sets each corresponding to centre $\overline{y}_k = -128 + (k-1)$ for $k = 1, \cdots, 256$. In this case the name of the sets are the same as the output centres $\overline{y}_k$ of the sets.

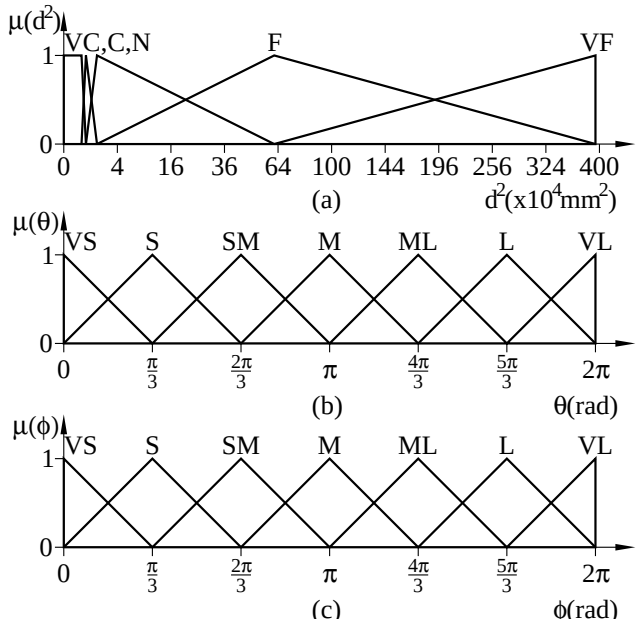


Figure 7.2: Fuzzy Input Sets

The purpose of taking 256 B_k output fuzzy sets instead of 255, $B_k \in [-127, 127]$, is a technical issue to allow the use of a binary mutation operator in the evolutionary algorithm. The velocities are in fact capped to $v_L, v_R \in [-127, 127]$ to simulate the transmission of data to the robots.

before transmission to the robots.

Taking a large number of output sets serves four purposes:

(i) It does not affect the computational cost of the fuzzy controller,

(ii) The solution can be as fine as it needs to be,

(iii) The $\overline{y}_k$ are the control values used for the left and right wheel motors—eliminating conversion, and

(iv) Reduces erratic behaviour of the evolutionary algorithm (finer control) during mutation.

There were $7 \times 7 \times 5 = 245$ rules in a complete fuzzy knowledge base for this system. In general, the ℓth fuzzy rule has the form:

If (x_1 is A_1^ℓ and x_2 is A_2^ℓ and x_3 is A_3^ℓ) Then (y_1 is B_1^ℓ and y_2 is B_2^ℓ).

where $A_k^\ell, k = 1, 2, 3$ are normalised fuzzy sets for input variables $x_k, k = 1, 2, 3$, and where $B_m^\ell, m = 1, 2$ are normalised fuzzy sets for output variables $y_m, m = 1, 2$.

Given a fuzzy rule base with M rules, a fuzzy controller as given in Equation 7.1 uses a singleton fuzzifier, Mamdani product inference engine and centre average defuzzifier to determine output variables.

$$y_k = \frac{\sum_{\ell=1}^{M} \overline{y}_k^\ell (\prod_{i=1}^{n} \mu_{A_i^\ell}(x_i))}{\sum_{\ell=1}^{M} (\prod_{i=1}^{n} \mu_{A_i^\ell}(x_i))} \tag{7.1}$$

where $\overline{y}_k^\ell$ are centres of the output sets B_k^ℓ

These values, 490 of them, are typically unknown and require determination in establishing valid output for controls to each wheel of the robot. Since there is lack of a priori knowledge about the system control, we used evolutionary algorithms [27] to search for an acceptable solution.

7.3 Evolutionary Learning

The objective here is to learn a rule base for the control of this system. The first problem is how to formulate the knowledge base as a string in the population.

Each output fuzzy set is represented by an integer in the interval $[-128, 127]$.

An individual string $\underset{\sim}{s}$ is formed as a string of $2M = 490$ consequents, (integers under the identification above).

$$\underset{\sim}{s} = \{s_1^1, s_2^1, \cdots, s_1^k, s_2^k, \cdots, s_1^M, s_2^M, \},$$

where $s_j, j = 1, 2$ is an integer in the interval $[-128, 127]$.

The population at generation t, $P(t) = \{\underset{\sim}{s}^n \ : \ n = 1, \cdots, N\}$, where N is the number of individuals in the population. The population at the next generation $P(t+1)$ was built using a full replacement policy. Tournament selection with n_T being the tournament size, determined two parent strings for mating in the current generation. One point crossover with probability p_c was used for generating two child strings from the parent strings, for insertion in the next generations population. In each string, the integer components were stored as two's complement byte sized quantities, and binary mutation was undertaken on each string in the new population with probability p_m. (Elitism was not used for it was found to cause premature convergence of the algorithm.)

Fitness evaluation of each individual was calculated by scribing a path using the fuzzy controller and stopping when:

(i) iteration (time steps) reached a prescribed limit (100), or

(ii) the path exceeded the maximum allowable distance from the ball (1954mm), or

(iii) the robot collides with the ball (see Section 5.4.3).

The final position of the path was used to evaluate the fitness of each individual

as given by Equation 7.2.

$$f_i = \sum_C (T_1 + T_2 + T_3 + T_4)\,. \tag{7.2}$$

The fitness is calculated as a sum of a number of different quantities, over a set C of initial starting configurations, each configuration specifying Robot coordinates x_R, y_R, and angle ϕ describing the orientation of the robot relative to the $\overline{BG}$ line.

The first quantity in the fitness sum is $T_1 = d^2(R, DP)$. It is the final squared distance between the robot centre R and the destination point $DP = (688.5, 650)$ when the path is terminated as described above. The term is used to determine accuracy of the fuzzy controller to control the system to the desired destination configuration.

The second term T_2 is the iteration count for each path. This quantity is used to minimise the time taken to reach the desired destination configuration.

The third quantity is $T_3 = 1000\sin^2(\phi)$ where ϕ is the final angle of the robot relative to line $\overline{BG}$. This term is included to enable forward facing and reverse facing solutions to be accepted at the final destination.

The fourth quantity T_4 is a penalty function which is **only** applied for those configurations $c \in [0, 28)$. It is described in Equation 7.3.

$$T_4 = \begin{cases} 10000 & \text{if } \theta \in [11\pi/12, 13\pi/12) \text{ and } (\sin^2(\phi) > 0.25), \\ 10000 & \text{if } \theta \notin [11\pi/12, 13\pi/12), \\ 0 & \text{otherwise.} \end{cases} \tag{7.3}$$

It is a constant penalty used to drive the solution away from paths that hit the ball when considering the first 28 initial configurations. Without this penalty,

the best solutions obtained via evolutionary learning are invariably ones that try to run through the ball.

There were 273 initial configurations. The first 28 are defined with the robot touching the ball on each of its four sides in seven different orientations θ around the ball. The remaining initial configurations were defined with seven θ angles on five distance rings from the ball with seven ϕ angles.

The first 28 configurations $c \in [0, 28)$, are given by:

$$x_R = x_B + 61.35\cos(\theta), \quad y_R = y_B + 61.35\sin(\theta), \quad \text{and} \quad \phi$$

with

$$\begin{aligned} \theta &\in \{0, \pi/3, 2\pi/3, \pi, 4\pi/3, 5\pi/3, 71\pi/36\}, \quad \text{and} \\ \phi &\in \{0, \pi/2, \pi, 3\pi/2\}. \end{aligned}$$

The remaining initial configurations $c \in [28, 273)$, are given by:

$$x_R = x_B + d\cos(\theta), \quad y_R = y_B + d\sin(\theta), \quad \text{and} \quad \phi$$

with

$$\begin{aligned} d &\in \{77.92, 122.75, 310.4, 784.94, 1900\}, \\ \theta &\in \{0, \pi/3, 2\pi/3, \pi, 4\pi/3, 5\pi/3, 71\pi/36\}, \\ \phi &\in \{0, \pi/3, 2\pi/3, \pi, 4\pi/3, 5\pi/3, 71\pi/36\}. \end{aligned}$$

The evolutionary algorithm was terminated after a prescribed number of generations. The best individual, that is, the one with the minimum fitness, is taken as the "best" fuzzy logic controller determined by the algorithm for this problem.

7.4 Discussion

The evolutionary algorithm was found to easily learn a fuzzy controller for when fitness was evaluated for a single initial configuration.

Establishing learning over a set of initial configurations from $c = 0$ to $c = 28$ where the robot was placed in contact with the ball was difficult; appropriate sets of evolutionary parameters needed to be defined with a mutation schedule to ensure diversity in the population for different stages in the learning, for example after every 1000 generations. The reason for the difficulty was that the algorithm tended to lock fuzzy control into always forward or always reverse motion of the robot, with the consequence that not the shortest distance path was achieved, and invariable penalty constraints were broken.

Learning the fuzzy control over the set all configurations incorporated the difficulties that had to be overcome for those configurations close to the ball. The algorithm tended to lock into local minima when considering multiple configurations. The local minima existed due to the algorithm finding a good single path amongst the many that influenced nearby paths.

Typical values in simulations were: the size of the population $N = 200$, probability of crossover $p_c = 0.6$ and the number of tournament contestants $n_T = 8$. Mutation probability was defined as a schedule: $p_m = 0.05 - 0.000048(gen \bmod (1000))$, which decreased mutation with increasing generation number and recommenced with high mutation every 1000 generations. The evolutionary algorithm was usually run for batches of 10000 generations.

7.4.1 Results: Fuzzy Control at a Distance

Results are shown in this section that are typical of the results obtained from evolutionary learning as described above across all initial configurations. Forty nine initial configuration at a constant radius from the ball are shown.

Figure 7.3 shows an overlay of the learnt configurations c077 to c125. The figure clearly shows the general form of the fuzzy controller from a ring of points from the ball. All robot paths terminate near to the destination point behind the ball facing forwards. To achieve a forward facing ball impact, the robot paths have in many instances used a cusp to change direction. A cusp requires a high change in momentum turn and should be minimised to cases when it is absolutely necessary and only near to the start or end position in a path. A cursory glance at the figure shows that some paths may contain more than one cusp in the path.

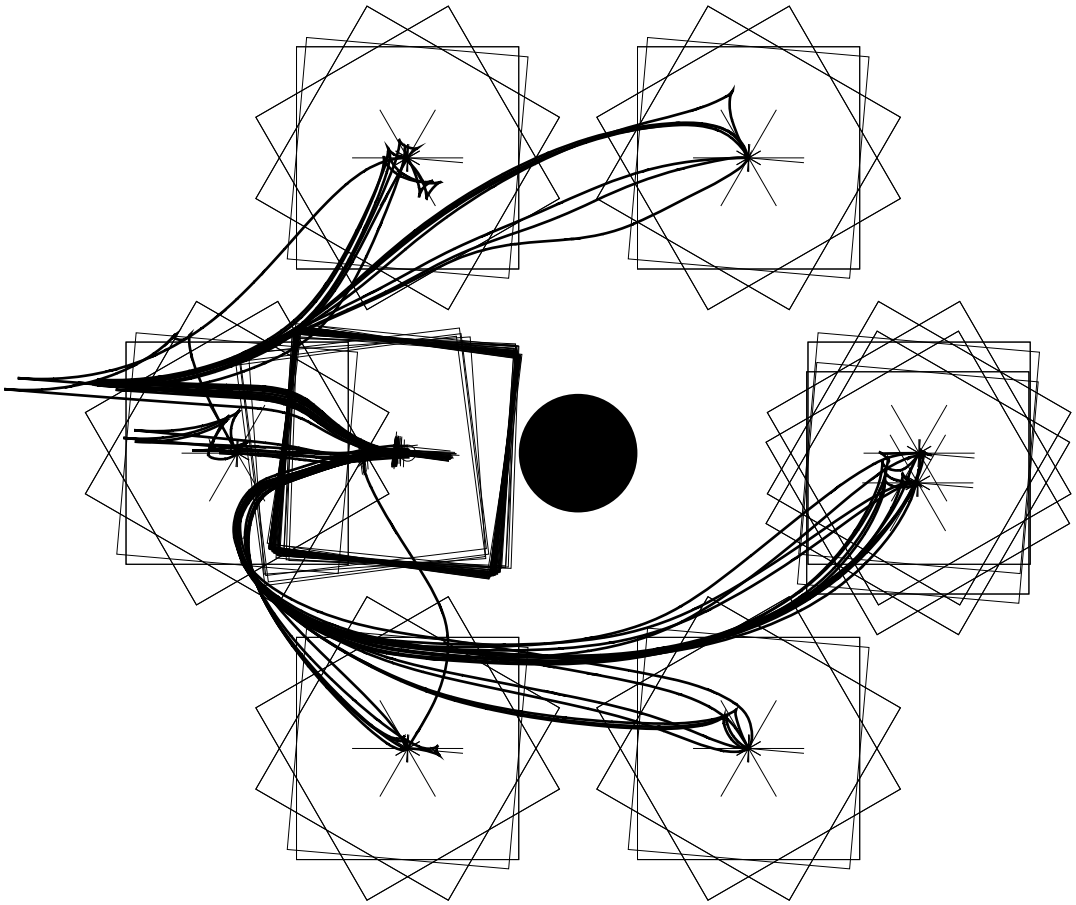


Figure 7.3: All Medium Distance Paths

Figure 7.7 shows in detail each paths from the learnt configurations c077 to c125. A close inspection of the paths reveals that some paths could be considered "good

paths" as they are smooth, minimise cusps, accelerate and decelerate towards the destination point. These paths are: c077, c079, c080, c081, c082, 083, c098, c104, and c105 to c125. A closer look at path from configuration c077 reveals a cusp near to the start and rapid acceleration to maximum speed around the ball, decelerating for the tight curve behind the ball, and then accelerating for the impact with the ball. The angle of the impact with the ball is of some concern, as the high restitution force between the golf ball and machined aluminium robot frame could have the ball moving in a direction outside the goal. The robot may not be able to recover this error and correct the angle on a subsequent impact. Loss of control of the ball due to impacts has been observed at tournaments.

Comparing the path from configuration c077 in Figure 7.7 to the velocity profiles of left wheel velocity (a) and right velocity (b) in Figure 7.8, very large acceleration components are seen. Looking back to the design of the controller, there was no control over the acceleration of the robot. Accelerations approaching infinity were allowed to be part of the solution. The programme generating this solution ran for twenty-eight continuous days on a Pentium 1.8GHz machine and was written in embedded assembly language for optimal processing speed. Modifying assembly code is a lengthy and arduous task taking approximately twelve times the man hours than C code and executes approximately six times faster (if using Visual C or True64 Unix cc, 12 times for Linux gcc). Modifying this code for another one month execution did not seem viable and other computing methods were sought. This programme is adequately demonstrate the computational power required to solve this type of problem.

The velocity profiles for configuration c077 to c125 are shown in Figures: 7.9, 7.10, 7.11, 7.12, 7.13, 7.14, and 7.15.

The statistics of the final position in each path is shown in Table 7.1 and Table 7.2.

These graphs are typical of the fuzzy control of the robot starting from initial positions at a "large" distance from the ball. Destination and final angle accuracy was excellent. Evolutionary learning was quite rapid, with acceptable solutions resulting in smooth paths to the destination started appearing within a few hundred generations. Further learning resulted in faster control to the destination.

7.4.2 Learning of Fuzzy Control Close to the Ball

Figure 7.4 shows the average of the minimums over eleven runs of the evolutionary learning process. A local minimum is found in 1 000 generations. The learning process stagnated until near to the end of 10 000 generations where another local minimum is found. A better fuzzy controller could not be found.

Figure 7.5 shows the average of averages over eleven runs of the evolutionary learning process. The "shaking" with the saw tooth mutation function can be seen in the averages. There is no discernable learning after the first 1 000 generations.

Learning of the fuzzy knowledge base close to the ball was more difficult. Figure 7.6 shows the paths obtained from learning the twenty eight initial configuration close and touching the ball.

Figure 7.16 shows each path in Figure 7.6 in detail.

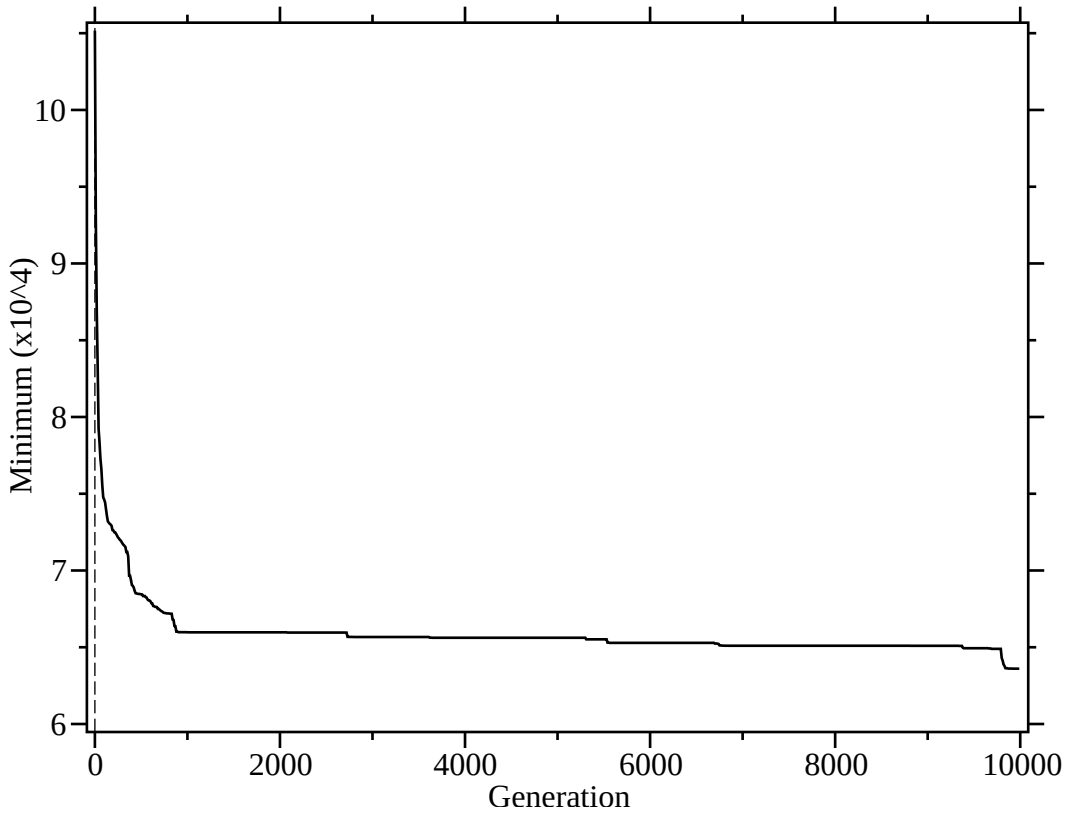


Figure 7.4: Evolutionary Statistics Average of Minimums over 11 runs

The velocity profiles for each path is shown in Figures 7.17, 7.18, 7.19 and 7.20.

The statistics of the final robot position are shown in Table 7.3.

All of the paths show chatter behind and close to the ball. The chatter is a product of the robustness of the evolutionary algorithm in obtaining the minimum objective value. Ignoring the chatter (assuming there is a time step close to the desired position), good paths are from initial configurations: c004, c005, c006, c007, c010, c011, c013, c022 and c023. The paths with looping behind the ball are considered as poor performing paths.

The starting initial configurations in which the robot was touching the ball were the most difficult to learn, for they were responsible for the majority of the

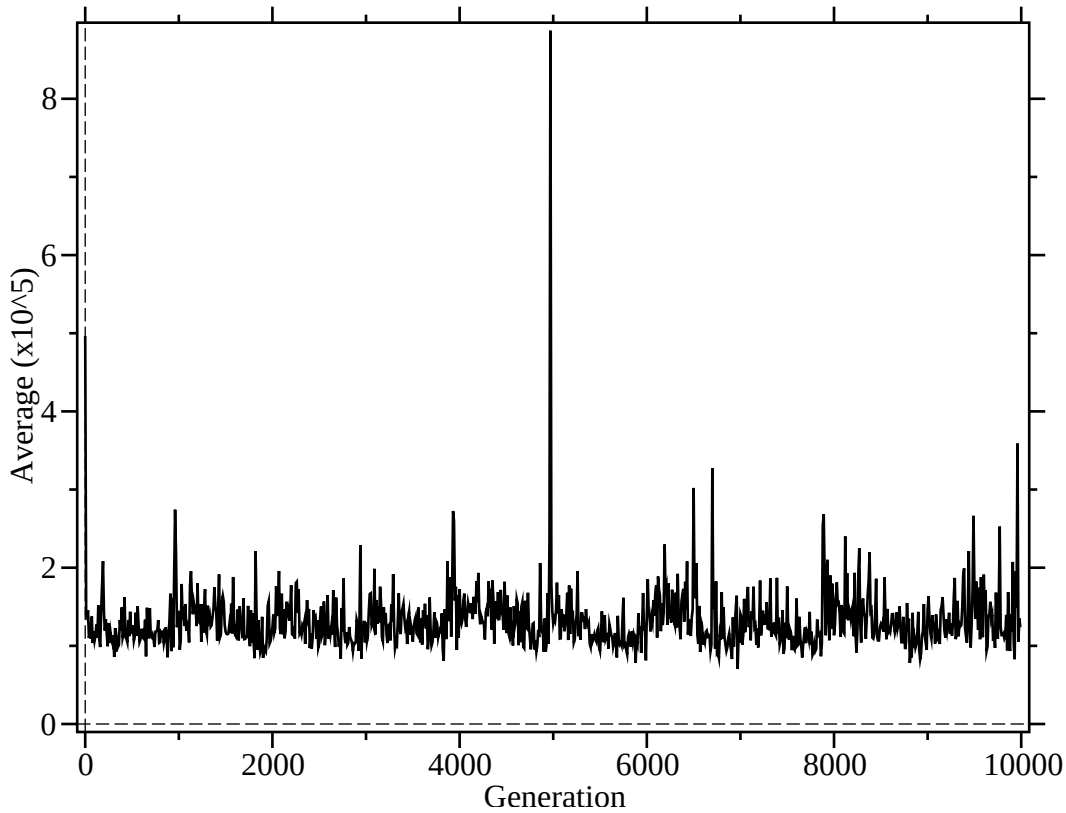


Figure 7.5: Evolutionary Statistics Average of Averages over 11 runs

penalty function evaluations in the fitness calculations for each individual of the evolutionary algorithm. The hardest initial configuration to learn was $c = 013$.

7.5 Comments

This chapter covered learning a small fuzzy system using minimal input data. It is clearly shown, that a 3I2O is fuzzy system is able to control the robot when at a distance and with minimal rule combinations. As the position of the robot entered areas where large combinations of rules acted, the fuzzy system would result in either a forward facing impact or a reverse facing impact with the ball. The obvious direction from here is two fold:

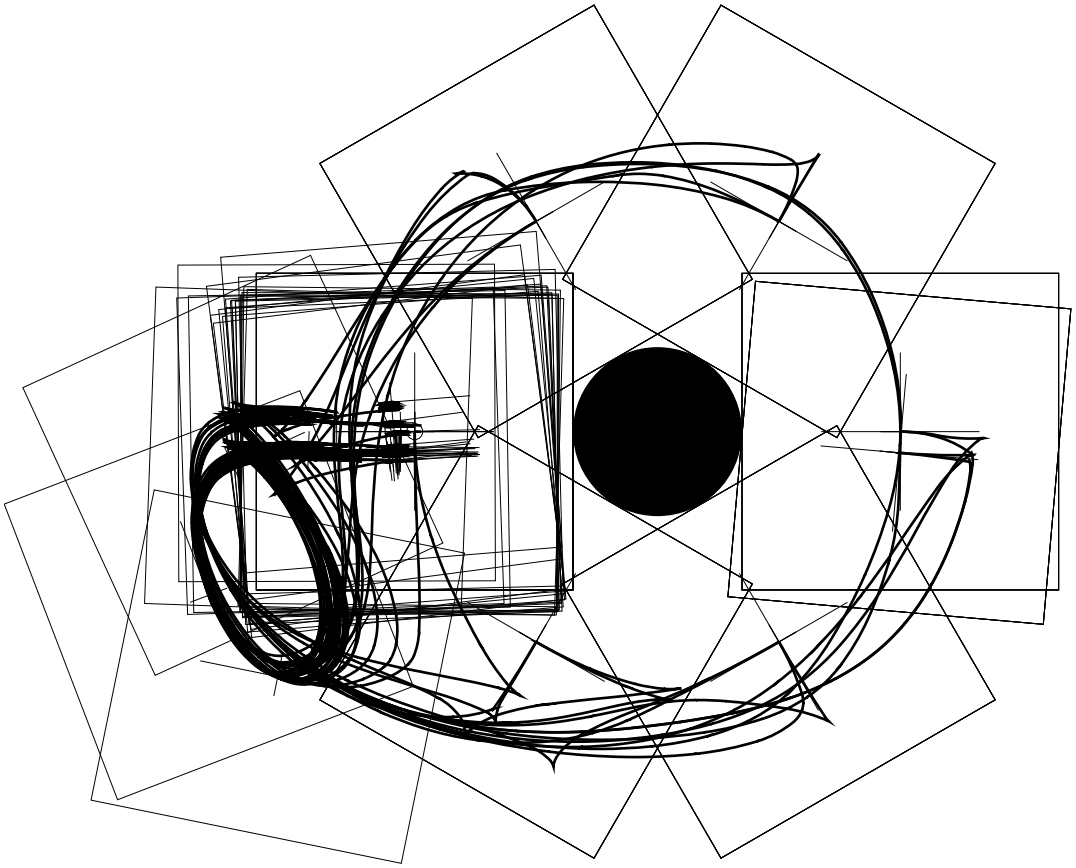


Figure 7.6: All Short Distance Paths

(i) Increase the number of variables input into the system, and/or

(ii) Increase the number of membership sets for each input variable.

A systematic approach of applying these ideas is to apply them individually in turn. The next chapter investigates an increase in input variables.

Table 7.1: Medium Path Final Robot Statistics

Config	x (mm)	y (mm)	ϕ_R (rad)	v_L (ms^{-1})	v_R (ms−1)	v_C (ms^{-1})	ω (rad s^{-1})
c077	683.97	650.25	6.1713	0.075	0.532	0.303	6.66
c078	683.83	650.31	6.1664	0.077	0.541	0.309	6.78
c079	683.72	649.78	6.1463	0.070	0.562	0.316	7.18
c080	683.63	649.93	6.1519	0.075	0.566	0.320	7.17
c081	683.73	650.54	6.1627	0.079	0.545	0.312	6.80
c082	684.22	650.41	6.1804	0.073	0.509	0.291	6.36
c083	683.74	650.57	6.1631	0.079	0.544	0.312	6.78
c084	683.79	650.92	6.1276	0.075	0.555	0.315	7.01
c085	685.34	650.56	6.1652	0.055	0.449	0.252	5.75
c086	684.49	650.42	6.1629	0.067	0.502	0.285	6.36
c087	684.59	650.70	6.1562	0.066	0.495	0.281	6.27
c088	684.29	650.74	6.1534	0.071	0.514	0.292	6.47
c089	683.86	650.87	6.1522	0.078	0.538	0.308	6.72
c090	683.94	650.53	6.1443	0.073	0.543	0.308	6.86
c091	683.68	650.30	6.1804	0.081	0.542	0.312	6.73
c092	684.54	650.00	6.2066	0.069	0.482	0.275	6.03
c093	684.44	650.44	6.2199	0.075	0.471	0.273	5.79
c094	684.07	649.24	6.2098	0.063	0.496	0.280	6.32
c095	684.95	650.12	6.2014	0.062	0.458	0.260	5.78
c096	684.31	650.86	6.1771	0.074	0.497	0.286	6.17
c097	684.03	650.25	6.1922	0.077	0.516	0.297	6.41
c098	687.29	649.92	6.2478	0.023	0.292	0.157	3.93
c099	686.25	651.00	6.2262	0.049	0.347	0.198	4.36
c100	674.04	649.15	0.0703	1.107	0.706	0.906	-5.85
c101	673.16	648.34	0.1254	1.188	0.679	0.933	-7.43
c102	672.41	650.50	0.1215	1.236	0.746	0.991	-7.16
c103	684.30	650.61	6.1511	0.070	0.517	0.293	6.54
c104	671.72	647.23	0.1307	1.298	0.824	1.061	-6.92
c105	686.19	650.29	6.2504	0.046	0.350	0.198	4.44
c106	684.59	649.94	6.1965	0.066	0.484	0.275	6.11
c107	684.16	652.79	6.1822	0.090	0.471	0.280	5.55
c108	683.73	649.92	6.1454	0.072	0.564	0.318	7.17
c109	684.01	649.92	6.1743	0.072	0.531	0.301	6.70
c110	683.61	649.96	6.1566	0.076	0.565	0.321	7.13
c111	684.23	649.81	6.1831	0.067	0.511	0.289	6.48
c112	685.73	651.17	6.2386	0.060	0.367	0.214	4.49
c113	684.38	651.39	6.1884	0.078	0.476	0.277	5.81
c114	684.85	650.76	6.2045	0.068	0.450	0.259	5.57
c115	683.76	651.16	6.1634	0.083	0.533	0.308	6.57
c116	684.23	651.12	6.1821	0.078	0.494	0.286	6.07
c117	685.16	651.17	6.2178	0.067	0.415	0.241	5.08

Table 7.2: Medium Path Final Robot Statistics (continued)

Config	x (mm)	y (mm)	ϕ_R (rad)	v_L (ms^{-1})	v_R (ms^{-1})	v_C (ms^{-1})	ω (rad s^{-1})
c118	685.02	651.26	6.2131	0.070	0.425	0.247	5.18
c119	684.48	650.15	6.1896	0.069	0.492	0.280	6.18
c120	685.05	650.06	6.2090	0.061	0.449	0.255	5.67
c121	683.62	650.31	6.1588	0.079	0.558	0.319	7.00
c122	684.16	650.02	6.1782	0.071	0.520	0.296	6.55
c123	683.58	650.53	6.1510	0.080	0.561	0.320	7.03
c124	683.53	650.76	6.1534	0.082	0.559	0.321	6.96
c125	684.96	650.16	6.2063	0.063	0.454	0.258	5.71

Table 7.3: Short Paths Robot Statistics

Config	x	y	ϕ_R	v_L	v_R	v_C	ω
	(mm)	(mm)	(rad)	(ms^{-1})	(ms^{-1})	(ms^{-1})	(rad s^{-1})
c000	654.051	588.103	2.9407	1.211	-0.638	0.287	-26.98
c001	683.137	642.418	0.0794	0.136	-0.045	0.046	-2.64
c002	681.842	644.842	0.1284	0.181	0.001	0.091	-2.62
c003	684.734	643.977	6.2755	-0.425	0.112	-0.157	7.84
c004	684.664	644.455	6.2831	-0.442	0.097	-0.173	7.87
c005	684.021	644.888	0.0418	0.140	-0.049	0.045	-2.76
c006	683.640	644.399	0.0750	0.121	-0.063	0.029	-2.67
c007	682.750	657.459	0.0816	0.101	-0.029	0.036	-1.90
c008	672.137	644.972	0.0169	1.133	0.901	1.017	-3.39
c009	685.157	644.791	6.2539	-0.383	0.126	-0.129	7.42
c010	682.320	642.545	0.1217	0.137	-0.036	0.050	-2.53
c011	685.130	650.008	0.0248	0.156	-0.028	0.064	-2.68
c012	688.650	650.000	3.1416	-0.035	-0.024	-0.030	0.17
c013	683.742	646.612	0.0756	0.132	-0.052	0.040	-2.67
c014	688.650	650.000	0.0000	0.189	0.000	0.094	-2.76
c015	669.871	645.164	0.0344	1.281	1.030	1.155	-3.67
c016	684.227	642.928	0.0422	0.111	-0.078	0.016	-2.75
c017	683.947	644.994	0.0476	0.139	-0.049	0.045	-2.74
c018	682.860	644.966	0.0984	0.150	-0.031	0.059	-2.65
c019	642.704	641.482	0.4311	1.096	-0.085	0.505	-17.24
c020	661.805	645.121	6.2472	-0.798	-0.510	-0.654	4.20
c021	685.035	646.190	6.2723	-0.423	0.087	-0.168	7.44
c022	668.928	652.198	0.0028	1.426	1.150	1.288	-4.03
c023	680.917	652.291	0.1310	0.263	0.086	0.175	-2.59
c024	636.604	608.669	1.9370	1.258	0.166	0.712	-15.94
c025	684.243	643.855	0.0219	0.141	-0.051	0.045	-2.81
c026	684.246	644.164	0.0235	0.141	-0.051	0.045	-2.80
c027	684.795	644.947	6.2783	-0.430	0.098	-0.166	7.71

