Artificial Intelligence in Robot Soccer

Bachelor of Software Engineering
Honours Year Project
Massey University

20 October 2006

A. P. Gerdelan
gerdelan@gmail.com

Supervised by

Dr. N. H. Reyes

Table of Contents

List of Figures

Index of Tables

Abstract

Our previous works have successfully developed novel intelligent navigation algorithms for the control of fast autonomous vehicles. The project presented in this report builds on these past works by presenting a new algorithm for the dynamic role assignment and coordination of a *team* of mobile robots operating in real-time environment against multiple intelligent adversaries. This report also presents a complete robot architecture, and introduces a new motor control system, specially designed for this system, and enabling more efficient, intuitive, high-speed robot motion.

Keywords: *Real-time autonomous agents, autonomous vehicle navigation, path planning, Fuzzy Logic, Robot Soccer, AI strategy, intelligent motor control*.

1. *Introduction*

The robot soccer game provides us with an effective development arena for a range of robot researches and associated technologies.

This project report describes the analysis, theory, and development of a new level of artificial intelligence, designed specifically for the coordination of soccer robots, but that is also highly applicable to other applications such as computer games.

We have not limited the scope of this project to theory alone; we intended to fully implement the new intelligence into the robot soccer control infrastructure.

In order to make implementation of the high level intelligence feasible and demonstrable, it was necessary to develop a complete range of supporting technologies to work in conjunction with the strategy system.

This report therefore also introduces the supporting technologies employed, the innovations that we have made in these areas during the course of the project presented herein, and the special robot architecture that we have developed to facilitate the combined operation of the involved technologies.

2. *Research Description*

2.1 Overview of the Problem Domain: Robot Soccer

The competitive nature of robot soccer drives researchers to produce algorithms that are not only adequate, but must also be continually more efficient and more capable than existing methods.

Robot soccer also requires the effective deployment of a complete 'kit' of robot systems, all of which must work in complement with the other systems; a considerable challenge for any research undertaking.

There are two prominent robot soccer *leagues*; FIRA and RoboCup. Both of these leagues host international competitions subdivided into events for a range of different categories of robot system.

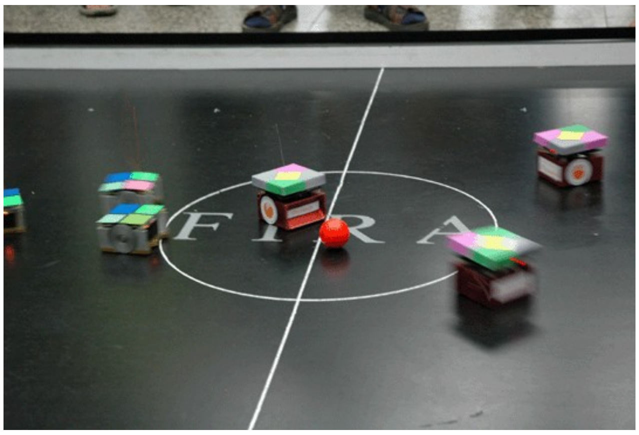


Figure 1: MiroSot Robots in Play at FIRA Dortmund, 2006.

We are building our research on the Micro Robot (MiroSot) soccer category (refer to *Figure 1*). These robots are extremely fast moving, and there are up to 22 robots in play at one time; which puts maximum emphasis on the efficiency requirements of vision and other AI algorithms, and also encourages us to inculcate a tactical, team element into our artificial intelligence.

For simplicity, the MiroSot league of robot soccer allows competing teams to control an entire team of these robots from a central computer *via* remote control. The entire team is able to use one overhead camera for sensing the environment, which can then detect the positions and headings of robots based on coloured patches on the tops of the robots (refer to *Figures 1 and 2*).

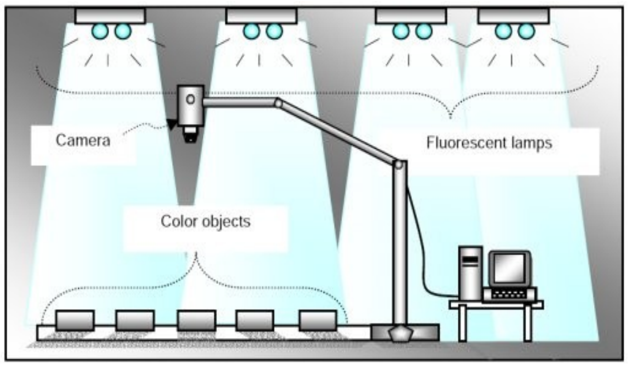


Figure 2: The Robot Soccer System

A competitive robot soccer team must then develop and combine their robot hardware with several cooperating areas of artificial intelligence:

- Machine Vision
- Path Planning and Navigation
- Motor Control
- Cooperative strategy
- Opponent prediction (hostile intelligence)

Where opponent prediction can be as simple as estimating the immediate path of movement of opposing robots to avoid collision, or go so far as to predict the pattern of strategy in use by the other team, which leads into the area of *game theory*.

2.2 Overview of the Current State of Technology

2.2.1 Existing State of Technology at Massey University

Significant resources were available within our department that could be used in development of our research.

We had at our disposal, the hardware for a complete MiroSot robot soccer system:

- Kit-set robots
- An analogue video camera
- A digital video camera
- Frame-grabber computer interface cards
- A robot soccer 'field'
- A computer workstation dedicated to robot control
- Laboratory space
- RF Transmitter Equipment

In addition, there had been significant work on previous projects that had resulted in the creation of a semi-completed robot control program, capable of controlling the robots and processing most of the essential robot system components in a cohesive approach.

The existing control program provided us with a working platform that we intended to use to test the implementation of our research. We had the complete source code for this control program, which was a valuable resource and reference.

The existing control program did not, however, incorporate any artificial intelligence, but some program components were useful, as developing these system elements was outside of the scope of this project:

- RF Remote Control Infrastructure
- Vision Processing Systems

With a program infrastructure for interfacing with the RF Transmitter equipment, we were able, for this project to output intended robot motor *velocities*, which removes the need to calculate the exact voltages required, and the corresponding signal, considerably simplifying the actuator outputs from our system.

An advanced machine vision system based on new Fuzzy Colour Classification techniques had been incorporated into the control program by Reyes [7], establishing a solid foundation for which we could reliably test our new research.

The Fuzzy Colour Classification system allows us to work in a much greater range of light conditions than popular vision systems currently facilitate, which tend to introduce high levels of error into the system.

By observation of the running robot soccer system resources, we can see that the robots have no effective navigation or coordination systems, as we commonly observe our robots on the same team colliding with each other and competing for the ball.

It was our intention to integrate our new research and algorithms into the existing architecture, reusing existing resources to as high a degree as was feasible.

2.2.2 State of Technology at the International Level

The chair of the previous FIRA games in Singapore was Dr N. H. Reyes, who collected a large amount of video footage of the modern robot soccer teams during competition that we were able to view.

Reviewing the footage from FIRA Singapore, we make the conclusion that most of the current teams have no central intelligence or coordinated path planning based on the following observations:

- Robots on the same team compete with each other for the ball
- Robots on the same team frequently collide with team members

We also make the key observation that modern teams have not developed any dynamic *role assignment* functionality; that is, we observe from the behaviour of individual team members that the roles of *defender, attacker,* and *goalie* are pre-set to individual vehicles before the game, and overall robots *do not* have the ability swap roles during play when it would be to their advantage.

There is therefore, a lack of dynamic central intelligence in robot soccer systems internationally.

2.3 Research Objectives

2.3.1 General Objectives

In order to both develop the robot soccer team at Massey University towards a competition-ready state, and to develop new algorithms designed to exploit the observed weaknesses of modern international robot soccer teams, the following were set as general project objectives:

1. To design a complete robot system architecture to combine the new research with existing technology.
2. The architecture must be flexible enough to support future change to the vision systems
3. To incorporate previous robot navigation works into this architecture
4. To develop a new strategy layer of intelligence for dynamic role assignment
5. To investigate a new intelligent motor control module to marry together the new intelligent layers with existing hardware.

2.3.2 Specific Objectives

The General Objectives would have to be developed on a range of different platforms and environments and can be further specified into sub-tasks:

1. Outline a new robot system architecture to support the requirements of the various layers of robot control systems.
2. Develop the *theory* for a strategy layer and investigate the application of new AI algorithms as part of this layer
3. Build a simulation to demonstrate theory in a prototype
4. Redesign the entire robot control program with a modular approach to support new architectural models.
5. Implement previously developed navigation layers.
6. Investigate methods for controlling the robot motors based on navigation layer outputs.

7. Build a real-time 3D simulation representing the game of robot soccer to combine new technologies and algorithms in a pseudo-control program.
8. Build a disembodied adaptation of the existing robot control program to trial implementation without hardware.
9. Design and implement a new motor layer.
10. Refine the integration of the different layers and identify any communication errors or inefficiencies.
11. Implement strategy layer (top brick of pyramid).
12. Test systems with full hardware implementation and tune system.

