MIROSOT TECHNICAL REPORT

CONTENTS

1.0 INTRODUCTION

This document is the MiroSot Technical Report, and aims to cover the knowledge and understanding to support the findings presented in the main group report.

1.1 OVERVIEW OF MIROSOT FOOTBALL

MiroSot Robot Football is an automated competition played between 2 teams of 5 cube-based robots, each 7.5cm Square, with no human interaction. Much like the traditional game, the aim of the competition is to score as many goals in the opposition's half as possible using pre-planned tactics and strategy, within a fixed time frame.

Played on an 220cm x 180cm pitch, each team uses an overhead vision system to accurately determine the location and orientation of the players based on colored ID barcodes positioned on the robots top. A computer system then runs a pre-written strategy to determine their next move and broadcasts the movement directions to the robots via wireless control.

This year, Warwick Mobile Robotics is continuing to evolve and develop computer strategies for MiroSot however is halting major mechanical hardware development. This decision was taken due to a multitude of reasons, the key factors being a lack of national based competition this year and limited scope for hardware improvement above that of last year's team.

1.2 REPORT STRUCTURE

The technical report is broken into the following sections, representing the various areas of work required to meet the objectives:

- Hardware and Systems
- Computer Strategy
    - Low Level Strategy (movement algorithms)
    - High Level Strategy (tactics, assigned roles and testing)

Each section opens with the concept, outlining the motivation and rationale behind many of the decisions taken. Following this are the technical diagrams, prose and calculations that form the main body of this report.

For the final conclusions, please see the main group document contained within the portfolio.

2.0 THE NEW COORDINATE SYSTEM

One of the initial problems taking over the project from the previous year was the general lack of documentation and poor labelling of the pitch coordinate system. It was clear that throughout the code various constants had been defined to represent the key x and y dimensions, however were often labelled ambiguously.

As a result, one of our first tasks was to completely redesign the naming system so that it was clear what coordinate value in the code referred to what physical dimension within the play area.

Not only did this allow for a faster coding process throughout the project, however future teams will benefit from the reduced learning curve associated with the pitch data.

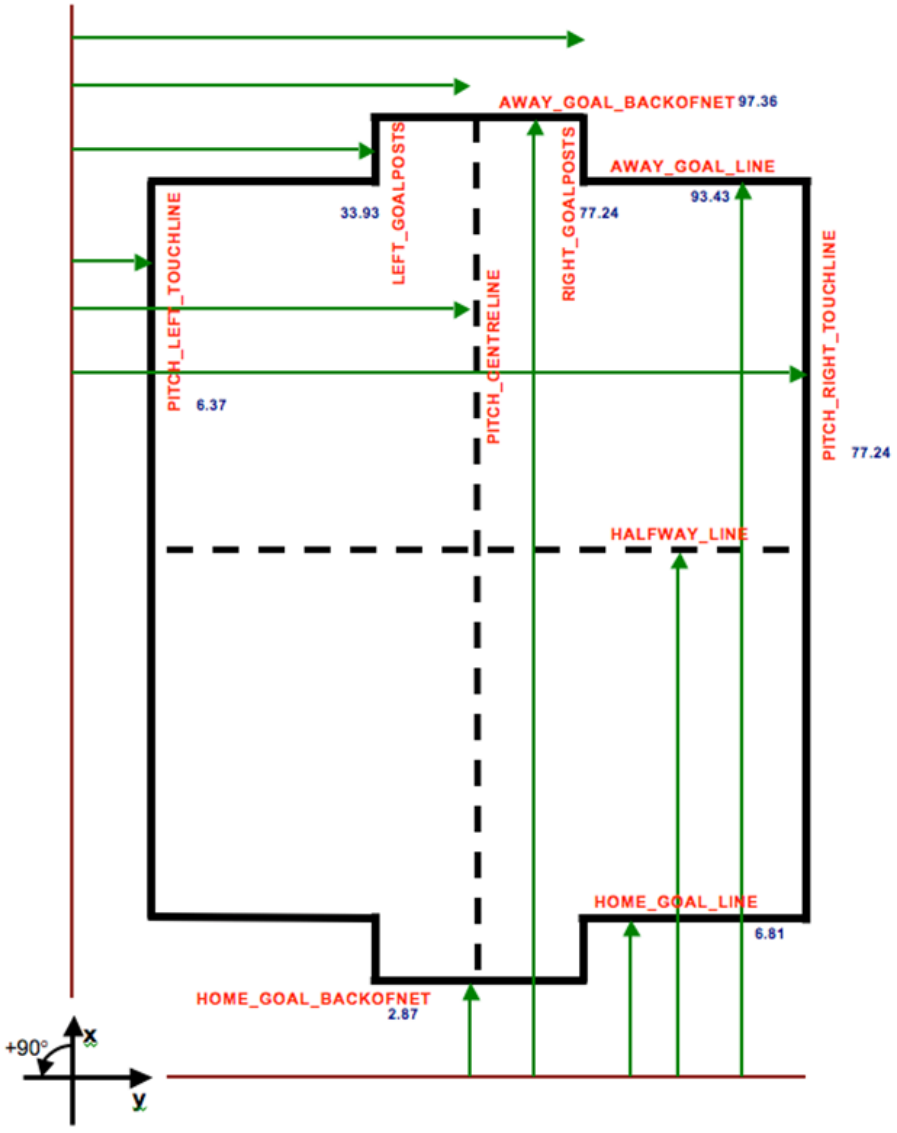

Figure 1 - New Coordinate System

3.0 HARDWARE

The current generation of MiroSot robots utilised in the project are the 2006/2007 designed Evo4's, part of the Warwick Evolution series that have been developed over the past four academic years.

Figure 2 – Evolution 4 Robot and timeline

As set out in the initial plan, the hardware would not prove the primary concern for development this year. Despite this however, basic maintenance and upkeep is still required and this section will address hardware modifications and additions that were performed over the past academic year.

3.1 BATTERIES

The primary hardware revision this year was the sourcing and replacement of all batteries in the Evo4 robot series.

Following heavy use and poor charging patterns over the previous year, none of the robots were holding significant charge to be able to perform an entire 10-minute game at near full performance.

Finding replacement Li-Po batteries that were both the correct size as well as providing sufficient mAh was a challenge due to the tight dimensions that the Evo4 chassis design provided.

The correct balance of size and power was found in the Kokam 7.4V 350mAh range. A selection of 10 was purchased to allow for not only a fresh set but to have spares as well that could be changed at half time, if the need occurred.

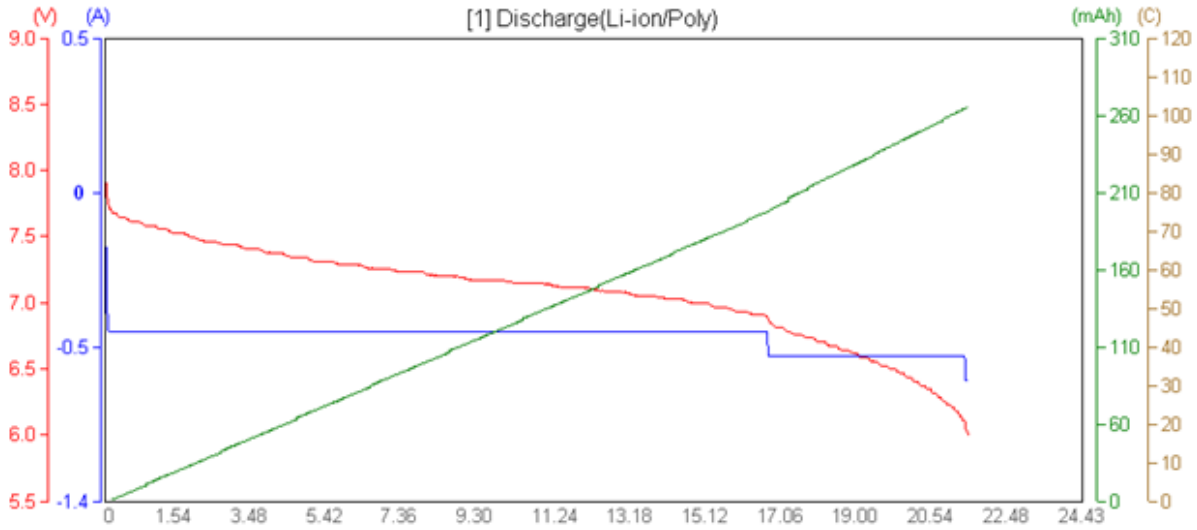

Figure 3 - Battery Test Graph

4.0 ID PATCHES

The ID patches integrity is essential to the correct operation of the vision feedback system. The method employed by the previous team was to use printed patches on standard paper and attach them to the top metal surface of the robot using an adhesive.

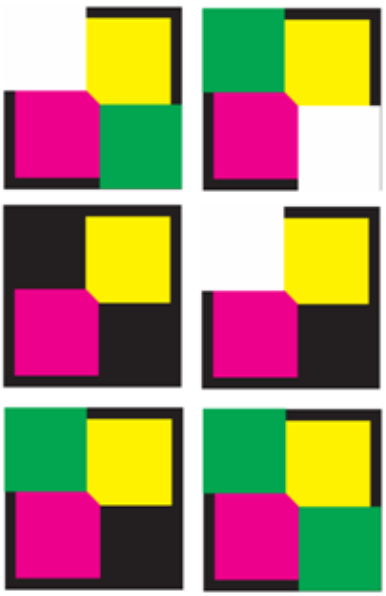

Figure 4 - Yellow ID card

The problem here was that after a number of games the patches soon became scuffed and torn. Having sustained prolonged damage, the vision system then fails to recognise the robots.

We investigated various more permanent solutions before settling on printed 3mm foamex board.

These were printed and manufactured using WarwickPrint at the University of Warwick. Whilst the quality, thickness and finish was excellent, following testing the solution was found to be subpar exhibiting the following problems:

- The foamex board used has a subtle reflective quality compared to the matt finish of using standard paper. As a result the colouring detected at various points on the pitch would vary and prove difficult to calibrate.