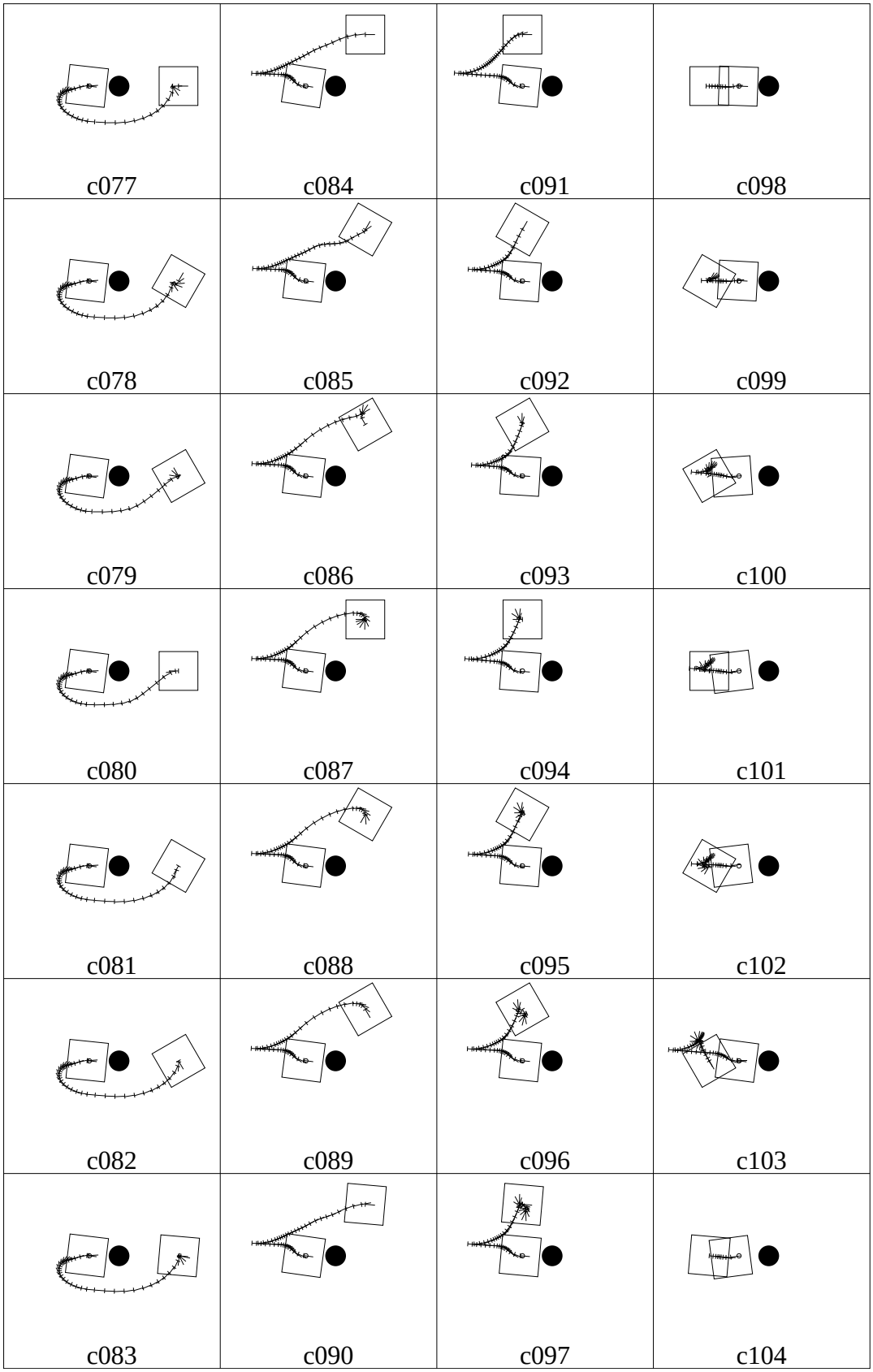


Figure 7.7: Medium Distance Paths: c077–c104

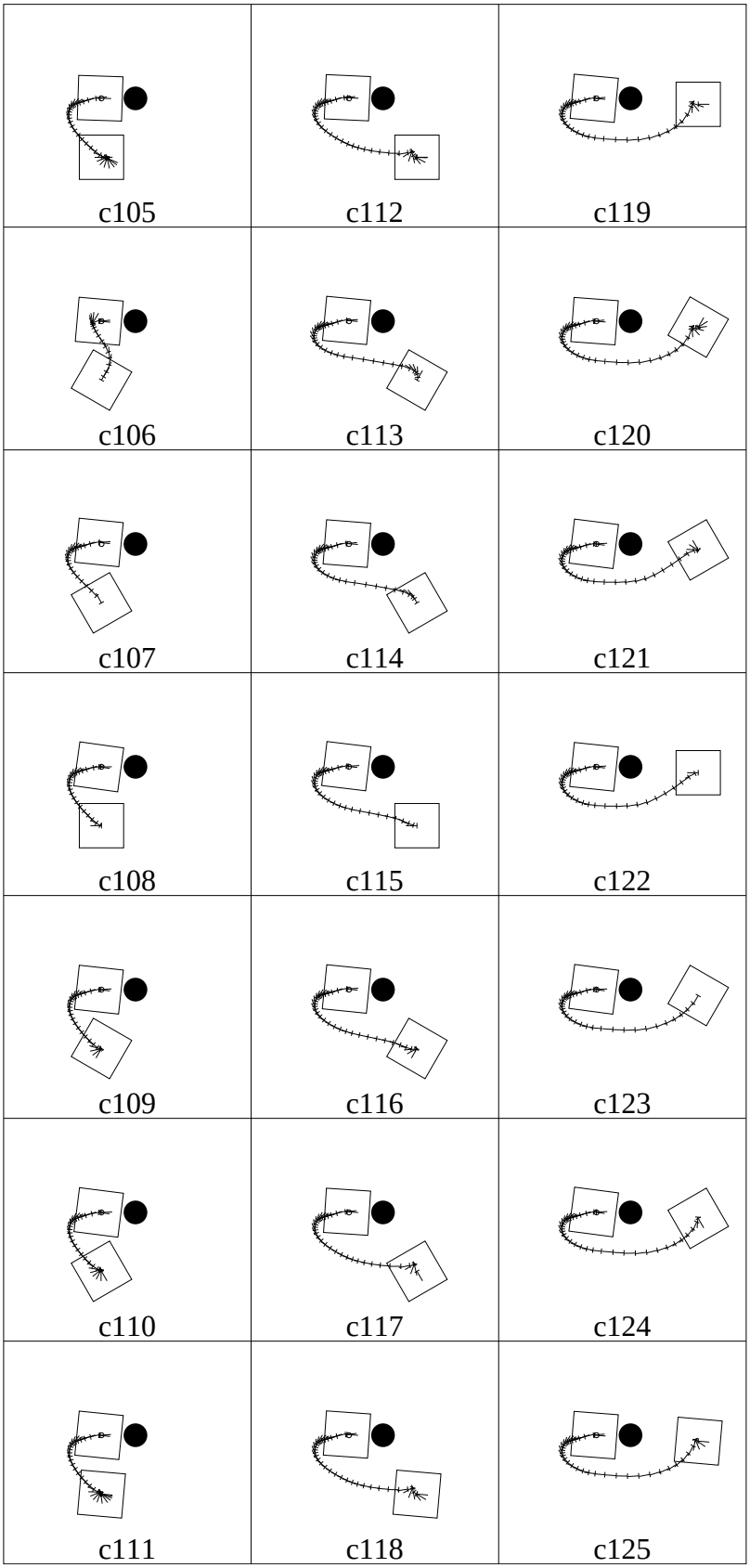


Figure 7.8: Medium Distance Paths: c105–c125

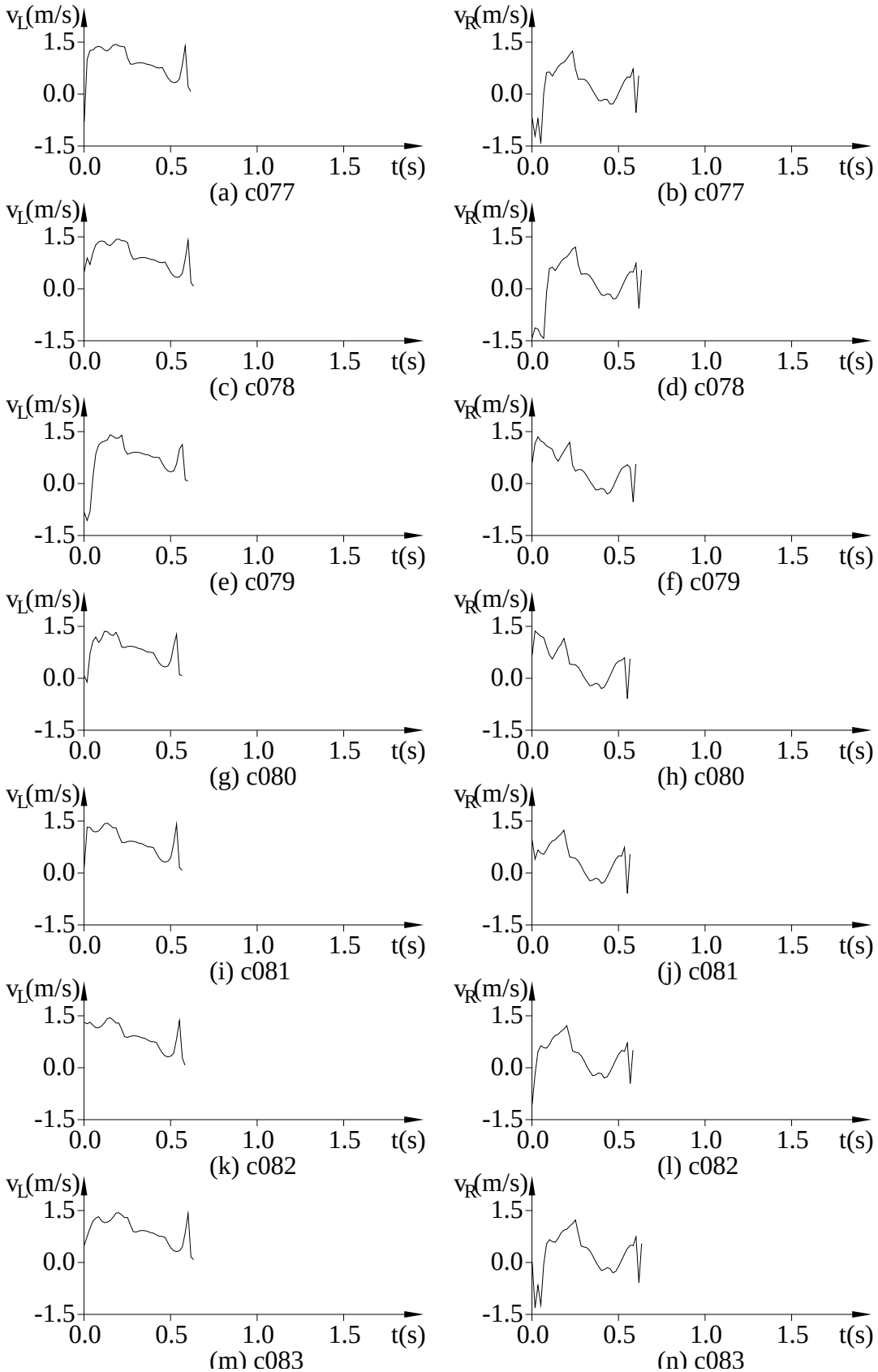


Figure 7.9: Medium Distance Velocity Profiles: c077–c083

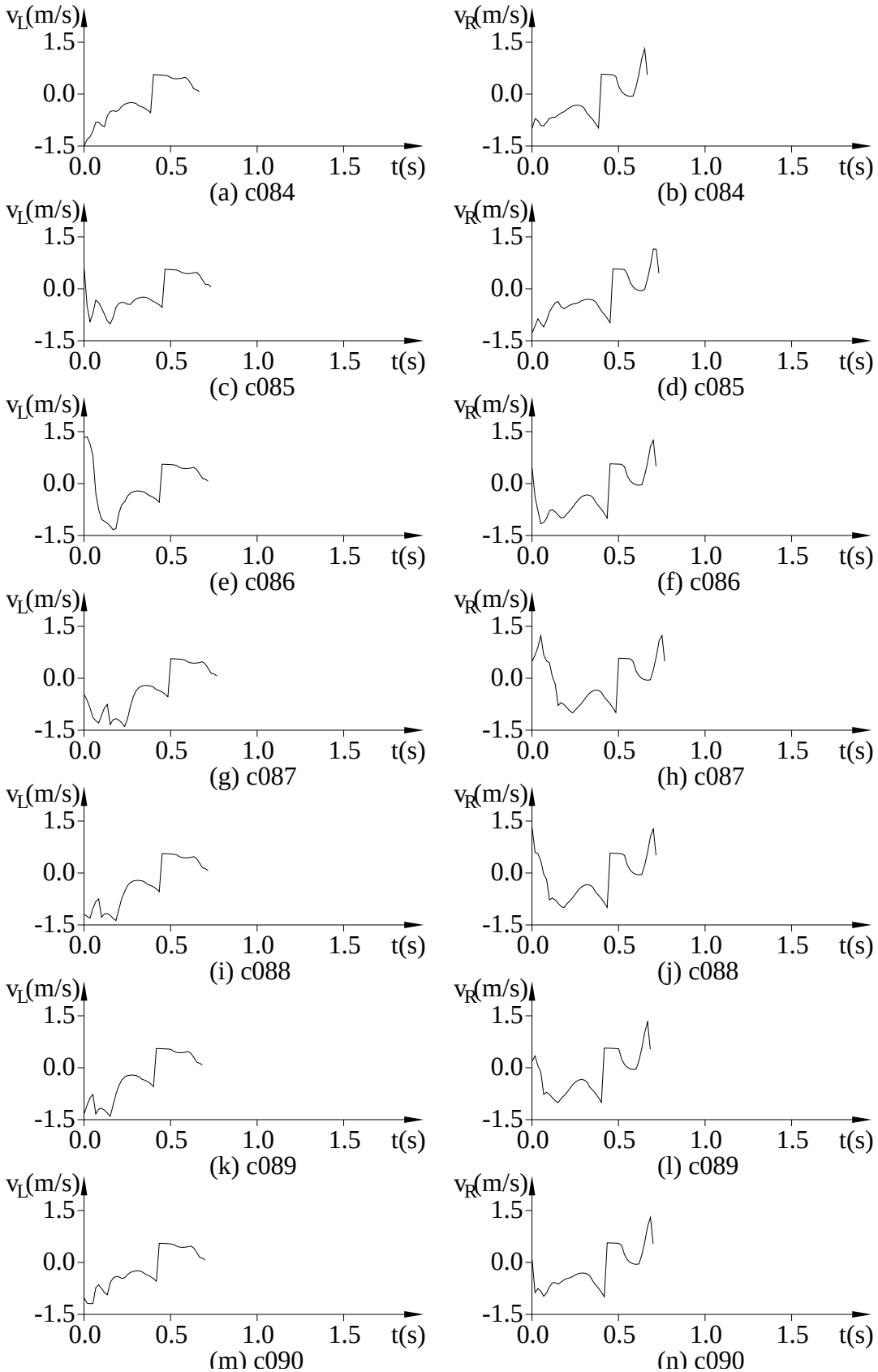


Figure 7.10: Medium Distance Velocity Profiles: c084–c090

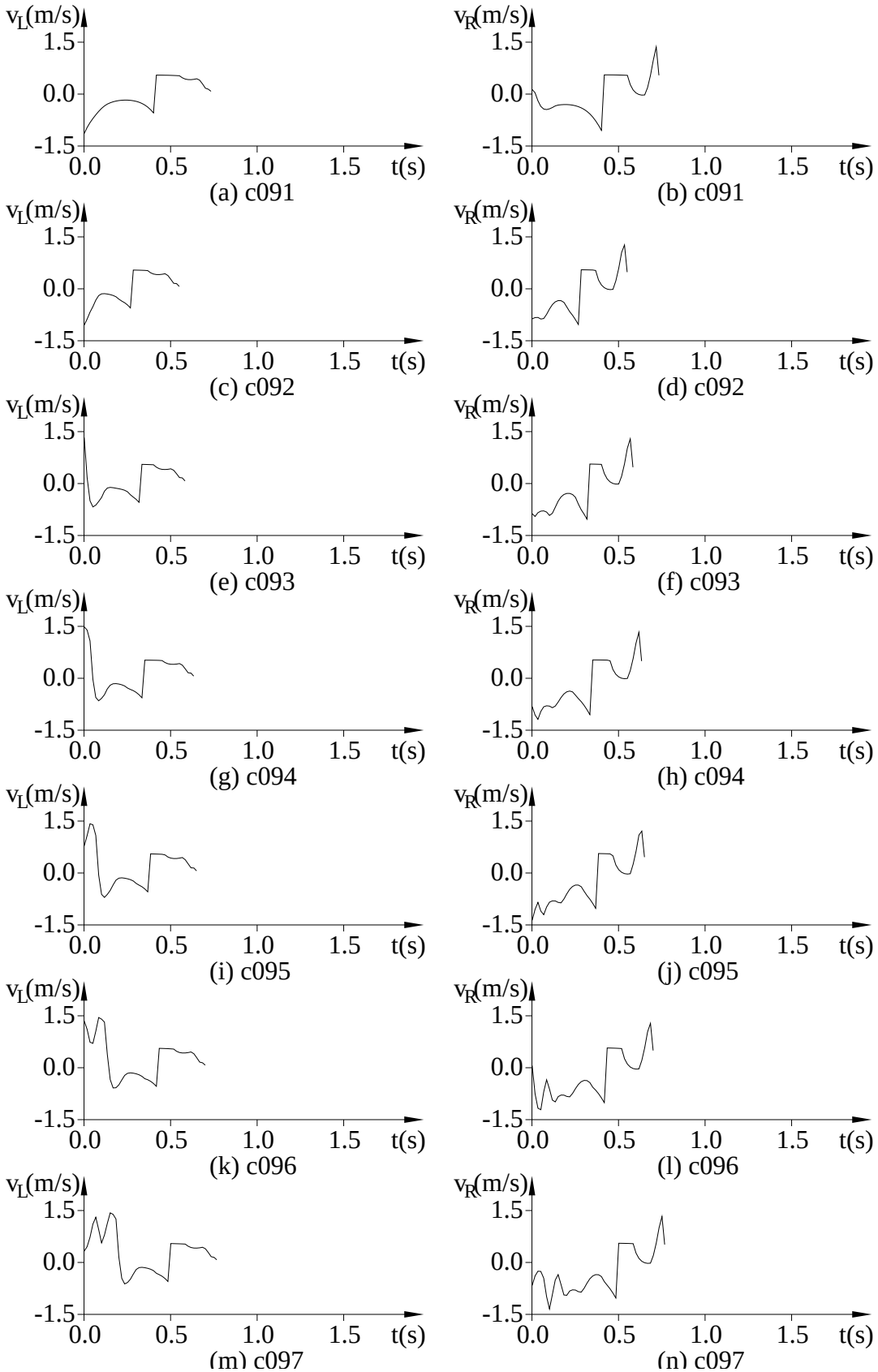


Figure 7.11: Medium Distance Velocity Profiles: c091–c097

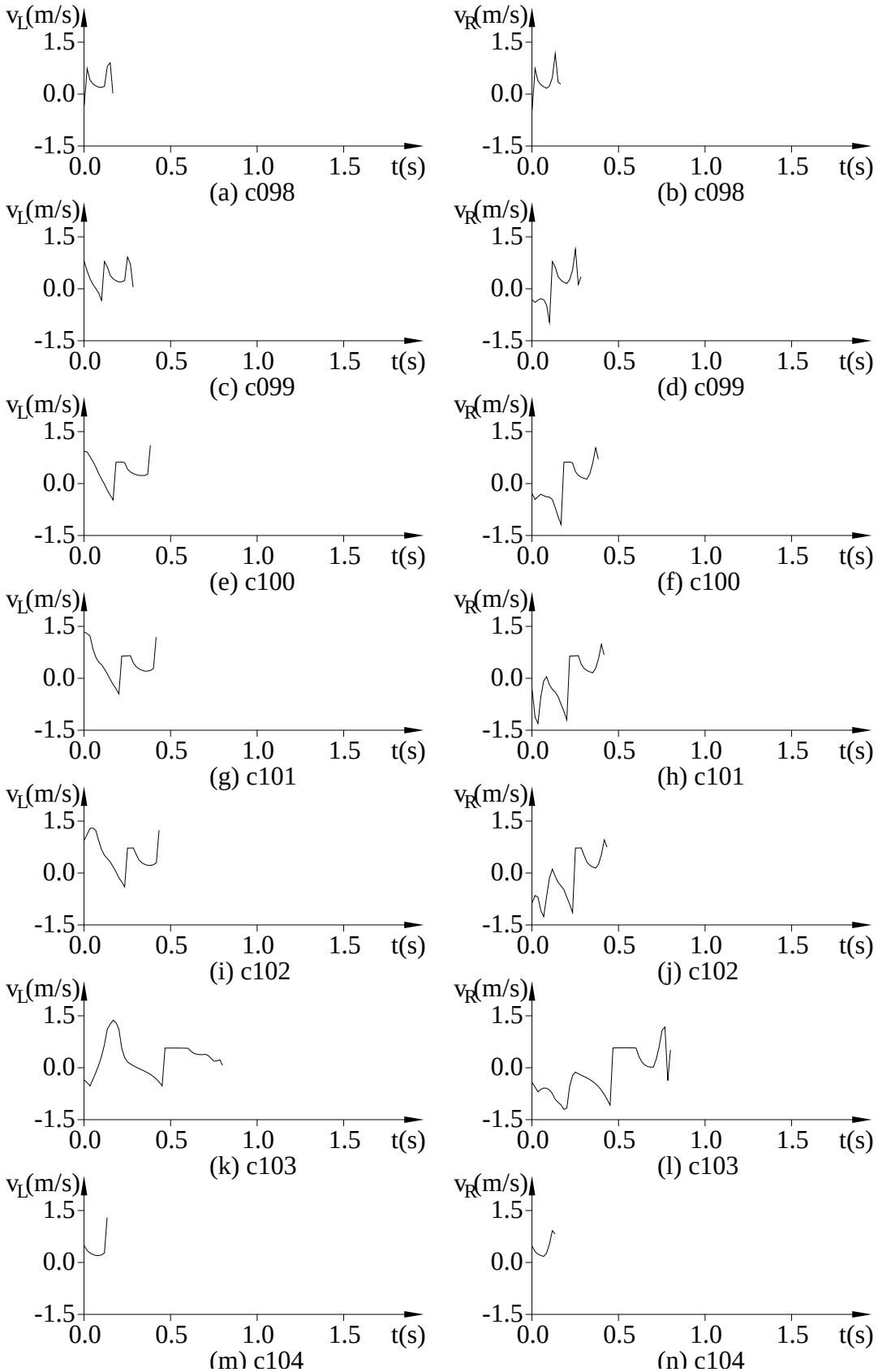


Figure 7.12: Medium Distance Velocity Profiles: c098–c104

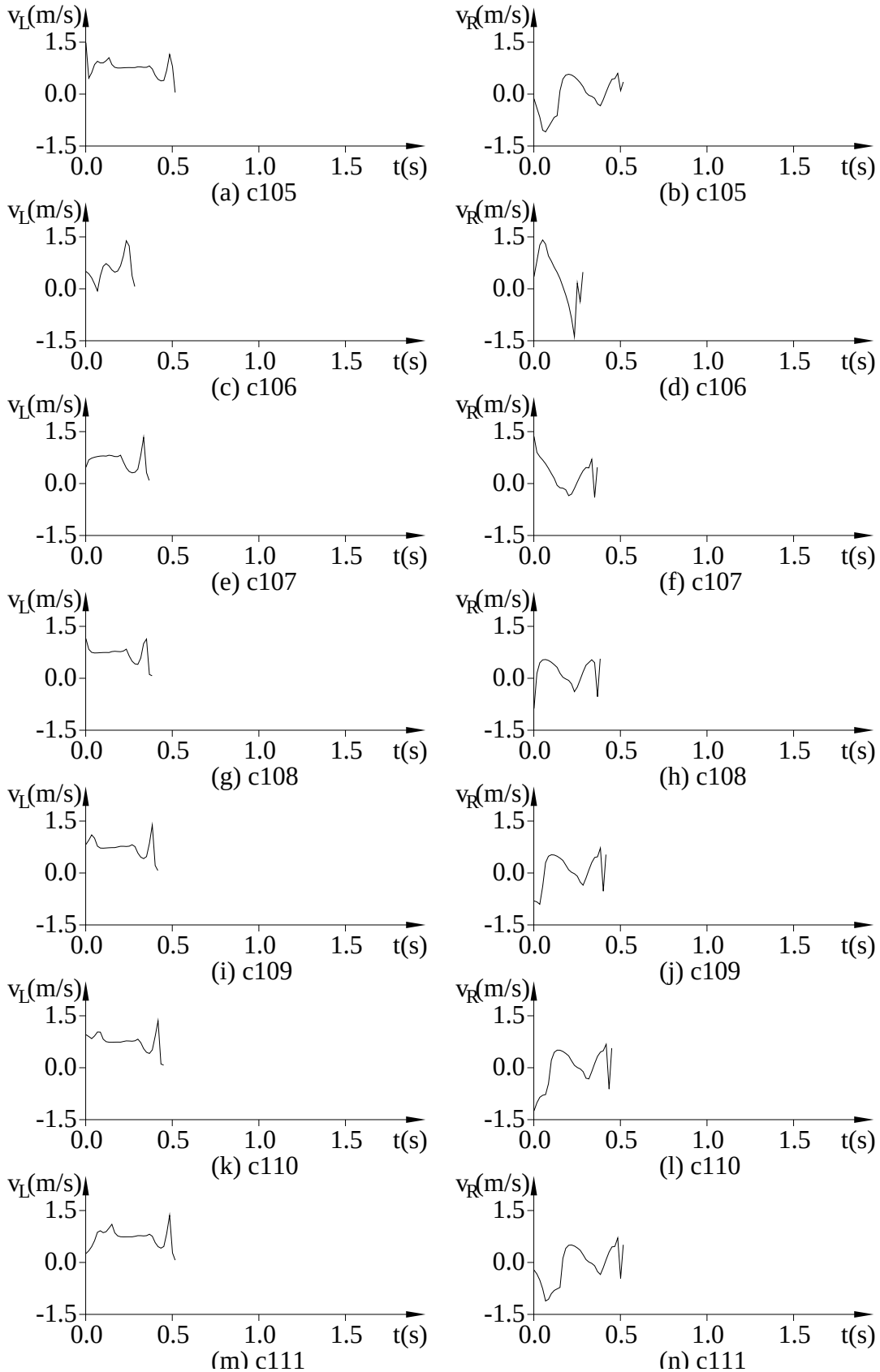


Figure 7.13: Medium Distance Velocity Profiles: c105–c111

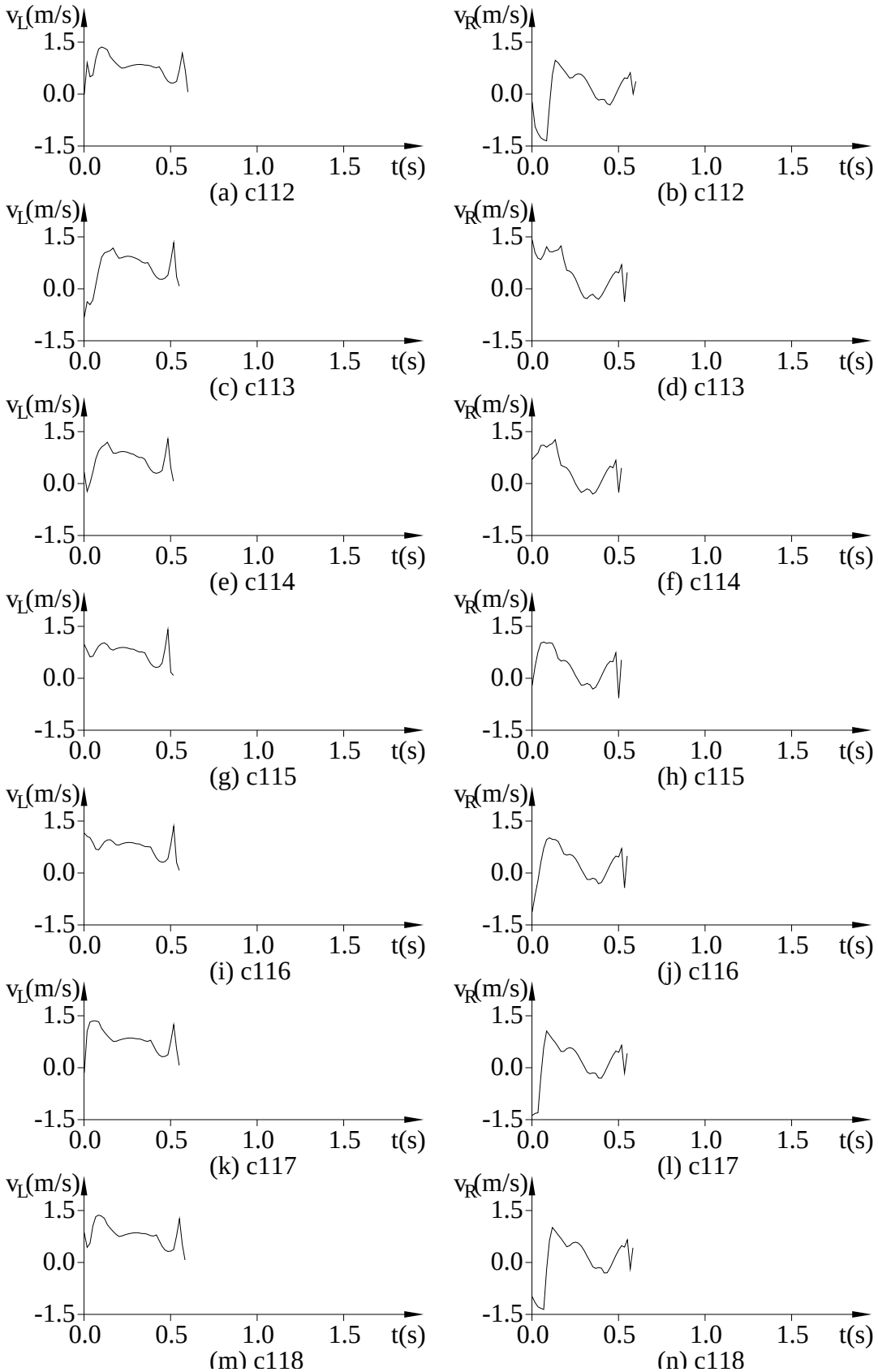


Figure 7.14: Medium Distance Velocity Profiles: c112–c118

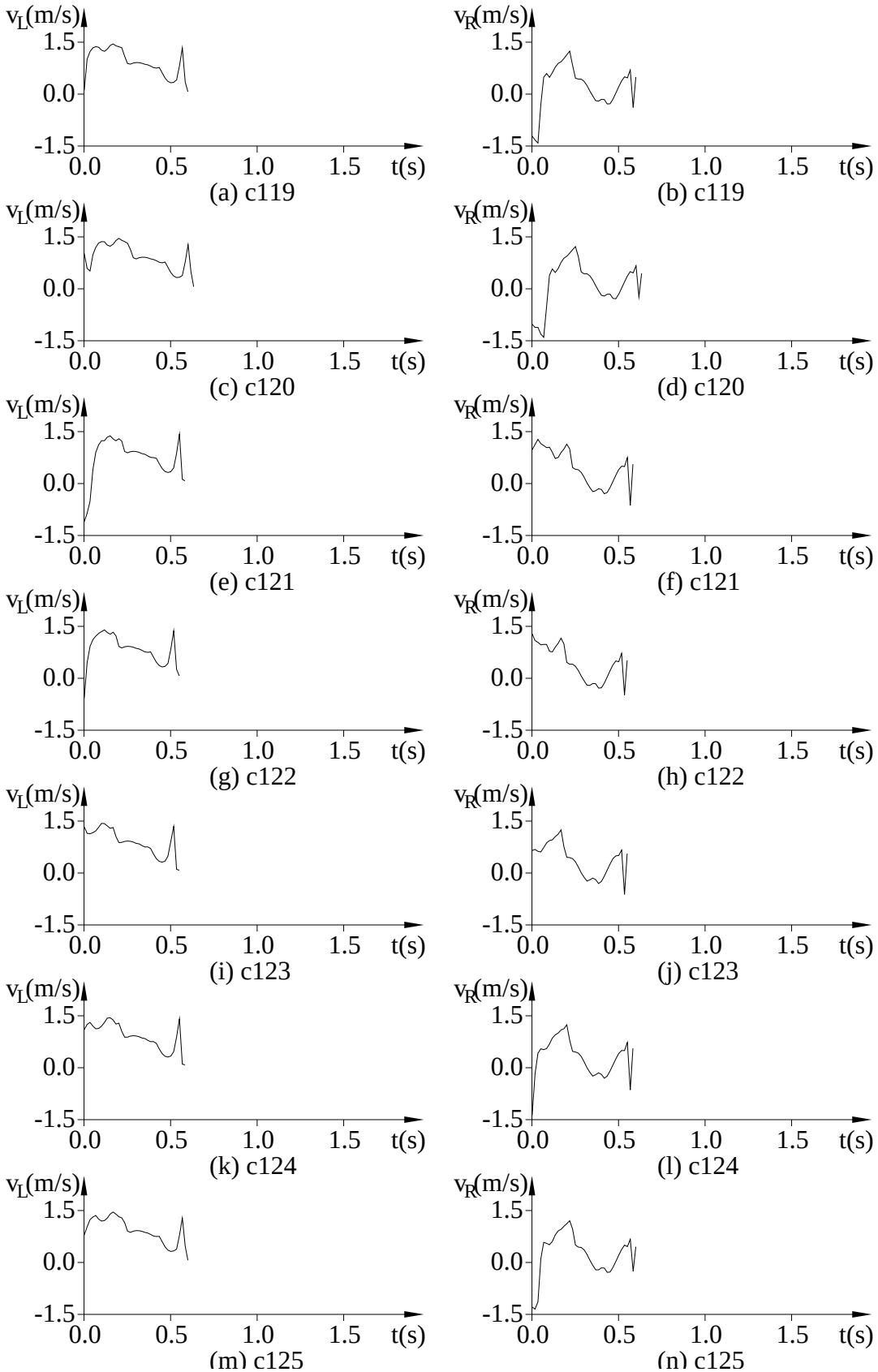


Figure 7.15: Medium Distance Velocity Profiles: c119–c125

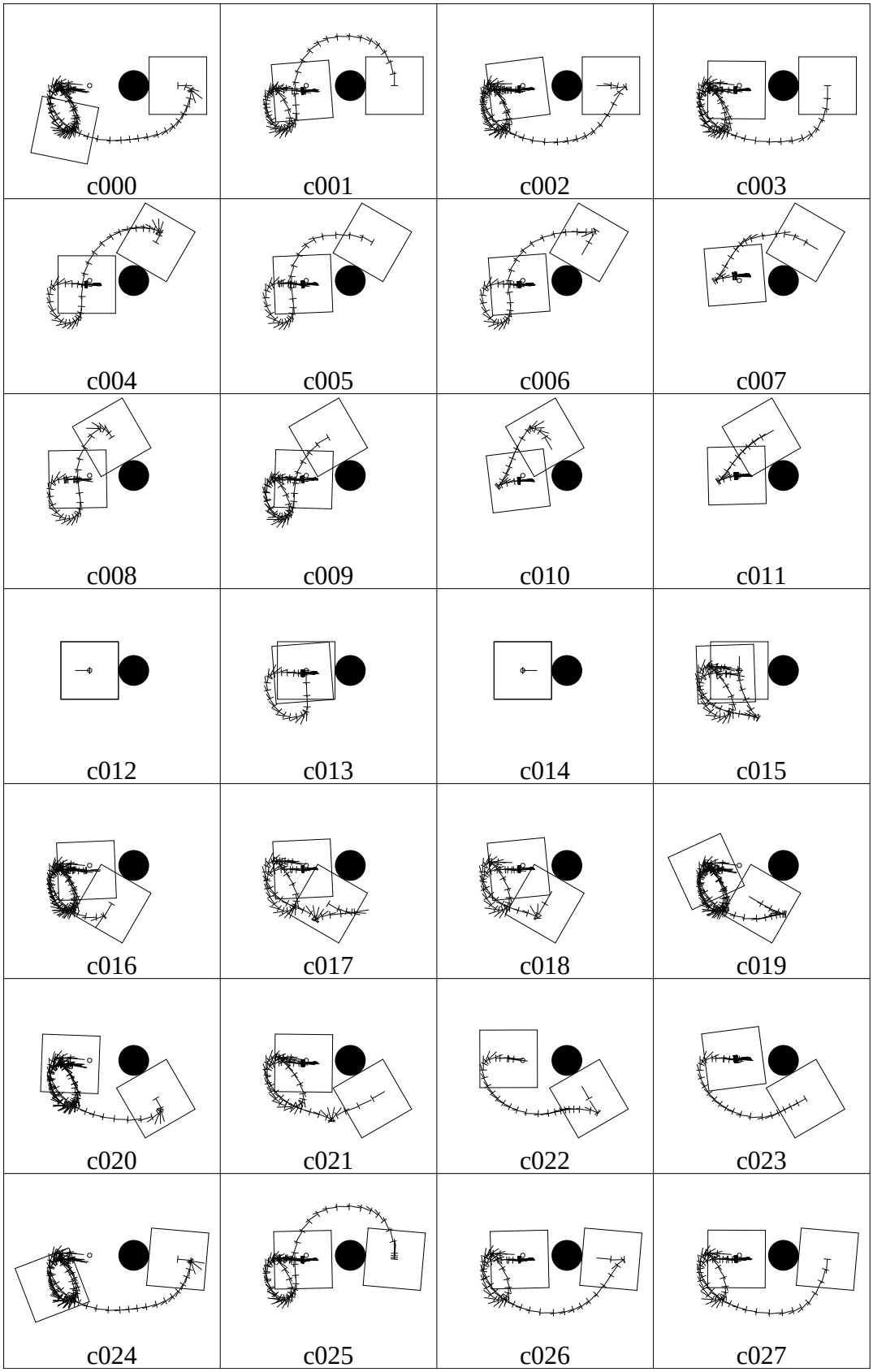


Figure 7.16: Individual short distance paths

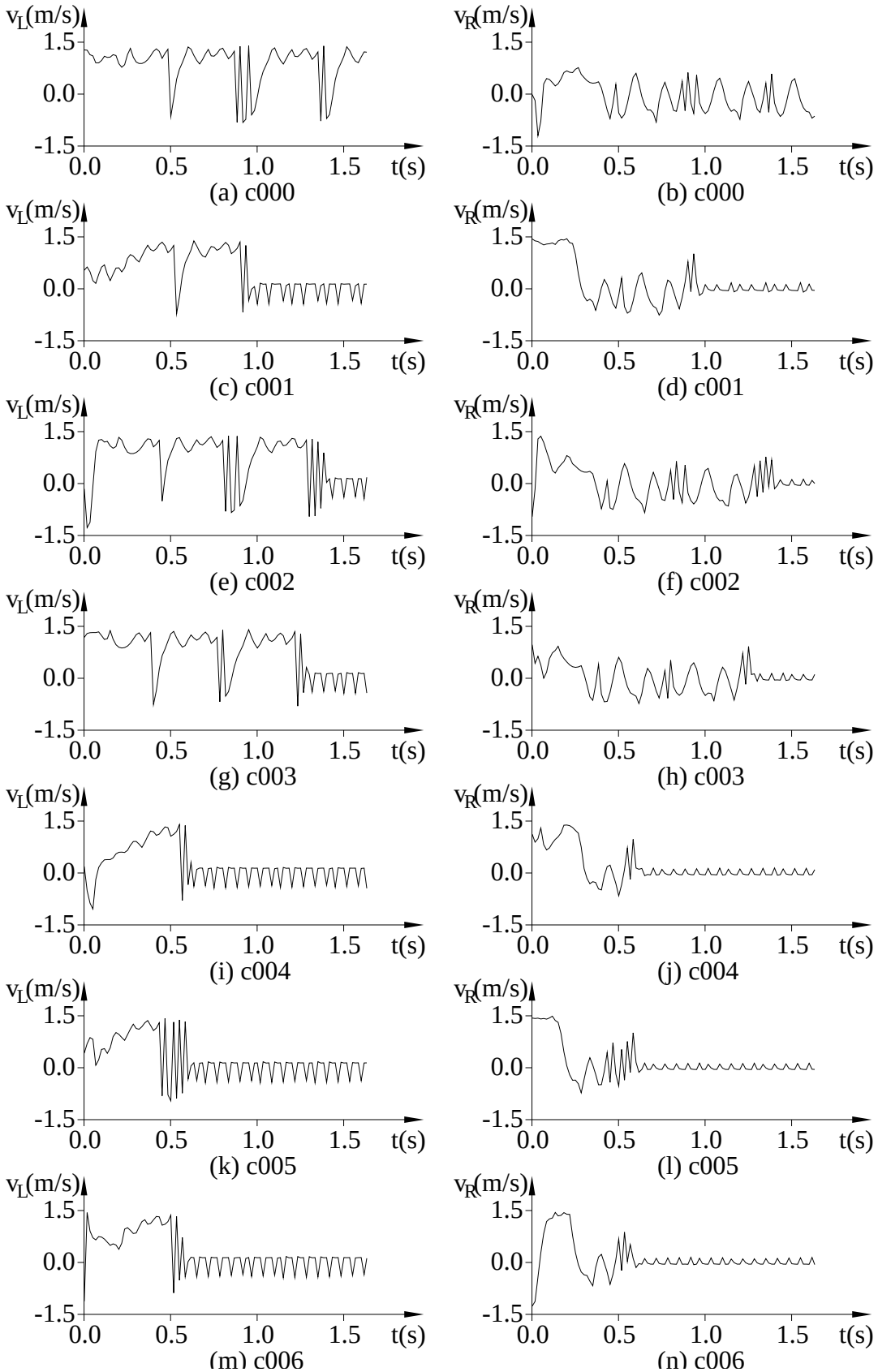


Figure 7.17: Short Distance Velocity Profiles: c000–c006

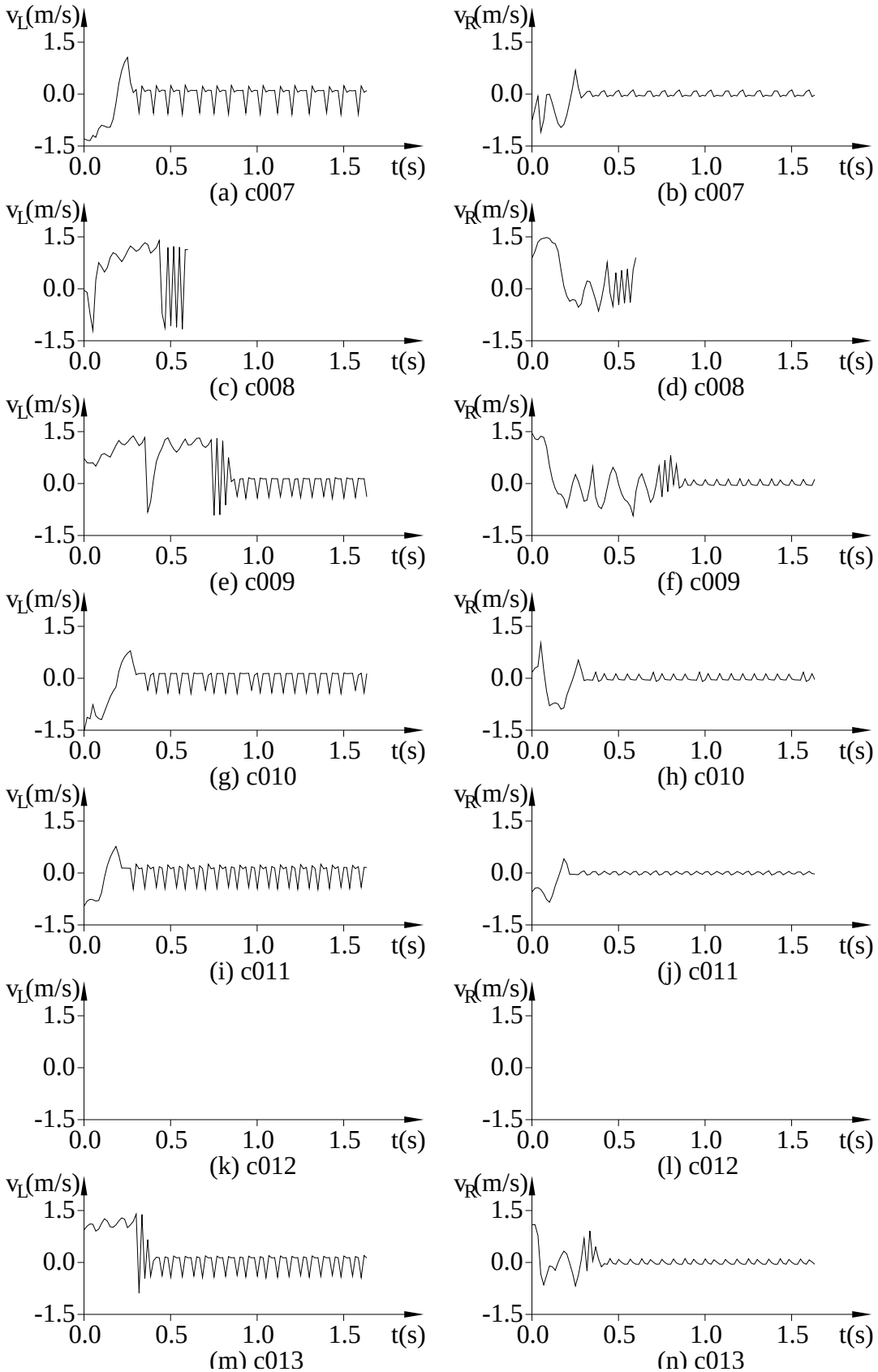


Figure 7.18: Short Distance Velocity Profiles: c007–c013

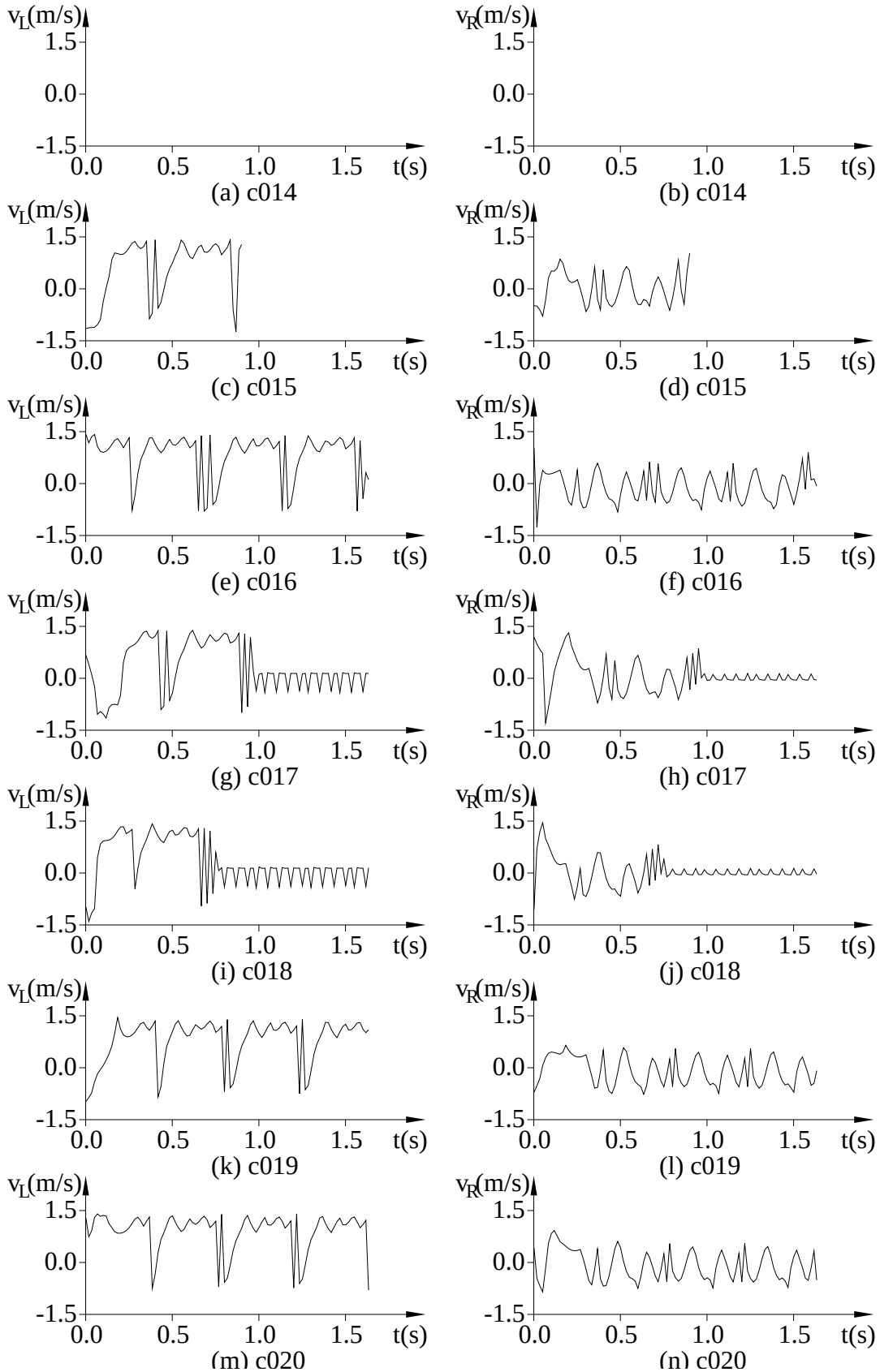


Figure 7.19: Short Distance Velocity Profiles: c014–c020

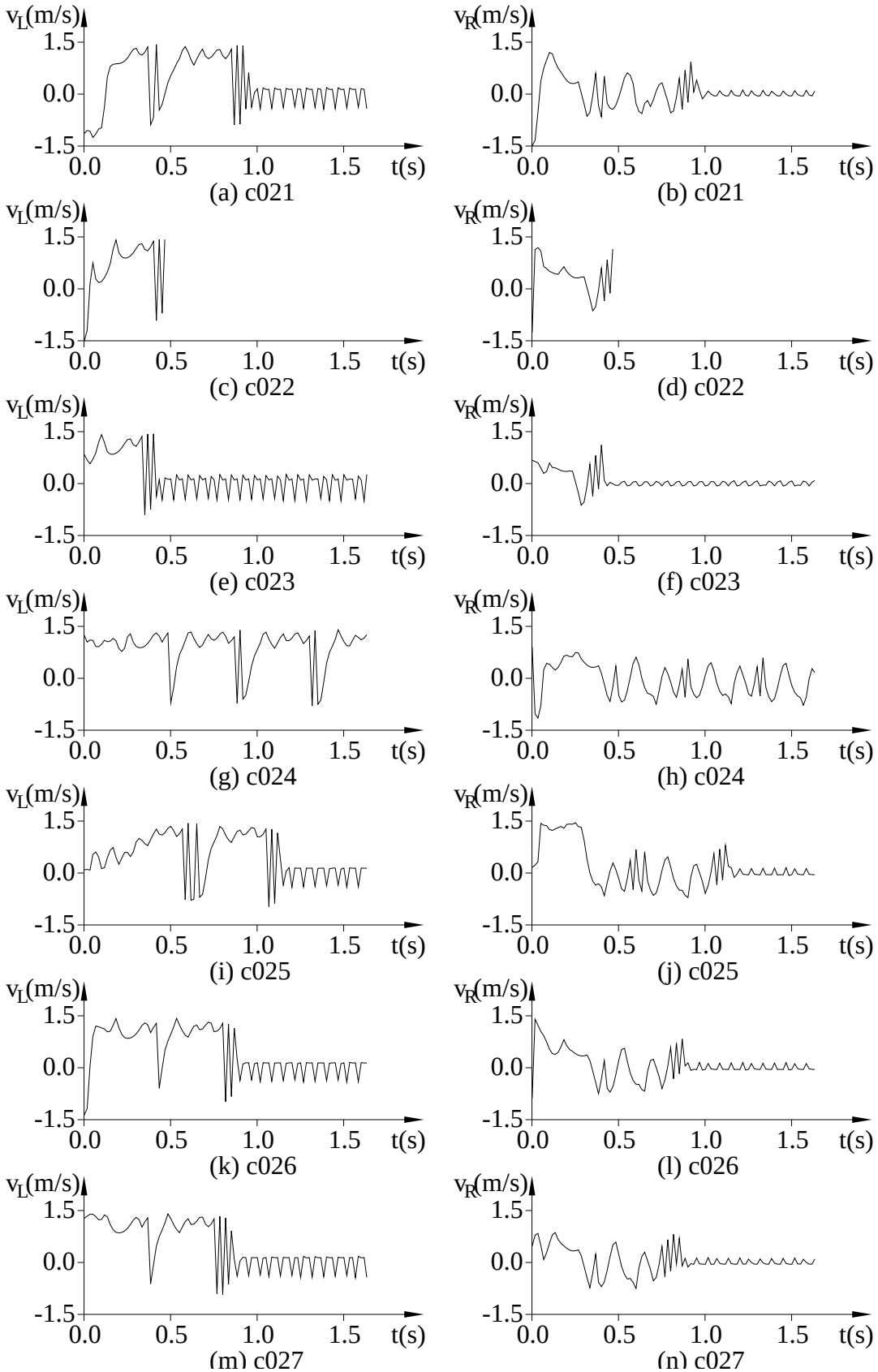


Figure 7.20: Short Distance Velocity Profiles: c021–c027

Chapter 8

Five Input Two Output (5I2O)

Substantial parts of this chapter were published in the proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI'2002) in Florida [28]. Development of a five input, two output fuzzy controller is considered. The results presented in this chapter are through simulation as the resulting controller was incomplete.

8.1 Introduction

Chapter 7 used just three variables to control the robot to a position near to the ball with non-zero velocity. Acceptable single paths were found, but had difficulty in finding a general fuzzy controller from the grid of initial configurations. No general set of parameters were found for the objective function in evolving a fuzzy knowledge base. The solutions from multiple starting initial configurations tended to be dominated by either forward facing or reverse facing robot and only half of the initial configurations made it to the destination point. The reason for these difficulties are two fold:

(i) Insufficient number of variables input to the fuzzy system, and

(ii) Granularity of the fuzzy membership sets.

This chapter will extend the three input variable fuzzy system to a five input fuzzy system. Granularity of the fuzzy system is addressed in Chapter 9.

The robot soccer system remains as described in the previous Chapter. With the three input fuzzy input system using d^2, θ and ϕ increased to five input by including the instantaneous left $v_L(t)$ and right $v_R(t)$ wheel velocities of the robot, as shown in Equations 8.1. Figure 8.1 shows the five input two output system in diagram from.

$$\begin{aligned} x_1 &= d^2 \\ x_2 &= \theta \\ x_3 &= \phi \\ x_4 &= v_L(t) \\ x_5 &= v_R(t) \end{aligned} \tag{8.1}$$

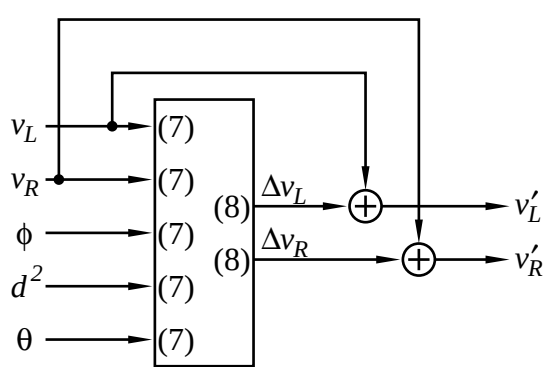


Figure 8.1: Five Input Two Output Model

The system and kinematics are the same as used in Chapter 7. The extension is in the fuzzy rule base and consequently the structure of the individuals in the evolutionary algorithm.

8.2 Fuzzy Control System Design

The inputs into the fuzzy knowledge base are now variables x_1, x_2 and x_3 defined previously and the two wheel velocities $x_4 = v_L$ and $x_5 = v_R$. Figure 8.2 shows the fuzzy input sets used for each variable. There are seven linguistic membership sets defined for each variable. For both angles θ and ϕ: **VS** is Very Small, **S** is Small, **SM** is Small Medium, **M** is Medium, **ML** is Medium Large, **L** is Large and **VL** is Very Large. Distance squared x_1: **VC** is Very Close, **C** is Close, **CN** is Close Near, **N** is Near, **NF** is Near Far, **F** is Far and **VF** is Very Far. For left wheel velocity v_L and right wheel velocity v_R: **FR** is Fast Reverse, **MR** is Medium Reverse, **SL** is Slow Reverse, **S** is Stationary, **SF** is Slow Forward, **MF** is Medium Forward and **FF** is Fast Forward.

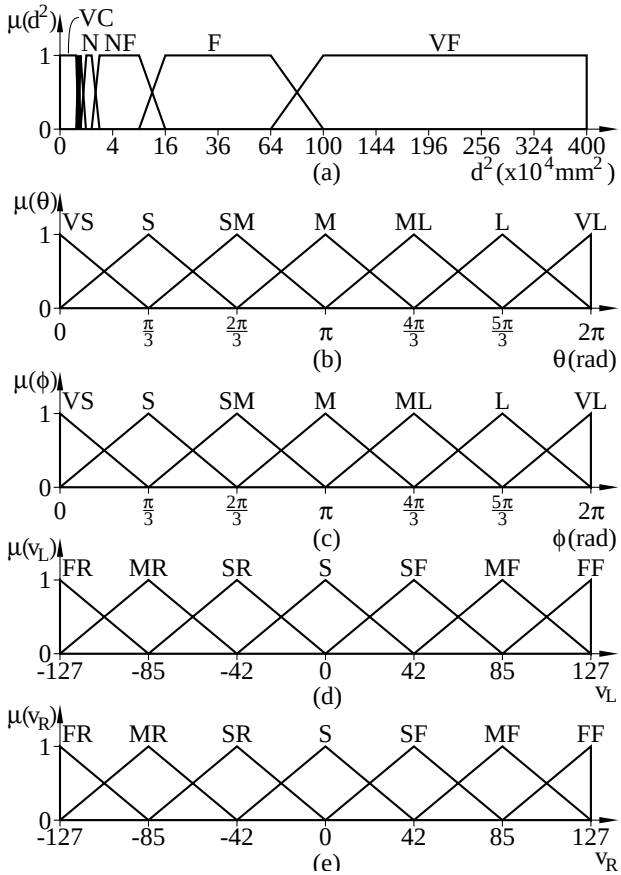


Figure 8.2: Fuzzy Input Sets

The new wheel velocities are calculated by adding the change in velocity output from the fuzzy controller $y_1 = \Delta v_L$ as in Equation 8.2, and $y_2 = \Delta v_R$ as in Equation 8.3.

$$v_L(t+1) = v_L(t) + \Delta v_L(t) \tag{8.2}$$

$$v_R(t+1) = v_R(t) + \Delta v_R(t) \tag{8.3}$$

Eight membership sets for each output variable, y_1 and y_2, are defined as: **VFR** is Very Fast Reverse, **FR** is Fast Reverse, **MR** is Medium Reverse, **SR** is Slow Reverse, **SF** Slow Forward, **MF** is Medium Forward, **FF** is Fast Forward, and **VFF** is Very Fast Forward. Equation 8.4 describes the position of each fuzzy output centre.

$$\overline{y}_k^\ell = -28 + 8k \quad \text{for} \quad k = 0, \cdots, 7 \tag{8.4}$$

In all there are now $7^5 = 16807$ rules in a complete fuzzy knowledge base for this system. In general, the ℓth fuzzy rule has the form:

If (x_1 is A_1^ℓ and x_2 is A_2^ℓ and x_3 is A_3^ℓ and x_4 is A_4^ℓ and x_5 is A_5^ℓ)
Then (y_1 is B_1^ℓ and y_2 is B_2^ℓ).

where $A_k^\ell, k = 1, \cdots, 5$ are normalised fuzzy sets for input variables $x_k, k = 1, \cdots, 5$, and where $B_m^\ell, m = 1, 2$ are normalised fuzzy sets for output variables $y_m, m = 1, 2$.