2.4 Scope and Limitations of Research

The scope of this project covers several complementary areas of artificial intelligence as components of a complete robot system:

- Fuzzy Strategy and Dynamic Role Assignment
- Real-Time Path Planning
- Opponent Prediction
- Fuzzy Obstacle Avoidance and Target Pursuit
- Intelligent Motor Control

These areas must also work with other layers of intelligence and components of the robot system. The following areas are, however, not the main focus of this research, and are considered outside of the scope this project:

- Camera Calibration
- Image Processing
- Other Machine Vision Systems
- Frame-grabber Interface
- Transmitter Interface
- Signal Processing
- Electronics
- Microelectronics
- Vehicle design

Where these systems that are out of project scope must be interacted with during the development of our new technologies, we will create *dummy* layers, where we can control error, or we shall simply appropriate those pre-developed systems that are already at our disposal.

2.5 Significance of Research

The research element of this project presents a novel architecture for a complete, competition-ready robot soccer system, that successfully combines several new intelligent algorithms and is designed to take advantage of current weaknesses observed in other robot soccer systems.

As part of the research for this project, we have devised a novel algorithm based on Fuzzy Set Theory, for dynamic allocation of behavioural *roles* based on the state of the robot soccer environment. This algorithm has been shown to operate in real time and is able to simultaneously control a large number of robots.

We also present in this report a new motor control layer, designed to map the abstractions of vehicle motion output by a Fuzzy Navigation Control Layer to real motor instructions for twin-motor vehicles.

2.6 Structure of Report Documentation

The core sections of this report are structured to reflect the architecture of our layered system; there is a section for each layer of the robot system that we have developed within the scope of the project.

We have chosen not to examine those layers that we have developed as part of previous works within the body of this report, but rather attached those works as appendices.

In each section we discuss the systems and algorithms that we are using for each layer, and introduce new algorithms that we have developed for the project presented in this report.

We have included additional sections, discussing the development of the 3D simulation environment that we have created for research and demonstration purposes.

3. Proposed System Architecture

We have devised a new robot system architecture for this project. The purpose of this architecture is to clearly separate different elements of the robot system, and to detail the relationships of these layers.

We have found no evidence from the literature of modern research of any explicitly designed robot system architecture having been produced. We have therefore designed our own architecture from scratch.

Our previous works have been focused on the development of single elements of the robot system. The primary objective of this project was to develop a strategy layer; a parent layer of intelligence to path planning and obstacle avoidance systems. In order to experiment with such as system beyond pure theory, it was necessary to construct the full robot system from individual components.

For simplicity we have created an autocratic control architectural, with a flow of information (or control) passed in only one direction. The flow through our complete architecture occurs regularly; all of the elements are recomputed with at each refresh of the sensory inputs at the frame-grabber level (approximately 33 iterations per second).

This one-dimensional structure gives us full visibility and control over our robot system against our time constraint; we can break down our CPU costs amongst the elements of our robot system.

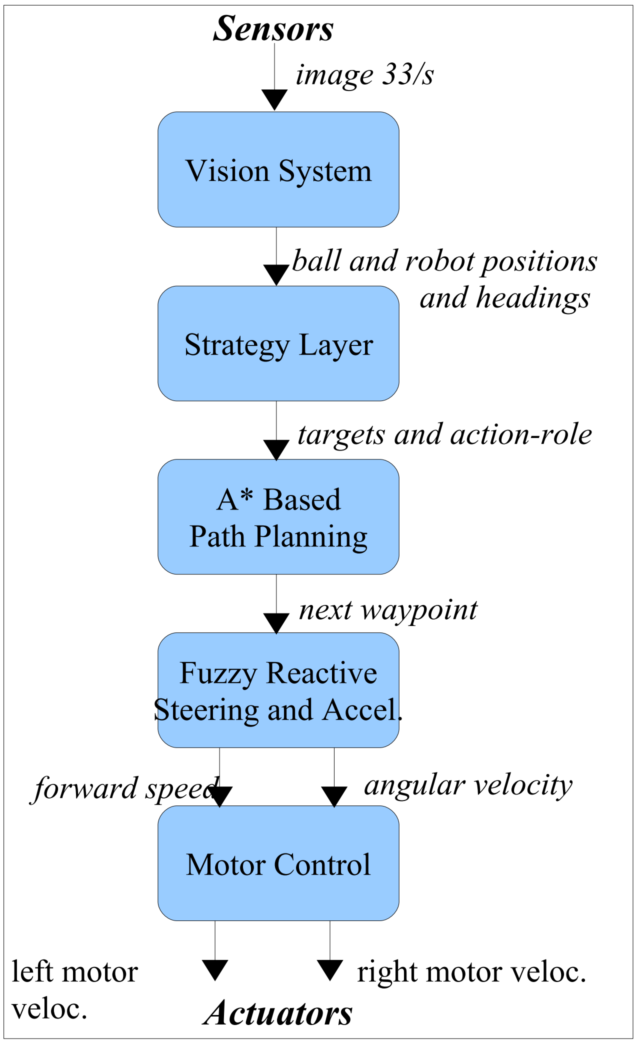


Figure 3: Our Layered Robot System Architecture

Figure 3 illustrates the layered architecture that we have devised for our robot system. The input to our system is provided by the *sensors*, which we consider to be the video camera and frame-grabber hardware and software interface.

The frame-grabber provides our system with an image of the robot soccer environment, as captured by the overhead camera, every 1/33s. This is fed as input to the first layer of our architecture – the *vision system*.

The vision system layer processes the image and produces the positions and headings of all of the robots and the ball. These are passed to the next layer of the architecture; the *strategy layer*.

The strategy layer collects the ball and robot positions and produces velocities, used to predict the path of movement for all of these elements. The strategy layer considers these factors and makes an intelligent tactical decision for the team of robots under its direction.

The strategy layer decides which robots to assign the roles of *attacker. defender*, and *goalie*, and determines the immediate action for each robot on the team based on these stereotypes. The immediate action for each robot is sent as a *destination of movement* to the *path planning layer* of each robot.

The path-planning layer breaks the entire environment into a conceptual grid, which corresponds to a graph of nodes, and searches for a path from the grid cell occupied by the robot in question to the grid cell in which its destination of movement lies.

Because we are operating in a dynamic environment and recalculating this path every 1/33s we are only interested in the *next* step (node) along the calculated path. The centre of this node is passed as input to the *fuzzy reactionary navigation layer.*

The Fuzzy reactive layer refines the path of movement of the robot. The reactionary navigation layer determines the steering and acceleration required for *target seeking* motion towards the *next* path waypoint, but switches its behaviour to *obstacle avoidance* motion in real time when an obstacle is encountered.

The Fuzzy reactive layer outputs the *forward speed* and *angular velocity* instructions required to achieve the desired motion to the *motor control* layer.

The motor control layer accepts *forward speed* and *angular speed* instructions, and contains an intelligent system for mapping this abstraction of motion to individual left and right motor instructions, based on the characteristics of the particular vehicle. These low-level motor instructions are sent via remote control to the motors of the robot.

4. A New Real-Time Strategy Layer AI Engine for Dynamic Role Assignment

The strategy layer designed for this project comprises several stages of operations:

1. Task identification

2. Role prioritisation
3. Role assignment
4. Robot destination assignment based on role stereotype

The strategy layer is intended for dynamic operation, that is, roles for each robot are recalculated with every iteration of the robot system; at approximately 30Hz. The strategy layer has therefore been designed to be adaptive to environment changes, and consume minimal processing time.

4.1 Initial Design and Prototype

4.1.1 Phase One: Dynamic Role Assessment

The initial design for the strategy layer was to use similar core inputs to those used by the strategy engine designed by Messom and Gupta [4]. These inputs are derived from the parent layer, the vision system, and are:

- *Ball speed towards own goal*
- *Ball distance from own goal*

These inputs would then be processed by a large, multi-dimensional decision matrix, that would also consider the heuristic-based success of previous matchings as internally-derived inputs, along with other, projected input-factors. The decision matrix would produce the following outputs:

- Heuristic weighting of importance of the *goalie* role
- Heuristic weighting of importance of the *attacker* role
- Heuristic weighting of importance of the *defender* role

4.1.2 Phase Two: Dynamic Role Allocation

Using the weighted importance of roles from the first phase of the strategy engine, we can assign roles to individual robots, starting with the highest-priority role, until all robots have been allocated a single role.

In choosing a robot to allocate each role we can consider the following pseudo-code:

- **for** (all robots on team)
    - **if** *unallocated* **then** evaluate role-allocation heuristic
    - **if** heuristic is lower than *lowest heuristic* **then** *lowest heuristic =* robot[*current*]
- allocate role to robot[*lowest heuristic*]

and change state of robot to *allocated*

Where the role-allocation heuristic is determined by a function specific to that role. The heuristics used are simply the Euclidean distance between the robot and the intended destination for that role, which we discuss in the following section.

4.1.3 Phase Three: Destination Assignment Based on Role

Each role represents a set of different behaviours. These behaviours are simply different paths of motion in relation to the input variables *ball speed* and *ball position*.