- The patches, whilst much more durable than paper, still succumbed to considerable damage in a game scenario.

The optimum solution was found to be using the foamex board as a mounting point for a matt printed-paper version. This was found to sustain the least damage for the longest period of time. Regular replacing of the ID cards is recommended for optimum vision performance.

5.0 GENERAL MAINTENANCE

In addition to the previous hardware upkeep tasks, throughout the year general maintenance was required on all the Evo4 and Evo3 series of robots.

This generally required replacing bolts and screws as appropriate and re-soldering loose wires.

6.0 LOW LEVEL STRATEGY

6.1 MOVEMENT ALGORITHMS

6.1.1 STRAIGHT LINE ALGORITHM

6.1.1.1 OVERVIEW

Coding the robot's straight-line movement was one of the most complex development tasks encountered throughout the course of the year.

Logically, setting both the left and right wheel velocities to the same speed would lead to straight-line movement. However, no system is perfect and due to various inconsistencies in motors, wheel tyre friction, pitch surfaces etc, it is necessary to write an algorithm that uses corrective feedback to achieve this aim. Therefore the following series of pages describe the key ideas behind this algorithm.

6.1.1.2 INTRODUCTION

There are 3 key stages to movement in a straight line:

- ***Orientation (to face the target)***
- ***Individual Wheel Velocity Adjustment (to maintain a fixed line trajectory)***
- ***Deceleration (to not overshoot the target)***

6.1.1.3 ORIENTATION

Travelling in a straight line can be achieved using either forward or reverse motion. As a result, the maximum angle the robot has to turn before being in an orientation that is capable of travelling to the target is 90°. The first stage in this process is therefore turning on the spot to face the destination.

Figure 5 - Straight line movement

6.1.1.4 INDIVIDUAL WHEEL VELOCITY ADJUSTMENT

Once moving towards the target, the next stage is maintaining the orientation. This is done via applying a modifier to increase the velocity of the outside wheel in the direction that requires correction. The current and desired angles are monitored and the modifier is scaled based on the difference between the two. The greater the angle is out by, the greater the modifier applied to correct it.

6.1.1.5 DECELERATION

To ensure the target isn't overshot, a deceleration modifier is required to slow down the robot upon approaching the desired location.

6.1.1.6 LOGIC DIAGRAM

These ideas can be pictured in an early concept logic diagram.

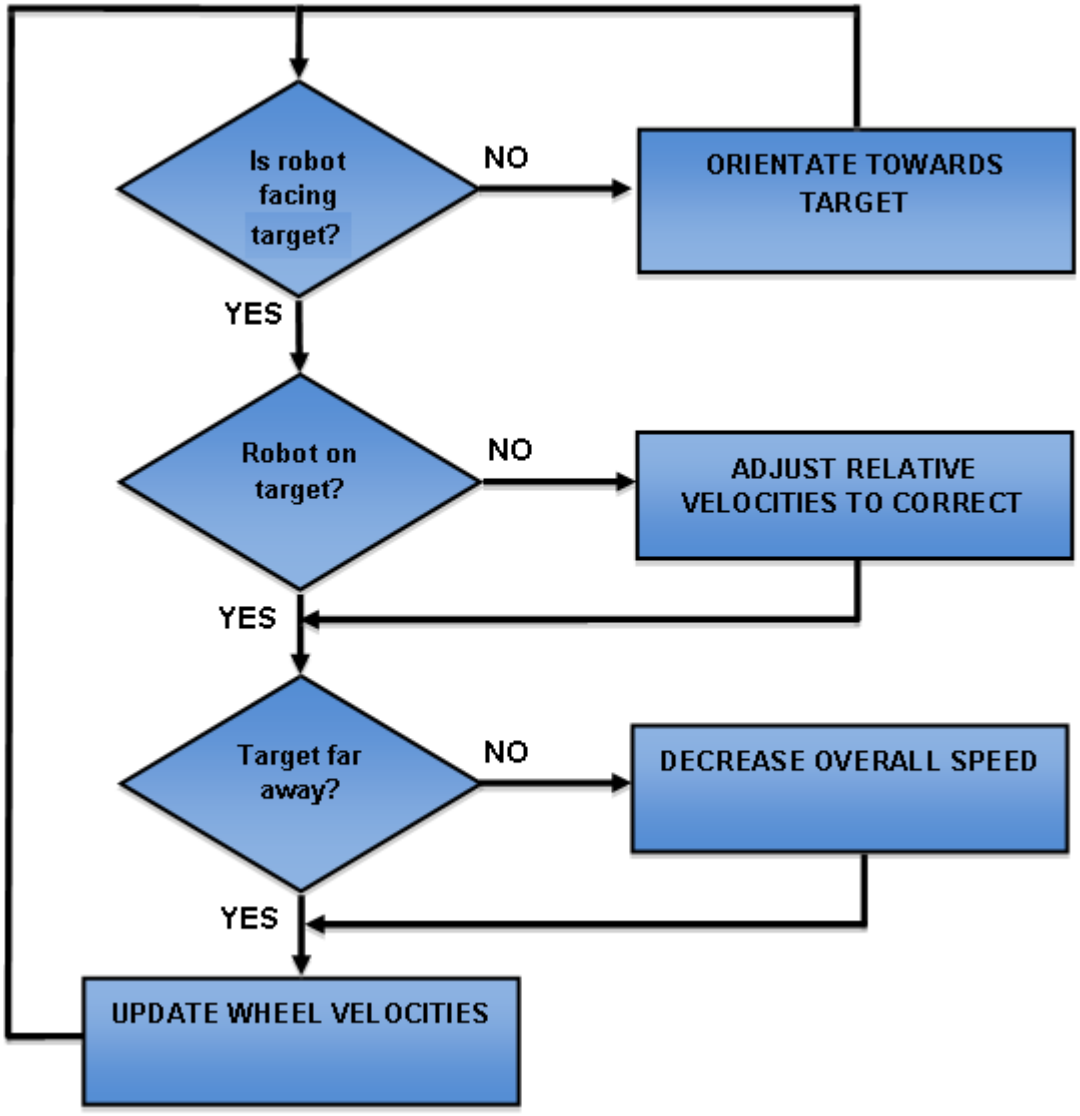

Figure 6 - Straight Line Logic Diagram

7.0 ORIENTATING THE ROBOT

When the algorithm is first run, the robot must orientate itself.

This stage is also run if at any point the difference between the current and the desired angle exceeds that of 25°. This was added as it was observed through testing that beyond this limit it was more efficient to stop the robot and orientate it, rather than attempt to correct it by modifying wheel velocities whilst moving.

Originally orientation was achieved by setting the wheel velocities to be an equal and opposite small value. This allowed relatively accurate turning before starting the straight-line movement. The main problem with this however was that it was slow and compromised the efficiency and purpose of the algorithm.

Likewise, orientating at too high a velocity faced similar problems. This scenario would often lead to overshooting and needing to re-orientate a second time, again causing an inefficient algorithm.

Taking the middle ground and applying a function based of the square root of the angle was found to achieve a good balance of speed and accuracy. The graph below indicates the "spin" value that will be applied equally and opposite on the relative wheels.

For example, when the angle is at a maximum of 90° off target, the velocities of both wheels would be 9.5 and -9.5 respectively. As the angle narrows, the speeds of both would be reduced as indicated by the curve pictured.

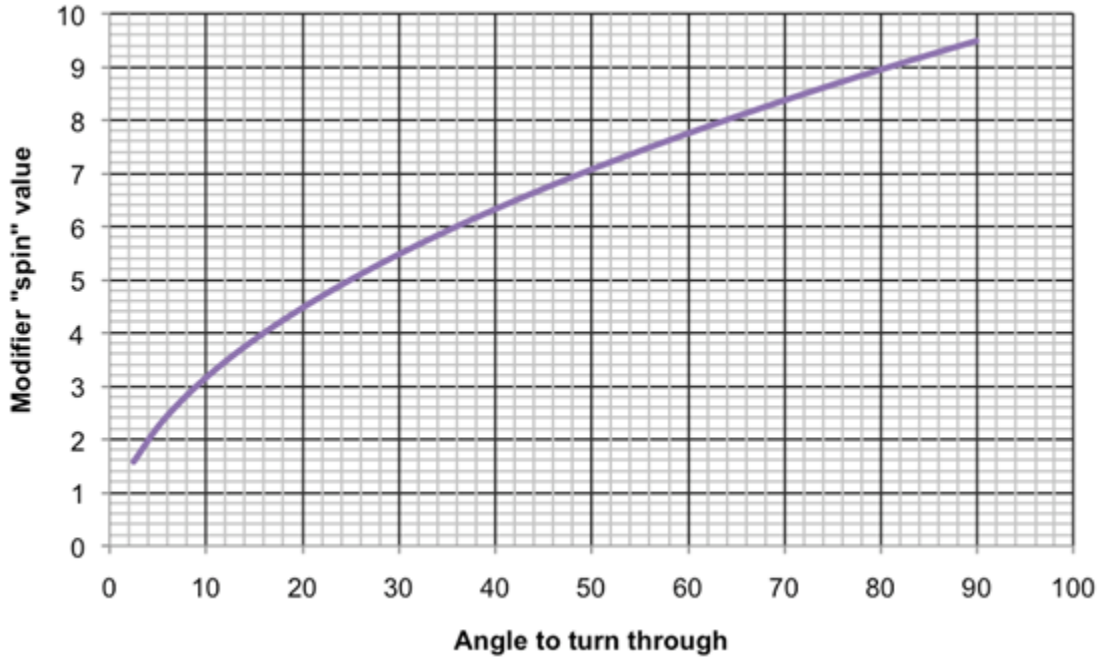

Figure 7 - Straight Line Orientation

7.1 INDIVIDUAL WHEEL VELOCITIES: ADJUSTMENT

Initially, when first moving, both the left and the right wheel velocities are set the same. As the robot moves towards its destination, the program monitors the difference between the chosen angle and the angle of movement.