Given a fuzzy rule base with M rules, a fuzzy controller as given in Equation 7.1 uses a singleton fuzzifier, Mamdani product inference engine and centre average defuzzifier to determine output variables.

These values, 33614 of them, are typically unknown and require determination in establishing valid output for controls to each wheel of the robot. As there is no a priori knowledge about the system control, an evolutionary algorithm (EA) [27] is used to search for an acceptable solution.

A simple method of implementing Equation 7.1 is to use nested loops for the summation and product terms. This simple encoding requires the loops to consider antecedents and consequents of 16807 rules.

A special feature of overlapping sets using the Mamdani product inference engine is that a minimum of one and a maximum of two membership sets for each input variable will fire. A vertical slice through any variable membership in Figure 8.2 illustrates this property.

A maximum of $2^5 = 32$ out of the $7^5 = 16807$ will fire for any input into the fuzzy controller. With a sparse access to the rule base, it makes sense to access the rule base by developing pointers to the rule being used.

The number of membership sets was set at seven for each input variable to make the calculation of the pointer to the rule an easy radix calculation. It is not necessary to use the same number of memberships per variable, it was done simply to reduce errors made in programming and ease debugging.

Each input variable is tried on the input membership sets to find the one or two memberships that fire. The set identification is stored in an array with the membership value(s). Each array is terminated by a pointer value of -1. Five input variables require five nested loops to calculate the pointer reference using the antecedents in accordance with Equation 7.1. The pointer refers to the

consequent $\overline{y}_k^\ell$ in the string for inclusion into Equation 7.1.

8.3 Evolutionary Learning

The objective here is to learn a rule base for the control of this system. The first problem is how to formulate the knowledge base as a string in the population.

Each output fuzzy set is represented by an integer in the interval $[0, 7]$ corresponding to the fuzzy output sets {**VFR**, **FR**, **MR**, **SR**, **SF**, **MF**, **FF**, **VFF**}. Equation 8.5 shows the construction of a potential unique knowledge base solution for an individual string $\underset{\sim}{s}$ containing $2M = 33614$ integers, where $s_j, j = 1, 2$ is an integer in the interval $[0, 7]$.

$$\underset{\sim}{s} = \{s_1^1, s_2^1, \cdots, s_1^k, s_2^k, \cdots, s_1^M, s_2^M, \} \tag{8.5}$$

The population at generation t, $P(t) = \{\underset{\sim}{s}^n \ : \ n = 1, \cdots, N\}$, where $N = 2000$ is the number of individuals in the population. The population at the next generation $P(t+1)$ was built using a full replacement policy, tournament selection with size $n_T = 3$, and one point crossover with probability $p_c = 0.6$. Elitism was used, with the 10 best individuals carried from population $P(t)$ to population $P(t+1)$. An incremental mutation operator with probability $p_m = 0.01$, replaced the binary mutation used previously. This mutation operator increments/decrements s_k by one with equal probability using bounds checking, that is, if $s_k = 0$, it was incremented to $s_k = 1$, and if $s_k = 7$, it was decremented to $s_k = 6$.

Fitness evaluation of each individual was calculated by scribing a path using the

fuzzy controller and stopping when:

(i) iteration (time steps) reached a prescribed limit (500), or

(ii) the path exceeded the maximum allowable distance from the ball, or

(iii) the robot collides with the ball.

As in the previous chapter, the physical size of the robot is considered in a collision. Scribing the path is stopped when the flag "HitBall" is true.

A grid of initial configurations are defined with (x, y, θ) coordinates as defined by Equations 8.6, excluding the ball position (x_B, y_B). The total number of initial configurations is therefore $C = 5(31 \times 27 - 1) = 4180$. All initial configurations start with zero, left and right, wheel velocity.

$$\begin{array}{rclcrcl} x & = & -750 + 100(k-1) & \text{for} & k & = & 1, \cdots, 31 \\ y & = & -650 + 100(k-1) & \text{for} & k & = & 1, \cdots, 27 \\ \theta & = & 2(k-1)\pi/5 & \text{for} & k & = & 1, \cdots, 5 \end{array} \tag{8.6}$$

The final position of the path obtained by running the fuzzy controller defined by the knowledge base (represented by each string) from a given grid configuration was used to evaluate the fitness of each individual as shown in Equation 8.7, where α_k for $k = 1, \cdots, 3$ are positive constants.

$$f_i = \alpha_1 T_1 + \alpha_2 T_2 + \alpha_3 T_3 \tag{8.7}$$

The first quantity in the fitness sum is $T_1 = d^2(R, DP)$. It is the final squared distance between the robot centre R and the destination point $DP = (688.5, 650)$ when the path is terminated as described above. It determines the accuracy of the fuzzy controller to control the system to the desired destination configuration.

Term T_2 is the iteration count for each path and is used to minimise the time taken to reach the desired destination configuration.

The third quantity $T_3 = \sin^2(\phi)$ where ϕ is the final angle of the robot relative to line $\overline{BG}$. This term is included to enable forward facing and reverse facing solutions to be accepted at the final destination.