In our system, the goalie should only move within the goal area. The goalie should move in-between the goal and the position of the ball. If the ball is moving towards the goal, based on the *ball speed*, then we can instruct the goalie to move into the goal area that is closest to the projected point of impact of the ball. Considering the goal area divided into three zones, we can considerably simplify this problem.

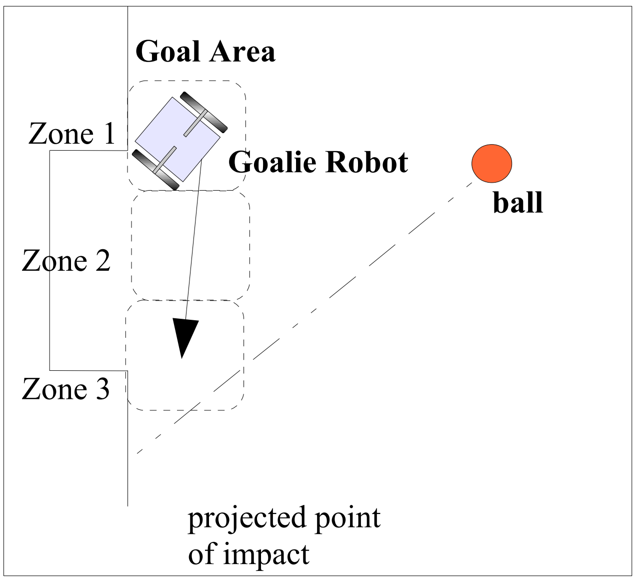


Figure 4: Example Target Destination for Goalie Role

Referring to *Figure 4*, we can see that as the projected impact for the ball is closest to *zone 3*, we will instruct the goalie robot to move into this zone. In this way we can inculcate a limited predictive behaviour into the goalie.

The common role of the *defender* is to block the ball and pass it forward to the attacker. Our system improves on this strict *defender-attacker* stereotype by allowing the defender robot to change roles and *become* the attacker robot once it has seized the ball. The role of the *defender* in our system, then, is only to manoeuvre a robot into a defensive position. This is perhaps a key advantage of our system, as static-role approaches do not achieve this smooth-transition duality of robot function.

The target destination for the defender robot is similar to that of the goalie, except that it is not limited to the goal area. Referring to *Figure 5*, we can consider the following simplified cases and rules:

- The defender should move to *closest point* to the robot, in-between the ball and the goal.
- If the robot is already in-between the ball and the goal, then it should head directly for the ball.
- If the ball is heading towards the goal, we can modify the target destination to be the closest point along the projected path of the ball and the defender.
- If the defender is not in-between the ball and the goal then it should move towards the goal area, avoiding the ball.

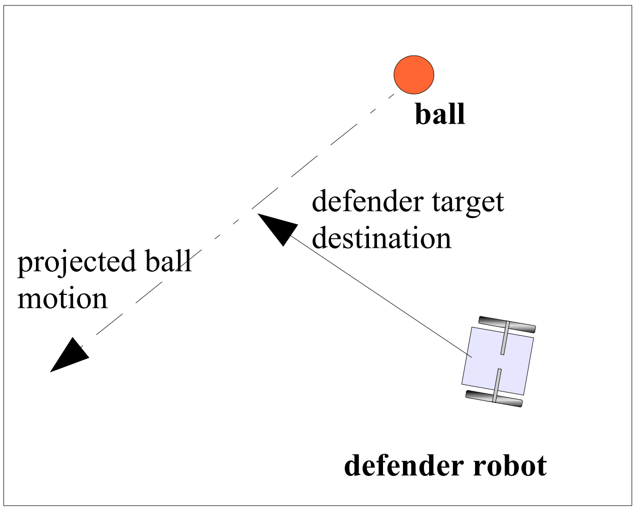


Figure 5: Example Target Destination for Defender Role

The *attacker* robot has two distinct states of behaviour in our system:

1. *Aggressive Behaviour* – head straight for ball
2. *Evasive Behaviour* – predict and dodge opponents

The first state – *aggressive behaviour* – is designed to allow the robot to move as quickly as possible to capture the ball. In this case the robot is always awarded the ball as target destination.

The second state – *evasive behaviour* – is a special feature of robot navigation developed in our previous work on that subject [3]. This mode of navigation is specially designed for a robot *carrying the ball*. In this case the robot's target destination is the *opponents' goal*.

Under *evasive* navigation, the robot not only avoids obstacles and heads towards the target, but uses a probabilistic approach to predict the intended movement of hostile robots, and steers clear of those areas *en route* to the goal. The internal state of the robot must be set to *evasive*, to facilitate the special behaviour at the navigational level.

4.1.4 Conclusions

Although the prototype for the initial design operated satisfactorily under simulation, there were several problems related to using large decision-matrix arrays that demanded revision before the system could be considered flexible or broadly applicable:

1. Inflexibility of fixed arrays to future additions
2. Lack of sensible interpolation policy for unknowns

If the decision matrix were to support a large number of inputs and decision criteria, as was intended, then we must construct, store and access a gigantic, multi-dimensional table in memory every 1/33s.

Although the space and memory constraints of the robot system discourage large tables, the greater problem is the inflexibility of this approach; the entire set of table matching and construction functions would have to be redesigned whenever a new input variable is added to the decision matrix.

The other core problem with the proposed decision-matrix design is the lack of an obvious interpolation policy. With smaller rule tables, it is acceptable to manually enter the decision outcomes into the table cells, however, with a very large, multi-dimensional decision-matrix, or indeed, a *dynamic* decision-matrix, that practice would be infeasible.

4.2 Strategy Layer Design Revision: The Best Man for the Job Algorithm

In order to address the problems with the initial design, we have devised a new decision-making engine, based on Fuzzy Logic.

4.2.1 Fuzzy Input Sets

The first step in the revised process is to *fuzzify* the *ball speed* and *ball position* inputs into *rough set memberships*.

Figures 6 and *7* present the Fuzzy Inputs Set Membership Functions that we have created. The domain of the input set functions covers the complete range of possible *ball speeds* and ball distances from goal (*ball positions*).

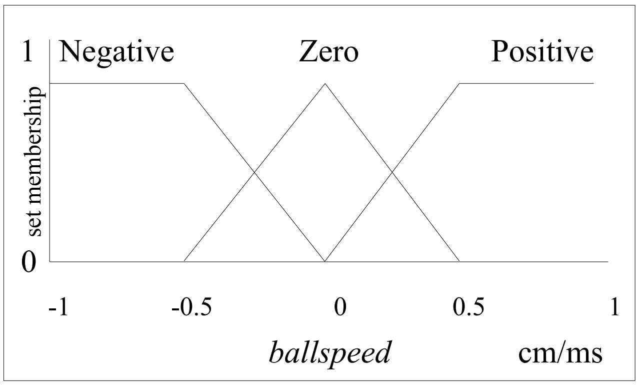


Figure 6: Sample Fuzzy Input Set membership function for ball speed

Referring to *Figure 6*, we can calculate *fuzzy set membership* values (between 0 and 1) for *ball speed* for each of the three fuzzy sets that we have designed. Our *ball speed* is then fuzzified into a partial member of the *Negative*, *Zero*, and *Positive* sets.

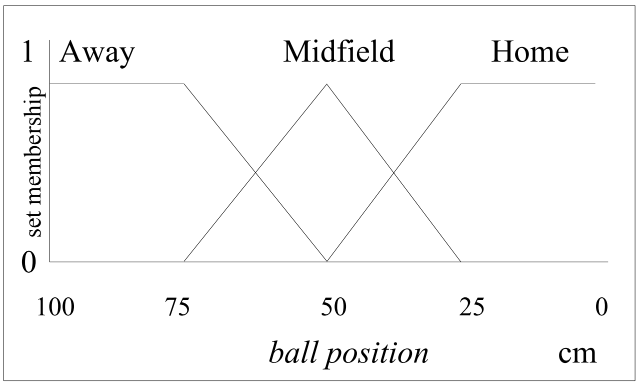


Figure 7: Sample Fuzzy Input Set membership function for ball position

Figure 7 illustrates our Fuzzy Set Membership function for the *ball position* input. We can fuzzify our ball position into a partial member of the *Away*, *Midfield*, and *Home* fuzzy sets in the same manner as *ball speed*.

Evaluation of real inputs against Fuzzy Input Set membership functions will produce a Fuzzy Input; a *set* of Fuzzy set memberships, which we can express in Fuzzy Set Notation:

$$\begin{gathered} \textit{Fuzzy Input} = \{ \textit{Fuzzy Set}_0 \textit{Membership Value}, \\ \textit{Fuzzy Set}_1 \textit{Membership Value}, \\ \textit{Fuzzy Set}_2 \textit{Membership Value} \} \end{gathered}$$

We have, for example, after evaluation of a ball speed of -0.25cm/ms, the following Fuzzy Input for Fuzzy Ball Speed:

$$\textit{Fuzzified Ball Speed} = \{0.5, 0.5, 0\}$$

Where the values for *Fuzzy Sets 0 , 1,* and *2* correspond to the membership values of the *Negative*, *Zero*, and *Positive* sets, respectively.