In order to stop the robot veering off course it is necessary to apply an angle correction modifier that will increase the speed of the outside wheel so that turning in motion is possible.

Many different complex power functions were trialled to see which would produce the smoothest motion and not send the robot either out of control or into a state of oscillation about the desired trajectory.

It was found that actually a simple inverse power function produced the most desirable result.

When the difference between the chosen and desired angle is between $-25^{o} \rightarrow 25^{o}$ the outside wheel velocity will have added to it the following value to steer the robot back on course.

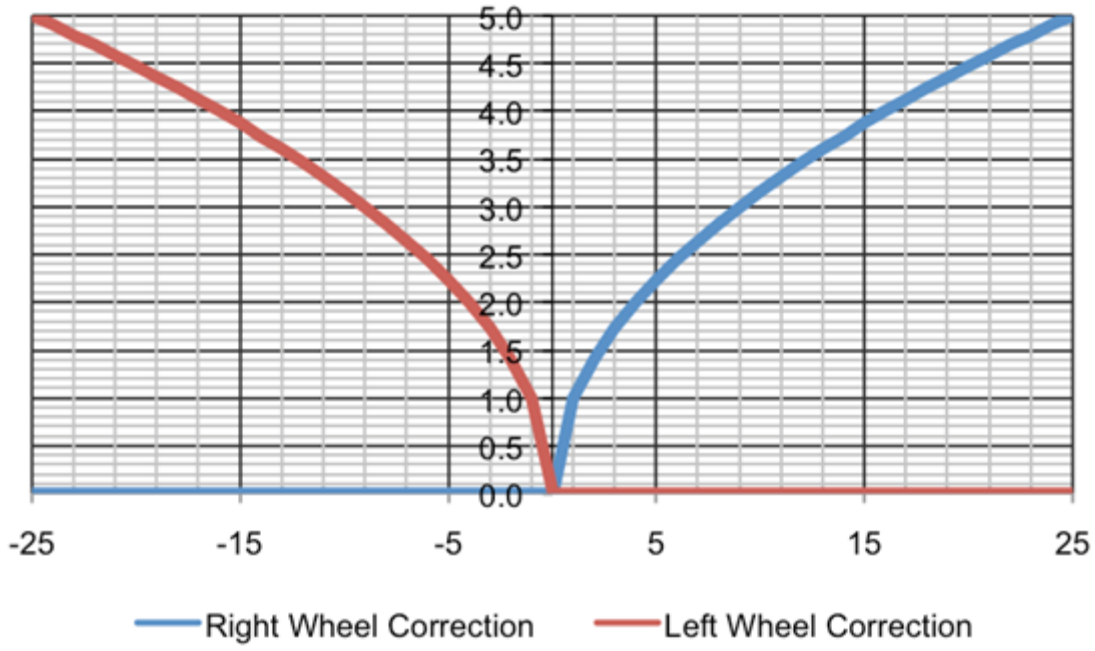

Figure 8 - Straight Line Wheel Correction

7.2 INDIVIDUAL WHEEL VELOCITIES: OVERALL REDUCTION

Modifying the relative wheel velocities has been seen to be an effective means to keep the robot moving in a straight line at low overall velocities. The problem occurs however when trying to run the same motion at higher speeds that are suitable for game play.

Raising the speed means that for the corrective measure to have the same impact, the adjustment will also need to be higher. This inevitably leads to the robot spinning out of control or "oscillating" about the line.

It was found that the most effective measure of compensating for this is to decrease the overall base speed, depending on the angle adjustment needed. The modifier that varies the individual wheel velocities can then be added on top of this.

The formula used to reduce the overall speed is the following:

Base Velocity = (Max Velocity – ((angle off target)^2/20))

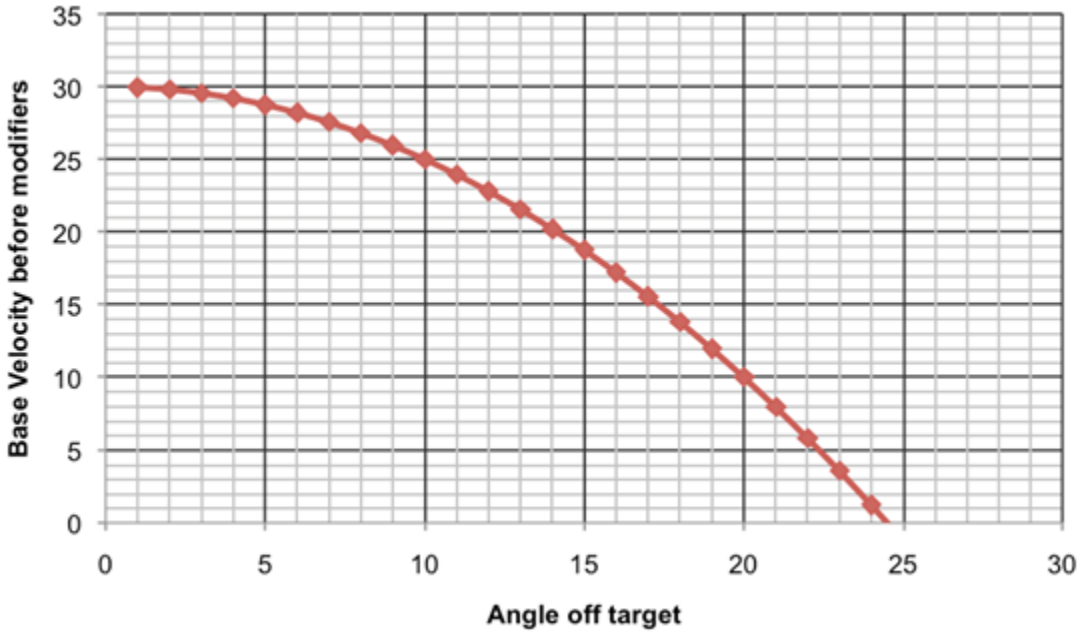

Figure 9 - Overall Straight Line Overall Velocities

As can be noted in the figure above, once the angle off target exceeds that of 25, the base velocity comes to a standstill. This was found to be necessary to prevent spinning out of control from being excessively outside of the angle tolerance.

8.0 STOPPING THE ROBOT

The task of getting the robot to a target location as quickly as possible poses various problems for actually stopping it:

- By not applying a stopping profile that reduces the overall speed upon approaching the location, the robot will overshoot.

- By applying an exaggerated stopping profile, the robot will reach its target too late, as it will be moving too slow on the approach.

As a result, a compromise solution is required that prevents overshooting, however keeps the velocity high enough to ensure the robot reaches the destination in the most efficient and quick manner.

Through testing, it was found the following proportional profile provided the best balance for the reduction of speed, without comprising the efficiency of the motion.

Note that the reduction only occurs within the last 5” to the target location.