Typical values of α_k used for the following results were: $\alpha_1 = 1.0$, $\alpha_2 = 1.0$, and $\alpha_3 = 100.0$. The terminal angle coefficient was heavily weighted to ensure the correct final alignment of the robot to the ball and the target goal.

The evolutionary algorithm was terminated after a prescribed number of generations. The best individual, that is, the one with the minimum fitness, is taken as the "best" fuzzy logic controller or defining the "best" knowledge base determined by the algorithm for this problem.

In Chapter 7, the learning of the knowledge base took place by summing the fitness evaluations for ALL starting configurations in the grid. This led to heavy computation in fitness evaluation, and difficulties in achieving convergence to acceptable paths in some instances from the initial configuration to the target point.

A new method for learning a knowledge base across the grid configuration, is employed and reported here. The evolutionary algorithm was run sequentially through the full number of initial configurations, being allowed to run for 10 generations at each configuration before moving to the next. That is, each individual in the population was evaluated with fitness derived from a single grid point for 10 generations, at which instance, fitness was then calculated at another point

in the grid for the next 10 generations and so on. The evolutionary process was stopped after a total of 500 000 generations in all were completed.

8.4 Discussion

The programme written for this chapter was written in Visual C with embedded assembly to improve the execution time. This programme has a graphical interface that displays the best solution every generation if the generation execution time is greater than 100ms. An entire run of 500 000 generations on a Pentium 4 1.8GHz machine took 30 wall clock days to complete. As the programme used a thread, this execution time is close to the total CPU time used. The GA statistics are also displayed on the graphical interface of the programme and were not written to file. As a consequence, the GA statics are not easily reproduced. However, the programme has since been converted to assembly for Linux and run for 50 000 generations to provide some statistics. The most interesting statistics to show are the average of the averages over eleven runs as shown in Figure 8.3. The average on minimum plot

Figure 8.3 shows a cyclic pattern of thirty-one (ignoring two spurious local maxima near 21 000 generations) local maxima every 41800 generations. There are thirty-one columns in the initial grid of starting configurations. Starting from generation zero, the initial configuration begins at the lower left corner of the grid. The average fitness improves until close to the next column of initial configurations near generation 1348. Then there is a rapid increase of the average fitness to the second local maxima at generation 1350. This indicates that the knowledge learnt from the previous column of initial configurations has been destroyed by the high mutation used in this programme. The local maxima show a

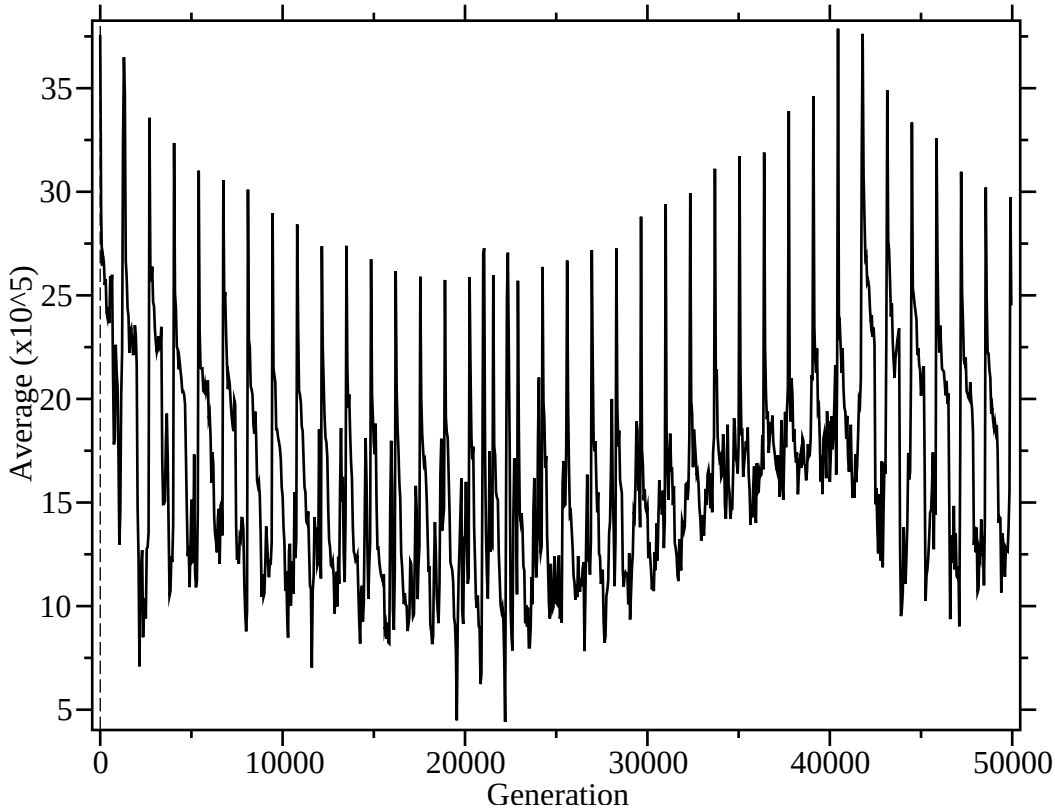


Figure 8.3: Average of averages over 11 runs for 50 000 generations

parabolic shape that indicates that the fitness is smaller for initial configurations close to the ball. The two spurious maxima around 21 000 generations indicates difficulties in finding suitable paths in the search space for initial configurations close to the ball. By the time all initial configurations have been tested, the average appears to shows no general improvement as shown in the figure at generation 41800.

To use statistical analysis of the final configuration of the robot, the outliers should be excluded from the data set. A plot of the d^2 parameter used in the fitness is shown in Figure 8.4.

Judging from this plot, a $d^2 = 4000$ can safely be considered as outliers. A plot of $100 \sin^2(\phi)$ looked like noise and did not hint towards a suitable threshold for

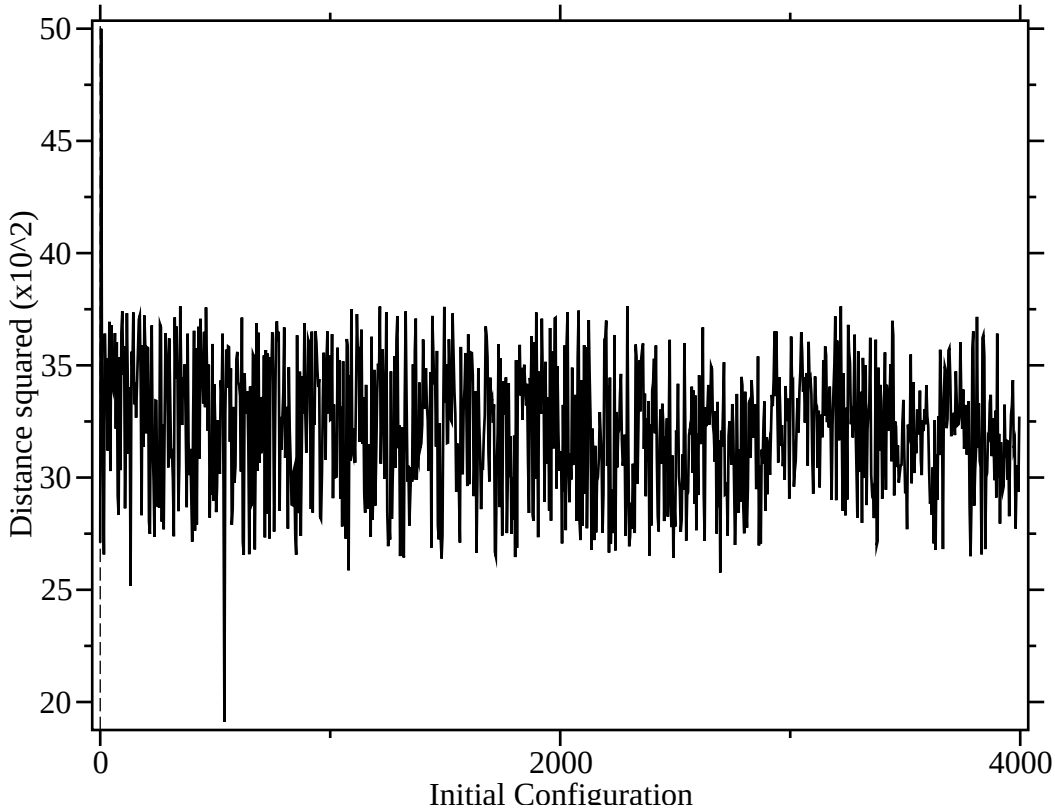


Figure 8.4: End point statistics of $\alpha_1 T_1$ term

an exclusion. However, considering that the angle needed to hit the ball into the goal is 15 degrees. Let us consider that angles greater than 10 degrees to be an outlier. This makes the threshold for the $100\sin^2(\phi)$ term in the fitness function to be greater than 3.0. Using these two condition for outliers, sixteen out of 4180 records are removed from the data set. Table 8.1 shows the statistics with the outliers removed.

Table 8.1: Statistics of 5I2O fuzzy controller

5I2O	d^2	iteration	$100sin^2(\phi)$
Min	191	6	1.4×10^{-11}
Max	397	415	1.7
Average	322	73	0.0084
Std. Deviation	287	25	0.058

Figure 8.5 shows a plot of the number of time steps of paths from each initial configuration. The time for each step is $\Delta T = 1/60$ seconds.

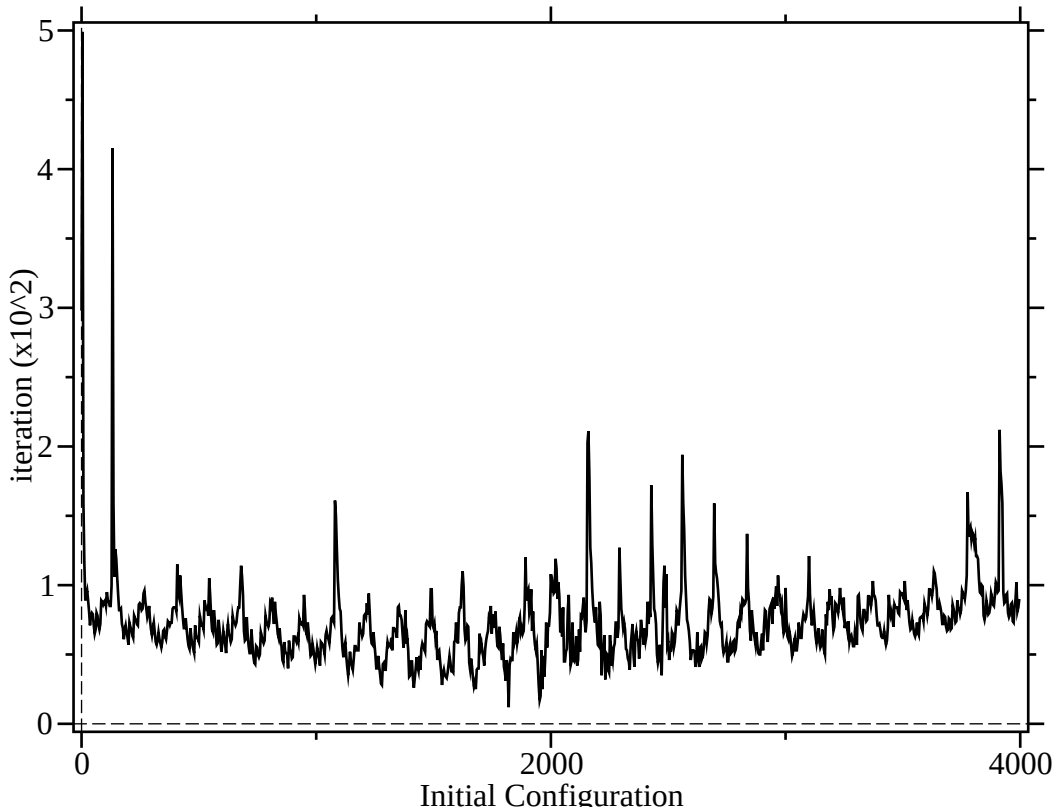


Figure 8.5: End point statistics of $\alpha_2 T_2$ term

Several mutation values were trialed before the full run of the programme. Although, this documentation cannot be retrieved due to a catastrophic hard drive failure. One would expect a mutation around 0.000007 to be more effective in improving the global fitness. The high mutation resulted from running the programme and inspecting the behaviour of the graphical display. Lower mutation rates appeared to fall into local minima and stagnate. In retrospect, changing the mutation from looking at the graphics of the paths was a local determination of a mutation rate.

Despite this evidence of poor GA convergence, the results obtained in the final

"best" fuzzy knowledge were excellent, obtaining very smooth continuous paths to the target with both forward and reverse facing in the final position depending on the initial configuration. Only a very small number of aberrations existed but the paths to the target were still acceptable. A number of the many images obtained are shown in Figures 8.6, 8.7, 8.8, and 8.9.

Figure 8.6 shows a long path from the left of the ball. The robot starts with zero velocity and executes a turn while accelerating. Then travelled towards the destination point at maximum velocity including two high speed turns. The robot reaches the destination point in $72/60 = 1.2$s with maximum velocity and excellent angle to hit the ball toward the goal. There are no cusps, or unnecessary loops in the path and it is considered to be an excellent path.

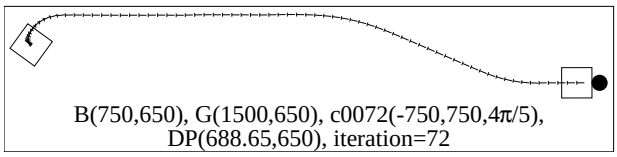


Figure 8.6: Long Distance Path from Left

Figure 8.7 shows a short path from above the ball. Initially the velocity is slow, then a small area of acceleration and deceleration within robot length occurs to settle to a slowly accelerating curve around to the destination point. Finally, the robot accelerates quickly and lines up to an angle to hit the ball to the goal. The path does not contain cusps or unnecessary looping. The final position, speed, and angle are excellent to hit the ball to the goal. The angular rotation of the robot at the destination point, may be too high for the robot to recover the ball quickly in the event of system errors. Time taken for this path is: $45/60 = 0.75$s.

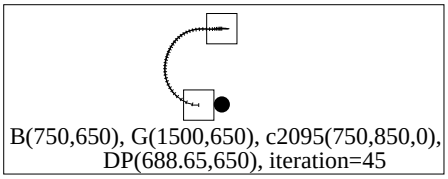


Figure 8.7: Medium Distance Path from Above

Figure 8.8 shows a medium length path starting below and to the right of the ball. The path starts off with slow acceleration and then quickly accelerates to a high speed "S" curve. The speed in the curve appears to be constant with a small deceleration towards the destination point.The angle of impact with the ball is good, but again the angular rotation of the robot may be too high to recover from system errors. Ideally, the angular velocity should be close to zero near to the destination point. This will allow the robot to recover in the presence of system errors such as determining the position of objects using vision processing, or another robot has hit the ball just before our robot does. The time take over this path is: $47/60 = 0.78$s.

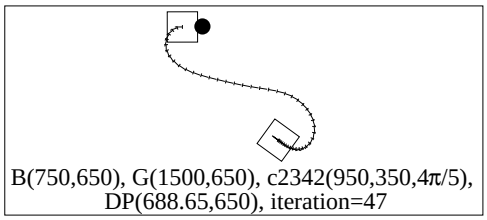


Figure 8.8: Medium Distance Path from Bottom Right

Figure 8.9 illustrates a long path from above and to the right of the ball. Similar to the other paths examined, the robot accelerates along a small radius curve until lined up to a maximum velocity gentle curve around the ball. The path decelerates to a small radius curve of medium velocity to line up with the destination point. Angle of impact is excellent, but the angular velocity of the robot may be to high to recover from system errors. There are no cusps or unnecessary

looping in this path. The time taken is: $86/60 = 1.4$s.

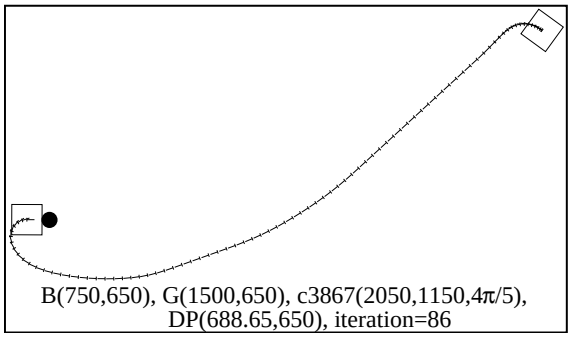


Figure 8.9: Long Distance Path from Right

A subset of the 4180 paths, that require 150 pages in print from, are discussed. Figures 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, and 8.16 show seven of these pages.

Figure 8.10 shows paths from initial configurations c0056 to c0083 inclusive. They are long paths from the left of the ball. These paths reach the destination point at at high speed and are correctly aligned to hit the ball to the goal. A few of the paths may have high angular velocity, such as c0078 to c0083 inclusive. All of the paths ended with forward facing impact with the ball.

Figure 8.11 shows paths from initial configurations c0560 to c0587 inclusive. They are medium to long paths from the left of the ball. All of these paths may have the high angular velocity close to the destination point problem. They are high speed and all reach the destination point in for a forward impact with the ball. There are no cusps or unnecessary looping in these paths. These paths are considered good.

Figure 8.12 shows paths from initial configurations c1148 to c1175 inclusive. They are medium paths from the left of the ball. The discussion is similar to that of

Figure 8.11 and are considered to be good paths.

Figure 8.13 shows paths from initial configurations c2072 to c2099 inclusive. They are short paths starting close to the ball. Chapter 7 found that learning a fuzzy controller from initial configurations close to the ball was difficult. It appears that increasing the number of inputs to the fuzzy controller has improved the learning of these configurations, but still presents some difficulties. Configuration c2072 uses a loop and a half before setting off towards the ball. Loops are found in configurations c2074, and c2076. While these paths have undesirable looping in them, it is not known if this behaviour is necessary for the controller to work, or is an area that needs more learning, or a better definition of the input sets for the fuzzy controller.

Configurations c2075, c2076, c2077, c2079, c2080, c2081, and c2093 contain a cusp early in the path. A cusp may be acceptable early in the path or near to the end of the path where speed is low. Cusps are undesirable in the high velocity parts of a path as it causes a large change in momentum. Rapid changes in momentum tend to cause instability and loss of control. Cusps are undesirable as it takes time to decelerate and then accelerate. As with loops, the inclusion of cusps may be necessary to allow control for other paths. They could indicate: that more learning is needed, different set definitions are needed in the fuzzy controller, or tuning of evolutionary parameters is necessary. From this discussion, configurations c2075 and c2081 are poor paths from this controller with the defined parameters. More learning or better definitions would need to be investigated.

Figure 8.14 shows close to medium paths to the right of the ball. Looping is found in configurations c2341, c2343, c2344, c2345, c2348, c2349, and c2351. The other

paths appear to be satisfactory.

Figure 8.15 shows medium to long initial configurations c2968 to c2995 inclusive. Generally, these paths are acceptable.

Figure 8.16 shows long paths to the right of the ball from initial configurations c3864 to c3891 inclusive. The shapes of the paths are close to straight lines, indicating high speed. All of the paths use a tight turn which may cause the "follow through" problem discussed above (caused by high angular velocity near impact). The angle of impact is good and these paths are considered to be good.

Compared to [50], the trajectories obtained are very smooth, there is much less tendency for the robot to execute high momentum turns resulting in a "cusp" along the trajectory.

A problem with premature convergence of the evolutionary algorithm was reported in [50]. The algorithm now outlined in this chapter has shown no premature convergence problems.

8.5 Comments

This chapter has dealt with learning a fuzzy rule base to be used in a fuzzy controller for micro-robots. This is a complex controller as it uses five input variables and produces control that allows for forward and reverse facing impact with the ball. Another criteria met with this controller is precise directional control to the destination point behind the ball from configurations far from the ball as well as close and touching the ball.

The fuzzy controller incorporates as input current left and right wheel velocities and it's output determines changes in these two velocities. Also shown, is a capability to learn the rules of such a knowledge base using an evolutionary algorithm.

The present analysis has been undertaken from initial configurations in which starting left and right wheel velocities are zero. To properly model this problem, initial configurations with non-zero velocities should be incorporated.

This requires extending the grid configuration to include the extra two variables of v_L and v_R, and in so doing drastically increases the already heavy computation in the evolutionary learning of the fuzzy rules.

It has already been noted that by increasing the number of input variables, the fuzzy rule base has increased in size considerably. The following chapter considers the same controller developed here, but using a three layered hierarchical fuzzy knowledge base [57]. A hierarchical system considerably reduces the size of the rule base being learnt.

 c0056	 c0057	 c0058	 c0059
c0060	c0061	c0062	c0063
c0064	c0065	c0066	c0067
c0068	c0069	c0070	c0071
 c0072	 c0073	 c0074	 c0075
 c0076	 c0077	 c0078	 c0079
 c0080	 c0081	 c0082	 c0083

Figure 8.10: Table of Results c0056–c0083

 c0560	 c0561	 c0562	 c0563
 c0564	 c0565	 c0566	 c0567
 c0568	 c0569	 c0570	 c0571
 c0572	 c0573	 c0574	 c0575
 c0576	 c0577	 c0578	 c0579
 c0580	 c0581	 c0582	 c0583
 c0584	 c0585	 c0586	 c0587

Figure 8.11: Table of Results c0560–c0587

c1148	c1149	c1150	c1151
c1152	c1153	c1154	 c1155
 c1156	 c1157	 c1158	 c1159
 c1160	 c1161	 c1162	 c1163
 c1164	 c1165	 c1166	 c1167
 c1168	 c1169	 c1170	 c1171
 c1172	 c1173	 c1174	 c1175

Figure 8.12: Table of Results c1148–c1175

 c2072	 c2073	 c2074	 c2075
 c2076	 c2077	 c2078	 c2079
 c2080	 c2081	 c2082	 c2083
 c2084	c2085	c2086	 c2087
 c2088	c2089	c2090	c2091
 c2092	 c2093	c2094	 c2095
 c2096	 c2097	 c2098	 c2099

Figure 8.13: Table of Results c2072–c2099

 c2324	 c2325	 c2326	 c2327
 c2328	 c2329	 c2330	 c2331
 c2332	 c2333	 c2334	 c2335
 c2336	 c2337	 c2338	 c2339
 c2340	 c2341	 c2342	 c2343
 c2344	 c2345	 c2346	 c2347
 c2348	 c2349	c2350	 c2351

Figure 8.14: Table of Results c2324–c2351

 c2968	 c2969	 c2970	 c2971
 c2972	 c2973	 c2974	 c2975
 c2976	 c2977	 c2978	 c2979
 c2980	 c2981	 c2982	 c2983
 c2984	 c2985	 c2986	 c2987
 c2988	 c2989	 c2990	 c2991
 c2992	 c2993	 c2994	 c2995

Figure 8.15: Table of Results c2968–c2995

 c3864	 c3865	 c3866	 c3867
 c3868	 c3869	 c3870	 c3871
 c3872	 c3873	 c3874	 c3875
 c3876	 c3877	 c3878	 c3879
 c3880	 c3881	 c3882	 c3883
 c3884	 c3885	 c3886	 c3887
 c3888	 c3889	 c3890	 c3891

Figure 8.16: Table of Results c3864–c3891

Chapter 9

Hierarchical Modelling

Substantial parts of this chapter were submitted to the International Conference on Evolutionary Computation (CEC2003) in Canberra, Australia. Development of a three layer, five input, two output fuzzy controller is considered. The results presented in this chapter are through simulation as the resulting controller was incomplete.

9.1 Introduction

Hierarchical Fuzzy Logic (HFL) systems have been used in work such as [84] and [85]. Hierarchical systems are usually used in fuzzy systems to reduce the number of fuzzy rules as increasing number of sets in each input universe gives rise to the "curse of dimensionality". Normally, the most important input variables [85] are chosen for the first layer. The second layer uses the second most important variables, and so on.

A fuzzy system using five input variables $x_1 = \phi$, $x_2 = d^2$, $x_3 = \theta$, $x_4 = v_L$ and

$x_5 = v_R$ using seven membership sets per input results in $7^5 = 16807$ rules. By using a three layer hierarchical fuzzy system the number of rules is reduced to $3 \times 7^3 = 1029$. Normally, this would be approximately sixteen times faster to compute. However, this is not the case because of a sparse matrix technique using only those sets that "fire". This technique was discussed in Chapter 8. A vertical line through complete fuzzy sets will have at least one non-zero membership to two non-zero membership values. The two sets that fire in each input universe is therefore $2^5 = 32$ calculations out of 16807. The same technique is used for the hierarchical fuzzy system using: $3 \times 2^3 = 24$ calculations. The hierarchical system has additional calculations (overheads) as variables are passed between each fuzzy system in the hierarchical layers and needed a slightly longer time for calculation than the five input model.

Allocation of the input variables to layers requires a choice of priority. The importance of the input variables was unknown and an arbitrary selection of importance chosen as: ϕ, v_L and v_R for the first fuzzy layer, $d, \Delta v_L^1$ and Δv_R^1 for the second and $\theta, \Delta v_L^2$ and Δv_R^2 for the third. Where Δv_L^k and Δv_R^k, for $k = 1, \cdots, 2$, are input from the output of the previous fuzzy layer. Figure 9.1 shows the hierarchical configuration used. The system and kinematics are the same as used in Chapter 7.

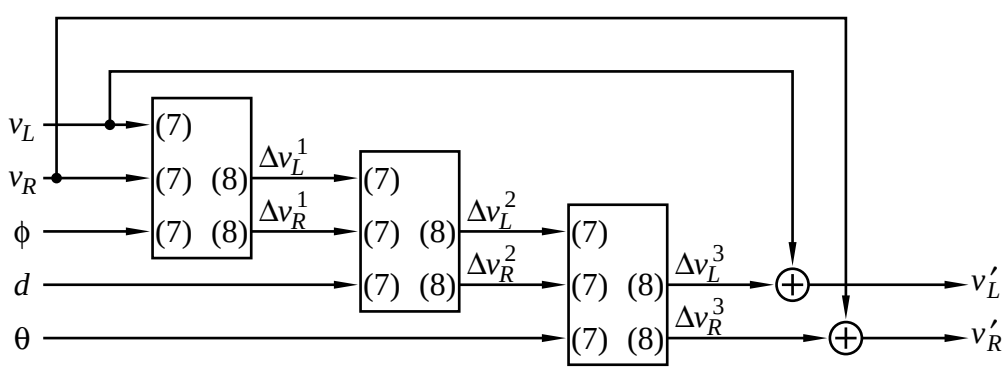


Figure 9.1: Hierarchical Model

9.2 Fuzzy Control System Design

In this section, each of the three fuzzy controllers as shown in Figure 9.1 will be discussed.

9.2.1 First Fuzzy System

Three inputs are used in the first layer of the fuzzy control system as described in Equations 9.1.

$$\begin{aligned} x_1 &= v_L \\ x_2 &= v_R \\ x_3 &= \phi \end{aligned} \tag{9.1}$$

Two outputs from the first fuzzy controller are described in Equations 9.2.

$$\begin{aligned} y_1 &= \Delta v_L^1 \\ y_2 &= \Delta v_R^1 \end{aligned} \tag{9.2}$$

There are seven linguistic membership sets defined for each input variable. The linguistic sets defined for angle ϕ are: **VS** is Very Small, **S** is Small, **SM** is Small Medium, **M** is Medium, **ML** is Medium Large, **L** is Large and **VL** is Very Large. Seven membership names for left v_L and right v_R wheel velocities are: **EN** is Extremely Negative, **VN** is Very Negative, **N** is Negative **Z** is Zero, **P** is Positive, **VP** is Very Positive, and **EP** is Extremely Positive. The membership functions used for the first layer of the hierarchical structure is shown in Figure 9.2.

Both output variables are change in velocity for left Δv_L^1 and right Δv_R^1 wheel velocities are defined with eight membership sets using linguistic names: **VLN** is Very Large Negative, **LN** is Large Negative, **MN** is Medium Negative, **SM** is Small Negative, **SP** is Small Positive, **MP** is Medium Positive, **LP** is Large

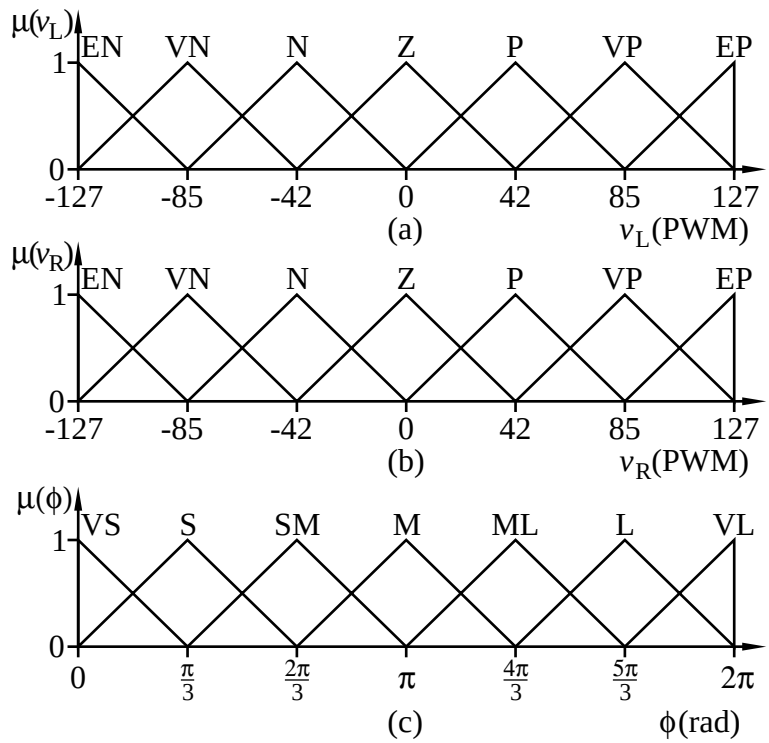


Figure 9.2: Membership Sets Defined for the First Hierarchical Layer

Positive, and **VLP** is Very Large Positive. Equation 9.3 describes the output centres for the first fuzzy controller.

$$\overline{y}_k^\ell = -28 + 8j \;\text{ for }\; k = 1, 2 \;\text{ and }\; j = 0, \cdots, 7 \tag{9.3}$$

The general form of the ℓth rule in the first rule base is:

$$\text{If } (x_1 \text{ is } A_1^\ell \text{ and } x_2 \text{ is } A_2^\ell \text{ and } x_3 \text{ is } A_3^\ell)$$
$$\text{Then } (y_1 \text{ is } B_1^\ell \text{ and } y_2 \text{ is } B_2^\ell).$$

where $A_k^\ell, k = 1, \cdots, 3$ are normalised fuzzy sets for input variables $x_k, k = 1, \cdots, 3$, and where $B_m^\ell, m = 1, 2$ are normalised fuzzy sets for output variables $y_m, m = 1, 2$.

Each layer output is assumed to be similar to the final layer for simplicity. The intermediate variables, $\Delta v_L^1, \Delta v_R^1, \Delta v_L^2,$ and Δ_R^2, could be non-physical variables, having unknown domain. The complexity of what happens in the intermediate layers is avoided by assuming that the domain of all layer output are the same. Research in this area is beyond the scope of this thesis.

9.2.2 Second Fuzzy System

The two outputs from the first layer are are fed as inputs to the second fuzzy controller as Equation 9.4 and Equation 9.5 respectively. The third input is the distance d as described in Equation 9.6.

$$x_4 = \Delta v_L^1 \tag{9.4}$$

$$x_5 = \Delta v_R^1 \tag{9.5}$$

$$x_6 = d \tag{9.6}$$

Two outputs from the second fuzzy controller are described in Equations 9.7.

$$\begin{aligned} y_3 &= \Delta v_L^2 \\ y_4 &= \Delta v_R^2 \end{aligned} \tag{9.7}$$

Seven membership sets are defined for change in left Δ_L^1 and right Δv_R^1 wheel velocities with linguistic names: **EN** is Extremely Negative, **VN** is Very Negative, **N** is Negative **Z** is Zero, **P** is Positive, **VP** is Very Positive, and **EP** is Extremely Positive. Membership sets defined for distance d are seven in number with linguistic names: **I** is Impact, **VC** is Very Close, **C** is Close, **N** in Near, **M** is Medium, **F** is Far, and **VF** is Very Far, as shown in Figure 9.3.