We can illustrate our new robot soccer environment, divided into overlapping fuzzy ball position sets. Refer to *Figure 8*.

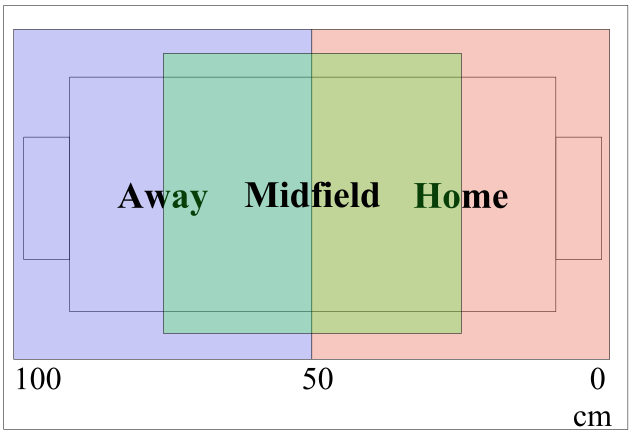


Figure 8: Fuzzy Sets for ball position superimposed over soccer field

4.2.2 Fuzzy Output Sets

Our Fuzzy Controller will output a *set of heuristic priorities*, ranking the relative importance of each of the robot roles.

For a greater degree of flexibility when deciding on the output of Fuzzy rules, we use 5 Fuzzy Output Sets. We have used the standard Fuzzy classifiers; *Zero, Small Positive, Medium Positive, Large Positive,* and *Very Large Positive*. Where output values can be expressed as partial members of each of these sets in the Fuzzy Set notation:

$$\textit{Fuzzy Output Value} = \{ ZE, SP, MP, LP, VP \}$$

Figure 9 illustrates the Fuzzy Output Set functions that we have used for our experiments. The centre definitions (the heuristic value that maps from a set membership value of 1) for each set are indicated by the dashed lines in *Figure 9*. These definitions will be used as weights in the *defuzzification* and aggregation process.

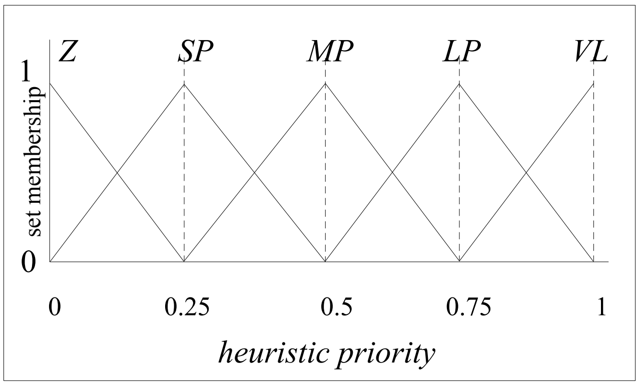


Figure 9: Fuzzy Output set membership function with dashed central values

The output from our system, is not a single *Fuzzy Output Value*, however, but are a set of these output values; one Fuzzy Output Value representing each of the three robot roles.

4.2.3 Fuzzy Rules and Fuzzy Associative Memory Matrix

We have arbitrarily defined the full complement of Fuzzy Rules that map every combination of Fuzzy Input Sets to a Fuzzy Output set *for every robot role*.

For simplicity, we have combined the rules for the Fuzzy Controllers of each robot role into one Fuzzy Controller, which we are able to do because each role is using the same Fuzzy Input Sets. Therefore our rules produce *sets* of Fuzzy Outputs, rather than a single output. Our outputs are expressed in the form:

$$Output = \{ \text{Attacker Fuzzy Output Set}, \text{Defender Fuzzy Output Set}, \text{Goalie Fuzzy Output Set} \}$$

We can express our Fuzzy Rules in *natural language*. Following are sample Fuzzy Rules from our system:

if *ball speed is negative* ***and*** *ball position is away* ***then*** *attacker priority is very large* ***and*** *defender priority is zero* ***and*** *goalie priority is small.*

(Fuzzy Rule 1).

if *ball speed is zero* ***and*** *ball position is away* ***then*** *attacker priority is large* ***and*** *defender priority is small* ***and*** *goalie priority is medium.*

(Fuzzy Rule 2).

Table 1 presents a Fuzzy Associative Memory Matrix (FAMM); a complete mapping of all the Fuzzy Rules in our controller.

	Away	***Midfield***	***Home***
Negative	*{VP,ZE,SP}*	*{LP,SP,MP}*	*{LP,MP,MP}*
Zero	*{LP,SP,MP}*	*{LP,MP,LP}*	*{VP,LP,LP}*
Positive	*{ZE,MP,MP}*	*{ZE,LP,LP}*	*{ZE,LP,VP}*

Table 1: FAMM matching ball speed and ball position to a set of role priorities

In First Order Logic, we would apply Aristotle's *law of the excluded middle*, which states that for all propositions *p*, either *p* or *~p* must be true, there being no middle true proposition between them. In a First Order Logic Inference Engine, we would evaluate only one of our rules.

We, however, are making use of Rough Set theory, which allows us to define propositions as a region between true and false (1 and 0). In our Fuzzy Controller, therefore, we evaluate *all* of our rules, combined using *Zadeh* operators, and produce an output *weight* for each rule output.

We can therefore define a map of weights, corresponding the the rule outputs in our FAMM. This matrix is illustrated in *Table 2.*

	Away	***Midfield***	***Home***
Negative	w_1	w_2	w_3
Zero	w_4	w_5	w_6
Positive	w_7	w_8	w_9

Table 2: Map of weights for FAMM

For each robot role, we can now evaluate a set of 9 weights based on the ball speed and ball position inputs.

In order to determine the output set membership values we make use of the *Zadeh AND* operator. *Fuzzy Rule 1* states that our Fuzzy Output Set for the *attacker* role is *Very Large Positive*. To determine the value for the *VL* output set, we therefore apply *Zadeh AND* to both the Fuzzy *Away* Input Set value and the Fuzzy *Negative* Input Set value:

$$Output_{VL} = min(Input_{Away}, Input_{Negative})$$

Where we can see that the *Zadeh AND* operator is equivalent to the mathematical *min* operation.

4.2.4 Defuzzification

In order to produce a single, real output value that we can use as a role priority heuristic we need to *defuzzify* our Fuzzy Output values.

Defuzzification involves use of an aggregation procedure to determine which output values to use, and then combining the sum of weighted averages of those values in a centre of gravity function to produce a single, crisp output value.

As our Fuzzy rule system is a clear series of 3 3*3 rules tables, we are able to see that there can be very little bias of one Fuzzy Output dominating the aggregation procedure. Therefore we have chosen not to use a *Fuzzy Aggregation Procedure*, but have rather used the First Order Logic *OR* operator, in order to incorporate the values of *all* Fuzzy Outputs into the sum of weighted averages and centre of gravity calculation. Thus our centre of gravity function, *for each robot role*, is:

$$\frac{(w_1 * v_1 + w_2 * v_2 + w_3 * v_3 + w_4 * v_4 + \ldots + w_9 * v_9)}{\sum_{i=1}^{9} w}$$

Where v_i is the centre value for the rule that corresponds to weight w_i. For example, for the *attacker* role, the weight w_1 corresponds to the output set *Very Large Positive*, which has a centre value of 1. The value of v_1 for the *attacker* role is therefore equal to 1.

4.3 Convergent Cascades of Fuzzy Associative Memory Matrices

One key advantage of the Fuzzy Controller approach to robot soccer strategy, and the reason for which we have re-designed our strategy system with the Fuzzy approach, is its very high degree of flexibility.

It is possible to incorporate a multitude of other input variables into the Fuzzy Strategy Layer with no added complexity to the decision making process. We no longer have to consider multi-dimensional decision arrays, and can sensibly interpolate unknowns by aggregation of Fuzzy outputs.

If we analyse pairs of real inputs using Fuzzy Associative Memory Matrices, we can pass the

Fuzzy Outputs (without defuzzification) as inputs to subsequent Fuzzy Associative Memory Matrices. *Figure 10* illustrates this approach.

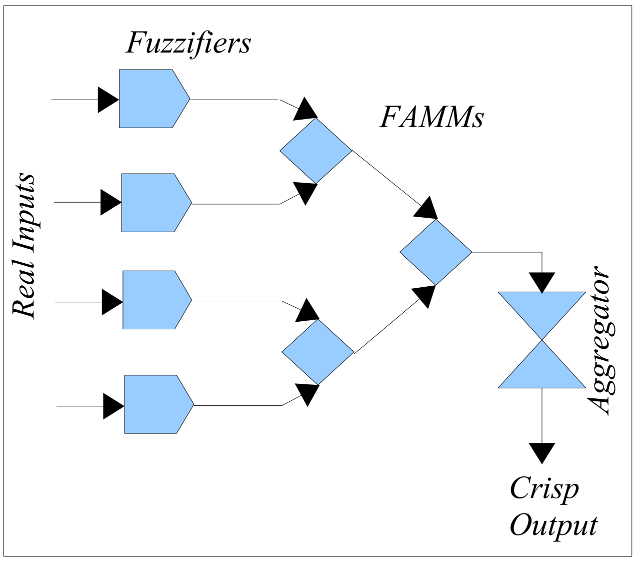


Figure 10: Convergent Cascade of Fuzzy Systems

We can see that would be possible to analyse a very large number of input variables in this manner, and that due to the nature of the Fuzzy process, that this would be very fast and consume very little memory because we must only consider one 2D FAMM at any one time. The inputs would converge to a final 2D FAMM, and a single, crisp output value would be produced through the aggregation procedure.