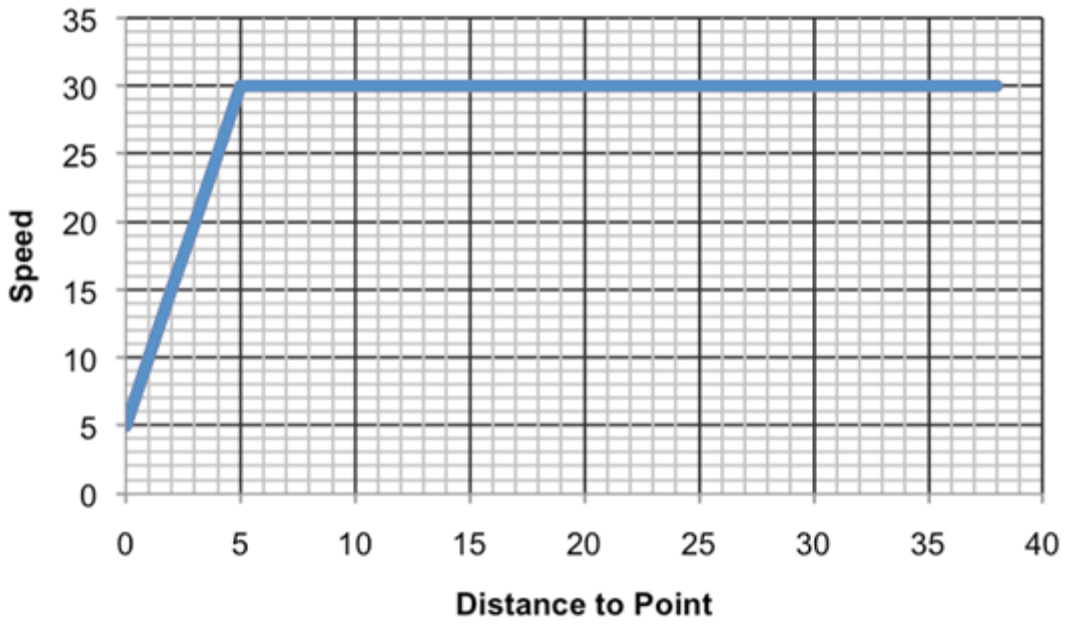

Figure 10 - Straight Line Approaching Location Velocity

8.1 MODIFIED LOGIC DIAGRAM

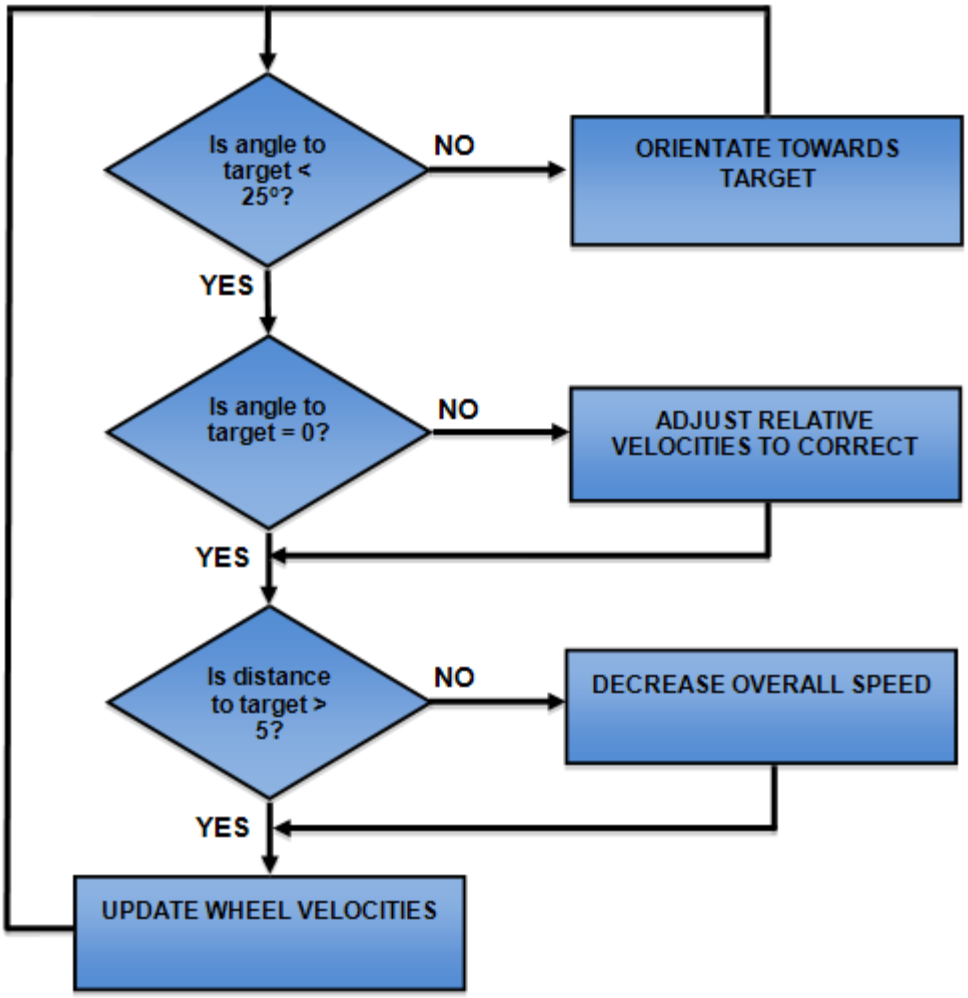

Figure 11 - Modified Straight Line Logic Diagram

STRAIGHT LINE MOTION CONCLUSIONS

Through extensive trailing in various scenarios, straight-line movement was found to be a very useful asset, however only in certain tactical plays.

Its strengths lie within defensive roles that are heavily based on fixed destinations, e.g. linear blocking. Straight-line movement in these situations is highly effective and can get to the target location faster and more efficiently than the past "PID" solution could achieve.

Its weaknesses lie in areas where chasing the ball is required. Due to the rapidly changing destination, the angle keeps exceeding the set limit of 25^{o} and is then required to stop and re orientate, wasting precious time.

In conclusion, a very powerful movement algorithm has been created, however it should be used with caution and only applied to roles that it is suited for, such as linear defence.

9.0 OTHER ALGORITHMS AND TACTICS

9.1 BALL PREDICTION

To compensate for delays within the system (vision, processing), it is necessary to predict the location of the ball. If prediction is not utilised then, by the time the strategy is implemented, the location data will be will be a significant number of frames old which could have important implications for tracking and interception algorithms.

The method utilised by the previous 06/07 team took the Occam's razor approach to ball prediction that the simplest solution could be the most successful and worked with a straightforward averaging calculation.

Previous teams recommended looking into the use of a Kalman-Bucy filter to improve ball prediction, however upon briefly studying the theory and the teams past work in this field it was deduced that improvements would be minor at the very best.

Through analysing the averaging technique during match play it was found that it indeed worked well and since it provided the required data, it was felt that attention was better focused elsewhere within the strategy and therefore only very minor changes were applied to get the functionality within our code.

In summary, the system takes the data from the past 5 frames analysed and stores them in a stack. When a new frame is produced it gets 'pushed' onto the top of the stack whilst the oldest gets 'popped' off at the bottom. To calculate a future coordinate location, an average is taken of the stack and then multiplied by the number of frames prediction that is required.

For the purposes of the 07/08 strategy it was found that 10 frames proved the optimal number to compensate for delays within the system.

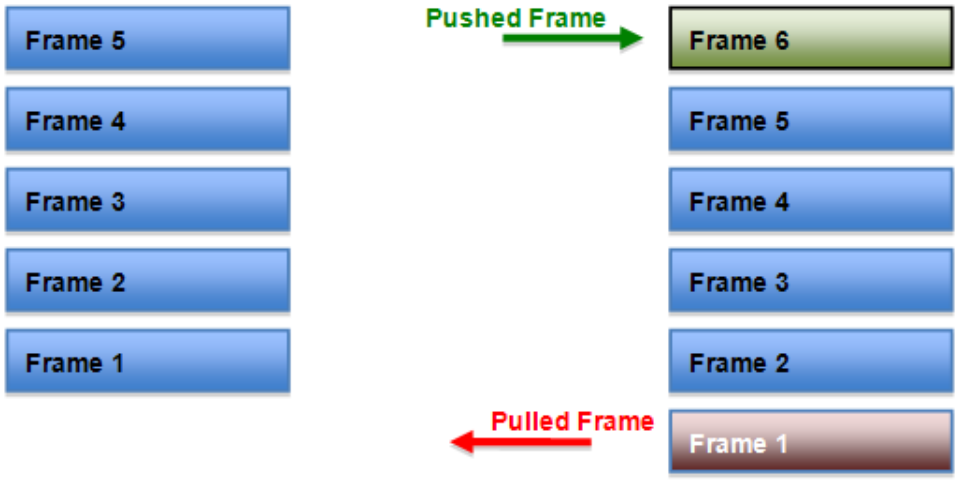

Figure 12 – Ball Prediction

9.2 STUCK ROBOTS

Robots can appear to be stuck during a game either against a wall or a fellow robot. In these situations the wheel velocities still behave as though they are moving towards their destination, therefore are both often at high speeds.

Stuck conditions occur either because both robots are trying to get past one another, or because the coordinate data is sending the player “beyond” the walls location.

In the case of the former since the robots are not aware of the others presence, they will simply continue to “barge” past one another. If the robots are equally matched in size and power they will become “deadlocked” for a wasted period of time.

In the case of the latter, whilst every effort has been made to make the maximum permitted coordinates equal to the pitch dimensions, there are occasions, perhaps due to calibration error, that this phenomena still occurs. As a result an “unsticking” algorithm is required.

This idea is not new; many teams preceding us have had similar concepts. In fact the algorithm used in the 07/08 strategy borrows heavily from old iterations with minor tuning required to ensure that not only it integrates with our code but also performs optimally.

Like with the ball prediction algorithm, it was not our intention to attempt to “reinvent the wheel”. If through extensive testing and investigating the alternate possibilities it was found that the current solution was fit for purpose then effort would be concentrated elsewhere in the project. In these two cases, this was found to be true. As a result, using partial code from the old “PID” algorithm alongside some minor modifications and variable alterations, the stuck code was implemented in the following manner:

9.2.1 WALL STUCK:

An area 4” wide has been defined inside each of the walls within the play area. If a robot is within this zone and hasn’t moved for a period of approximately 2 seconds it will initiate a curved reversing motion to try and back out of the situation. If this fails, it will attempt the same motion in the opposite direction.

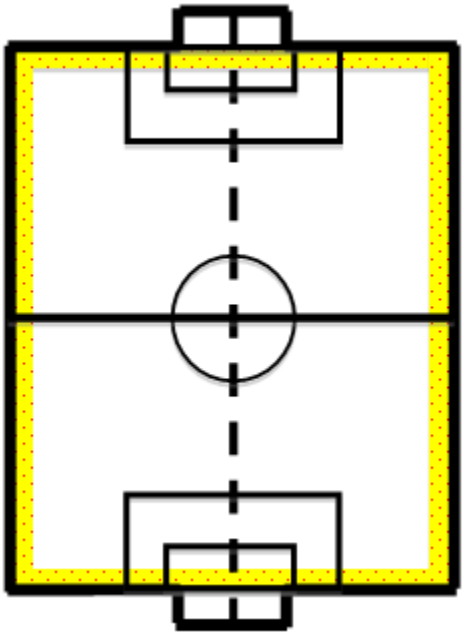

Figure 13 – “Stuck” Zones in Play Area

9.2.2 ROBOT STUCK:

Much like the correction applied to “unstuck” a robot from a wall, a similar algorithm is applied to the problem of deadlocked players. In these situations, if it is detected that robots are not moving and are located less than 2” away from one another, a series of motions will be applied to attempt to correct the situation.

These motions range from reversing at varying wheel velocities to a quick spin on the spot to break the deadlock.

9.3 ZONE PLACEMENT

For the purpose of defensive tactics it is often required to modify the behaviour of the players based on the location of the ball. To do this effectively the home side of the play area was divided into 8 zones, the details of which are stated below.