Both output variables are change in velocity for left Δv_L^2 and right Δv_R^2 wheel

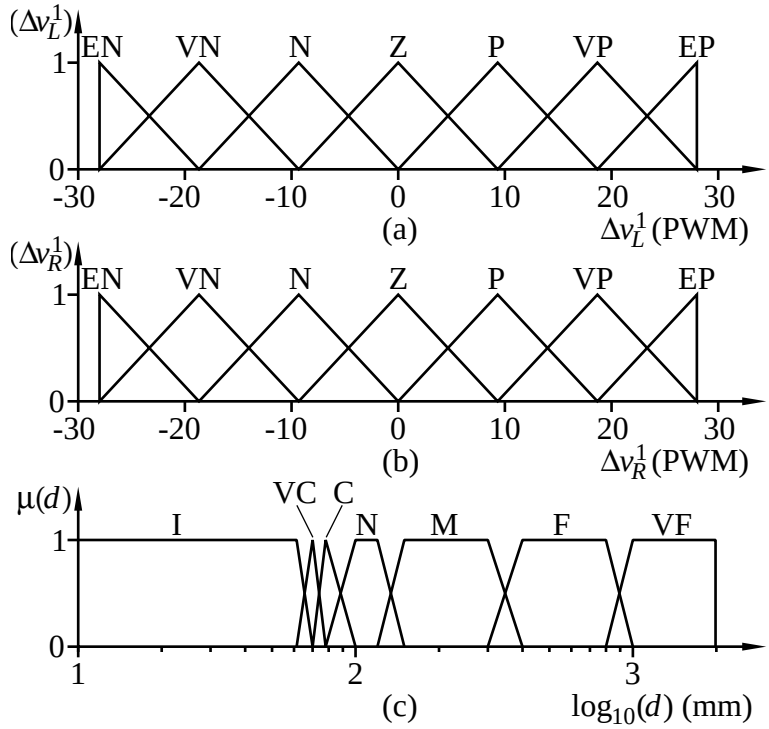


Figure 9.3: Membership Sets Defined for the Second Hierarchical Layer

velocities are defined with eight membership sets using linguistic names: **VLN** is Very Large Negative, **LN** is Large Negative, **MN** is Medium Negative, **SM** is Small Negative, **SP** is Small Positive, **MP** is Medium Positive, **LP** is Large Positive, and **VLP** is Very Large Positive. Equation 9.8 describes the output centres for the second fuzzy controller.

$$\overline{y}_k^\ell = -28 + 8j \quad \text{for} \quad k = 3, 4 \quad \text{and} \quad j = 0, \cdots, 7 \tag{9.8}$$

The general form of the ℓth rule in the second rule base is:

If (x_4 is A_4^ℓ and x_5 is A_5^ℓ and x_6 is A_6^ℓ)

Then (y_3 is B_3^ℓ and y_4 is B_4^ℓ).

where $A_k^\ell, k = 4, \cdots, 6$ are normalised fuzzy sets for input variables $x_k, k =$

$4, \cdots, 6$, and where $B_m^\ell, m = 3, 4$ are normalised fuzzy sets for output variables $y_m, m = 3, 4$.

9.2.3 Third Fuzzy System

The two outputs from the second layer are are fed as inputs to the third fuzzy controller as Equation 9.9 and Equation 9.10 respectively. The third input is the angle θ as described in Equation 9.11.

$$x_7 = \Delta v_L^2 \tag{9.9}$$

$$x_8 = \Delta v_R^2 \tag{9.10}$$

$$x_9 = \theta \tag{9.11}$$

Two outputs from the second fuzzy controller are described in Equations 9.12.

$$\begin{aligned} y_5 &= \Delta v_L^3 \\ y_6 &= \Delta v_R^3 \end{aligned} \tag{9.12}$$

Seven membership sets are defined for change in left Δ_L^2 and right Δv_R^2 wheel velocities with linguistic names: **EN** is Extremely Negative, **VN** is Very Negative, **N** is Negative **Z** is Zero, **P** is Positive, **VP** is Very Positive, and **EP** is Extremely Positive. Angle theta θ is defined using seven sets with linguistic names: **VS** is Very Small, **S** is Small, **SM** is Small Medium, **M** is Medium, **ML** is Medium Large, **L** is Large and **VL** is Very Large as shown in Figure 9.4.

Both output variables are change in velocity for left Δv_L^3 and right Δv_R^3 wheel velocities are defined with eight membership sets using linguistic names: **VLN** is Very Large Negative, **LN** is Large Negative, **MN** is Medium Negative, **SM** is Small Negative, **SP** is Small Positive, **MP** is Medium Positive, **LP** is Large

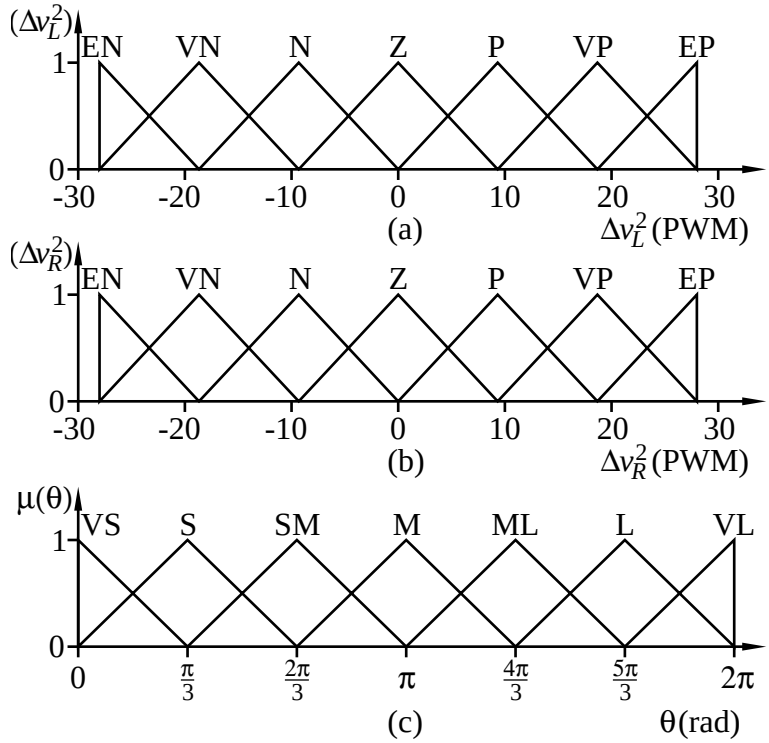


Figure 9.4: Membership Sets Defined for the Third Hierarchical Layer

Positive, and **VLP** is Very Large Positive. Equation 9.13 describes the output centres for the second fuzzy controller.

$$\overline{y}_k^\ell = -28 + 8j \quad \text{for} \quad k = 5, 6 \quad \text{and} \quad j = 0, \cdots, 7 \tag{9.13}$$

The first fuzzy system contains $7^3 = 343$ rules, the second fuzzy system contains $7^3 = 343$ rules, and third fuzzy system in the hierarchical structure contains $7^3 = 343$ rules giving a total of $3 \times 7^3 = 1029$ rules.

9.2.4 System Design

Given a fuzzy rule base with M rules and n antecedent variables, a fuzzy controller as given in Equation 7.1 uses a singleton fuzzifier, Mamdani product inference engine and centre average defuzzifier to determine output variables.

These values, 2058 of them, are typically unknown and require determination in establishing valid output for controls to each wheel of the robot. As there is no a priori knowledge about the system control, an evolutionary algorithms (EA) [27] is used to search for an acceptable solution.

A simple method of implementing Equation 7.1 is to use nested loops for the summation and product terms for each hierarchical stage. This simple encoding requires the loops to consider antecedents and consequents of $M = 343$ rules.

A special feature of completely overlapping sets [8] using the Mamdani product inference engine is that a minimum of one and a maximum of two membership sets for each input variable will fire. A vertical slice through any variable membership in Figure 9.2 illustrates this property.

A maximum of $2^3 = 8$ out of the $7^3 = 343$ rules will fire for any input into any of the hierarchical stages. With a sparse access to the rule base, it makes sense to access the rule base by developing pointers to the rule being used.

The number of membership sets was set at seven for each input variable to make the calculation of the pointer to the rule an easy radix calculation. It is not necessary to use the same number of memberships per variable the same, it was done simply to reduce errors made in programming and ease debugging.

Each input variable is tried on the input membership sets to find the one or two memberships that fire. The set identification is stored in an array with the membership value(s). Each array is terminated by a pointer value of -1. Five input variables require five nested loops to calculate the pointer reference using the antecedents in accordance with Equation 7.1. The pointer refers to the

consequent $\overline{y}_k^\ell$ in the string for inclusion into Equation 7.1.

9.3 Evolutionary Learning

Construction of the fuzzy rule base cannot be made by heuristic formulation as there is no a priori knowledge and high dimensionally. The fuzzy rule base to control the robot is found by an evolutionary algorithm (EA) [27].

Each output fuzzy set is represented by an integer in the interval $[0, 7]$ corresponding to the fuzzy output sets: **VLN** is Very Large Negative, **LN** is Large Negative, **MN** is Medium Negative, **SM** is Small Negative, **SP** is Small Positive, **MP** is Medium Positive, **LP** is Large Positive, and **VLP** is Very Large Positive. A potential knowledge base solution can be uniquely represented as a concatenated string $\underset{\sim}{s} = \{\underset{\sim}{s}_1, \underset{\sim}{s}_2, \underset{\sim}{s}_3\}$, where $s_j, j = 1, 2, 3$ represents the rule base corresponding to each of the three fuzzy systems in the hierarchical model. Each sub-string contains $2M = 686$ integers, representing consequents of the form:

$$\underset{\sim}{s}_k = \{s_1^1, s_2^1, \cdots, s_1^k, s_2^k, \cdots, s_1^M, s_2^M, \},$$

where $s_j, j = 1, 2$ is an integer in the interval $[0, 7]$.

The population at generation t, $P(t) = \{\underset{\sim}{s}^n \ : \ n = 1, \cdots, N\}$, where $N = 2000$ is the number of individuals in the population. The population at the next generation $P(t+1)$ was built using a full replacement policy, tournament selection with size $n_T = 3$, and one point crossover with probability $p_c = 0.6$. Elitism was used, with the 10 best individuals carried from population $P(t)$ to population $P(t+1)$. An incremental mutation operator with probability $p_m = 0.01$, increments, or decrements s_k by one with equal probability using bounds

checking. That is if $s_k = 0$, it was incremented to $s_k = 1$, and if $s_k = 7$, it was decremented to $s_k = 6$, otherwise incremented or decremented with probability 0.5. The GA was run for a set number of generations 500 000.

Fitness evaluation of each individual was calculated by scribing a path as shown in Chapter 8. The fitness is also evaluated over the same initial configurations as shown in the previous chapter.

9.4 Discussion

The programme written for this chapter was written in Visual C with embedded assembly to improve the execution time. This programme has a graphical interface that displays the best solution every generation if the generation execution time is greater than 100ms. An entire run of 500 000 generations on a Pentium 4 1.8GHz machine took 32 wall clock days to complete. As the programme used a thread, this execution time is close to the total CPU time used. The GA statistics are also displayed on the graphical interface of the programme and were not written to file. As a consequence, the GA statics are not easily reproduced. However, the programme has since been converted to assembly for Linux and run for 50 000 generations to provide some statistics. The most interesting statistics to show is the average of the averages over eleven runs as shown in Figure 9.5.

To use statistical analysis of the final configuration of the robot, the outliers should be excluded from the data set. A plot of the d^2 parameter used in the fitness is shown in Figure 9.6. Judging from this plot, a distance measures $d^2 > 4000$ can be safely considered as outliers. A plot of $100 \sin^2(\phi)$ looked like noise and did not hint towards a suitable threshold for an exclusion. However, a guide

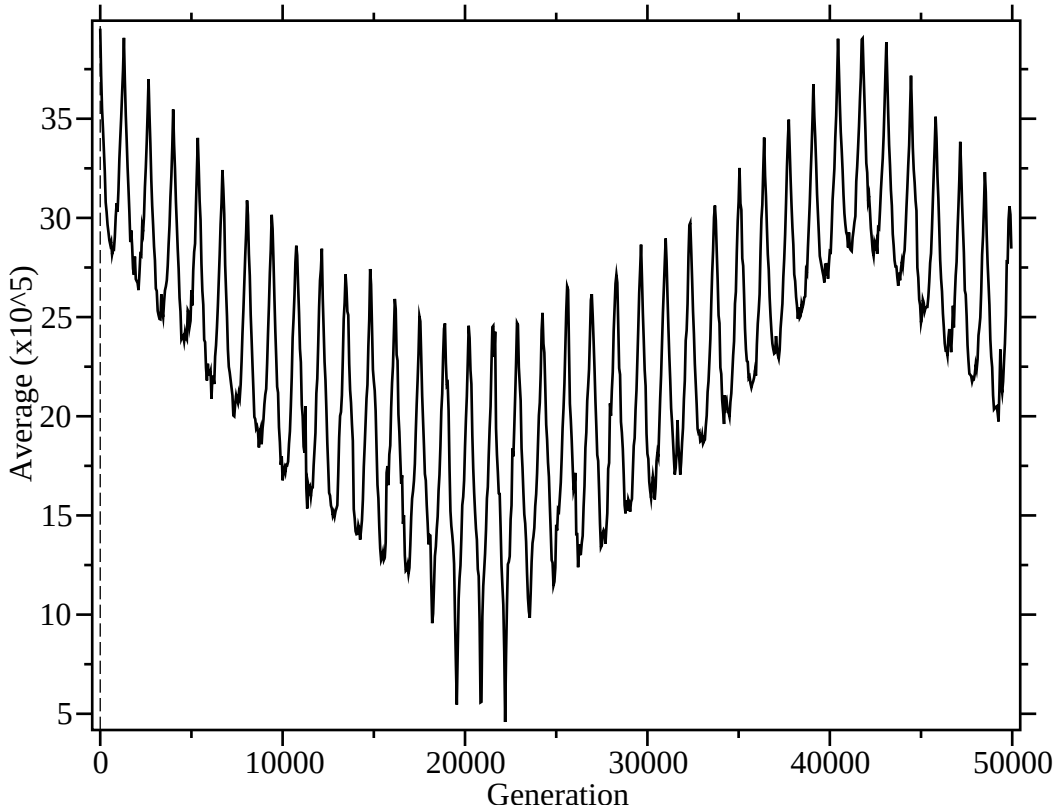


Figure 9.5: Average of averages over 11 runs for 50 000 generations

is that the angular error needed to hit the ball into the goal is better than 15 degrees. Let us consider that angles greater than 10 degrees to be an outlier. This makes the threshold for the $100\sin^2(\phi)$ term in the fitness function to be greater than 3.0. Using these two conditions for outliers, 187 out of 4180 records are removed from the data set. Table 9.1 shows the statistics with the outliers removed.

Table 9.1: Statistics of Hierarchal fuzzy controller

Hierarchical	d^2	iteration	$100sin^2(\phi)$
Min	1600	5	5.8×10^{-11}
Max	3960	485	3.0
Average	3210	65	0.042
Std. Deviation	324	22	0.21

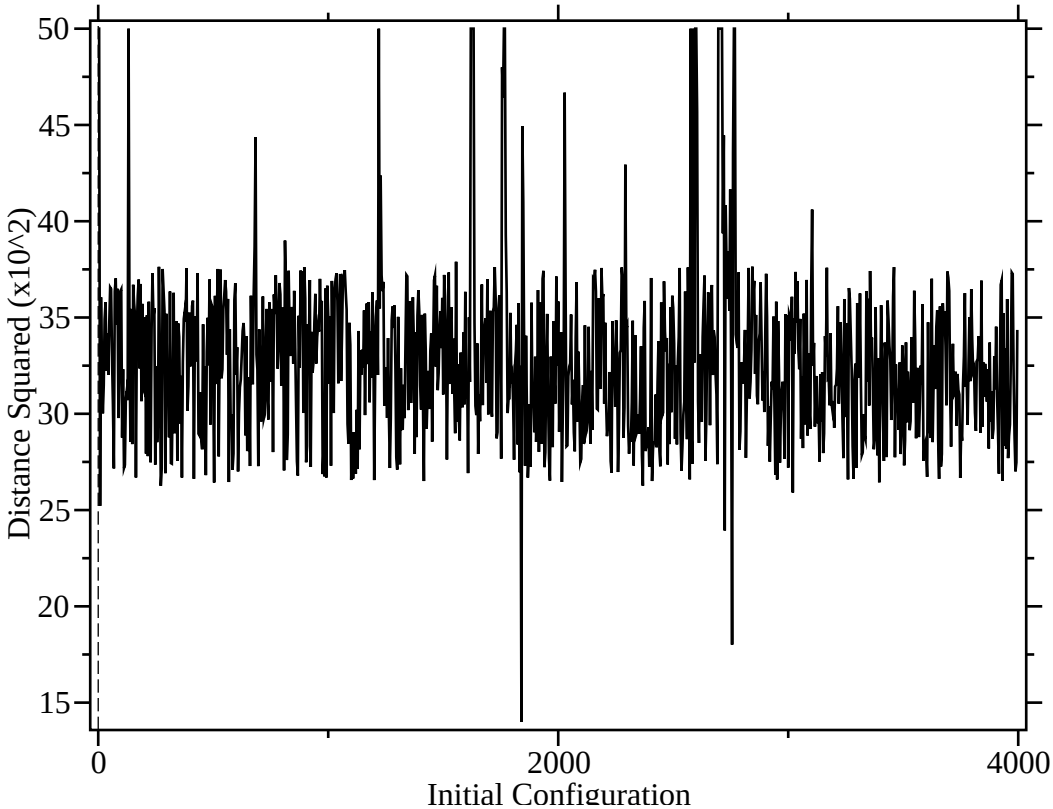


Figure 9.6: End point statistics of $\alpha_1 T_1$ term

Figure 9.7 shows a plot of the number of time steps of paths from each initial configuration. The time for each step is $\Delta T = 1/60$ seconds.

Figures 9.5 and 9.7 show a cyclic pattern of thirty-one local maxima every 41800 generations. There are thirty-one columns in the initial grid of starting configurations. Starting from generation zero, the initial configuration begins at the lower left corner of the grid. The average fitness improves until half way through the column at around generation 670 and then gets progressively worse until the next column starts at generation 1350. The shape between generation 0 to generation 1350 looks like a quadratic type of function. The improvement from generation 0 to 670 shows that the learnt search space from the previous generations is retained. From generation 671 to 1349, the average fitness gets worse as the search goes into unfamiliar territory. When the next column of initial configurations is

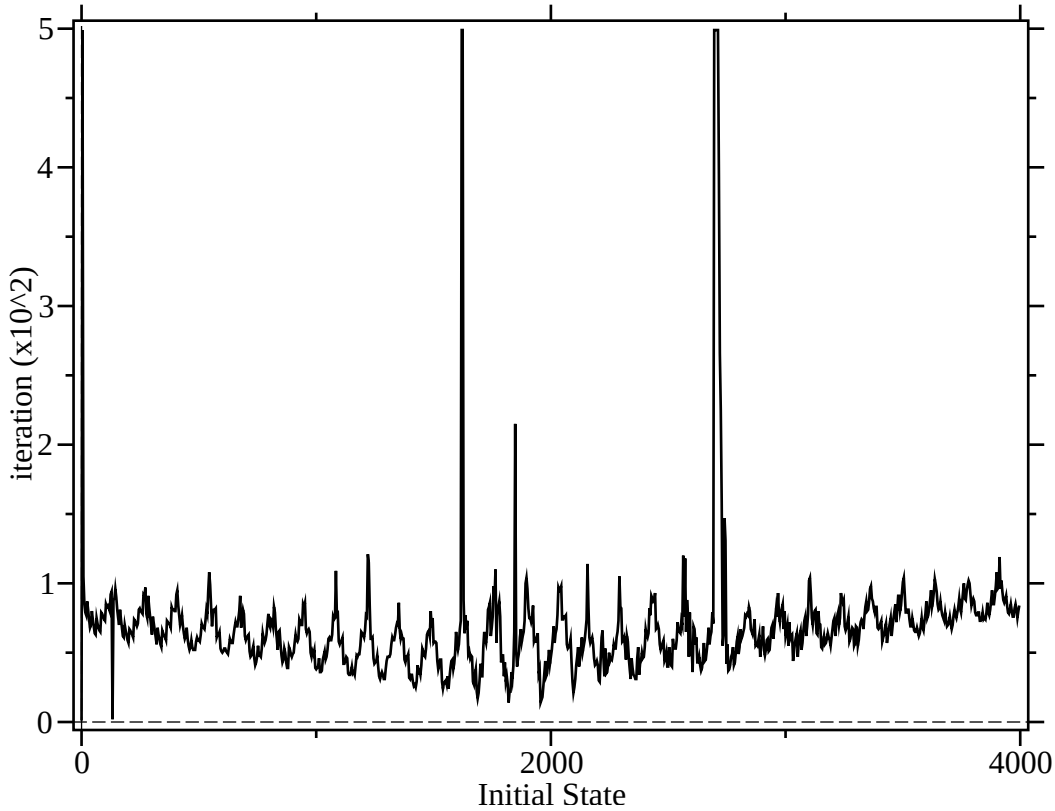


Figure 9.7: End point statistics of $\alpha_2 T_2$ term

started at generation 1350, the learnt space has been destroyed. When all of the initial configurations have been tested, the average fitness at generation 41800 does not seem better than generation 0 (showing no net decrease in the average fitness). As in the previous chapter, the mutation seems too high and should be near 0.0001 (1/(4 * 2058)). However, for comparison reasons, the mutation rate was set to that of the previous chapter to 0.01.

Scribed paths are considered satisfactory if they have the following attributes:

- Plot of path appears smooth,
- Can contain a cusp near to initial configuration or final configuration of the robot in a path, and

- Should not have unnecessary looping or cusps in the middle of a path.

Despite this evidence of poor GA convergence, the results obtained in the final "best" fuzzy knowledge were excellent, obtaining very smooth continuous paths to the target with both forward and reverse facing in the final position depending on the initial configuration. Only a very small number of aberrations existed but the paths to the target were still acceptable. A select few of the many paths obtained are compared to the results found in Chapter 8.

Comparing trajectories in Figure 9.8 to Figure 8.6, we see they are close in their properties. They both accelerate smoothly from the initial configuration to a straight section. The straight sections are traversed at maximum speed, but Figure 9.8 has one less turn than Figure 8.6. Both pass through the destination point at maximum speed with a very good angle of approach to hit the ball to the goal. Both paths are forward facing for impact with the ball. The time take for the path in Figure 9.8 is: $69/60 = 1.15$s, which is 0.05s faster than the paths in Figure 8.6.

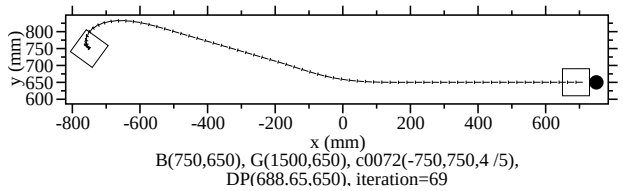


Figure 9.8: Long Distance Path from Left

The scribed path in Figure 9.9 show some improvement over the path in Figure 8.7. Both accelerate and scribe a smooth curve to the destination point. The average speed of the robot in Figure 9.9 is higher than in Figure 8.7. The impact with the ball in Figure 9.9 is not a good as in Figure 8.7. Although, the impact

shown in Figure 9.9 could place the ball on a trajectory towards the goal as the acceptable angle form the ball position is 0 ± 15 degrees. The time taken for the robot to reach the destination point in Figure 9.9 is: $24/60 = 0.4$s. This is 0.35s faster than the robot path in Figure 8.7. As with Figure 8.7, the robot angular velocity at point of impact with the ball in Figure 9.9 could be too high for recovery from system errors.

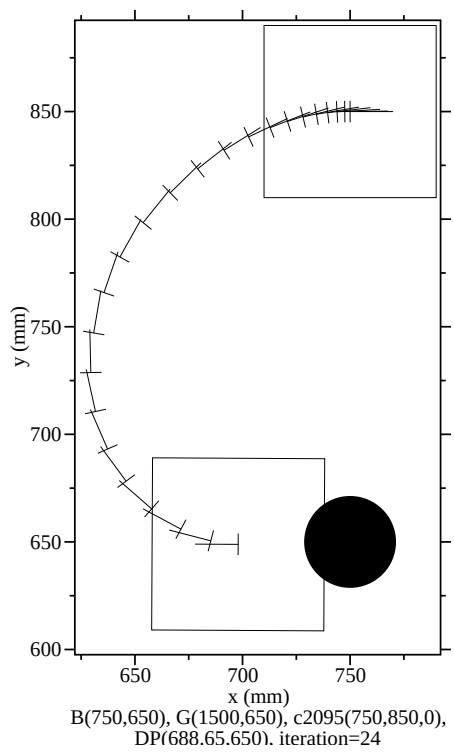


Figure 9.9: Medium Distance Path from Above

Compared to Figure 8.8, the scribed path in Figure 9.10 shows some improvement. Both paths accelerate to a high speed curve and arrive at the destination point with an acceptable angle of attack between the robot and ball. The path in Figure 9.10 is a shorter path then in Figure 8.8. In both paths, the robot may have a high angular velocity at the destination point that could prevent follow through in case of system error. The time taken for the robot to scribe the path in Figure 9.10 is: $37/60 = 0.6$s and is 0.17s faster then the path in Figure 8.8.

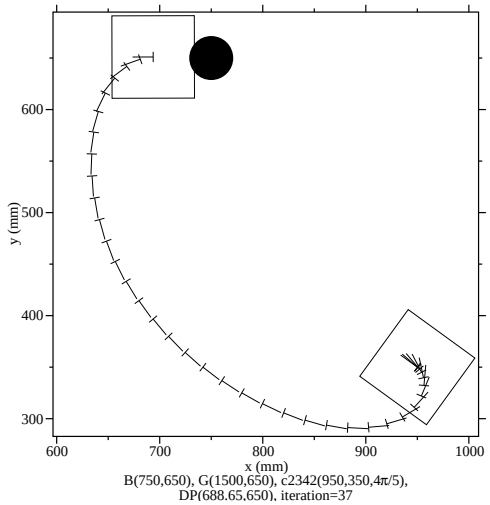


Figure 9.10: Long Distance Path from Bottom Right

The scribed paths are similar between Figures 9.11 and 8.9. Both accelerate to a high speed curve passing under the ball and decelerate on a tight curve to the destination point. Their impact angle with the ball is good, though problems may arise with follow through after impact. Neither have selected the shorter path passing over the ball. The time taken for the robot to follow this path is: $81/60 = 1.35$s, which is 0.08s faster then the path in Figure 8.9. The similarity between these paths is very high.

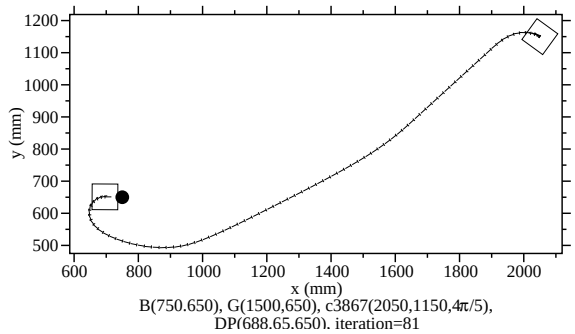


Figure 9.11: Long Distance Path from Right

A subset of the 4180 trajectories are presented in Figures 9.12 to 9.18. Presenting the 4180 paths in print form requires 150 pages.

Figure 9.12 shows long paths from the left of the ball. Compared to the corresponding paths in Figure 8.10, the robot ends each path with zero angular velocity. Both figures present acceptable paths for the robot to to hit the ball towards the goal. However, a notable exception is that some of the paths in Figure 8.10 have a high angular velocity just prior to impact with the ball.

Medium to long length paths from the left of the ball are presented in Figure 9.13. Compared to the corresponding paths in Figure 8.11, the distance travelled by the robot is longer. All of the paths in Figure 9.13 finish with zero angular velocity, while those in Figure 8.11 have some angular velocity at impact with the ball.

Figure 9.14 shows medium length paths from the left of the ball. These paths are comparable to the paths in Figure 8.12 in many respects. Path lengths are similar, approach to the destination point is similar and angular velocity at impact. A notable difference among the paths is that the robot tends to rotate clockwise at the beginning of each path in Figure 8.12 and tends rotate anti-clockwise in Figure 9.14.

Paths with initial configurations close to the ball are presented in Figure 9.15. Compared to the corresponding paths in Figure 8.13, the paths do not exhibit unfavourable behaviour such as as cusps and unnecessary looping. The path from initial configuration c2072 in Figure 8.13 has a couple of loops at the start of the path while there is no unnecessary looping in the same initial configuration in Figure 9.15. The cusp in initial configuration c2081 in Figure 8.13 is absent in the same configuration in Figure 9.15. The path from initial configuration c2089 is longer in Figure 9.15 than the corresponding configuration in Figure 8.13.

Short to medium length paths are presented in Figure 9.16. Compared to cor-

responding paths in Figure 8.14, there is an improvement in the reduction of unnecessary looping. The path from configuration c2324 in Figure 9.16 shows a loop at the start of the path, whereas there is no loop in the same configuration in Figure 8.14. Length of path from initial configuration c2325 is longer than the respective path in Figure 8.14. The loop at the start of the path from initial configuration c2345 in Figure 8.14 is absent form the respective configuration in Figure 9.16.

Long paths to the right and below the ball are presented in Figure 9.17. A notable quality is the length of the paths are shorter to the respective paths in Figure 8.15. Path from initial configuration c2981 contains a loop at the beginning of the path in Figure 9.17, but not in Figure 8.15.

Long paths to the right and above the ball are presented in Figure 9.18. Their attributes are similar to the corresponding paths in Figure 8.16. A notable tendency is that the robot passes under the ball in both figures. The angular velocity at the robot-ball impact position shows high angular velocity in both figures.

Compared to the previous chapter, the trajectories obtained are very smooth, there is much less tendency for the robot to execute high momentum turns resulting in a "cusp" along the trajectory.

A problem with premature convergence of the evolutionary algorithm was reported in [50]. The learning of the hierarchical fuzzy controller did not show signs of premature convergence.

9.5 Comments

This chapter shows how to design a complex fuzzy rule base for a hierarchical fuzzy controller. As in the previous chapter, the controller adequately satisfies the criteria:

- Forward and reverse impact with the ball,
- Precise directional control to the destination point behind the ball, and
- Control of short and long distant paths.

The results are very similar to those obtained with the 5I2O fuzzy controller. However, the paths produced by the hierarchical fuzzy controller are all shorter in time than the 5I2O counterparts. This is not a significant result indicating that the hierarchical fuzzy controller is better than the 5I2O controller, because the paths are shorter due to large steps towards the wheel lift constraint regions. If the wheel lift constraints were applied, the difference between the hierarchical and 5I2O controller should be minimal.

The hierarchical fuzzy controller behaved much like its 5I2O counterpart. The calculation speed is the same as the 5I2O counterpart, because of a method of using indexing to the fuzzy rule base. The advantage is a smaller memory size for the population. The rule base for the 5I2O model cannot be practically extended because of the effect of the "curse of dimensionality" causing difficulties in execution time. Also, the size of the population make its use in concurrent programming on a Mosix Linux cluster impractical. The significantly reduced population size in the hierarchical model makes concurrent programming on the

Mosix cluster a practical exercise. Although, the execution time could be an issue.

An alternative to learning a large fuzzy rule base with increased membership sets is to learn velocity profiles, convert to input/output data and use neighbourhood clustering.