4.4 Other Enhancements Possible with the Strategy Layer

Use of the strategy layer that we have introduced in this report opens further extensions to the robot soccer system.

The role heuristics could easily be stored in memory, and passed to the path-planning layer, where they could be used again to prioritise the order of path planning for the robots on the team.

The highest-priority robot would have its path planned before the other robots, and path points would be stored in memory with an *estimated time of arrival* for each point. Subsequent paths, for the other robots on the team, would be planned to *avoid crossing* the paths of higher-priority robots at the same time.

A simple enhancement such as this would considerably improve the cohesive ability of the robot system, and reduce the occurrence of own-team collisions.

5. A Novel Motor Control Layer for Mapping a Fuzzy Abstraction of Motion to Twin-Motor Instructions

5.1 Background

Presented here is a new algorithm for mapping the new Fuzzy Logic robot navigation layer outputs into usable motor control instructions for vehicles with a twin-drive actuator configuration. This section is written as a practical guide for immediate implementation of a capable motor control layer into an intelligent vehicle, and includes complete derivation of relevant formulae.

Of interest to us here is the bottom layer in *Figure 3* – the motor control system. As input, the motor control layer is passed a defuzzified *forward speed* and *rotational speed* from the Fuzzy control layer. The motor control layer maps the defuzzified speeds to specific motor velocities, expressed in corresponding voltages, which it outputs to each motor (the actuators in *Figure 3*).

5.2 Aim

The aim of the motor control system outlined in this section is to facilitate a smooth, arcing path of motion for two-wheeled, fixed-steering, and tracked autonomous vehicles – allowing these vehicles to steer whilst driving forwards by blending left and right motor speeds.

The advantage presented by the proposed system is that vehicles of this type are able to move more efficiently; in terms of robot soccer, this means that robots are able to move much more quickly than those robots that are constrained to turning on the spot before moving.

The approach we present here enables *smooth-path* motion calculation that not only supersedes the discontinuity typical during transition between curved and straight components of motion in the popular Reeds-Shepp method [6], and the Circle-Line method, reported by J-H Kim [1] as being the most popular method of robot motion, but also requires none of the calculation overheads.

5.3 Assumptions

Because the time period considered between motor control calculations is very small (approximately 0.0303 seconds in our system) we can disregard acceleration, and assume that the arc of motion for the vehicle within our time period will always be an approximately circular sub-section of a larger parabolic path.

Disregarding that the arc segment is actually parabolic allows us to greatly reduce the amount of calculation required and therefore also the latency between calculation and action.

We have also disregarded inclusion of friction, robot momentum, angular motor momentum, inertia and other forces in our motor-control calculations, as it is not feasible in our system to gather measurements for all of these external and internal forces.

These additional forces can, however, be accounted for implicitly to some degree by tuning the fuzzy sets of the fuzzy logic layer of the robot system – ideally for individual robots.

There is scope for motor control refinement for individual robots, to take into account variable or unmeasurable forces, by adjusting fuzzy sets through learning algorithms.

5.4 Method

The properties of the circular arc of motion of the robot; that is, the angular velocity *ω* and radius *r* of the arc, are determined by the magnitude and ratio between magnitudes of the inner and outer wheel and track velocities V_O and V_I, proportionate to the wheel separation distance *d*. Refer to *Figure 11* for illustration.

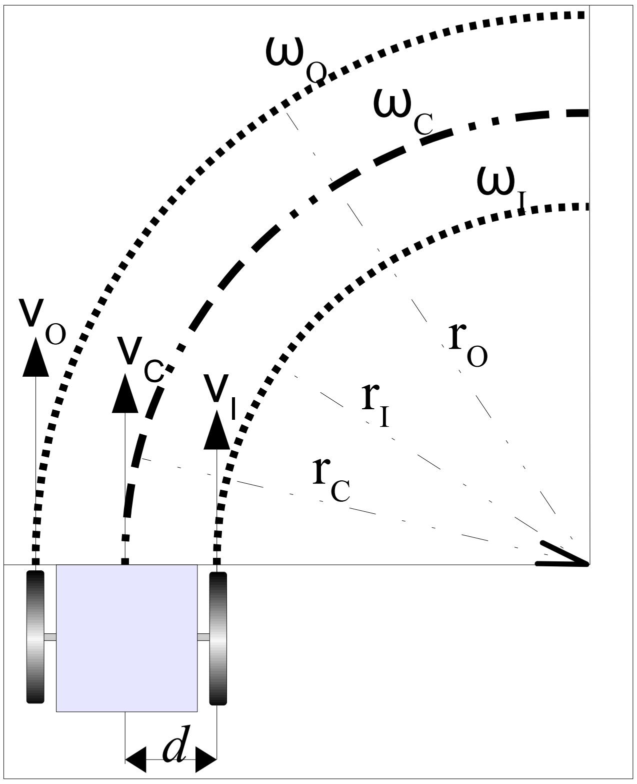


Figure 11: Rotational variables for a twin-drive vehicle

Rotational physics formulae can be used to determine the left and right motor velocities required to reproduce the forward velocity and angular velocity required by the fuzzy layer's abstraction of motion.

We can consider the following linear-angular relation - the definition of angular measure in radians:

$$s=\theta r \quad \text{(eq. 1)}$$

Where *s* represents the distance a point in moved along a circular arc of angle *θ* and with radian measure *r*, we can derive the equation with respect to time *t*, with *r* set constant:

$$\frac{\delta s}{\delta t}=\frac{\delta \theta}{\delta t}r$$

We can see that *ds/dt* is the linear speed – the magnitude of our linear velocity *v,* and *dθ/dt* is our angular speed. We can now express our equation in the form:

$$v = \omega r$$

Again, we express *ω* in terms of the radian measure. We can now derive our clockwise and anticlockwise motor instructions.

The Fuzzy Reactionary Layer of the system (*see Figure 3*) must feed information indicating the direction to turn the vehicle, either as a signed output angle value, or as the value of a flag variable. Knowing the direction to turn, we can simply apply the *outermost* and *innermost* motor instructions to the appropriate motor with a simple switching statement.

Taking the inputs passed from the Fuzzy layer; *forward speed* and *angular velocity*, we can say that:

$$v_C = \textit{forward speed}$$
and
$$\omega_C = \textit{angular velocity}$$

Where v_C and ω_C are the forward speed and angular velocity for the centre of the vehicle. We however, are interested in the variables at the *actuators* (wheels or tracks) of the vehicle.

We next establish the radius for the arc motion of from the centre of the vehicle:

$$r_C = \frac{v_C}{\omega_C}$$

With the radius for the arc of motion for the centre of the vehicle we can calculate the other radii using a known actuator distance *d* from the centre of the vehicle. We are assuming here that the actuators are equally separated from the centre of the vehicle.

We know that all three radial velocities (ω_C, ω_O, and ω_I) must be equal. Therefore we can now establish the required motor velocities using our derived *r* and *ω* values:

$$v_O = \omega_C \left(\frac{v_C}{\omega_C} + d \right)$$
and
$$v_I = \omega_C \left(\frac{v_C}{\omega_C} - d \right)$$

Which can be further reduced:

$$v_O = v_C + d \cdot \omega_C \quad \text{(eq. 2)}$$
and
$$v_I = v_C - d \cdot \omega_C \quad \text{(eq. 3)}$$

For cases where the Fuzzy layer dictates that the required required linear velocity is very low (close to zero) we can take advantage of the special ability of some twin-drive vehicles to oppose the direction of the left and right motors.

The behaviour can not be smoothly blended with the *turn and drive* behaviour (*equations 2 and 3*) as the inertia and stress created on a motor during transition from forward to reverse motor direction is very difficult to manage.

The change in relative location for centre of rotation reverses direction as one of the motors is reversed, which requires another dimension of complexity to be accounted for by the Fuzzy navigation layer.

The number of potential mappings for motor instructions is greater than one for each forward speed and angular velocity pair. Because our system must be very fast, it benefits from the reduction of this sort of complexity.

For these reasons, we have completely separated our navigation logic into two cases where our vehicle must either:

1. Simultaneously steer and drive
2. Turn on the spot

This separation requires a high-level switching statement, based on the angular velocity and linear speed requirements, to decide which means of motivation to employ at each considered instance.

For the case where the vehicle must turn on the spot, we can say based on our derivation of *equation 1* that:

$$v_O = -v_I = d \cdot \omega \quad \text{(eq. 4)}$$

Where we have substituted our radius *r* for *d*, because we are turning about the centre of our vehicle. We must then simply set the left motor equal to minimum of the two velocities (v_O and v_I from *equation 4)* if we are turning anticlockwise, and vice versa for the case when we are turning clockwise.