0 – Home goal

1 – Penalty box left

2 – Penalty box middle

3 – Penalty box right

4 – Left of Penalty box

5 – Right of Penalty box

6 – Remainder of home side, Left

7 – Remainder of home side, Right

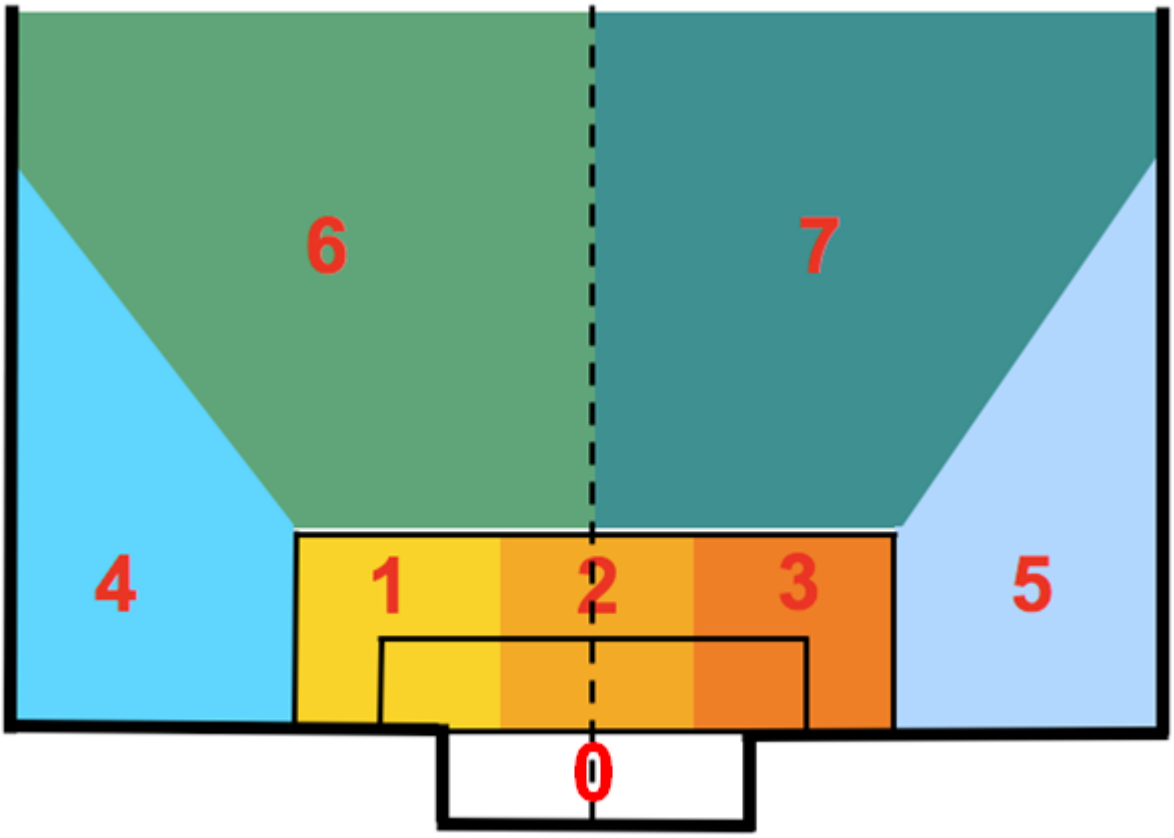

Figure 14 – Defensive Zones

Whenever the ball is in the home play area it will be located within one of the 8 zones. All of the defending robots to some extent vary their tactics depending on the zone the ball resides.

9.4 INTERCEPTION ALGORITHM

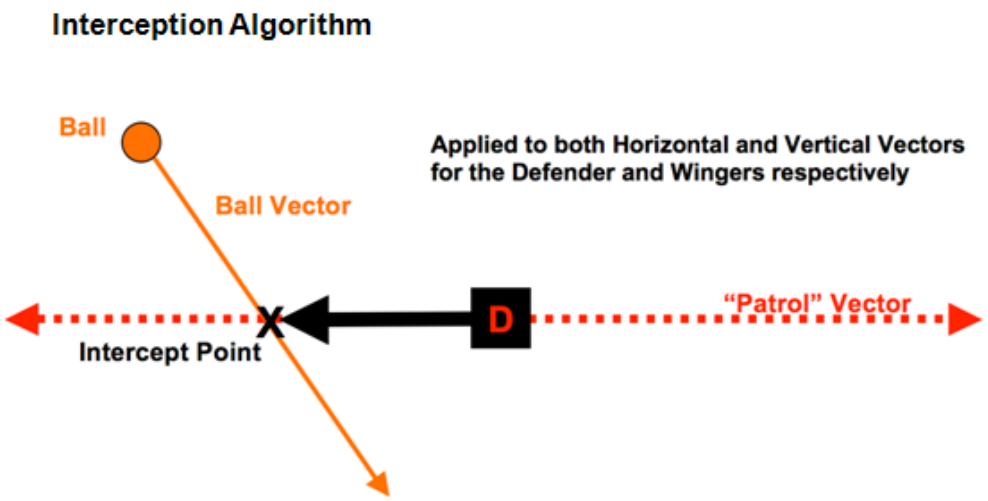

Figure 15 – Linear Interception

Interception is vital for a strong defence. Using ideas developed from previous years, interception takes the concept of ball prediction and extends it through the following process:

- Predicted ball location and trajectory calculated (from an earlier explanation).
- Trajectory extended to form a vector within the play area.
- Vector analysed to determine if and at what point it crosses a particular intersection line.
- If found to intersect, the location is determined as well as the approximate time to intersection.

Utilised from the work of the 06/07 team, the concept was developed and extended to take into account vertical as well as horizontal axis predication. As a result this allowed for a wider variety of roles to take on this ability.

This method is used in the goalkeeper, defender and multi-tasker functions and is essential in the overall concept of the defensive strategy.

10.0 HIGH LEVEL STRATEGY

10.1 STRATEGY TACTICS OVERVIEW: PHASE 1

10.1.1 STRATEGY NAME: PID++

The strategy initially developed for the year 2007/2008 utilises all 5 robots in the following configuration:

The numbers in brackets refer to the robot ID number that the role is assigned to.

Defensive	Misc	Offensive
Goalkeeper **(0)**	Multi-tasker **(2)**	Striker 1 **(3)**
Defender **(1)**		Striker 2 **(4)**

A simple overview of the details of the various roles follows.

10.1.1.1 GOALKEEPER

The goalkeeper maintains a fixed defence in front of the goal line and uses prediction and straight-line movement to prevent the scoring of goals.

10.1.1.2 DEFENDER

The defender maintains a fixed defence at the top of the penalty box and uses prediction and straight-line movement to prevent the ball entering the home goal area.

10.1.1.3 STRIKER 1/2

The striker's objective is to score goals. It does this through angle prediction to get behind the ball and shoot in the direction of the opposing goal. The two strikers work together so they don't both attack the ball at the same time and an expert system algorithm decides which robot has control of the ball.

10.1.1.4 MULTI-TASKER

The multi-tasker has the most varied role and takes on a variety of different tactics depending on the current zone the ball is currently in. In various situations it either takes a defensive role, a ball-clearing role or as an additional attacker.

10.2 DEFENSIVE

10.2.1 GOALKEEPER

The goalkeeper's primary objective is to prevent the ball passing the goal line, as a result movement and strategy is quite limited in scope.

The goalkeeper uses the interception algorithm along the horizontal goal box axis to calculate the point of intersection then uses straight-line movement control to block the ball from entering the goal.

Straight line movement control is excellent for this fixed axis approach as since the robot can move both forwards and backwards along the line, very little angle adjustment is needed to maintain a horizontal vector.

Previous years strategies have incorporated kicking into various forms of goalkeeping. After prolonged testing however it was found this caused more own goals than it saved and as a result was removed and goal clearance delegated to other robots.

The goalkeeper is used purely in a blocking capability in this strategy therefore was one of the simplest roles to create once the lower level movement fundamentals were established.

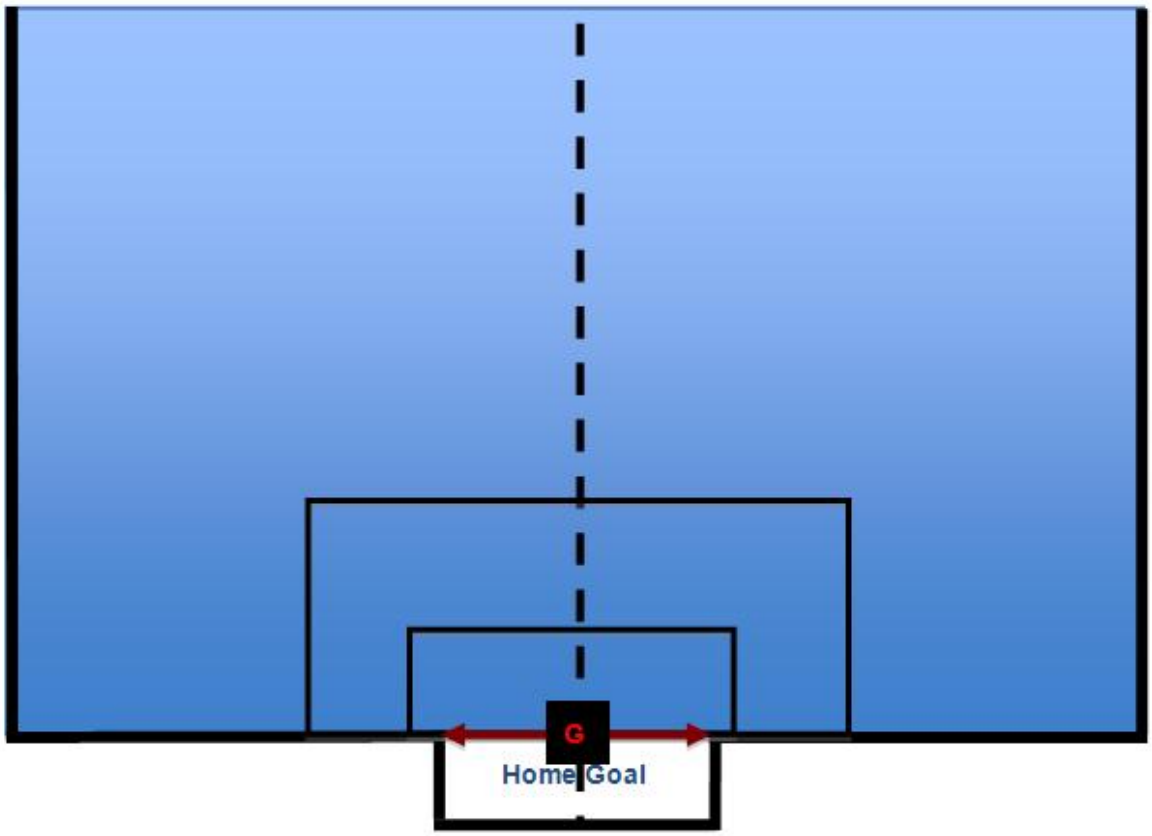

Figure 16 – Goalkeeper Strategy

10.2.2 DEFENDER

The defender works along the same principles as the goalkeeper, maintaining a fixed linear path and acting purely within blocking capabilities. This keeps movement highly efficient as turning is minimized and allows the defender to quickly reach the point of interception and successfully block the ball.