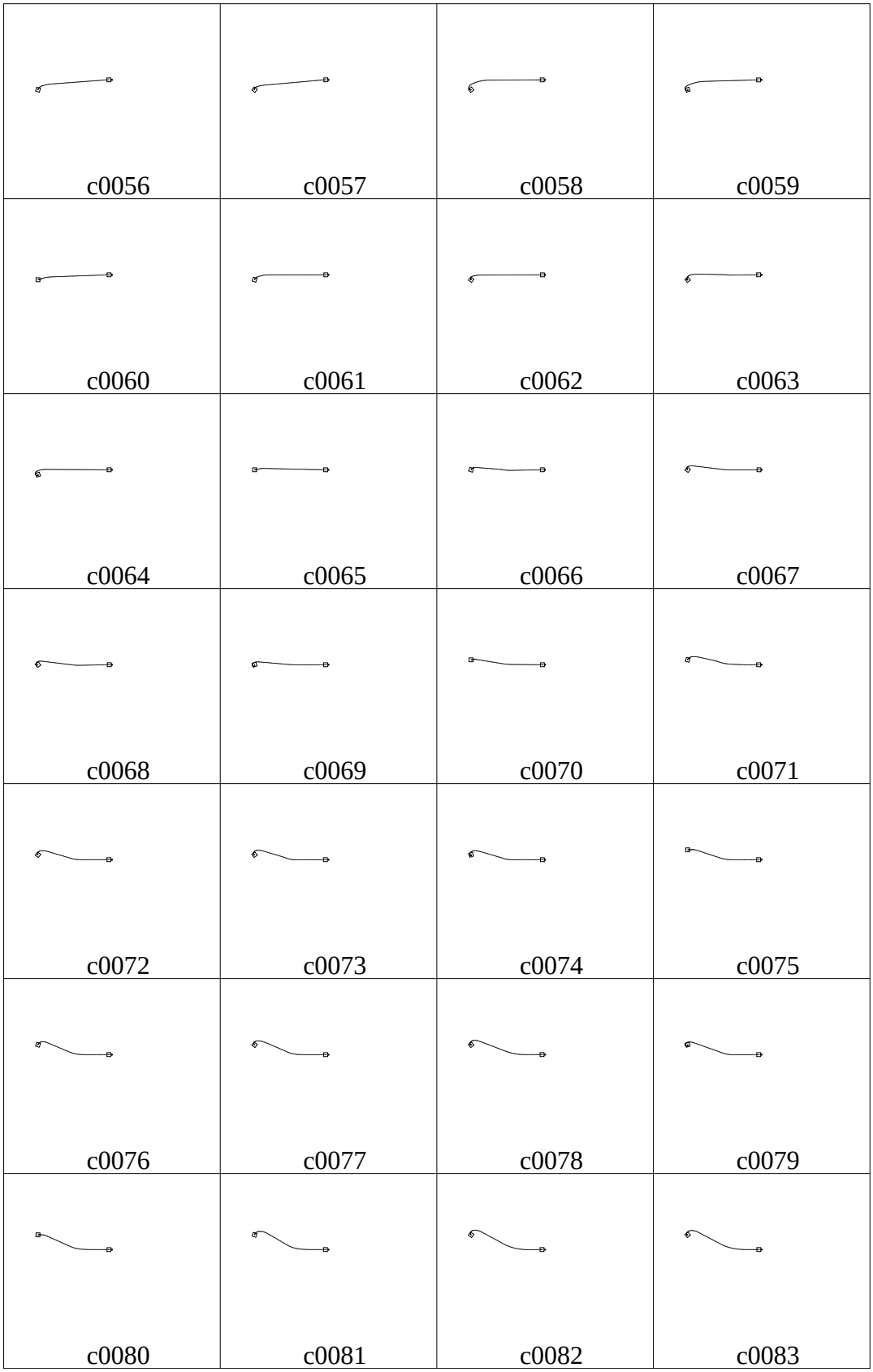


Figure 9.12: Table of Results c0056–c0083

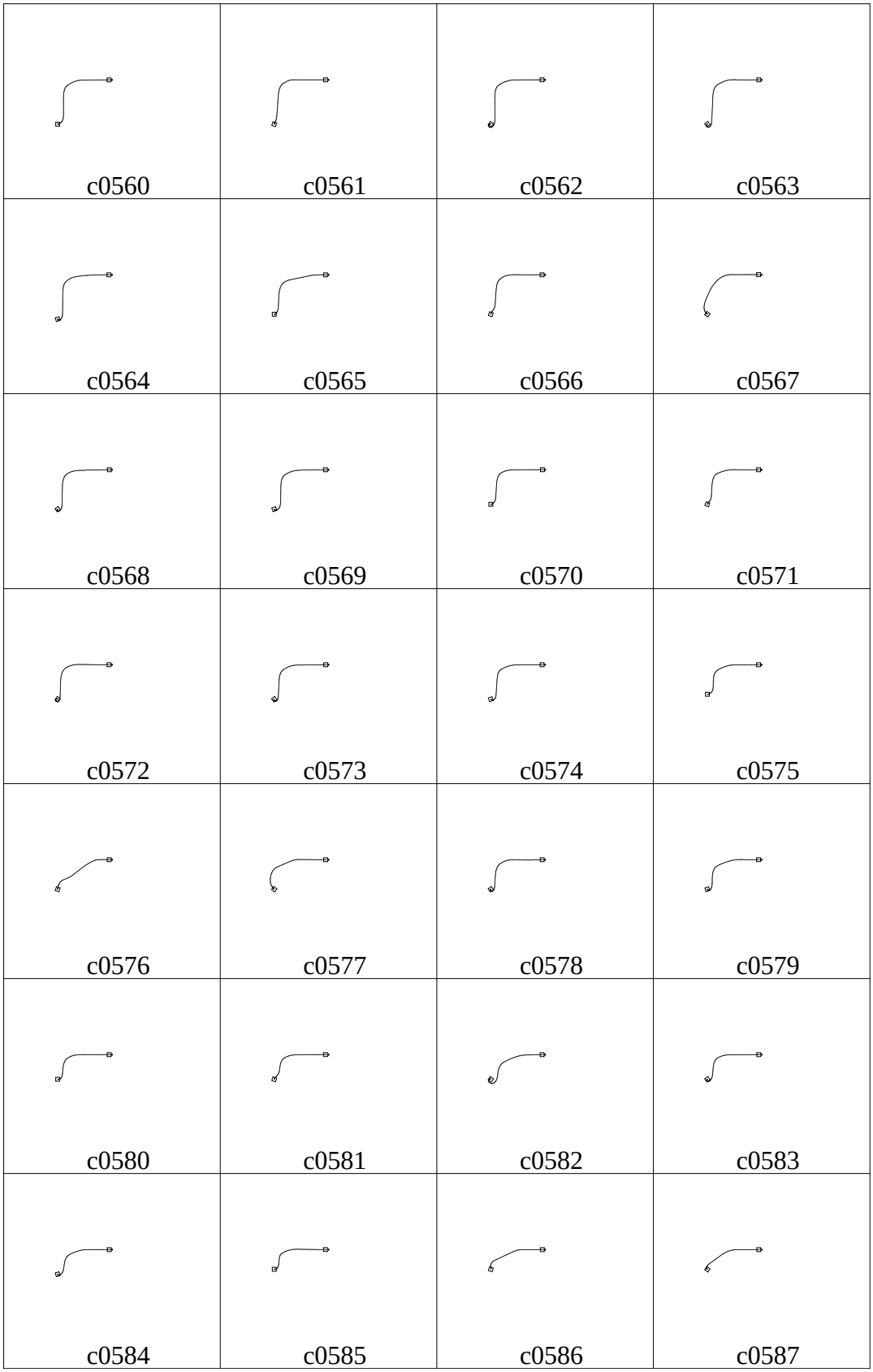


Figure 9.13: Table of Results c0560–c0587

c1148	c1149	c1150	c1151
c1152	c1153	c1154	 c1155
 c1156	 c1157	 c1158	 c1159
 c1160	 c1161	 c1162	 c1163
 c1164	 c1165	 c1166	 c1167
 c1168	 c1169	 c1170	 c1171
 c1172	 c1173	 c1174	 c1175

Figure 9.14: Table of Results c1148–c1175

c2072	c2073	c2074	c2075
c2076	c2077	c2078	c2079
c2080	c2081	c2082	c2083
c2084	c2085	c2086	c2087
c2088	c2089	c2090	c2091
c2092	c2093	c2094	c2095
c2096	c2097	c2098	c2099

Figure 9.15: Table of Results c2072–c2099

 c2324	 c2325	 c2326	 c2327
 c2328	 c2329	 c2330	 c2331
 c2332	 c2333	 c2334	 c2335
 c2336	 c2337	 c2338	 c2339
 c2340	 c2341	 c2342	 c2343
 c2344	 c2345	 c2346	 c2347
 c2348	 c2349	c2350	 c2351

Figure 9.16: Table of Results c2324–c2351

 c2968	 c2969	 c2970	 c2971
 c2972	 c2973	 c2974	 c2975
 c2976	 c2977	 c2978	 c2979
 c2980	 c2981	 c2982	 c2983
 c2984	 c2985	 c2986	 c2987
 c2988	 c2989	 c2990	 c2991
 c2992	 c2993	 c2994	 c2995

Figure 9.17: Table of Results c2968–c2995

 c3864	 c3865	 c3866	 c3867
 c3868	 c3869	 c3870	 c3871
 c3872	 c3873	 c3874	 c3875
 c3876	 c3877	 c3878	 c3879
 c3880	 c3881	 c3882	 c3883
 c3884	 c3885	 c3886	 c3887
 c3888	 c3889	 c3890	 c3891

Figure 9.18: Table of Results c3864–c3891

Chapter 10

Evolving Velocity Profiles

Substantial parts of this chapter were published in the IEEE proceedings of the International Symposium on Computational Intelligence in Robotics and Automation, Kobe, Japan, [58]. A different approach is taken from the previous chapters. The fuzzy rule base is developed by evolving robot velocity profiles and combining the resultant input/output pairs using neighbourhood clustering. The results presented in this chapter are through simulation as the resulting controller was incomplete.

10.1 Introduction

An important aspect of fuzzy logic application is the determination of a fuzzy logic Knowledge Base (KB) to satisfactorily control the specified system, whether this is derivable from an appropriate mathematical model or just from system input-output data. There are two main problems. The first is to obtain an adequate knowledge base for the controller and second is that of selection of key parameters defined in the method.

The three input, two output fuzzy system presented in [50] and the five input, two output fuzzy system [86] suffered from a uncooperative grouping of terms in the objective function. The time term was needed to minimise path lengths, however, this usually interfered with other terms. The effects of time could not be adjusted by a multiplying factor. The 'curse of dimensionality' also prevented obtaining a complete fuzzy controller from all initial configurations. An estimated twenty-five sets per input variable (25^5) are needed to achieve the required granularity of a fuzzy controller to control the system. A three layer hierarchical fuzzy system will require $25^3 \times 2(25 \times 9^2)$ rules to control the system.

The issues mentioned can be addressed separately by decomposing the system into smaller parts. Individual optimal paths can be found by evolving velocity profiles. The optimised paths can be combined by a clustering technique.

Evolutionary learning of a knowledge base of fuzzy controllers over a set of initial configurations has been obtained using a *fuzzy amalgamation* process that fuses KB's for each initial configuration or by developing an evolutionary algorithm to learn directly the KB by itself over the region of initial configurations, [69].

In this research we still incorporate special attributes to cover the difficult cases for control when the robot is close and touching the ball. A relative coordinate system is used and terms are introduced into the fitness evaluations which allow both forward and reverse motion of the soccer robot.

The kinematics of a robot have been discussed in Chapter 8. The equations used are: Equation 5.10, Equation 5.12, and Equation 5.13.

Relative coordinates for a five input fuzzy control systems are used as discussed

in Chapter 9. Wheel lift constraint 5.4.4 is used in learning the velocity profiles of the robots. Skidding of the wheels on the playing surface limits the amount of motor torque that can be translated into linear propulsion. The maximum acceleration as determined in Section 5.2 is used.

10.2 Evolutionary Learning of Paths

By using a relative coordinate system relative to the ball position, the ball can be considered as a stationary object and therefore the end configuration becomes a stationary point behind the ball. The desired position of the robot at the end of the path is $(x_{DP}, y_{DP}) = (688.65, 650)$ when the ball is at $(x_B, y_B) = (750, 650)$.

To find paths directly we use an evolutionary algorithm [27] to evolve a velocity profile that takes the robot from each initial starting configuration to the end configuration. The length of the string K is allowed to vary from between one velocity pair to a maximum of M velocity pairs. As time corresponds to position in the string, time is allowed to vary from $t = 0$ to $t = t_{K-1} \leq t_{M-1}$.

String lengths are initialised with sufficiently large $2M = 300$ to enable velocity paths to reach the destination point early in the evolutionary process.

An individual string $\underset{\sim}{s}$ contains $s_1^k = \Delta v_L(t)$ and $s_2^k = \Delta v_R(t)$ pairs in the form:

$$\underset{\sim}{s} = \{s_1^0, s_2^0, \cdots, s_1^k, s_2^k, \cdots, s_1^{K-1}, s_2^{K-1}\},$$

where $s_j^k, j = 1, 2$ are integers in the interval $[-4, 4]$, corresponding to each $t^k = k\Delta t$.

Change in wheel velocity, Δv, has been set to the range $[-4, 4]$, determined experimentally in Section 5.2 for our physical robots, and are in motor control units (PWM). Velocity is proportional to angular speed of the armature in a DC motor [87]. Pulse Width Modulation (PWM) averaged over the pulse width gives a proportional decrease in voltage.

A path is constructed from time $t = 0$ at the initial starting configuration until:

1. The robot position exceeds maximum distance from ball,
2. The robot collides with the ball,
3. Time reaches the maximum time t_M allowable, or
4. A constraint is broken.

The velocity is updated by the Equations 10.1.

$$\begin{aligned} v_L(t+1) &= v_L(t) + \Delta v_L(t) \\ v_R(t+1) &= v_R(t) + \Delta v_R(t) \end{aligned} \tag{10.1}$$

for $t = 0, \cdots, t_{K-1}$.

As in previous chapters, the physical size of the robot is considered in a collision. Scribing the path is stopped when the flag "HitBall" is true.

Fitness evaluation of each individual was calculated by scribing a path using the velocity pairs in the string until one of the stopping configurations were met.

A grid of initial configurations was defined as (x, y, θ, v_L, v_R) according to the Equations 10.2 (excluding the ball position), with sixty-one had selected velocity

pairs.

$$\begin{aligned} x &= 350 + 100(k-1) \;\text{ for }\; k = 1, \cdots, 9 \\ y &= 250 + 100(k-1) \;\text{ for }\; k = 1, \cdots, 9 \\ \theta &= 2(k-1)\pi/9 \;\text{ for }\; k = 1, \cdots, 9 \end{aligned} \tag{10.2}$$

The sixty-one hand selected velocity pairs are defined within the wheel lift constraint. The sixty-one velocity pairs are chosen as a grid of points avoiding the wheel lift constraint and with velocity pairs close to the wheel lift constraint boundary. These selected velocity pairs (v_L^0, v_R^0) are: (0,0), (27,0), (27,27), (0,27), (-27,27), (-27,0), (-27,-27), (0,-27), (27,-27), (54,0), (54,27), (54,54), (27,54), (0,54), (-27,54), (-54,54), (-54,27), (-54,0), (-54,-27), (-54,-54), (-27,-54), (0,-54), (27,-54), (54,-54), (54,-27), (78,56), (78,78), (56,78), (-56,78), (-78,78), (-78,56), (-78,-56), (-78,-78), (-56,-78), (56,-78), (78,-78), (78,-56), (103,88), (103,103), (88,103), (-88,103), (-103,103), (-103,88), (-103,-88), (-103,-103), (-88,-103), (88,-103), (103,-103), (103,-88), (127,115), (127,127), (115,127), (-115,127), (-127,127), (-127,115), (-127,-115), (-127,-127), (-115,-127), (115,-127), (127,-127), (127,-115) In total, the number of initial configurations $C = (9^2 - 1) \times 9 \times 61 = 43920$. A large number of initial configurations are used to enable poor performing paths to be deleted and have enough information to build the fuzzy KB.

The population at generation t, $P(t) = \{\underset{\sim}{s}^n \;:\; n = 1, \cdots, N\}$, where $N = 500$ individuals in the population. The next generation population $P(t+1)$ was built using full replacement policy, tournament selection with $n_T = 4$, and one point crossover with probability $p_C = 0.6$. Elitism was not used as it has not been successful in previous simulations. The mutation operator was a random change to an gene as $\text{int}(4 * \text{random}()) - 2 \in [-2, 2]$ with wrap-around. Wrap around is calculated by:

$$\begin{aligned} \text{gene} &= \text{gene} + \text{random}(-2, 2) \\ \text{gene} &= \text{gene} + 9, \;\; \text{if (gene} < -4) \end{aligned}$$

$$\text{gene} \quad = \quad \text{gene} - 9, \quad \text{if (gene} > 4)$$

The simple one point crossover was used in its standard form. Parents were crossed if the random cross probability was less than $p_C = 0.6$. The crossing point was a random position in the range $1, \cdots, M-1$. No adjustment was provided for that fact that the strings are of varying length up to a maximum length of M. One would expect that the standard one point crossover would introduce random data at the end or mid-position of the children during the early stages of the run. This is also the normal behaviour of an evolutionary algorithms with fixed length strings. As the run matures, an optimal velocity profile string emerges and becomes dominant. As the dominant string propagates through the population, strings become a fixed length path for an optimised solution to the problem. If the mature string is of short length, the effectiveness of producing new children by crossover decreases as crossing location is being chosen beyond the solution space at a rate of $p_C = 0.6 \times K/M$. No adverse effects of p_C degeneration were observed in the simulations.

Considerable effort to ensure success of the evolutionary algorithm was made in developing:

1. Mutation schedule and
2. Appropriate time to implement variables in the objective function to encourage paths to lengthen before optimising.

A linearly decreasing mutation schedule was used so that the mutation operator had a chance of changing genes in small length strings early in the evolutionary

process. The mutation schedule used is:

$$p_m = \begin{cases} 0.224875 - 0.000024875 \times \text{gen} & \text{if gen} < 9000, \\ 0.001 & \text{otherwise.} \end{cases}$$

where: gen is the generation count.

Initially the most stubborn string with a length of $t = 1$ will be mutated on average once in four to five strings with $p_m = 0.224875$. The high mutation usually helps strings to reach the destination target within 1000 generations. As the mutation schedule decreases, the evolutionary algorithm settles on suitable strings that best reach the target position with forward or reverse facing end point and $w = 0$ in minimum time. Stable long term mutation occurs when mutation is changing one gene in four or five strings. As the mutation falls below this value, the solution appears to be converged and shows no noticeable changes for the remainder of the run. Any affects of the mutation beyond this point are small changes that are no longer seen on the graphics.

The fitness of each individual is a sum of terms evaluated at the termination of scribing a robot path as shown in Equation 10.3.

$$f_i = \alpha_1 T_1 + \alpha_2 T_2 + \alpha_3 T_3 + \alpha_4 T_4 + \alpha_5 T_5 \tag{10.3}$$

where: α_k for $k = 1, \cdots, 5$ are positive constants.

Term $T_1 = d^2(R, DP)$ is the distance of the final robot position R and the destination point $DP = (688.5, 650)$ behind the ball squared.

Term $T_2 = |\sin(\phi)|$ where ϕ is the final angle of the robot to the $\overline{BG}$ line. This allows minimisation for both forward and reverse facing robot solutions.

Term T_3 is the count of time intervals used to scribe the path until some termi-

nating condition as described by Equation 10.4. The count could not be used directly as it caused premature convergence to short path lengths. The path needs to grow initially before optimisation causes the path to shrink.

$$T_3 = \begin{cases} \text{count} & \text{if } d^2 < 100 \text{ and } |\sin(\phi)| < 0.15, \\ M & otherwise. \end{cases} \tag{10.4}$$

Term T_4 is used to minimise the angular velocity of the robot towards zero when close to the ball as shown in Equation 10.5. This term only makes sense when applied to paths ending near to the destination point.

$$T_4 = \begin{cases} |v_L - v_R| & \text{if } d^2 < 100 \text{ and } |\sin(\phi)| < 0.15, \\ 254 & otherwise. \end{cases} \tag{10.5}$$

Term T_5 is a penalty value of $1.0e6$ applied when the wheel constraint is broken and also terminates scribing the path. It is a heavy penalty applied to discourage any wheel lift in a path.

Typical values of α_k used for the following results were: $\alpha_1 = 0.6$, $\alpha_2 = 150.0$, $\alpha_3 = 1.0$, $\alpha_4 = 0.5$ and $\alpha_5 = 1.0$.

The evolution of each path is terminated when the number of generations has reached a maximum prescribed value. The individual with the best fitness value is considered to be the best velocity profile from a particular initial configuration.

10.3 Discussion

Paths are declared satisfactory that have the following attributes:

- Plotting of the path appears smooth,

- Can contain a cusp near to initial configuration or final configuration of the robot in a path,
- Should not have unnecessary looping or cusps in the middle of a path.

The path should:

- End close to the destination point,
- End with angle to the $\overline{BG}$ near to 0 rad or π rad,
- End with angular velocity w close to zero so that the robot can chase the ball after impact,
- Not break wheel lift constraint, and
- Be minimum time to destination point DP.

Many of the paths were found to be satisfactory. However, some paths broke the wheel lift constraint in the first step. These paths were on the outermost positions of the initial configurations with high velocity towards the boundary. The allowable acceleration limit of $[-4, 4]$ (PWM) did not allow enough deceleration for the robot to avoid hitting the boundary. A sub-optimal velocity profile producing a path with wheel constraint broken and shortest distance to the destination point dominated the population.

Similarly, robots starting configurations close to the ball with high velocity towards the ball did not have enough time to decelerate and turn before hitting the ball.

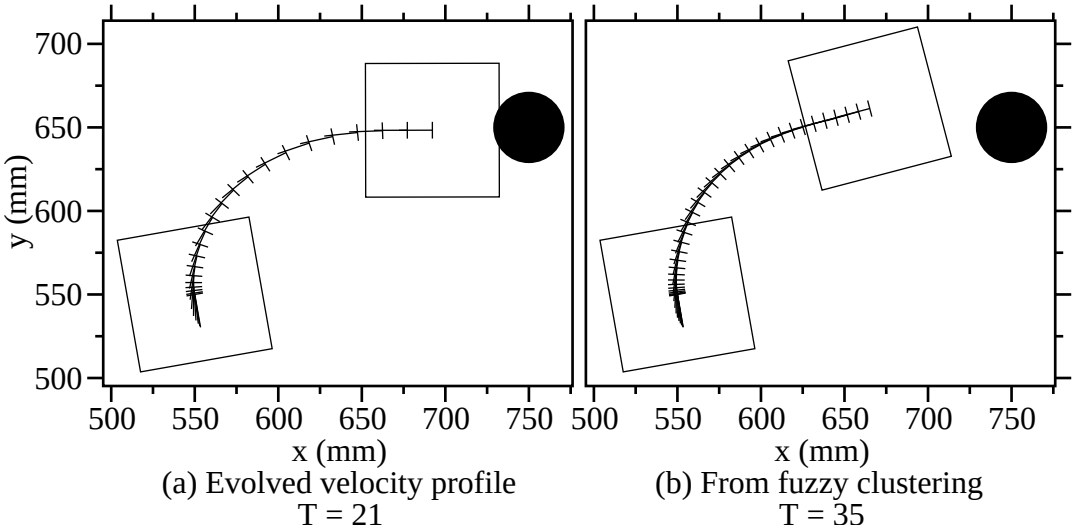
Paths with final robot position to ball distance $d_{BR} \geq 25$ or poor final angle $|\sin(\phi_R)| \geq 0.175$ were classed as unsuitable and removed (stripped) from the solution file. This process deleted 15789 out of 41040 paths leaving 61.5% good paths that were confirmed using a graphical windows programme. These raw paths are then used to build a fuzzy KB to control the robot. The time step used in the simulation was: $\Delta T = 1/60$ seconds. Time for path is calculated from $T \times \Delta T$ from each figure. Some of these paths are shown in Figures: 10.1(a), 10.2(a), 10.4(a), 10.5(a), 10.6(a), 10.7(a), 10.8(a), 10.9(a), 10.10(a), and 10.11(a). Each figure identifies the initial configuration as c *configuration*$(x_R, y_R, \phi_R, v_L, v_R)$. Part (a) of the figures are the scribed paths from evolving the robot velocity profile. Part (b) of the figures are the scribed paths obtained from using the clustered centres in the fuzzy controller.

10.4 Fuzzy Clustering

As a first approach, a neighbourhood clustering [8] method is used to combine the good raw path information to form the fuzzy KB. A description of the algorithm follows.

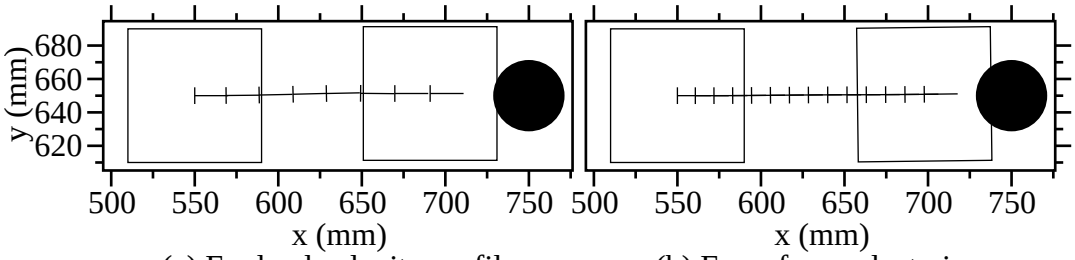
The objective is to develop a universe of fuzzy sets using Mamdani product inference, Gaussian fuzzifier and centre average defuzzification. This type of fuzzy system is described by Equation 4.10.

We have N input-output pairs $(\underset{\sim}{x}^{\ell}, \underset{\sim}{y}^{\ell})$, $\ell = 1, \cdots, N$. These input-output pairs are to be clustered to $\underset{\sim}{x}_C^k$, $k = 1, \cdots, M$ using neighbourhood clustering.
The process for nearest neighbourhood clustering is as follows:



(a) Evolved velocity profile T = 21 (b) From fuzzy clustering T = 35

Figure 10.1: Path Plots from Configuration c08400(550, 550, $7\pi/9$, 0, 0)



(a) Evolved velocity profile T = 7 (b) From fuzzy clustering T = 13

Figure 10.2: Path Plots from Configuration c08490(550, 650, 0, 54, 54)

Step 1: $\ell = 1$, $M = 1$, and Select radius r.

Step 2: $\underset{\sim}{x}_C^M = \underset{\sim}{x}^\ell$, $A^M = \underset{\sim}{y}^\ell$, $B^M = 1$.

Step 3: $\ell = \ell + 1$.

Step 4: Find $\min_{j=1}^{M} |\underset{\sim}{x}^\ell - \underset{\sim}{x}_C^j|$.

Step 5: If $|\underset{\sim}{x}^\ell - \underset{\sim}{x}_C^j| > r$, Then $M = M + 1$, $\underset{\sim}{x}_C^M = \underset{\sim}{x}^\ell$, $A^M = \underset{\sim}{y}^\ell$, $B^M = 1$.

Else $A^j = A^j + \underset{\sim}{y}^\ell$, $B^j = B^j + 1$.

Step 6: While $\ell < N$, Goto Step 3

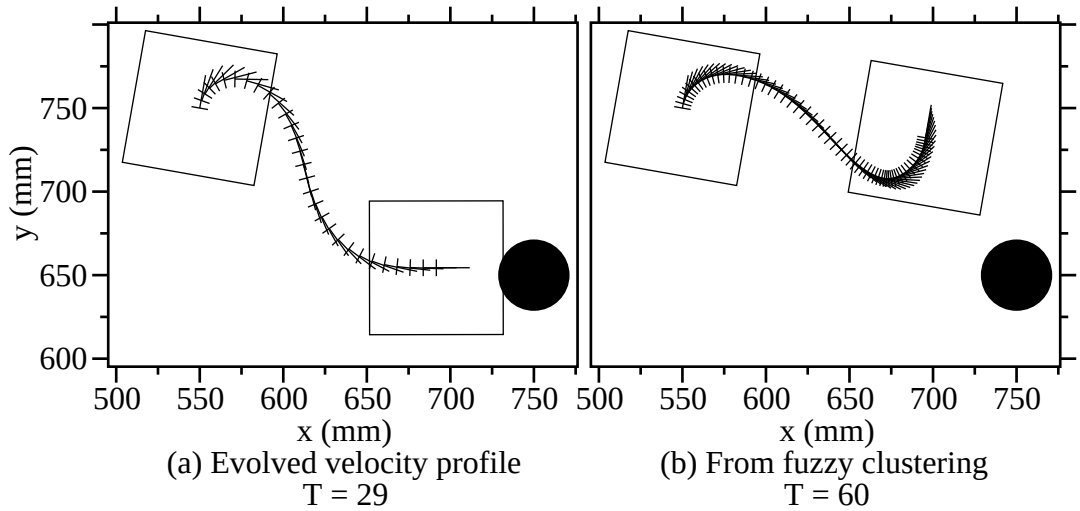
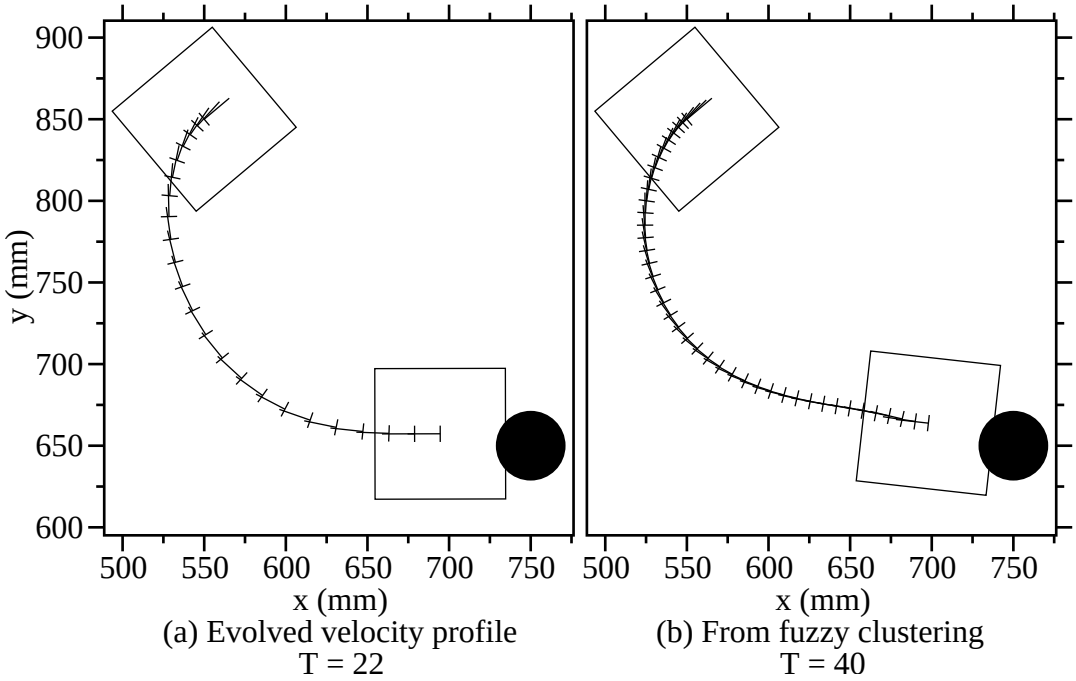


Figure 10.3: Path Plots from Configuration c08900(550, 750, $4\pi/9$, 27, 0)

Obtaining a fuzzy controller with the required number of membership sets is determined by the radius r. A large r results in a small number of membership sets with coarse control. A small r results in a large number of membership sets with fine control. The best radius gives a reasonable number of memberships that provide adequate control. The smoothness of the transition between fuzzy membership sets is controlled by σ. A large σ gives smooth transition between fuzzy memberships and reduces the amount of $NULL$ fuzzy sets for some untrained input. Fine control is given with small σ, however, may allow $NULL$ fuzzy memberships to appear for untrained inputs.

A few results are shown in Figures: 10.1(b), 10.2(b), 10.3(b), 10.4(b), 10.5(b), 10.6(b), 10.7(b), 10.8(b), 10.9(b), 10.10(b), and 10.11(b).

Figure 10.1(a) shows the optimised robot path from configuration c08400(550, 550, $7\pi/9$, 0, 0). The robot accelerated smoothly to the destination point in $21/60 = 0.35$s along a smooth curved path. Figure 10.1(b) shows the path from the same configuration using the clustered fuzzy controller. The robot accelerated to a constant medium velocity that started following the optimum trajectory



(a) Evolved velocity profile
T = 22

(b) From fuzzy clustering
T = 40

Figure 10.4: Path Plots from Configuration c09187(550, 850, $2\pi/9$, -27, 0)

and slowly diverged. The path stopped when it entered an unlearnt region in $35/60 = 0.58$s.

Figure 10.2(a) shows the optimised path from configuration c08490(550, 650, 0, 54, 54). The robot reached the destination point in $7/60 = 0.12$s along a straight path. Figure 10.2(b) shows the path using the clustered fuzzy controller. The fuzzy controller reached the destination point in a straight line in $13/60 = 0.22$s.

Figure 10.3(a) shows the path from optimising the velocity profile from configuration c08900(550, 750, $4\pi/9$, 27, 0). The robot accelerates to a constant medium velocity along a smooth curve to the destination point in $29/60 = 0.48$s. Conversely, the same configuration using the clustered fuzzy controller entered an unlearnt region and was stopped. The time for the robot to reach this position was $60/60 = 1$s.