We can compound our working into a consolidated algorithm (*Algorithm 1*).

Algorithm 1 *Motor Control Algorithm for Two-Motor Autonomous Vehicles Using a Fuzzy Navigation Abstraction*

The input is a *forward speed* v_C and an *angular velocity* ω from the Fuzzy navigation layer. The Fuzzy layer must also appropriately set a flag *TURNLEFT*. We must also know the distance from the centre of the vehicle to each wheel *d*.

```
1. if the linear velocity value $v_C$ is close to
   zero then jump to step 4

2. if the angular velocity value $\omega$ is close to
   zero then
      set both $v_L$ and $v_R$ equal to $v_C$
      and terminate algorithm
   else
         Proceed to step 3.

3. Using equations 2 and 3 calculate linear
```

velocities v_A and v_B:

$$v_A = \omega \cdot (v_C / \omega + d)$$
$$v_B = \omega \cdot (v_C / \omega + d)$$

Jump to step 5.

4. If the vehicle is turning on the spot, we calculate opposing velocities for the motors using *equation 4*:

$$v_A = -v_B = d \cdot \omega$$

Proceed to step 5.

5. Determine the motor velocities for specific motors (v_L and v_R):

```
if TURNLEFT then
    $v_L = min(v_A, v_B)$
    $v_R = max(v_A, v_B)$
else
    $v_L = max(v_A, v_B)$
    $v_R = min(v_A, v_B)$
```

5.5 Considerations

The are several factors to consider in implementation of the motor control algorithm presented in this report:

- Disparity in performance between left and right motors
- Variations is motor response from vehicle to vehicle
- Variation in response of motors during operation
- Left and right actuators operating on different surfaces

Depending on the magnitude of error experienced by a particular vehicle, additional coping mechanisms may or may not be required to correct the motion of the vehicle.

Disparity between left and right motor response is inevitable, and may be correcting for by adding some additional offset to the velocity instruction sent to a particular motor.

Individual vehicles will exhibit different operating characteristics, re-enforcing the need for fuzzy set definitions to be adjustable for individual vehicles.

However, it is likely that the variation in response for a particular motor will be non-linear over a range of velocity instructions (Voltages) and indeed also over a period of operation as the motor temperature increases. This will result in some error being expressed in the path of motion of the vehicle.

This error is typically only minor in nature in a Fuzzy-controlled system, due to its dynamic nature, as the vehicle will continue to be sent course correcting instructions as long as it is not positioned with the desired orientation.

In environments more complex than the robot soccer field, it is certain that actuators will at some point be operating, without prior information, on different types of terrain, and will produce unintended motion. Although this scenario should not prevent the vehicle from operating, again due to the dynamic nature of the Fuzzy system, the effectiveness of the system could be improved with an intelligent actuator-monitoring and adjustment approach.

5.6 Results

We have implemented the new motor control layer in the robot soccer game and observe that the system is successful. The robots retain the ability to quickly turn on spot when advantageous, but otherwise produce more effective motion *en route* to a destination.

The motor control algorithm presented here has driven the soccer robots to another evolutionary level; the time taken to complete a path of motion can be reduced significantly, which is all important to speed-critical scenarios such as robot soccer.

It must be noted that the specific motor response of the soccer robots requires significant tuning to not only the Fuzzy navigation layer but also to the path planning layer before the desired motion can be achieved.

The degree of tuning and optimisation required due to non-simulated real-world factors presents scope for a machine learning or Genetic Algorithm. Messom [5] discusses Genetic Algorithms for motion control, tuning the same twin-motor actuators that we are using in our system.

The motor control algorithm presented here consumes minimal processing time, and has been proven to successfully map a fuzzy abstraction of motion to real left and right motor instructions for two-wheeled soccer robots.

This research presents considerable potential for application to other vehicles of this type, particularly tracked vehicles using two motors, and there is scope for several optimisation and adaptive extensions to this system.

6. Simulation

The real robot control program is a highly unreliable and unstable environment, and does not provide a controlled environment beneficial for testing.

To experiment with and develop the robot system presented in this report, it was necessary to create a simulation of the robot soccer environment. This would allow us to simulate the entire layered robot system prior to implementation.

We have created a new 3D simulated

environment for the development and demonstration of new robotics algorithms. A 3D simulation gives us the advantage of being able to model 3D ball physics, and is also a very portable presentation piece, that can communicate research in an entertaining and visually appealing manner to any audience.

Building a 3D simulation required the creation of a graphics engine able to take spatial information about the environment, project that information into a 3D *scene* based on a perspective of the environment, and apply the appropriate mathematical matrix transformations to flatten that scene into a 2D picture that could be output to a monitor display.

To produce a *real-time* effect, the graphics engine would have to make these transformations on a regular basis; at least 33 times per second on an average PC.

6.1 Graphical Environment

6.1.1 Graphics Libraries

To support the creation of the 3-D graphics engine, there were several major programming libraries available:

- *OpenGL*
- *DirectX*
- *Java3D*

We decided to use DirectX libraries for the graphics engine, based on the availability and reliability of documentation.

We were also able to program with DirectX using C++, which we as we could then develop functional robot control modules in simulation, and directly import these into the real robot control program.

We developed the simulation program (illustrated in *Figure 12*) with an approach that would make it relatively simple for future Massey students to interpret, should they choose to use it as a platform for future projects.

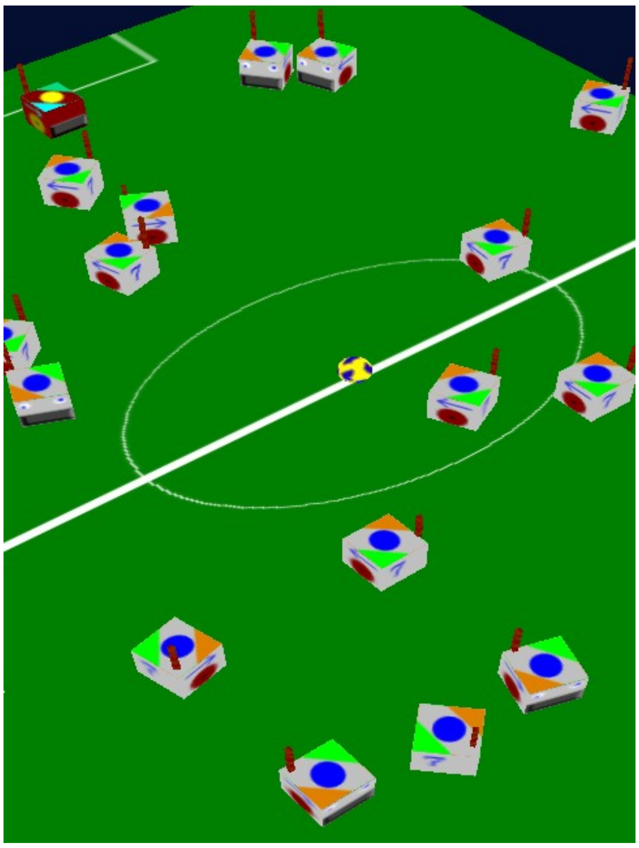


Figure 12: The 3D Robot Soccer Simulation

6.1.2 Graphical Effects and Features

For the demonstration version of the simulation, we developed and incorporated several modern graphical effects that make the rendered output more visually appealing:

- Alpha-Testing
- 32-bit Anti-Aliasing
- 1280x1024, 32-bit Texture Rendering
- MIP Mapping
- Improved Triangle Occlusion Culling
- Field-of-View Frustum

With the exception of MIP Mapping all of these effects are minor enhancements to the graphics engine.

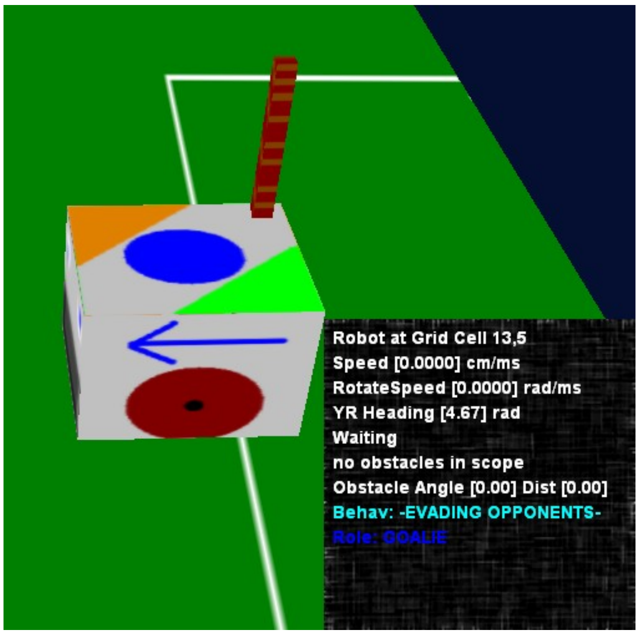
MIP Mapping, which is from the Latin *multim im parvo*, meaning "*many things in a small space*". MIP Mapping is a technique that reduces the grainy appearance of textured objects that are farther away from the viewpoint by scaling textures to a range of different resolutions. The textures then appear to 'blur into the distance' as they are farther away from the viewpoint.