Unlike the goalkeeper however, the defender has two different states depending on if the ball is in front or behind the robot:

Whenever the ball is in front of the penalty box (as pictured, light blue)

- Uses the interception algorithm along the horizontal axis of the penalty box to determine the point of intersection.
- Uses straight-line motion algorithm to intercept ball at crossing location.

Whenever the ball is behind the penalty line (as pictured, dark blue)

- Performs as above, however with a 5” offset towards the side of the pitch the ball isn’t in.
- This is to allow the ball to be cleared from the goal and if it gets hit towards the defender it will not rebound to score against us.

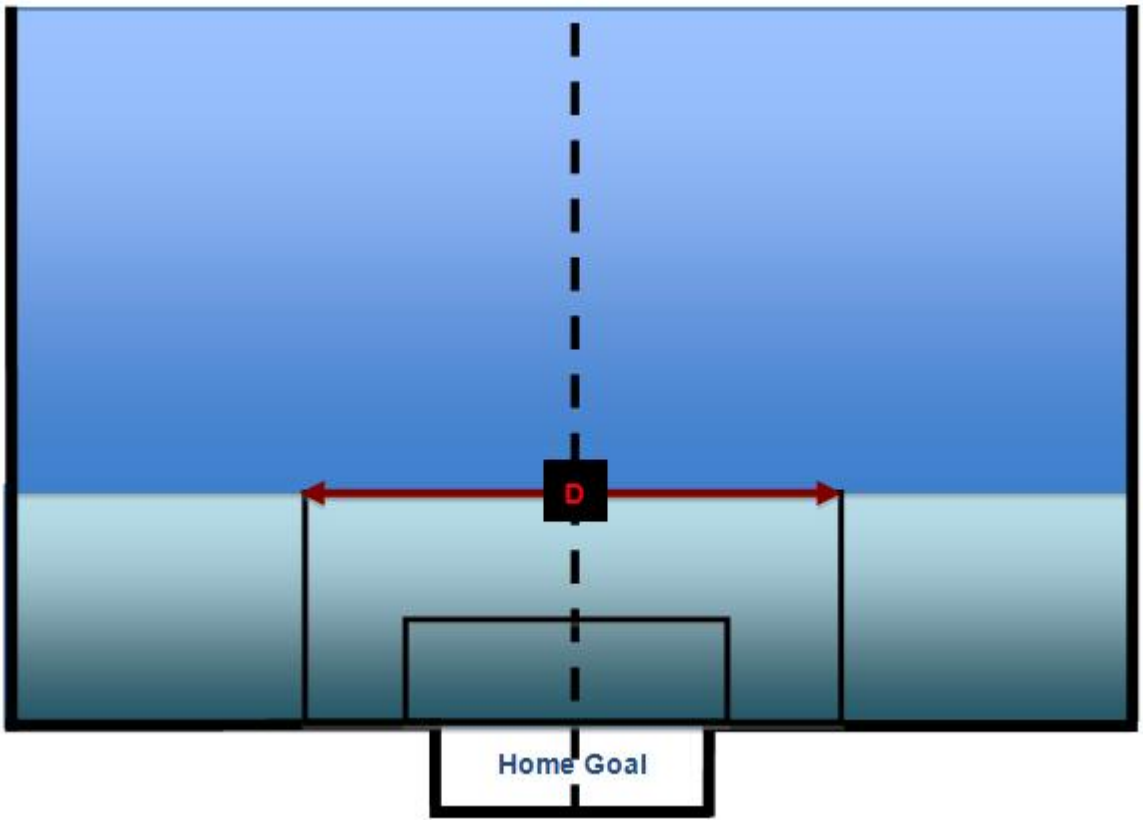

Figure 17 – Defender Strategy

The combination of a linear defence using the goalkeeper and defender works highly effectively within testing in real game scenarios at stopping most direct attacks at goal.

10.2.3 MULTI-TASKER

The role of the multi-tasker is the most complex element of the defence as the robot takes on different roles depending on the ball's location.

10.2.3.1 MULTI-TASK: DEFEND THE WINGS

The multi-tasker takes on this role when the ball is in the home side of the play area and located within zone 4/5.

Original ideas for the defensive strategy involved using 3 robots to form a protective "shield" around the penalty box during all times the ball was located within our half. While this provided excellent defence capabilities, it had the following significant problems with its implementation:

- Too many robots were being used to defend along a fixed axis as opposed to attacking and scoring goals.
- More than 3 robots were often within the penalty box, thus conceding needless penalties against us.

As a result a hybrid solution was found in the role of the multitasker. Whenever the ball is located in a position that a has a direct line of sight towards the goal through the side of the penalty box, the multitasker assumes a defensive position along the vertical axis as pictured.

Much like the main defender a combination of intersection location and straight-line movement is utilised to prevent the ball entering the penalty box.

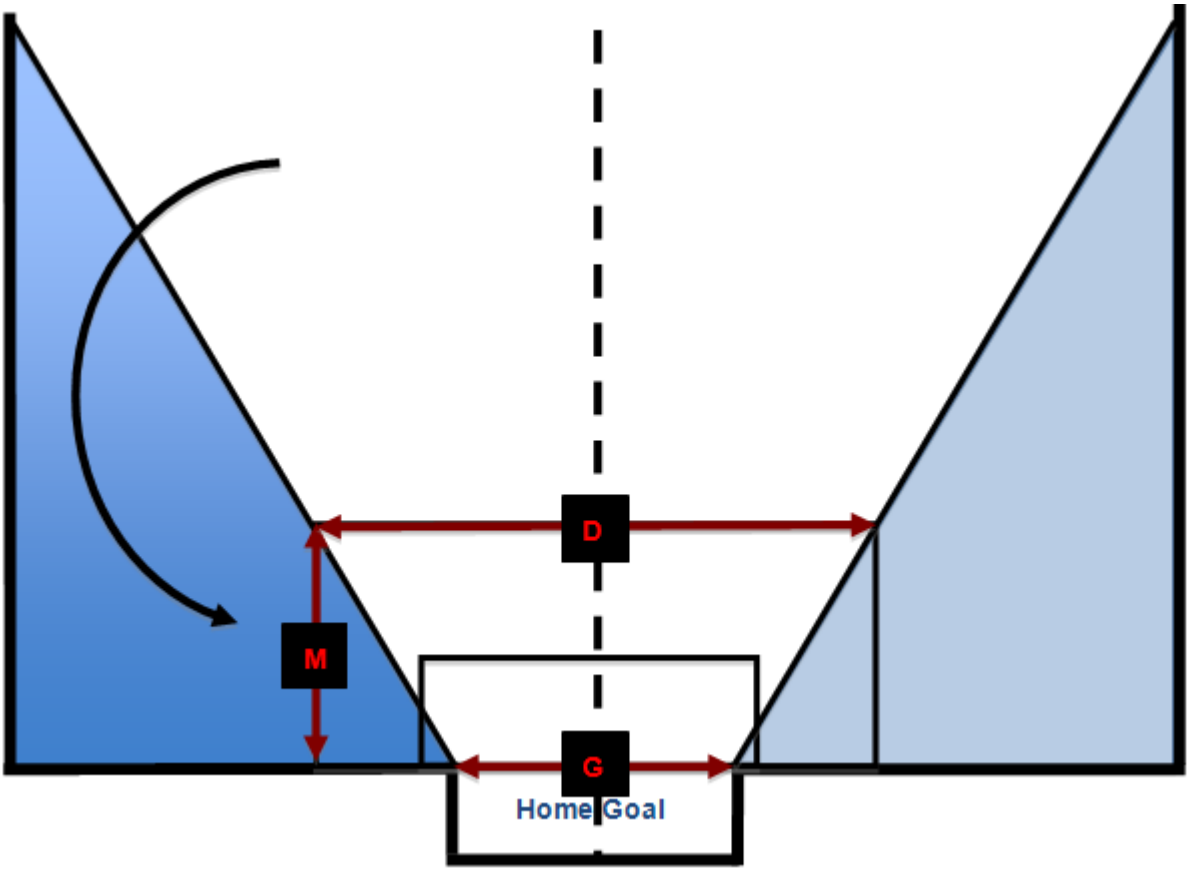

Figure 18 – Multitasker Wing Defence

One downside to this current method is the stationary ball situation. Defending the vertical side of the penalty box works well if the ball is moving with potential to enter it, however if the ball is stationary then there are currently no robots assigned to clear it out of the area. This scenario was found through testing to particularly leave us exposed to attack.

As a result, the strategy calculates the balls velocity for this purpose. If the ball is moving at a speed below a set threshold and is determined to either be “very slow” or “stationary” the multitasker will move to “nudge” the ball outside of the zone.

Once the ball is outside of either zone 4/5 (i.e. no longer has line of sight through the side of the penalty box), the multitasker will resume its duties as an attacking robot (see later) and chase the ball outside our half.