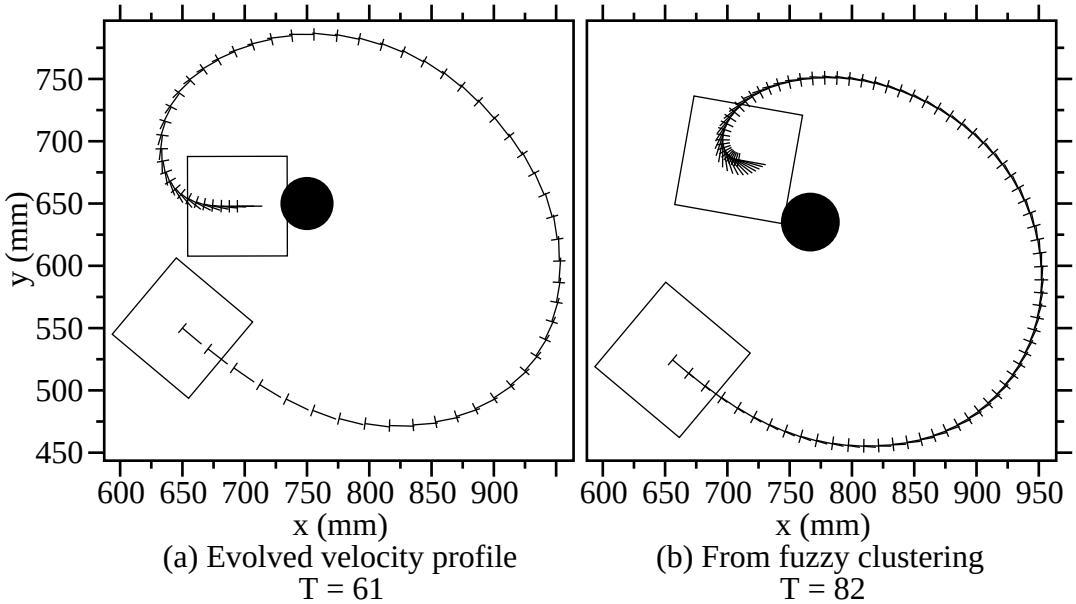


Figure 10.5: Path Plots from Configuration c11587(650, 550, $16\pi/9$, 78, 78)

Figure 10.4(a) shows a path from optimised velocity profile from configuration c09187(550, 850, $2\pi/9$, -27, 0). The robot accelerated smoothly to a high speed impact the the ball in $22/60 = 0.37$s. In comparison, the clustered fuzzy controller accelerated to a medium velocity along a similar curved path. The robot came close to the destination point at an non-optimum angle in $40/60 = 0.67$s.

Figure 10.5 shows paths from configuration c11587(650, 550, $16\pi/9$, 78, 78). The path found by optimising the velocity profile is shown in part (a) of the figure. The robot decelerates smoothly along a smooth path to the destination point in $61/60 = 1.0$s. In comparison, part (b) shows the path from the clustered fuzzy controller. The robot decelerated along a smooth curved path similar to part (a) of the figure, but entered an unlearnt region and was stopped. The time taken by the robot in part (b) took $82/60 = 1.4$s.

Figure 10.6(a) shows the path from configuration c14587(750, 550, $4\pi/3$, 103, -103) using the optimised velocity profile. The robot started from a pirouette,

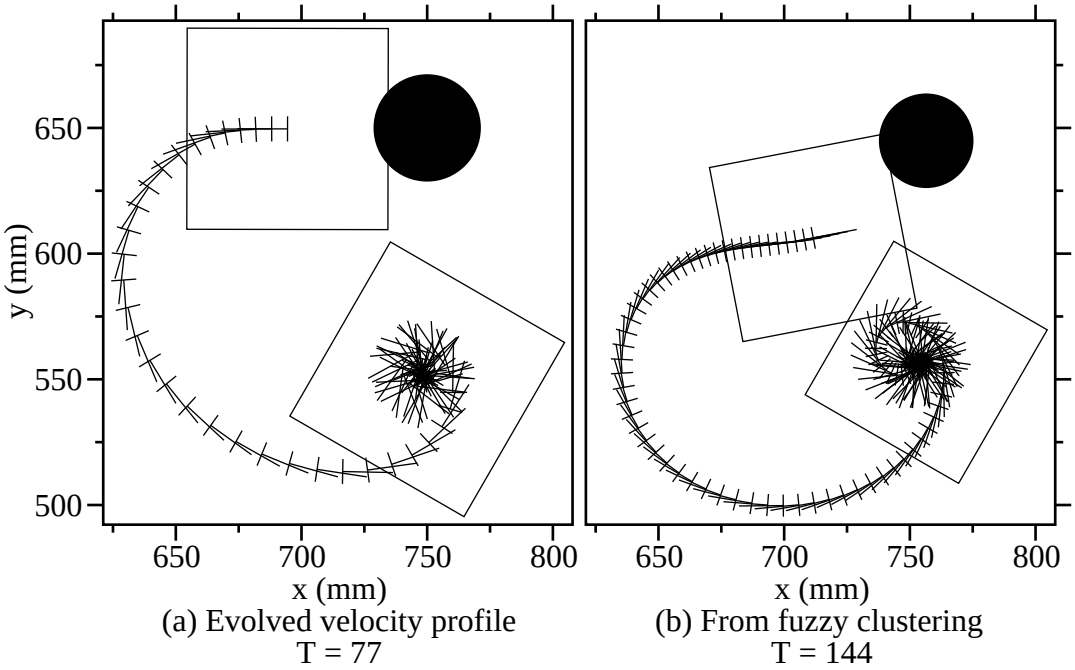


Figure 10.6: Path Plots from Configuration c14587(750, 550, $4\pi/3$, 103, -103)

slowed the spin and followed a smooth curve to the destination point in $77/60 = 1.3$s. In comparison, part (b) of the figure is the path from the clustered fuzzy controller. The path starts from the same pirouette and slows the spin. The exit of the spin enters a tight loop before following the smooth curve around to the ball. The robot finishes some distance from the destination point at a non-optimal angle to hit the ball to the goal. The time for the robot to scribe this path is $144/60 = 2.4$s.

Figure 10.7 shows the respective paths from configuration c14687(750, 750, 0, 56, 78). Part (a) of the figure shows the path from optimising the velocity profile. The robot takes a smooth curve to the destination point in $53/60 = 0.88$s. The path in part (b) enters a region that is not covered by a fuzzy set and is stopped prematurely.

Figure 10.8 show the paths from configuration c16887(850, 450, $2\pi/3$, -78, 56).

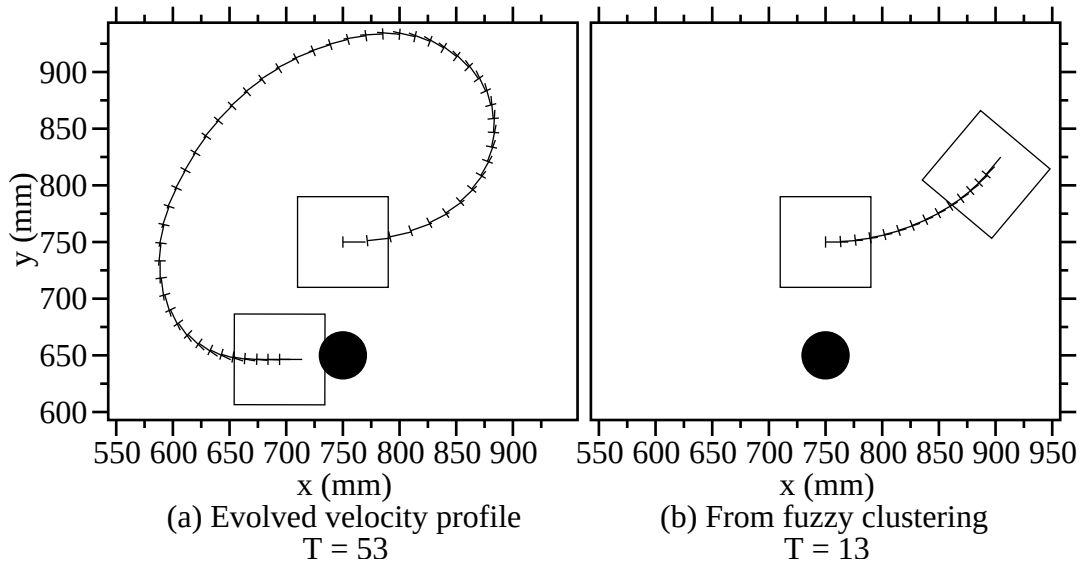


Figure 10.7: Path Plots from Configuration c14687(750, 750, 0, 56, 78)

Path (a) quickly exits the extremely tight curve to follow a smooth path to the destination point in $72/60 = 1.2$s. In comparison, the clustered fuzzy controller follows a similar path at medium velocity. The robot gets close to the destination point, but at a poor angle of attack. The time for the robot to scribe this path is $99/60 = 1.7$s.

Figure 10.9(a) shows the robot path scribed from using the evolved velocity profile from configuration c17087(850, 550, 0, -27, 54). The robot follows a smooth path around to the destination point in $43/60 = 0.72$s. In comparison, the path in part (b) of the figure has not follow the curve as tightly and hit the robot towards it's own goal.

Figure 10.10(a) shows the scribed robot path using the evolved velocity profile from configuration c17287(850, 550, $10\pi/9$, 54, 0). The robot followed a smooth curve to the destination point in $50/60 = 0.83$s. In comparison, the path scribed from using the clustered fuzzy controller (part (b)) contains unnecessary looping and takes a long time to reach the target configuration. The robot took $153/60 =$

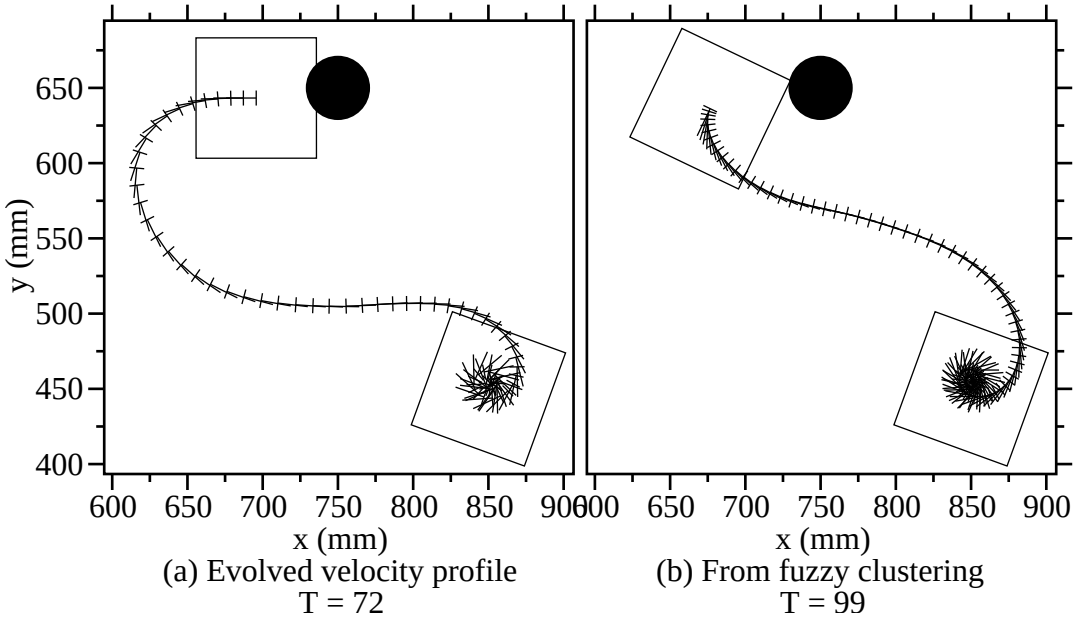


Figure 10.8: Path Plots from Configuration c16887(850, 450, $2\pi/3$, -78, 56)

2.6s to reach the target. The angle of the robot at impact will place the ball on a trajectory outside of the goal.

Figure 10.11(a) shows the scribed robot path using the evolved velocity profile from configuration c18099(850, 850, $2\pi/9$, -54, -27). The robot follows a smooth curve to the destination point in $32/60 = 0.53$s. Figure 10.11(b) shows the scribed robot path from the same initial configuration. The path produce from the fuzzy controller takes twice the time to teach the target and hits the ball at an unacceptable angle.

All paths from fuzzy clustering take more time than those used to build the fuzzy KB. No fuzzy controlled path impacts with the ball at a reasonable angle to enable the robot to continue along the current path and come close to the ball again. The fuzzy controller takes a bad turn in Figure 10.3(b) and finds a *NULL* set. The path is stopped at this point as the denominator of the fuzzy system becomes zero.

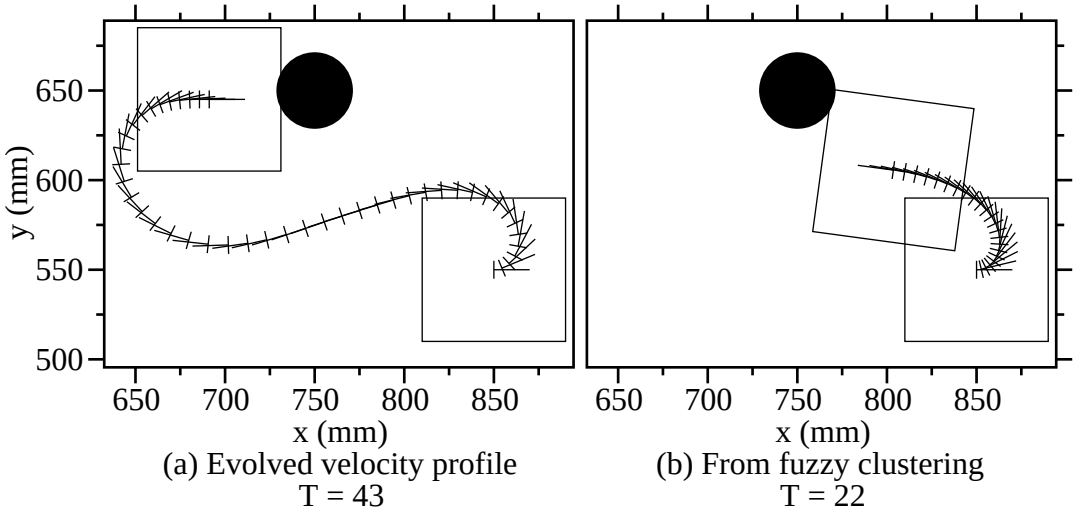
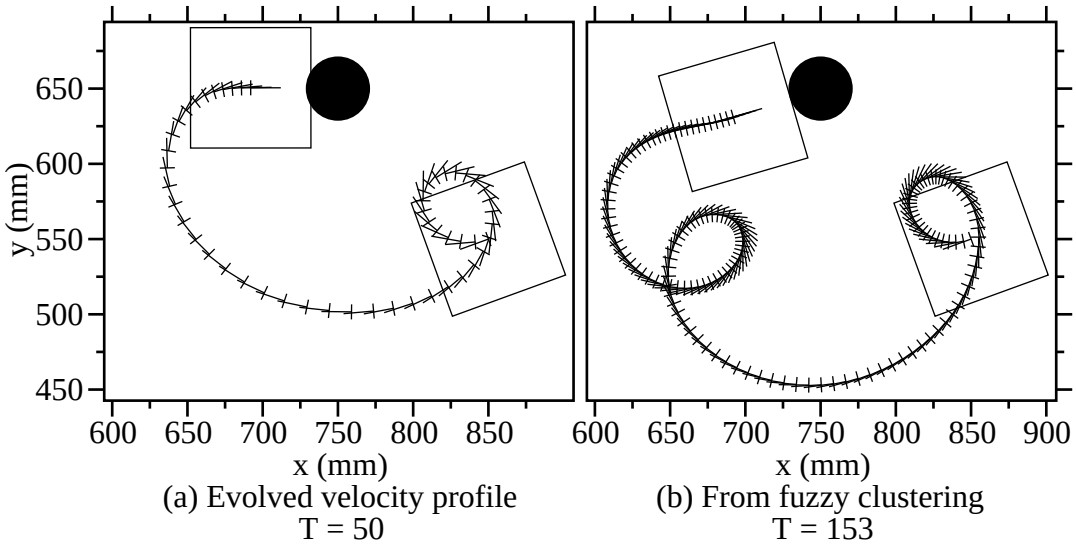


Figure 10.9: Path Plots from Configuration c17087(850, 550, 0, -27, 54)

10.5 Comments

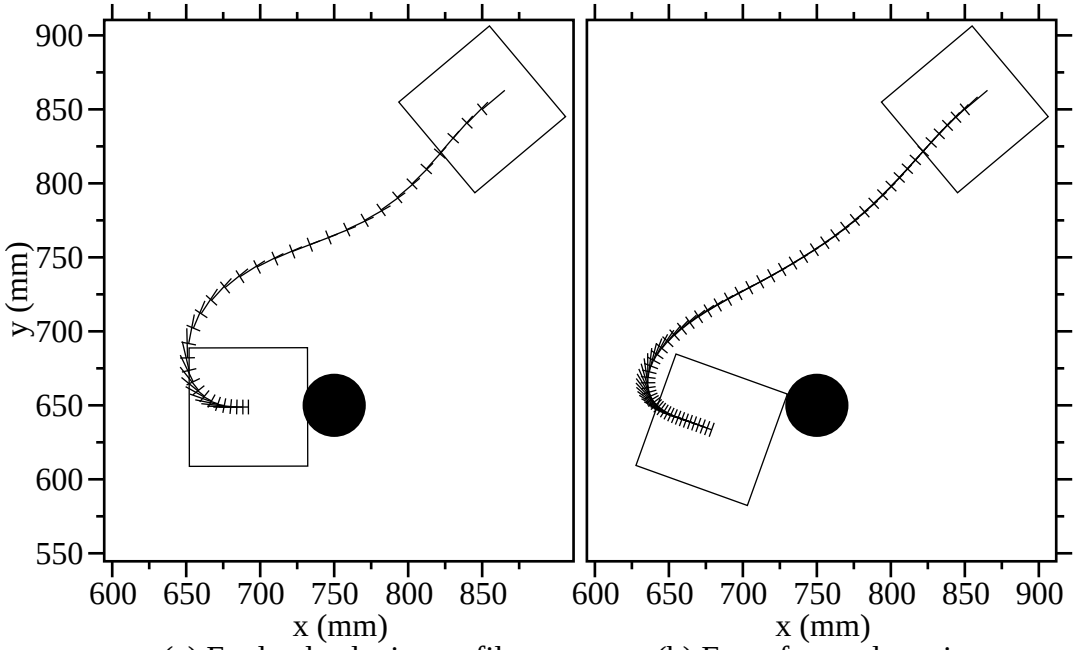
Although 43920 configurations were used to develop the fuzzy KB by clustering, there was not enough information to build a complete fuzzy controller. Using optimised paths helped to reduce the amount of information available for clustering as optimised paths avoid the end-position shown in Figure 10.3(b). To build a fuzzy controller by clustering, an extremely high number of input/output data is required that finely covers the space of the variables.



(a) Evolved velocity profile
T = 50

(b) From fuzzy clustering
T = 153

Figure 10.10: Path Plots from Configuration c17287(850, 550, $10\pi/9$, 54, 0)



(a) Evolved velocity profile
T = 32

(b) From fuzzy clustering
T = 60

Figure 10.11: Path Plots from Configuration c18099(850, 850, $2\pi/9$, -54, -27)

Chapter 11

Colour Space

Substantial parts of this chapter were published [9] in the conference proceedings: The Seventh International Conference on Control, Automation, Robotics and Vision (ICARCV'2002) in Singapore. Detection of colour patches and the ball are considered.

11.1 Introduction

In robot soccer, for example the Mirosot [1] game category of the Federation of International Robot Soccer Association (FIRA) [78], determination of a robot's position and orientation is achieved primarily by computer based image analysis of the colour identification patch on each robot using an overhead vision camera. Image enhancement is not used due to computational loading. Other manipulations and transformations of the captured images are usually not performed because they are also computationally intense. Colour recognition systems include YUV, RGB and HSI and colour separation is visualised in YUV, YIQ and HSI coordinate systems [25].

The vision systems available are as varied as the imagination of the researchers, most have used commercial image capture cards. Image capture cards differ widely in the information presented to the computer and allow or inhibit user control over programmable features on the card. They are capable of providing YUV and/or RGB with RGB derived from YUV through translation of the video composite signal. Every translation to a different colour coordinate system includes some translation error. The effect is to add more noise to a noisy signal. Noise is accumulated in RGB to YUV translation inside the camera, encoding and filtering for video composite, transmission through cables, decoding, A/D conversion at image capture card to name a few.

Some Korean teams [88, 89, 90] use the RGB colour representation. One such team [89] expressed problems in using RGB because of luminance sensitivity. An inherent component of RGB is luminance for luminance changes over the field area and different lighting at venues cause the system to miss classify the position of the robots. Some teams [91, 93, 92] translated the RGB output of the image capture card to another colour coordinate system. Two teams [91, 93] converted RGB to HSI for colour detection. Another team [92] developed a colour system similar to HSI, translated from the RGB output of the image capture card. Still other teams [94, 95] placed LED light sources in an attempt to overcome glare and colour identification matching problems.

A growing vision processing technique is by an in-line processor between the camera and computer. Newton Labs [96] developed the "Cognachrome" vision processing system. The Cognachrome was superior to image capture cards because it processed images at a rate of 60 fields per second. The image capture cards are limited by the operating system of the computer to significantly lower the rate that images could be processed.

This first Cognachrome system was limited to four colours and the ball. Processing of each interlaced frame at full rate of 60 frames per second. The "Cognachrome" system was sold and purchased by other teams [97]. In-line systems became an advantage due to problems caused by the event-based multi-tasking Windows operating system which was not designed for real time processing.

In the next section we discuss the YUV colour recognition system, then examine difficulties in colour detection of the ball. We then define the pie slice decision region method for colour detection in the UV colour map. Experimental observations are given.

11.2 YUV

The exemplar software purchased with the Micro-Adventure soccer robots used rectangular decision regions, with a choice of YUV or UV for colour detection.

Equation 11.1 shows the equation used to convert from the RGB to YUV coordinate system [98].

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ -0.147 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{11.1}$$

The mapping of RGB to YUV coordinate system results in the RGB cube begin mapped to a distorted cube with white and black at (0, 0) in UV coordinates. The Y axis is in line with the black to white diagonal of the cube. Table 11.1 shows the positions of the colours in UV coordinate system.

Figure 11.1 shows the mapping of RGB colours on UV coordinates.

Table 11.1: RGB Colour Mapping onto YUV Coordinates

	R	G	B	Y	U	V	UV polar
Red	255	0	0	76	-37	157	$161\angle 167°$
Green	0	255	0	150	-74	-131	$151\angle -151°$
Blue	0	0	255	29	111	-26	$114\angle -77°$
Yellow	255	255	0	226	-111	26	$114\angle 103°$
Magenta	255	0	255	105	74	131	$151\angle 29°$
Cyan	0	255	255	179	38	-157	$161\angle -14°$

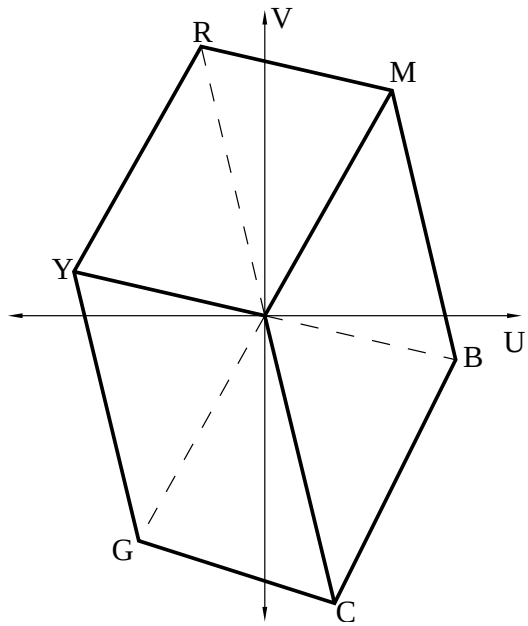


Figure 11.1: RGB Colours Mapped on UV Coordinates

The luminance (Y) part of the YUV can be ignored if the only interest is in hue and saturation of the colours, as in the case of robot soccer. It is convenient to make use of the YUV output of composite video:

- Because there are six colours on the perimeter of the chrominance/hue diagram this means that YUV components are a little more separable than HSI colour components and should allow easier separation of colours. The more saturated the colour, the better the classification into decision regions.

- Using the YUV output of a video camera minimises the accumulation of video error and of conversion to another system such as RGB or HSI.

11.3 Colour Detection of the Ball

The orange golf ball is the hardest object to correctly identify and track. It's dimpled surface causes glare on parts of the ball regardless of the angle of incident light. The ball consists of many colours ranging from pink through orange to yellow. The dimpled surface separates these colours into small blotches.

The cells of a CCD video camera are arranged in a rectangular grid and averaging occurs when two or more colours are incident on a single CCD cell. Figure 11.2 shows the averaged pixels on the boundary between red and yellow. The averaged pixels contain all of the colours along the line connecting red and yellow.

R	R	R	R	R	R	R	R	R	R
R	R	R	R	R	R	R	R	R	R
R	R	R	R	R	R	R	R	R	R
R	R	R	R	R	R	R	R	R	R
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Figure 11.2: Averaged Pixels caused by Red-Yellow Boundary

Figure 11.3 shows the averaged bi-colour distribution appearing on the UV colour map. All colours along the line connecting red and yellow can appear in the image at the red-yellow boundary. One of the colours on this line is orange—the colour

of the ball.

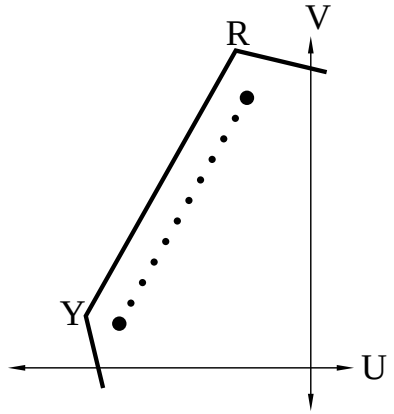


Figure 11.3: Averaged Colours on UV Map

Typical colour identification involves gathering pixels of the same colour into blobs—"blob construction". The many colours on the ball under different lighting conditions can cause blob construction problems, such as identifying two blobs over the ball, or not being able to identify the ball at all. Further, in analysing the colour of the ball, it is typical to select a patch over the ball and then determine a rectangular parallelepiped in the RGB cube which contains the colours in the patch, see Figure 11.4. To make this parallelepiped large enough to recognise the ball results in enveloping neighbouring colours such as pink, red and yellow. This then creates misclassification problems such as one or even more robots being classified as the ball.

11.4 Pie Slice Decision Regions

The robot soccer vision is very dependent on the assigned colour of the robot colour patches. Determination of these colours by rectangular decision regions is constrictive, as any incident luminosity changes and glare from incorrect field

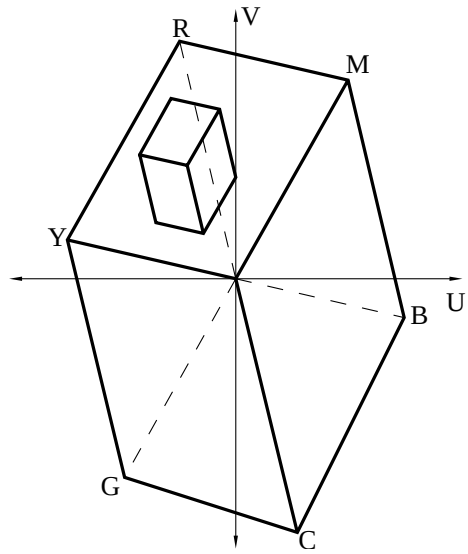


Figure 11.4: RGB Rectangle in YUV Space

lighting produce colour deviations beyond rectangular decision regions.

A system should be insensitive to colour drift and glare. Temperature variations of the incident light causes hue drift which rotates the colour around the Y-axis. Saturation of a colour (in YUV) is the distance of the point from the Y-axis. Changes in saturation of the object's colour is caused by variation of the incident light. Glare is produced by reflecting the major component of the incident light into the camera lens. It is a major problem in the detection of colour in mobile objects.

Two methods came with the Micro Adventure system for colour classification. The first enabled selection of a rectangular parallelepiped in YUV space and the second the selection of a rectangle in UV space. considering hue changes as rotating around the Y axis. Figure 11.5 illustrates a rectangular decision region taken in the UV plane. The larger it is made, the more it will include pink and yellow.

Our objective is to produce a robust colour definition ignoring the luminance component of the YUV signal, and by separating the colour decision regions as far as possible to minimise hue and saturation drift. The proposed methodology described below does not eliminate the effect of glare, but reduces it considerably.

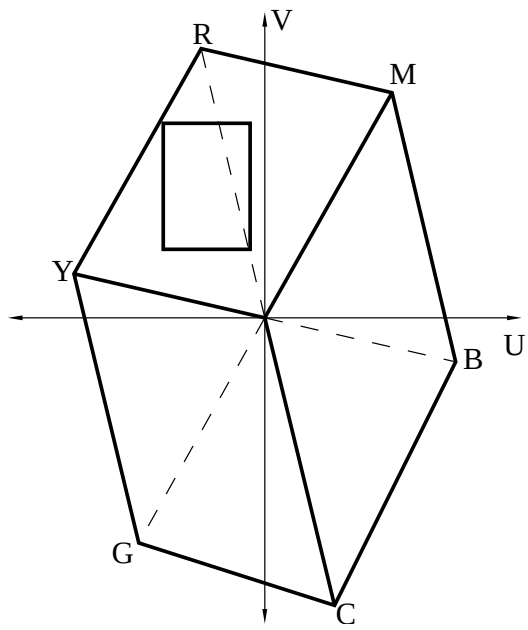


Figure 11.5: UV Rectangle over YUV

A more suitable decision region is a pie segment radiating from the origin, see Figure 11.6. Calculation of this segment would become prohibitive without the use of a Look-Up-Table.

The Look-Up-Table is constructed by assigning a number to each colour and filling the region of the table using rules of the following form:

$$\textbf{If } ((\phi_{UV} > \phi_1) \;\&\&\; (\phi_{UV} \leq \phi_2) \;\&\&\; (dist_{UV} > r)) \textbf{ Then } (\text{colour} = 1).$$

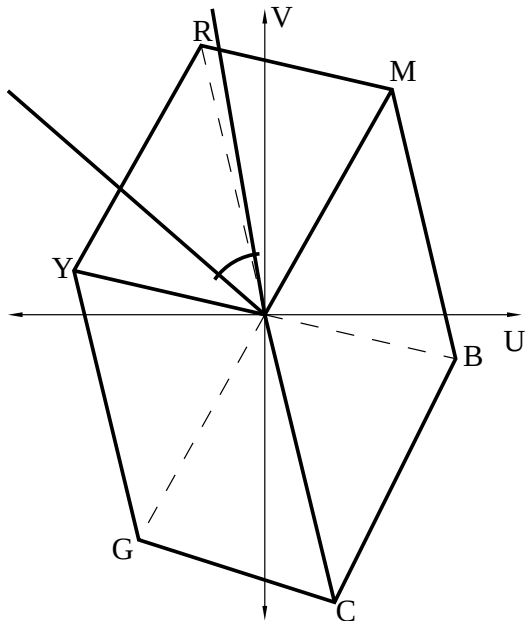


Figure 11.6: Pie Segment

Here, ϕ_{UV} is the angle of the chrominance point anti-clockwise from U axis, ϕ_1 is the first angle of colour region, ϕ_2 is the second angle of colour region, $dist_{UV}$ is the distance of the chrominance point from the origin, and r is the radius of exclusion arc removing white-grey-black from colour decision region.

All pixels in each interlaced frame are converted to a UV colour map. For example, Figure 11.7 shows the position of the ball in the left window, and the UV colour map in the right window. The ϕ_1, ϕ_2 and r values are then determined interactively by entering values or moving the sliding bars in the "Colour Limit" section. The best settings for the ball segment shown has ϕ_1 (Minimum Angle) set to 316 degrees, ϕ_2 (Maximum Angle) set to 350 degrees and r (Minimum Magnitude) set to 40. It should be obvious that these values maximise the pie segment capturing the colour of the ball without any overlap into the pink region above, or the yellow region below. The exclusion radius r is set to maximise the area of the image over the ball in the left window. If r is set too low, various images in the field marking will be classified as the object. If r is set too high,

the number of pixels covering the ball will be insufficient to identify the ball.

11.5 Experimental Observations

This classification method worked very well in identifying the two specified team colours: blue and yellow. We were able to identify a pink colour patch on a robot that proved difficult for other teams to identify. Three different green patches were easily classified using this method and so could be used for identifying our own three robots.

During actual game trials, changing lighting conditions from Quartz-Halogen to fluorescent, did not affect classification of the colour patches on the robots, and classification of the ball. Hue drift appeared not to cause any problems.

11.6 Comments

In this research we have described a method for detection and classification of the ball and the colour patches on micro-soccer robots using pie slice decision regions in a YUV colour map, as opposed to rectangular regions in RGB or YUV colour maps. Luminance is ignored as hue and saturation are sufficient in classifying a colour. Use of the pie slice decision regions enables a more accurate definition of colour patches on the robots and colour of the ball. The method was found to be robust under changing lighting conditions in experimental trials.