The Field of View *Frustum* is a performance-enhancer that calculates which objects are outside of the field of view of the viewpoint, and then excludes those objects from the rendering pipeline.

6.1.3 Graphical Debugging

Debugging DirectX-based graphical programs is exceedingly difficult, as they do not have the ability to print out simple error message text in the manner of common console applications.

We found that it was beneficial to have a whole range of internal data printed out during simulation, and so devised a system to write important debug output to a log file. To display data on screen during program runtime we created an additional module that would display text on special overlay panels.



```
Robot at Grid Cell 13,5
Speed [0.0000] cm/ms
RotateSpeed [0.0000] rad/ms
YR Heading [4.67] rad
Waiting
no obstacles in scope
Obstacle Angle [0.00] Dist [0.00]
Behav: -EVADING OPPONENTS-
Role: GOALIE
```

Figure 13: Panel displaying internal variable values for selected robot

We created a panel to display various internal states and variables for a selected robot (*Figure 13*), including all of the special variables relating to the algorithms in the robot navigation system.

We created other panels to display all of the general debug information to the screen, including the number of 3D models successfully loaded, files in use, and other information.

6.2 3D Models

In order to create a representation of the soccer robots for the 3D space of the simulation environment it was necessary to create 3D models using CAD (*Computer Assisted Design*) software.

The challenge, we discovered, was to find a CAD package that was able to export 3D models to the DirectX-compatible format that we were using in the graphics engine of the simulation.

We were able to configure the CAD packages that we used so that metrics used by the modelling packages to determine the dimensions of the robot models agreed with the metrics we were using in the simulation.

Because of this we were able to create accurate scale models of the real robots, and thereby improve the accuracy of the simulated robot soccer environment.

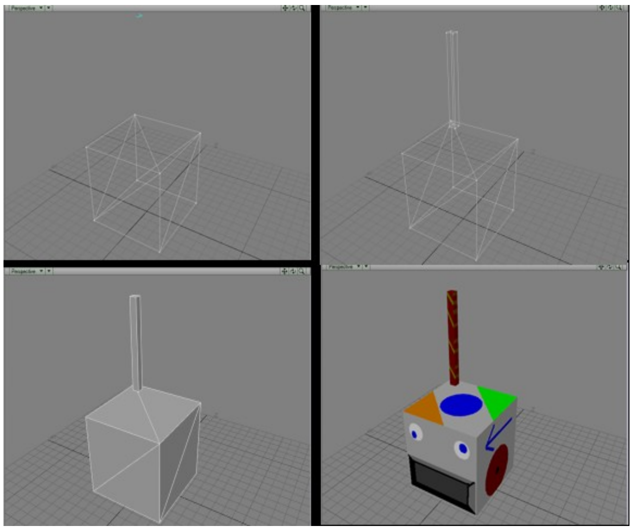


Figure 14: Constructing soccer robot models for simulation in a CAD program

Figure 14 illustrates the steps in the design process followed to create robot models. We wanted to keep the models as simple as possible to reduce the CPU requirements of the rendering engine.

You can see in *Figure 14* that we created a basic cube to represent the robot, and added several more polygons to represent the antenna, and finally applied simple bitmaps to all of the surfaces to provide recognisable detail.

6.3 Algorithm Implementation

Implementing the hybrid navigation system within the 3-D simulation proved to be a considerable challenge. Implementing the Fuzzy Logic and A* navigation systems separately proved to be challenging, however combining the algorithms to form a hybrid system required extensive modification to both systems, and an extensive volume of testing and refinement.

In order for the A* algorithm to operate in a simulation of the real environment, we had to create a supplementary algorithm to generate a map of the robot soccer environment, based on the dimensions of the playing field, and the known sizes and positions of all of the robots in play (which would be identified by the machine vision system in the real game).

The A* algorithm could then use the constructed map as a large graph to search; generating paths for the robots. *Figure 15* illustrates an example of a path, as a series of points, generated by the dark robot which is trying to reach the area to the bottom of the picture, and avoid colliding with the other robots.

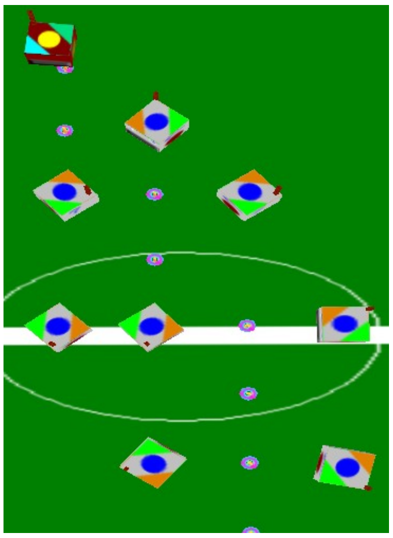


Figure 15: Path generated by the A algorithm*

Initially the two navigations systems conflicted over feasibility of calculated paths of movement. Extensive refinement was required to both systems in order to ensure that paths generated by the A* algorithm would be accepted as feasible paths of motion by the Fuzzy Logic system.

Implementing the algorithm for opponent *evasion* required additional infrastructure; another supplementary algorithm was required, to create a map of relative areas of *undesirability.*

The opponent evasion algorithm inculcates into the navigation system a predictive ability; areas that opposing robots are more likely to move into, such as the area immediately in front of a moving hostile robot, are rated at a higher level of undesirability than those areas less likely to be moved into; reflecting the reduced threat to those areas.

Robots planning paths with evasive behaviour can now make a prediction as to a *best* path balancing total path length with level of threat. Evasive robots continually re-assess the path to take into account the activities of opposing robots.

We were also able to implement our *Best Man for the Job* role-assignment algorithm into the 3D simulation. To better facilitate this layer of intelligence, we created a tessellated 3D ball, and a simple ball-physics engine into the simulation to allow the ball to be rolled and bounced about the field by the robots.

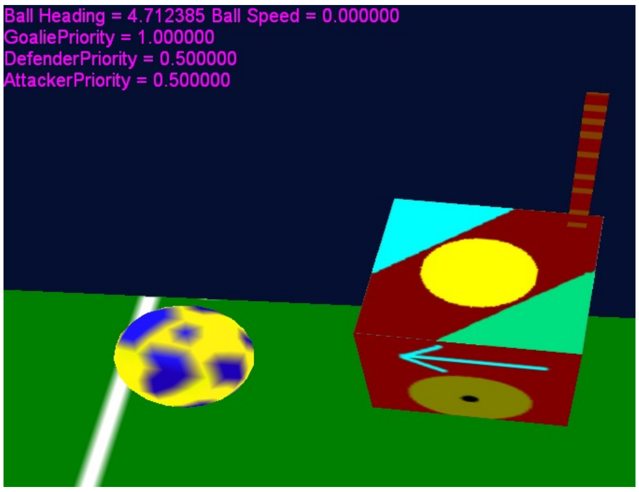

Ball Heading = 4.712385 Ball Speed = 0.000000
GoaliePriority = 1.000000
DefenderPriority = 0.500000
AttackerPriority = 0.500000

Figure 16: The Simulated Goalie Robot with the Ball

With a dynamic, moving ball, we were able to extract meaningful *ball speed* and *ball position* variables that could be used by the role-assignment algorithm. *Figure 16* illustrates the ball addition to the simulation with printed dynamic ball-related data appearing in the top of the figure.

6.4 Extensions to the Simulation

It was our intention to create a robot soccer simulation that would be useful beyond the scope of the research project that we undertook. We created the entire program in a modular fashion so that different modules could be added to or removed from the program.

6.4.1 All-Terrain Simulated Robot Environment

We managed to develop a parallel implementation of the simulation concurrently to the robot soccer simulation. This second simulation was used to test the feasibility of our navigation and other robot systems in an all-terrain environment. We were able to successfully demonstrate working implementations of our robot system in environments with uneven terrain (refer to Figure 17).