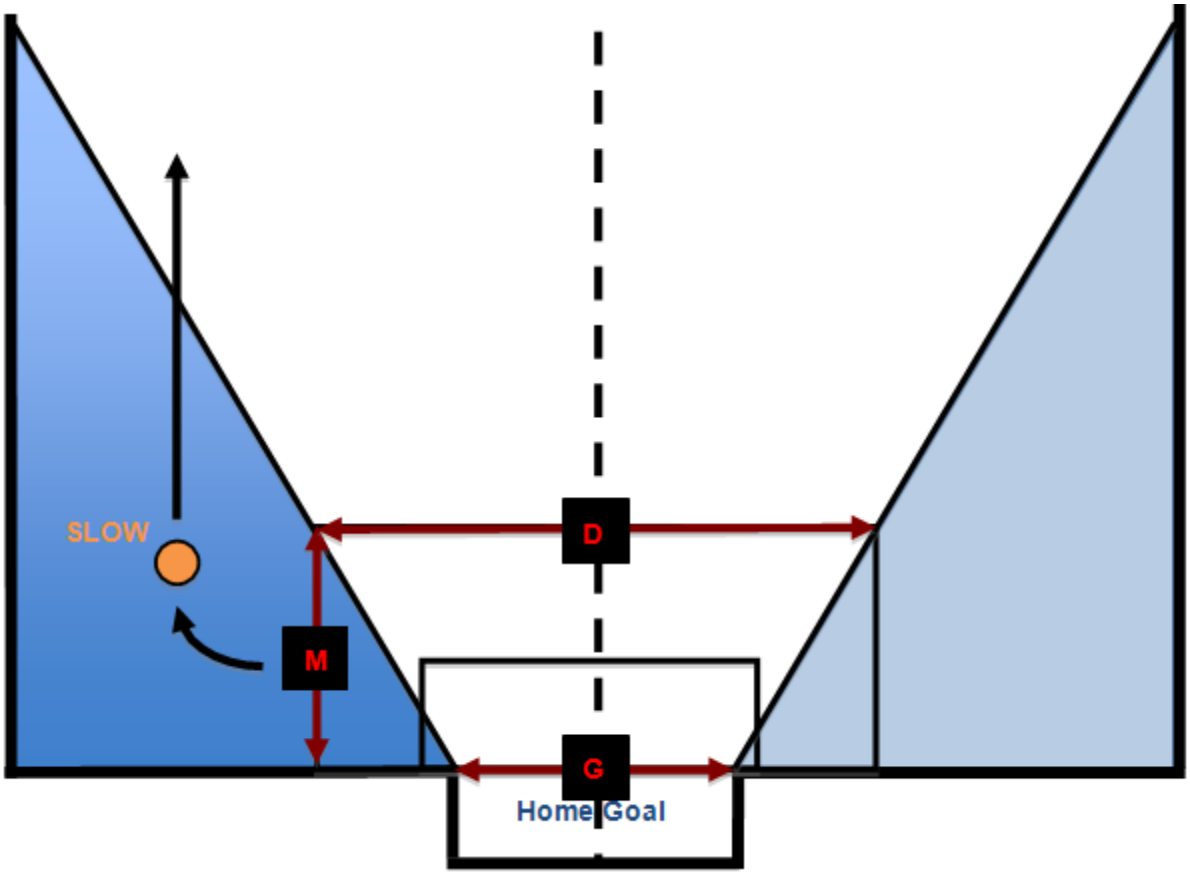

Figure 19 – Multitasker Wing Clearance

10.2.3.2 MULTI-TASK: CLEAR THE BOX

The multi-tasker takes on this role when the ball is in the home side of the play area and located within zones 1/2/3.

The main problem with using linear blocking defence methods is the problem of clearing the ball out of the penalty area if it manages to get through. Previous strategies have concentrated on getting either the goalkeeper to clear it by means of a spinning kick, or moving the defender to get it out of the box. More often than not these either result in own goals, or in the case of the latter, leave the goal exposed to attack.

In this years strategy the multitasker is used to clear the ball any time it enters the penalty box. It does this by entering the zone in a curved motion, to avoid the defender and then match the vertical coordinates of the ball before sweeping it out in a horizontal movement. This means the ball is never pushed closer to the goal and near eliminates the possibility of own goals through ball clearance.

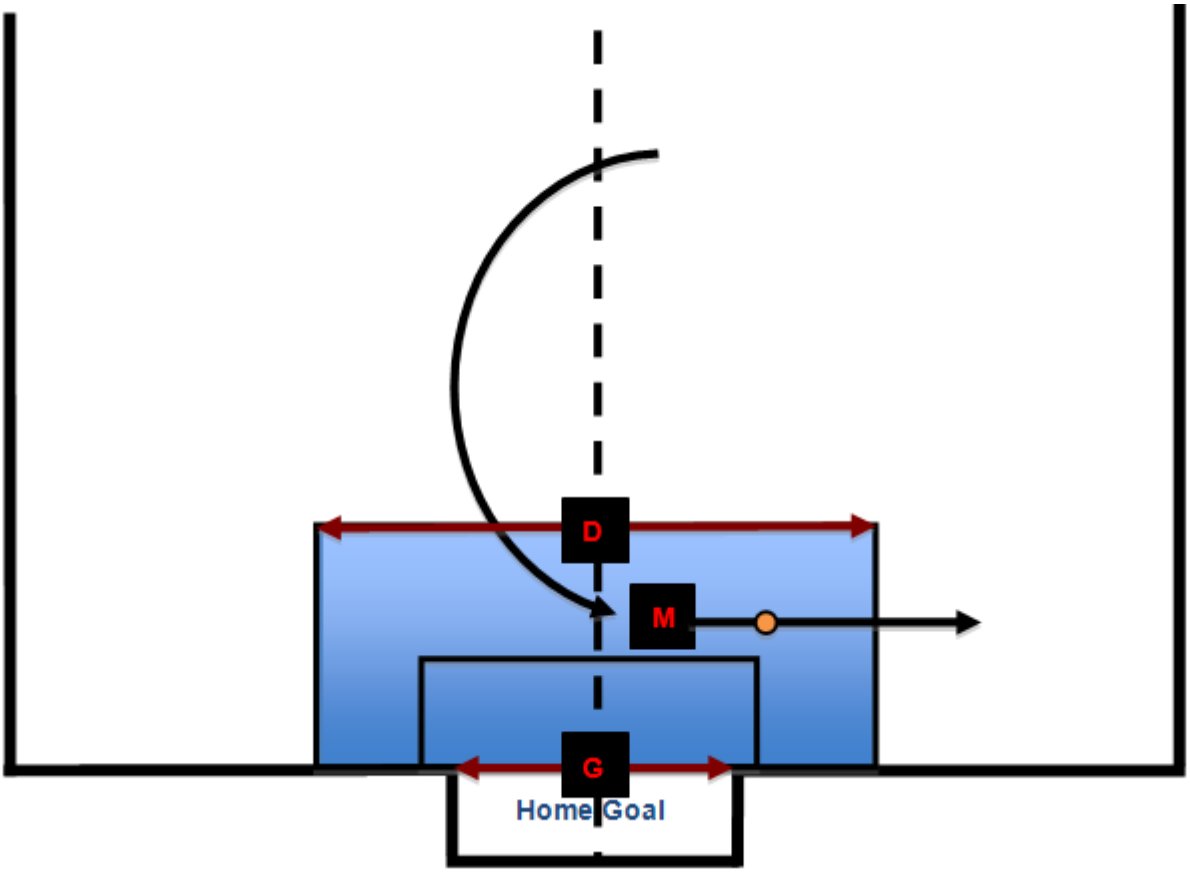

Figure 20 – Multitasker Goal Clearance

While ambitious on paper, through real match situations, this ability has proven to be highly effective and has not yet conceded a goal through poor control.

10.3 OFFENSIVE: PHASE 1

The offensive strategy has been through two main evolutions to reach the code used in the 2008 competition. Information has been included with regards to "Phase 1" for information purposes as many of the ideas in "Phase 2" build upon these original plans. In addition, a significant amount of thinking and planning went into devising these tactics, perhaps future years can iterate upon some of these ideas and improve them to the point they work better than the final solution that was chosen.

10.3.1 THE STRIKE CONE (ORIGINAL)

The concept of the strike cone is not an original one; it dates back a number of iterations of the Evolution robot football program.

The theory behind the cone is simple, by calculating the angle from the ball to each of the goalposts, a zone can be defined in which a robot has direct line of sight to the goal and movement straight forward with the ball would lead to scoring.

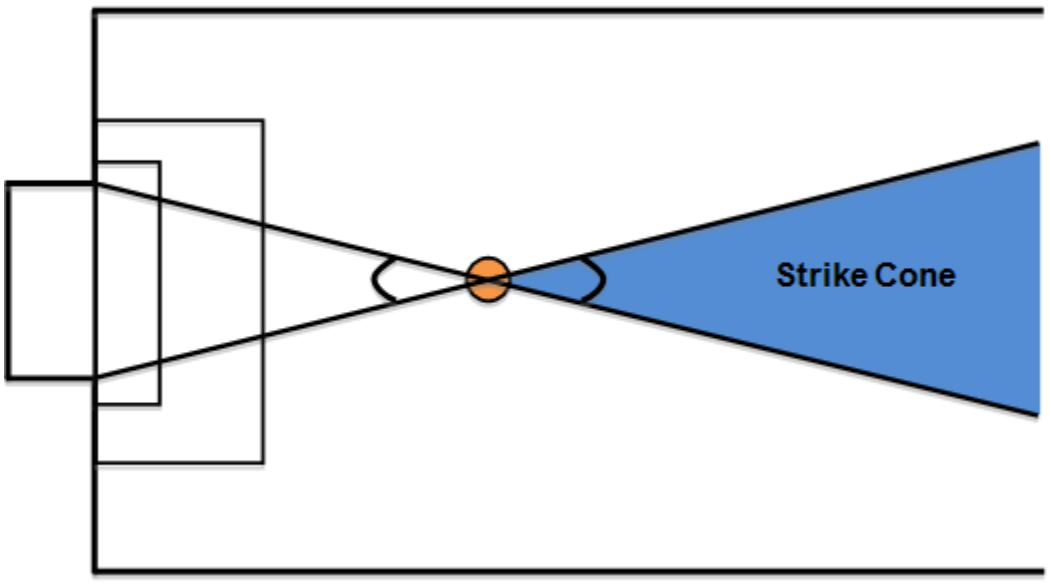

Figure 21 – Strike Cone Overview

The first attempt utilising this algorithm within our strategy used the "PID" movement protocol to position the ball within the strike cone and then switch to our straight-line movement to attempt to score.