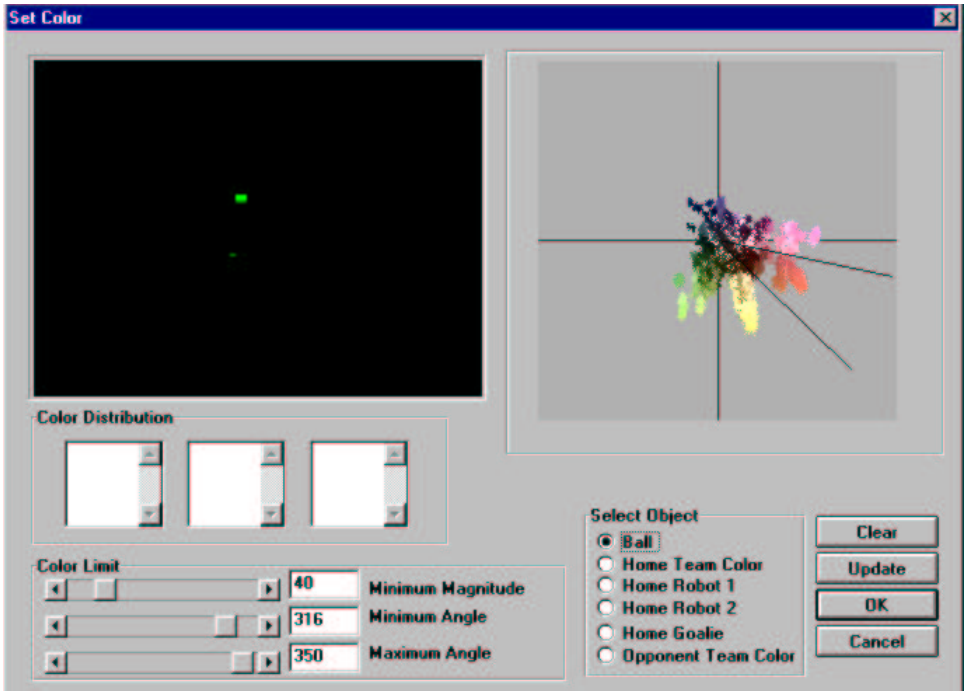


Figure 11.7: Screen Capture of Window using Pie Segment

Chapter 12

Conclusions and future directions

This thesis has examined the development of the fundamental controls for a robot soccer team in simulation. Observations of the robot performances at international competitions has shown that the fundamental controls are poorly done. The use of boundary avoidance, obstacle avoidance and defence strategies only work when the basics of robot control are functional. As robot soccer is a system, there is a large control loop including:

- Image capture,
- Robot colour patch identification,
- Robot control, and
- Radio transmission of control to the robots.

A number of issues in the control of robots in the robot soccer scenario have been examined:

- Boundary avoidance using a potential field,
- Proportional Cosine Control,
- A 3I2O fuzzy controller,
- A 5I2O fuzzy controller,
- A hierarchical fuzzy controller, and
- Fuzzy control from clustering of I/O pairs.

Boundary avoidance is simply that—the avoidance of defined boundaries and restricted regions of the playing field. When a potential field was used to stop the robot hitting the physical boundary, its movement tended to meander down the wall. The meandering was caused by the potential field forcing the robot away from the wall whilst the controller was forcing the robot towards the wall. To minimise this contradiction in control, a piece-wise linear profile was used to limit the angle between the robot and wall. This proved successful.

A control mechanism called a Proportional Cosine Controller was developed as a basic ball attack strategy. The cosine controller was a modification to the proportional control of the PID controller by including the cosine of the angular error ϕ. The resulting controller allowed the robot to turn toward a target position much faster, effectively allowing the robots to turn on the spot and move toward the target in one continuous motion. This feature made cosine control more suitable for use in positioning algorithms that guide the robot to points close to the robots centre. However, the use of a PID controller brought strict mathematical equations to an area that is not very precise. The PID controller is a precise mathematical model that requires exactness with input variables and produces a precise numerical output. Another problem in using PID controllers

is overshoot and misalignment. As a consequence, the cosine controller is not entirely effective in a non-linear environment as in robot soccer.

The control system developed in Chapter 6 was used on the soccer-robot hardware. It has been used for demonstrations and was prepared for the Paris competition in 1997. Transportation of the computer to Paris caused hardware problems that prevented playing in the competitive arena.

Research into these problems indicated that non-linear controllers would perform better in a non-linear environment. A fuzzy approach was chosen as this could show some insight into the strategies used in control through the fuzzy rule base.

Previous work in robot control [45, 46, 52, 67, 68, 69, 70, 71] used a simulated point mass robot. However, this research cannot be applied directly to the robot soccer area as the robots have a physical shape and include dynamic limitations such as wheel lift. The control of a robot in the physical environment needs to be determined before higher level control such as boundary and obstacle avoidance can be applied. Initially a minimal system was developed using only three inputs.

The 3I2O model used two angles and distance as inputs, derived from a relative coordinate system. The outputs were the left and right wheel velocities for the robot. These input variables describe positional information of the robot and are the minimum number variables that can be used for controlling the robot. An acceptable robot path can be found when evolving a fuzzy rule base from any single initial configuration. However, the evolutionary parameters and weightings of terms in the objective function had to be adjusted for convergence in reasonable time. Figure 12.1 shows the 3I2O system in diagram form.

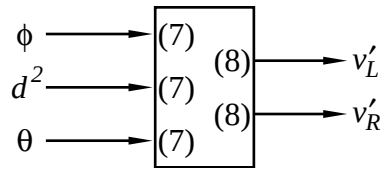


Figure 12.1: Three Input Two Output Model

Problems were encountered when evolving the fuzzy rule base for several initial configurations. Convergence of the algorithm was slow and was easily caught in local minima. A wide selection of parameters were trialed to obtain a reasonable solution. When developing a rule base over all input configurations, one disadvantage that occurs is the multi-criteria optimisation problem. Another drawback was the graphical plots of the robot paths showed the fuzzy controller produced either forward or reverse facing impact with the ball, but not both. Also, robot soccer is not about hitting a stationary ball, the game is very dynamic and the ball is, usually, moving at high speed. Increasing the number of fuzzy controller inputs to include left and right wheel velocities will increase the number of rules in the search space and provide a suitable solution for chasing a moving target.

The evolutionary algorithm in convergence did not reach an acceptable local minimum as each objective evaluation included the sum of many terms. The difficulties in obtaining convergence for such multi-criteria optimisation problems by all methods is well documented in the literature.

It was clear that information in terms of the current wheel velocities was necessary to overcome some of the problems observed. In an attempt to overcome the problems observed with the 3I2O model, the current left and right wheel velocities were included as inputs to the fuzzy controller.

With the current left and right wheel velocities added as input, the fuzzy system became a 5 input 2 output system. This 5I2O model for a controller gave a much better fuzzy rule base than the 3I2O model. The fuzzy controller was able to include both forward and reverse facing impact with the ball. Figure 12.2 shows the 5I2O system in diagram form.

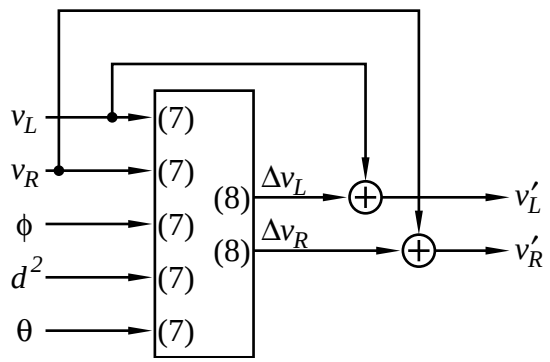


Figure 12.2: Five Input Two Output Model

However, the multi-criteria problem was still prevalent. A new method was trialed, evolving a path from each configuration separately for ten generations, and then moving onto the next initial configuration. The idea was the mass of the population would not move too much during small perturbations by concentrating on each initial configuration separately. However, this is an evolutionary algorithm where only the top individuals hold the best individual with other individuals ranging from mediocre to very poor. As there was only a small mass of good performing individuals, the knowledge base in other areas was destroyed. Another problem with the increase in the number of input variables was the looming "curse of dimensionality". This considerably increased the time taken for convergence of the evolutionary algorithm to produce an acceptable fuzzy controller.

Since the access to the fuzzy rule base was sparse, the programme established pointers to the relevant parts of the rule base before running the fuzzy system. This sparse matrix access method sped up the fuzzy control calculation. Large sections of the programme were rewritten in assembly to achieve a run time of one month on a single machine (P4 1.8GHz). The time for evolutionary algorithm convergence to an acceptable fuzzy controller was again high due to the "curse of dimensionality".

A three layer, five input, two output hierarchical system was developed to reduce the number of rules in the fuzzy controller. There was no decrease in the convergence time because the sparse matrix access to the rule base caused the fitness calculations to be the same as the 5I2O controller. Figure 12.3 shows the hierarchical system in diagram form.

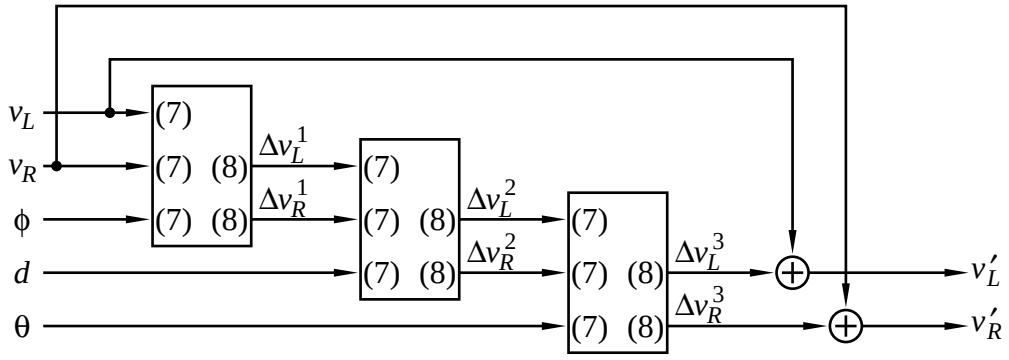


Figure 12.3: Hierarchical Model

The resulting controller performed the same as the 5I2O controller when viewing the paths produced. The only benefit of the hierarchical model was a smaller population size. This small population size has an advantage when considering distributed concurrent programming. Extending the number of rules in the 5I2O model make it impossible to use parallel techniques on a computing cluster as the problem becomes I/O bound. However, the hierarchical model is favourable

and will work on a cluster of machines as the I/O is greatly decreased.

Since the multi-objective criteria was still an issue, a different tack was taken in an attempt to reduce this problem. A reduction in the number of terms used in the objective function decreases the effect of multi-criteria optimisation. Therefore, evolutionary algorithms can be used to optimise velocity profiles from each initial configurations separately. This can then be converted to input/output pairs suitable for fuzzy clustering.

Distributed concurrent programming was easily implemented in this situation as each initial configuration used an individual evolutionary algorithm. Each optimised solution was stored in a separate file as a security measure. The files were collated into a single file of velocity profiles. A simple ending criteria was used to remove "bad" paths from the file. Each time step of the robot in the velocity profile path was extracted and sorted as input/output data for use in fuzzy clustering; approximately one and a half million input/output pairs.

The use of a fuzzy clustering algorithm was not entirely successful because the number of clusters used was small. In terms of each dimension in the input space of a 5I2O controller, this represents seventeen sets for each input universe: 17^5. To obtain the precise control needed for the system, the possible number of clustered centres could be as large as: 100^5 or larger. The problem with using clustering is that a very fine coverage of the input/output space is required to produce a controller with the required precision for controlling the robot.

As optimised velocity profiles were used, there were areas in the search space that had no information for the clustering algorithm to resolve appropriate action. The effect of untrained areas in the space, resulted in the wheel lift constraint being

broken. The clustering did not produce a controller that could satisfy all initial conditions and did not cover the entire problem space. Many more input/output pairs will be required to obtain a complete fuzzy controller. This makes this method somewhat infeasible as the number of input/ouput pairs increases substantially. However, the input/output data could be used to reduce the objective function to a sum of squared errors in developing a 5I2O fuzzy controller. The resultant system can then be checked for accumulation of errors and plotted in a graphics programme to identify where more membership sets are required. This is a future direction to be investigated.

As robot soccer is a complete system incorporating vision, control and hardware, it is natural to develop other areas to make a working system. A simple colour classification system was developed that improved searching for the ball and identification of the coloured robot patches on the field.

A major problem in any control system is the accumulation of delays in a loop of the system. The large control loop in this system incorporates many significant delays that make the system unsuitable for competition. An "off the shelf" video camera provides thirty frames per second (NTSC). The Windows (TM) programme used was found to process eighteen frames per second. The ball moving at 5ms^{-1} travels 278mm in $1/18^{th}$ of a second. Obviously, this contributes to the observed overshoot and misalignment problems. Overshoot and misalignment were also observed at low speed indicating issues with control. Image processing can be improved by processing each interlaced field at $1/60^{th}$ of a second with a dedicated image processor connected to the computer. The resolution of an interlaced field, chrominance portion is 320×240 assuming YUV 4:2:2. Projecting this resolution onto the field gives the smallest element of detection to be a square of 5.5×5.5mm. To detect a 5ms^{-1} object moving across 5.5mm of field

would require a frame capture rate faster than $1/1000^{th}$ of a second. There are a few image capturing systems that can provide images at this rate, however, they are expensive and one of the objectives of Mirosot is to use "off the shelf" components.

An in-line image processing system is in construction. It is designed to process images at sixty fields per second from a commercial video camera source.

Finally, the thesis has made valuable contributions in the study of high dimensionality and high precision problems using fuzzy control. These contributions have been presented at international conferences and some of these papers have been actively selected for a book chapter and journal publications.

Bibliography

[1] http://www.fira.net

[2] Russell, S. and Norvig, P. *Artificial Intelligence—A Modern Approach*, 1995, Prentice Hall, ISBN 0-13-103805-2.

[3] Lin, C. T. and Lee, G. C. S. *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, 1996, Prentice Hall P T R, ISBN: 0-13-235169-2.

[4] Goldberg, D. *Genetic Algorithms in Search, Optimisation and Machine Learning*, 1989, Addison-Wesley.

[5] Kosko, B. *Fuzzy Thinking*, 1993, Harper-Collins Publishers, ISBN 0-00654-713-3.

[6] Zurada, J. M. *Introduction to Artificial Neural Systems*, 1992, West Publishing Company, ISBN 0-314-93391-3.

[7] Gaweda, A. E. and Setiono, R. and Zurada, J. M. *Rule Extraction from Feedforward Neural Network for Function Approximation*, 2000, Proceedings of the Fifth Conference Neural Networks and Soft Computing, Zakopane, Poland, June 6–10, pp. 311–316.

[8] Wang, L. X. *A Course in Fuzzy Systems and Control*, 1997, Prentice-Hall Inc., ISBN 0-13-593005-7.

[9] Thomas, P. J. and Stonier, R. J. and Wolfs P. J. *Robustness of Colour Detection for Robot Soccer*, 2002, The Seventh International Conference on Control, Automation, Robotics and Vision (ICARCV'2002), Singapore.

[10] Kowalik, J. and Osborne, M. R. *Methods of unconstrained optimisation problem*, 1973, Modern analytical and computational methods in science and mathematics, R. Bellman (Ed.), American Elsevier Pub. Company, pp. 29–55.

[11] Fletcher, R. and Powell, M. J. D. *A rapidly convergent descent method for minimisation*, 1963, Computer Journal, Vol. 6, pp. 163–.

[12] van Laarhoven, P. J. M. and Aarts, E. H. L. *Simulated Annealing: Theory and Applications*, 1987, Kluwer Academic Publishers, ISBN: 90-277-2513-6.

[13] Shore, M. and Coddington, P. D. and Fox, G. C. and Marinari, E. and Wu, C. H. *A New Automatic Simulated Annealing-Type Global Optimization Scheme*, 1993, Journal of Undergraduate Research in High-Performance Computing, pp. 59–71.

[14] Kirkpatrick, S. and Gelatt, C. D. and Vecchi, M. P. *Optimization by Simulated Annealing*, 1983, Science, Vol. 220, No. 4598, pp. 671–680, http://citeseer.nj.nec.com/kirkpatrick83optimization.html.

[15] Kruglinski, D. J. *Inside Visual C++ Version 1.5*, 1994, Microsoft Press, ISBN 1-55615-661-8.

[16] Microsoft Corporation *Microsoft Visual C++ MFC Library Reference*, 1997, Part 1, Microsoft Press, ISBN 1-57231-518-0.

[17] Microsoft Corporation *Microsoft Visual C++ MFC Library Reference*, 1997, Part 2, Microsoft Press, ISBN 1-57231-519-9.

[18] Microsoft Corporation *Microsoft Visual C++ Run-Time Library Reference*, 1997, Microsoft Press, ISBN 1-57231-520-2.

[19] Microsoft Corporation *Microsoft Visual C++ Language Reference*, 1997, Microsoft Press, ISBN 1-57231-521-0.

[20] Petzold, C. *Programming Windows—The Microsoft Guide to Writing Applications for Windows 3*, 1990, 2nd Edition, Microsoft Press, ISBN 1-55615-264-7.

[21] Springer, G. and Wolfs, P. and Hannah, P. and Stonier, R. J. and Smith, S. *Simple Strategies for Collision-Avoidance in Robot Soccer*, 1996, of Micro-Robot World Cup Soccer Tournament Proceedings, KAIST, Korea, pp. 64–68.

[22] Thomas, P. J. and Stonier, R. J. *Evolutionary Learning of Fuzzy Control in Robot-Soccer*, 2003, Computational Intelligence in Control, M. Mohammadian (Ed.), Idea Publishing, USA, pp. 88–103.

[23] Thomas, P. and Springer, G. and Stonier, R. J. and Wolfs, P. *Boundary-Avoidance and Attack Strategy in Robot Soccer*, 1989, FIRA Robot World Cup France'98 Proceedings, Ed. Russel Stonier and Jong-Hwan Kim, Federation of International Robot Soccer Association, ISBN: 89-7163-102-3, pp. 27–32.

[24] Philips Semiconductors *Data Sheet SAA7110A One Chip Front-end 1 (OCF1) Product Specification*, 1995, Philips Semiconductors.

[25] Gonzalez, R. C. and Woods, R. E. *Digital Image Processing*, 1993, 3rd Edition, Addison-Wesley Publishing Company, ISBN 0-201-50803-6.

[26] Levine, J. *Programming for Graphics Files in C and C++*, 1994, John Wiley and Sons Inc.

[27] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*, 1994, 2nd Edition, Springer Verlag.

[28] Thomas P. J. and Stonier R. J. *Fuzzy Control in Robot-Soccer, Evolutionary Learning in the First Layer of Control*, 2002, Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI'2002), Florida, July, Vol. XII, pp. 181–186.

[29] Gill, P. E. and Murray, W. and Wright, M. H. *Practical optimization*, 1981, Academic Press, London, ISBN: 0122839501.

[30] ten Hagen, S. and l'Ecluse, D. and Kröse, B. *Q-Learning for Mobile Robot Control*, 1999, Proc. of the 11th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'99), 3–4 November, Maastricht, pp. 203–210.

[31] Stone, P. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*, 2000, MIT Press, ISBN: 0-262-19438-4.

[32] Sardis, G. *Toward the Realization of Intelligent Controls*, 1979, Proceedings of the IEEE, Vol. 67, No. 4, pp. 1115–1133.

[33] Freund, E. and Hoyer, H. *Collision Avoidance in Multi-Robot Systems*, 1984, Robotics Research, Second International Symposium, MIT Press, Cambridge, pp. 135–146.

[34] Freund, E. and Hoyer, H. *Real-Time Pathfinding in Multirobot Systems Including Obstacle Avoidance*, 1988, The International Journal of Robotics Research, Vol. 7, No. 1, Massachusetts Institute of Technology.

[35] Khatib, O. *Real-time Obstacle Avoidance of Manipulators and Mobile Robots*, 1985, The International Journal of Robotics Research, Vol. 5, No. 1, pp. 90–97.

[36] Newman, W. S. and Hogan, N. *High Speed Control and Obstacle Avoidance using Dynamic Potential Function*, 1987, IEEE International Conference on Robotics and Automation, pp. 14–24.

[37] Jarvis, R. A. *Growing Polyhedral Obstacles for Planning Collision Free Paths*, 1983, The Australian Computer Journal, Vol. 15, No. 3, pp. 103–111.

[38] Ardema, M. D. and Skowronski, J. M. *Coordination Controllers for Multi-arm Manipulators—a case study*, 1990, Advances in Control and Dynamical Systems, pp. 34.

[39] Stonier, R. J. *Use of Liapunov Techniques for Collision-Avoidance of Robot Arms*, 1990, Advances in Control and Dynamic Systems, Vol. 35, No. 2, pp. 185–215.

[40] Stonier, R. J. *Collision-Avoidance of two PR Manipulators*, 1992, Lecture Notes in Control and Information Sciences, No. 170, pp. 248–264.

[41] Lee, J. and Bien, Z. *Collision-free Trajectory Control for Multiple Robots based on Neural Optimisation Network*, 1990, Robotica, Vol. 8, pp. 185–194.

[42] Baba, N. and Kubato, N. *Collision Avoidance Planning of a Robot Manipulator by using Genetic Algorithm*, 1994, Proceedings of the First IEEE Conference on Evolutionary Computation, Florida, pp. 714–719.

[43] Lin, H. and Xiao, J. and Michalewicz, Z. *Evolutionary Algorithm for Path Planning in Mobile Robot Environment*, 1994, Proceedings of the First IEEE Conference on Evolutionary Computation, Florida, pp. 211–215.

[44] Xu, H. Y. and Vukovich, G. *Fuzzy Evolutionary Algorithms and automatic Robot Trajectory Generation*, 1994, Proceedings of the First IEEE Conference on Evolutionary computation, Florida, pp. 595–600.

[45] Mohammadian, M. and Yu, X. H. and Smith, J. D. *A Case Study of Knowledge Acquisition: from connectionist learning to an optimised fuzzy knowledge*, 1993, Proceedings of the 2nd International Workshop on Emerging Technologies and Factory Automation, Cairns Australia, pp. 106–111.

[46] Mohammadian, M. and Stonier, R. J. *Integrating Fuzzy Logic and Genetic Algorithms for Intelligent Control and Obstacle Avoidance*, 1994, Second National Conference on Complex Systems, Rockhampton, Australia, pp. 149–156.

[47] Shibata, T. and Fukuda, T. *Coordination in Evolutionary Multi-Agent Robotic System using Fuzzy and Genetic Algorithm*, 1994, Control Engineering Practice, Vol. 2, No. 1, pp. 103–111, Printed in Great Britain.

[48] Stonier, R. J. and Mohammadian, M. *Intelligent Hierarchical Control for Obstacle-Avoidance*, 1995, Computational Techniques and Applications: (CTAC95), Melbourne, Australia, July.

[49] Stonier, R. J. and Mohammadian, M. *Self Learning Hierarchical Fuzzy Logic Controller in Multi-Robot Systems*, 1995, Proceedings of the IEA Conference Control95, Melbourne, Australia, pp. 381–386.

[50] Thomas, P. J. and Stonier, R. J. *Evolutionary learning of fuzzy control in robot-soccer*, 2001, Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA'2001), ISBN 0 85889847 0 on CD, M. Mohammadian (Ed.), Las Vegas, pp. 494-503.

[51] Cordón, O. and Herrera, F. and Hoffmann, F. and and Magdalena, L. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases (Advances in Fuzzy Systems—Applications and Theory)*, World Scientific Publishing, Vol. 19, ISBN: 981-02-4017-1, 2001.

[52] Mohammadian, M. and Yu, X. H. and Stonier, R. J. *Tuning and Optimisation of Membership Functions of Fuzzy Logic Controllers by Genetic Algorithms*, 1994, Proceedings of 3rd International Workshop of Robot and Human Communication, Nagoya, Japan, pp. 356-361, July 18-20th.

[53] Homaifar, A. and Tunstel, E. and Dozier, G. and Battle, D. *Genetic and Evolutionary Methods for Mobile Robot Motion Control and Path Planning*, 2003, Intelligent Control Systems Using Soft Computing Methodologies, Edited by Ali Zilouchian and Mo Jamshidi, CRC Press, ISBN: 0-8493-1875-0, pp. 437–454.

[54] Ogata, K. *Modern Control Engineering*, 1986, Prentice-Hall, Englewood Cliffs, NJ.

[55] Yan, J. and Ryan, M. and Power, J. *Using Fuzzy Logic*, 1994, Prentice Hall, NY.

[56] Raju, S. and Zhou, J. and Kisner, R. A. *Fuzzy logic control for steam generator feedwater control*, 1990, Proceedings of American Control Conference, San Diego, CA, pp. 1491–1493.

[57] Raju, G. V. S. and Zhou, J. *Adaptive Hierarchical Fuzzy Controller*, 1993, IEEE Transactions on Systems, Man and Cybernetics, Vol. 23, No. 4, pp. 973–980.

[58] Thomas, P. J. and Stonier, R. J. *Using Evolved Paths for Control in Robot-Soccer*, 2003, IEEE Proceedings on Computational Intelligence in Robotics and Automation (CIRA2003), Kobe, Japan, ISBN: 0-7803-7867-9.

[59] Holland, J. H. *Adaptation in Natural and Artificial Systems*, 1975, Univ. Michigan Press, Ann Arbor, MI.

[60] Bäck, Th. *Evolutionary Algorithms in Theory and Practice*, 1996, Oxford University Press, NY.

[61] Fogel, D. B. *Evolutionary computation: Toward a New Philosophy of Machine Intelligence*, 1995, IEEE Press, Piscataway, NJ.

[62] Bäck, Th. and Schwefel, H. P. *Evolutionary Computation: An Overview*, 1996, Proceedings ICEC'96, Nagoya, Japan, May, pp. 20–29.

[63] Michalewicz, Z. *Evolutionary Computation: Practical Issues*, 1996, Proceedings ICEC'96, Nagoya, Japan, pp. 30–39.

[64] Karr, C. L. *Applying genetics*, 1991, AI Expert, pp. 38–43.

[65] Thrift, P. *Fuzzy logic synthesis with genetic algorithms*, 1991, Proceedings of the 4th International Conference on Genetic Algorithms, pp. 509–513.

[66] Cordón, O. and Herrera, F. *A general study on genetic fuzzy systems*, 1995, Genetic Algorithms in Engineering and Computer Science, G. Winter, J. Periaux, M. Galan, P. Cuesta (Eds.), John Wiley and Sons, pp. 33–57.

[67] Mohammadian, M. and Stonier, R. J. *Hierarchical Fuzzy Control*, 1998, Proceedings of the 7th International conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, Paris, pp. 621–629.

[68] Mohammadian, M. and Stonier, R. J. *Fuzzy Rule Generation by Genetic Learning for Target Tracking*, 1996, Proceedings of the 5th International Intelligent Systems Conference, Reno, Nevada, pp. 10–14, June 19-21st.

[69] Stonier, R. J. and Mohammadian, M. *Knowledge Acquisition for Target Capture*, 1998, Proceedings of the International Conference on Evolutionary Computing ICEC'98, Anchorage, Alaska, pp. 721–726.

[70] Mohammadian, M. and Stonier, R. J. *Hierarchical Fuzzy Control*, 1998, Proceedings of the 7th International conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, Paris, pp. 621–629.

[71] Stonier, R. J. *Evolutionary learning of fuzzy logic controllers over a region of initial states*, 1999, Proceedings of the 1999 Congress on Evolutionary Computation, Washington, Vol. 2, pp. 2131–2138.

[72] Zadeh, L. A. *Fuzzy Sets*, 1965, Information and Control, Vol. 8, pp. 338–353.

[73] Strum, R. D. and Kirk, D. E. *First Principles of Discrete Systems and Digital Signal Processing*, 1989, Addison-Wesley Publishing Company, ISBN 0-201-09518-1.

[74] De Jong, K. A. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*, 1975, Thesis, University of Michigan.

[75] Whiddett, D. *Concurrent Programming for software engineers*, 1987, Ellis Horwood Books, ISBN: 0-7458-0036-X.

[76] Bauer, B. E. *Practical Parallel Programming*, 1992, Academic Press, San Diego, California, ISBN: 0-12-082810-3.

[77] http://www-unix.mcs.anl.gov/mpi/mpich/

[78] Kim, J. H. *Multi-Robot Cooperative System Development*, Lecture Notes, Green Publishing Co., Feb., 1998.

[79] Kim, S. H. and Choi, J. S. and Kim, B. K. *Development of BEST Nanorobot Soccer Team for FIRA'98*, 1998, FIRA Robot World Cup France'98 Proceedings, La Cité des Sciences et de l'Industrié, Paris, France, ISBN: 89-7163-102-3, pp. 7–16.

[80] Zill, D. G. and Cullen, M. R., *Advanced Engineering Mathematics*, 1992, PWS Publishing Company, ISBN: 0-534-92800-5, pp. 465–467.

[81] Jung, M. J. and Shim, H. S. and Kim, H. S. and Kim, J. H. *The Omni-directional mobile robot OmniKity-Y(OK-I) for RoboSOT category*, 1999, Proceedings of the FIRA Robot World Cup 1998, R. J. Stonier and J. H. Kim (Eds.), pp. 37–42.

[82] Contact MA0307@DACOM.CO.KR, Tel: +82 42 864-0307, 0308, Fax: +82 42 864-0309

[83] http://www.dooin.co.kr

[84] Anver, M. M. and Stonier, R. J. *Design of an Hierarchical Fuzzy Filter for Removal of Heavy Impulse Noise from Corrupted Images*, 2002, accepted for presentation at The Seventh international Conference on Control, Automation, Robotics and Vision (ICARCV 2002), Singapore, December.

[85] Cordón, O. and Herrera, F. and Zwir, I. *Linguistic Modelling of Hierarchical Systems of Linguistic Rules*, 2002, IEEE Transactions on Fuzzy Systems, Vol. 10, No. 1, pp. 2–20.

[86] Thomas, P. J. and Stonier, R. J. *Evolutionary Learning of Fuzzy Control in Robot-Soccer*, 2003, In M. Mohammadian (Ed.), Computational Intelligence in Control, Idea Publishing, USA, pp. 88–103.

[87] Sen, P. C. *Principles of Electric Machines and Power Electronics*, 1989, John Wiley & Sons, ISBN: 0-471-61717-2.

[88] Shim, H. S. and Jung, M. J. and Kim, H. S. and Choi, I. H. and Han, W. S. and Kim, J. H. *Designing Distributed Control Architecture for Cooperative Multiagent Systems*, 1996, Micro-Robot World Cup Soccer Tournament Proceedings, KAIST.

[89] Kim, S. S. and Shin, W. G. and Lee, W. G. and Seo, S. H. and Moon, J. K. and Hong, S. J. *Development of Multiple Micro-Robot Playing Soccer and Supervisory System*, 1997, Micro-Robot World Cup Soccer Tournament Proceedings, KAIST.

[90] In, C. S. and Choi, Y. H. and Lee, J. S. *Implementation of Multi-target Tracker for Mirosot*, 1997, Micro-Robot World Cup Soccer Tournament Proceedings, KAIST.

[91] Kim, S. H. and Kim, J. K. and Choi, J. S. and Kim, B. K. *Development of Cooperation System for Robot Soccer Team*, 1996, Micro-Robot World Cup Soccer Tournament Proceedings, KAIST.

[92] Blank, R. and Koller, J. M. and Lauria, M. and Conti, F. *Omnidirectional robot and fast positioning systems*, 1997, Micro-Robot World Cup Soccer Tournament Proceedings, KAIST.

[93] Oller, A. and García, R. and Ramon, J. A. and Figueras, A. and de la Rosa, J. Ll. *Co-operation among Robots by Means of Multi-Agent Decision Making Framework under MATLAB/SIMULINK*, 1997, Micro-Robot World Cup Soccer Tournament Proceedings (MIROSOT'97), Taejon, KAIST.

[94] Kim, K. H. and Ko, K. W. and Kim, J. G. and Lee, S. H. and Cho, H. S. *Multiple Micro Robots Playing Robot Soccer Game*, 1996, Micro-Robot World Cup Soccer Tournament Proceedings, KAIST.

[95] Park, S. W. and Kim, J. H. and Kim, E. H. and Oh, J. H. *A study on the Development of Micro-Robots and Basic Strategy for Micro-Robot Soccer Tournament*, 1996, Micro-Robot World Cup Soccer Tournament Proceedings, KAIST.

[96] Sargent, R. and Witty, C. and Bailey, B. and Wright, A. *Fast Vision Tracking and Coordinated Control for Soccer-playing Robots*, 1997, Micro-Robot World Cup Soccer Tournament Proceedings, KAIST, 1997.

[97] Hong, S. G. and Eom, T. D. and Lee, C. Y. and Kim, M. S. and Sugisaka, M. and Lee, J. J. *Designing Soccer-Playing Robot Team (F. B. I.) Based on the Centralized Approach*, 1997, Micro-Robot World Cup Soccer Tournament Proceedings, KAIST.

[98] Hutson, G. H. and Shepherd, P. J. and Brice, W. S. J. *Colour Television*, 1990, 2nd Edition, McGraw-Hill Book Company (UK) Limited, ISBN: 0-07-084199-3, pp. 121.