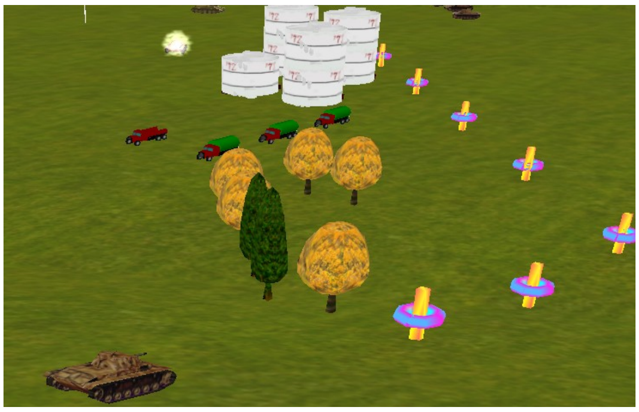
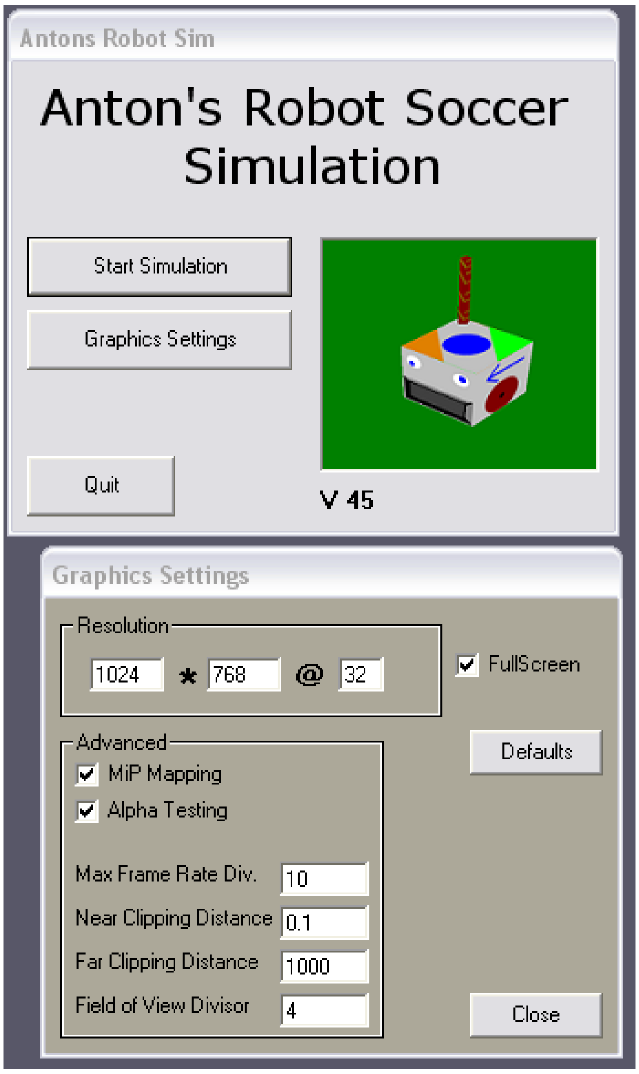


Figure 17: A robot navigates a hill using Fuzzy-A in the all-terrain simulation*

6.4.2 Graphical User Interface


Antons Robot Sim

Anton's Robot Soccer Simulation

Graphics Settings

Resolution: 1024 * 768 @ 32

[✓] FullScreen

Advanced

[✓] MiP Mapping

[✓] Alpha Testing

Max Frame Rate Div. 10

Near Clipping Distance 0.1

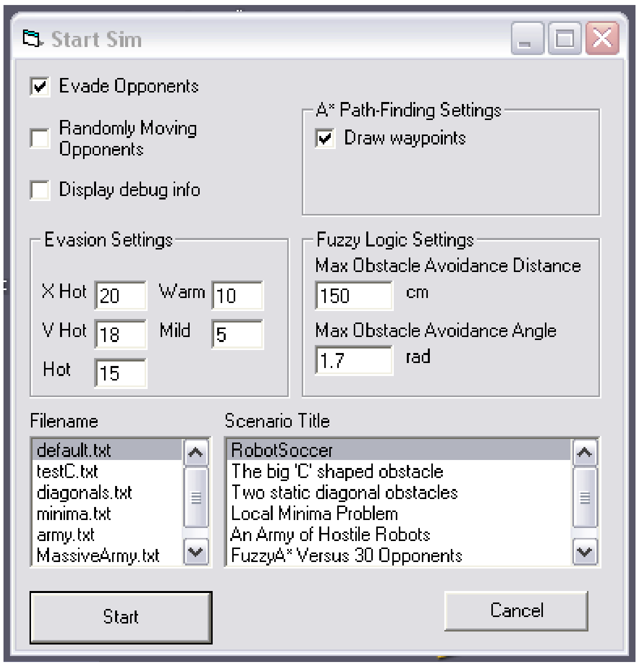
Far Clipping Distance 1000

Field of View Divisor 4

Figure 18: The Graphical User Interface

Because the simulation will be used by other students, for development or for demonstration, we developed a Graphical User Interface (GUI) (refer to *Figure 18)*, to allow very simple customisation of the simulation variables and graphical settings.

In order to accommodate repeated, comparative testing of the same situation, or demonstration of a particular scenario, we developed module to save and load specific soccer scenarios to and from simple flat files (*Figure 19*). We have also created a similar module for loading sets of 3D models into the simulation.


Start Sim

[✓] Evade Opponents

[ ] Randomly Moving Opponents

[ ] Display debug info

A* Path-Finding Settings

[✓] Draw waypoints

Evasion Settings

X Hot 20 Warm 10

V Hot 18 Mild 5

Hot 15

Fuzzy Logic Settings

Max Obstacle Avoidance Distance 150 cm

Max Obstacle Avoidance Angle 1.7 rad

Filename	Scenario Title
default.txt	RobotSoccer
testC.txt	The big 'C' shaped obstacle
diagonals.txt	Two static diagonal obstacles
minima.txt	Local Minima Problem
army.txt	An Army of Hostile Robots
MassiveArmy.txt	FuzzyA* Versus 30 Opponents

Figure 19: The Scenario Loader Front-End

7. Results

We have managed to implement and demonstrate our robot system architecture as both feasible and cohesive in simulation.

We managed to adapt the real robot control program to interface with our new robot system, based on a façade design paradigm (refer to *Appendix 5* for our working schematic diagram), and we have been able to demonstrate that our system works, and is able to control the real soccer robots at Massey University.

We believe that we would need to write a new robot control program from scratch in order to provide the necessary empirical and conclusive results that would attest to the superiority of our new algorithms and robot system.

The ultimate test, however, would be for our system to be taken to international competition against the world leaders in this area.

8. Conclusions

We have successfully developed a complete robot system incorporating new hybrid Fuzzy intelligent navigation systems.

We were able to design and produce all of the layers of our proposed robot model, and achieve an acceptable level of cohesion between these elements.

We have developed a new 3D robot simulation, incorporating cutting-edge graphics technology.

We have designed a unique strategy-level Artificial Intelligence for coordinating mobile robots dynamically. We believe that this model warrants further additions, and is transferable to other, even more complex applications.

We have designed a new, working motor control system, specially designed for the unique Fuzzy Logic based navigation layer that we are using in our system.

9. Future Works

We have proposed to develop novel, efficient hybrid intelligent algorithms for fast autonomous vehicles simultaneously navigating through and mapping unexplored terrain as part of PhD study (*refer to Appendix 4* for full proposal).

The proposed research will focus on vehicles mounting intelligent stereo-vision systems, and will target the development of new real-time mapping algorithms and, in particular, investigate the application of Fuzzy Logic to this research.

10. References

[1] J-H Kim, “*Robot Control Program III*”, Intelligent Control Lab, Korea Advanced Institute of Science and Technology, 1999.

[2] A. P. Gerdelan, N. H. Reyes “*A Novel Hybrid Fuzzy A* Robot Navigation System for Target Pursuit and Obstacle Avoidance*”, Proceedings, First New Zealand Korean Joint Workshop on Advance of Computational Intelligence Methods and Applications, 2006, AUT University, Auckland, New Zealand.

[3] A. P. Gerdelan, N. H. Reyes "*Synthesizing Adaptive Navigational Robot Behaviours using a Hybrid Fuzzy A* Approach*", Journal, 'Computational Intelligence, Theory and Applications', Springer, (Berlin&Heidelberg), 2006, pp. 699-710.

[4] G. Sen Gupta, C. Messom, S. Demidenko, "*State Transition Based (STB) Role Assignment and Behaviour Programming in Collaborative Robotics*", Proceedings of the Second International Conference on Autonomous Robots and Agents, 2004, pp 385- 390.

[5] C.H. Messom, "*Genetic Algorithms for Autotuning Mobile Robot Motion Control*", Res. Lett. Inf. Math. Sci, 2002, (3), pp 129-134, ISSN 1175-2777.

[6] J. A. Reeds, L.A. Shepp, "*Optimal paths for a car that goes both forwards and backwards*". Pacific Journal of Mathematics, 145(2):367-393, 1990.

[7] N. H. Reyes, "*Color-Based Object Recognition – Analysis and Application*", PhD dissertation, De La Salle University, Manila, Philippines, 2004.

11. List of Appendices

***Appendix 1* – Unpublished Paper**

A. P. Gerdelan "*A Novel Motor Control Algorithm for Two-Wheeled and Caterpillar-Tracked Autonomous Vehicles Using a Fuzzy Navigation Abstraction*".

***Appendix 2* – Conference Paper**

A. P. Gerdelan, N. H. Reyes "*A Novel Hybrid Fuzzy A* Robot Navigation System for Target Pursuit and Obstacle Avoidance*".

***Appendix 3* – Journal Paper**

A. P. Gerdelan, N. H. Reyes "*Synthesizing Adaptive Navigational Robot Behaviours using a Hybrid Fuzzy A* Approach*".

***Appendix 4* – PhD Proposal**

A. P. Gerdelan "*Stereo Vision Based Hybrid Intelligent Mapping: Proposal for PhD Research for 2007*".

***Appendix 5* – Diagram of Our Robot Control Program Façade Merger**