Whilst the principle is sound, opposition robots, imprecise striking velocity and slight angle inaccuracies in the feedback method prevent as many goals being scored as would have initially been hoped. Despite these flaws, the general tactic is successful in moving the ball in the right direction on the pitch and provides a reasonable method of shooting at goal.

One of the greatest flaws of combining this technique with a typical movement algorithm is the logic relating to positioning before the strike and the moment at which the decision to strike is actually made.

The simplest solution is the following logical flow:

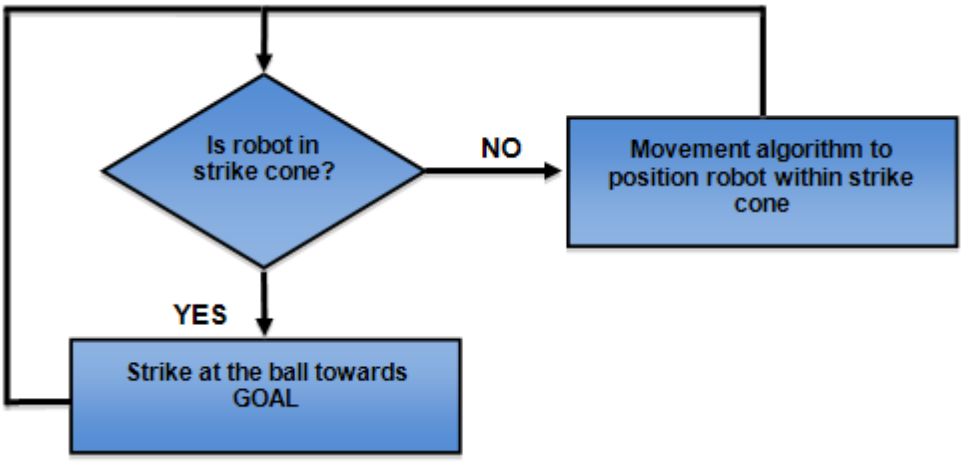

Figure 22 – Initial Striking Logic Diagram

The problem with only using one constraint on deciding whether to strike is that the robot will make an attempt the moment it enters the strike cone. While this may be acceptable when the ball is stationary, more often than not in a game situation, the ball will be moving at high speeds. As a result the situation occurs where the robot appears indecisive as will enter the cone but by the time has turned to strike, the ball has moved away and is no longer within the assigned area.

In game scenarios very few attempts to score were being made and therefore striking was quickly becoming our biggest weakness. To correct this problem, the next logical step was to add an additional constraint in the form of a second strike cone.

10.3.2 THE STRIKE EXTENDED STRIKE CONE

The diagram below illustrates the use of a secondary strike cone, alongside the modified dimensions of the original.

Note the original cone now takes its reference points approximately 5” **outside** of the goalposts, widening the permitted area. In contrast, the newly created cone is much narrower, taking its anchor points approximately 5” **inside** the goal posts.

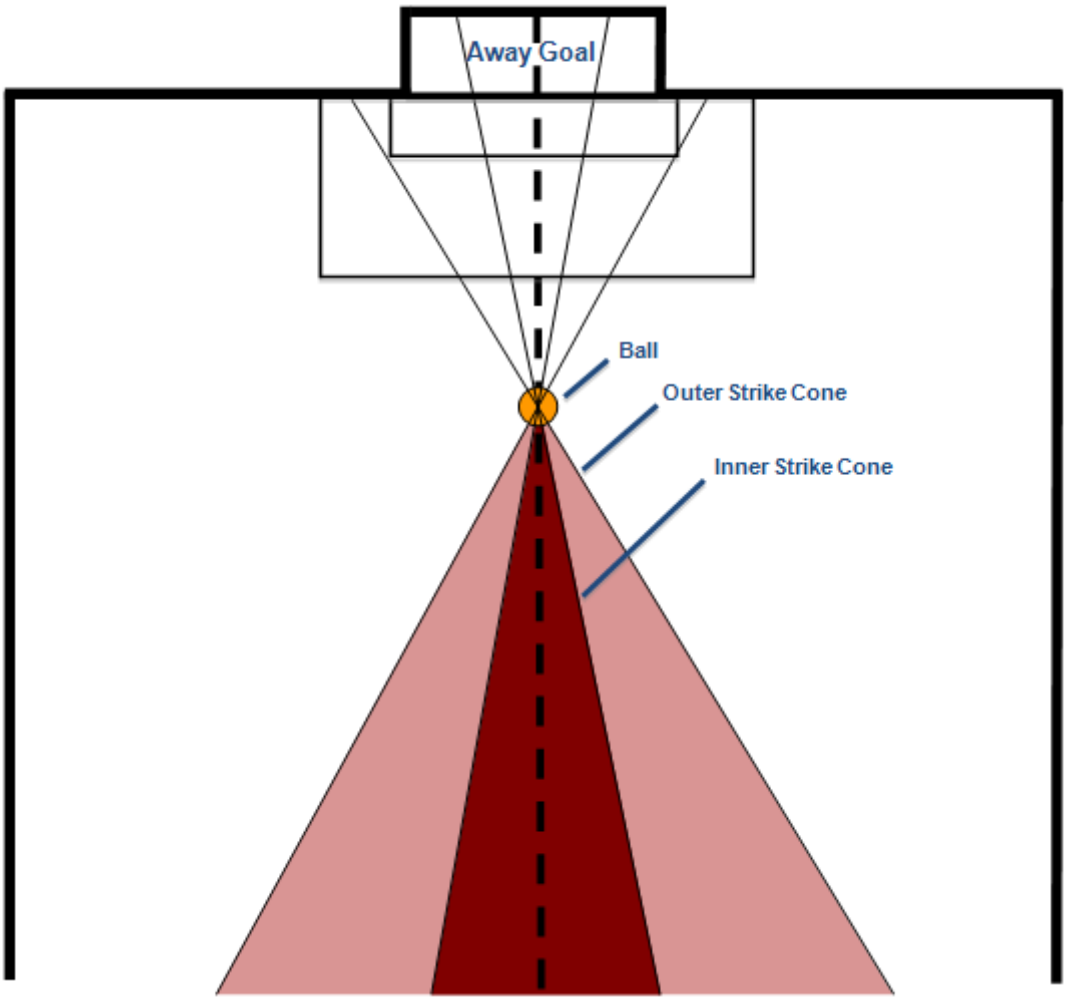

Figure 23 – Extended Strike Cone Overview

By widening the original cone, this creates a much larger possible area in which the robot is permitted to strike. Due to various inaccuracies inherent in the system, it was found through testing that more goals were scored with this method, despite the mathematics proving there is not direct line of sight at the extremities.

The logic flow with the addition of a secondary cone now works in the following manner:

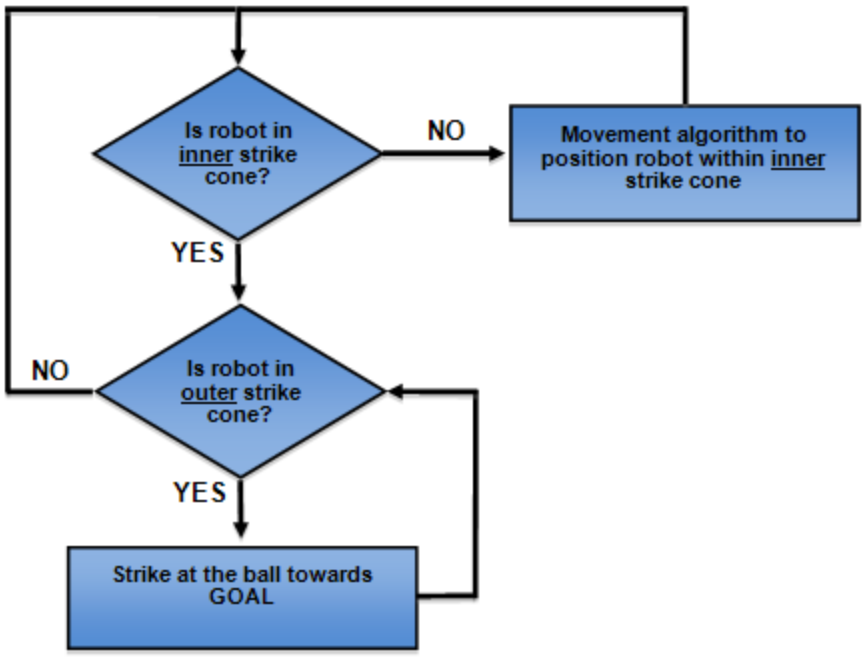

Figure 24 – Extended Strike Cone Logic Diagram

Note that now the robot must first enter the inner strike cone before being permitted to strike at the ball. It will then continue to do so for as long as contact with the outer strike cone is maintained.

This is to ensure the robot is first positioned in the best possible position to strike, so that after turning to shoot, despite the ball moving, the robot is still in a position to shoot.

10.3.3 STRIKER 1/2

The original plan for both the striker and the support striker was as follows:

- Only one robot at any one time would have 'control' of the ball. This was designed to prevent clashes and the other robot interrupting the strike attempt of the one currently shooting.
- The robot that would have control of the ball would be decided by a complex expert system algorithm, the details of which can be found discussed in great depth overleaf.
- The robot in control would use the 'PID' motion algorithm to position itself within the inner strike cone and then use linear motion to hit the ball into the opposition goal.
- The robot that currently isn't in control is assuming a waiting position on the opposite side of the pitch, approximately 5" in front of the halfway line, waiting for a possible rebound opportunity.

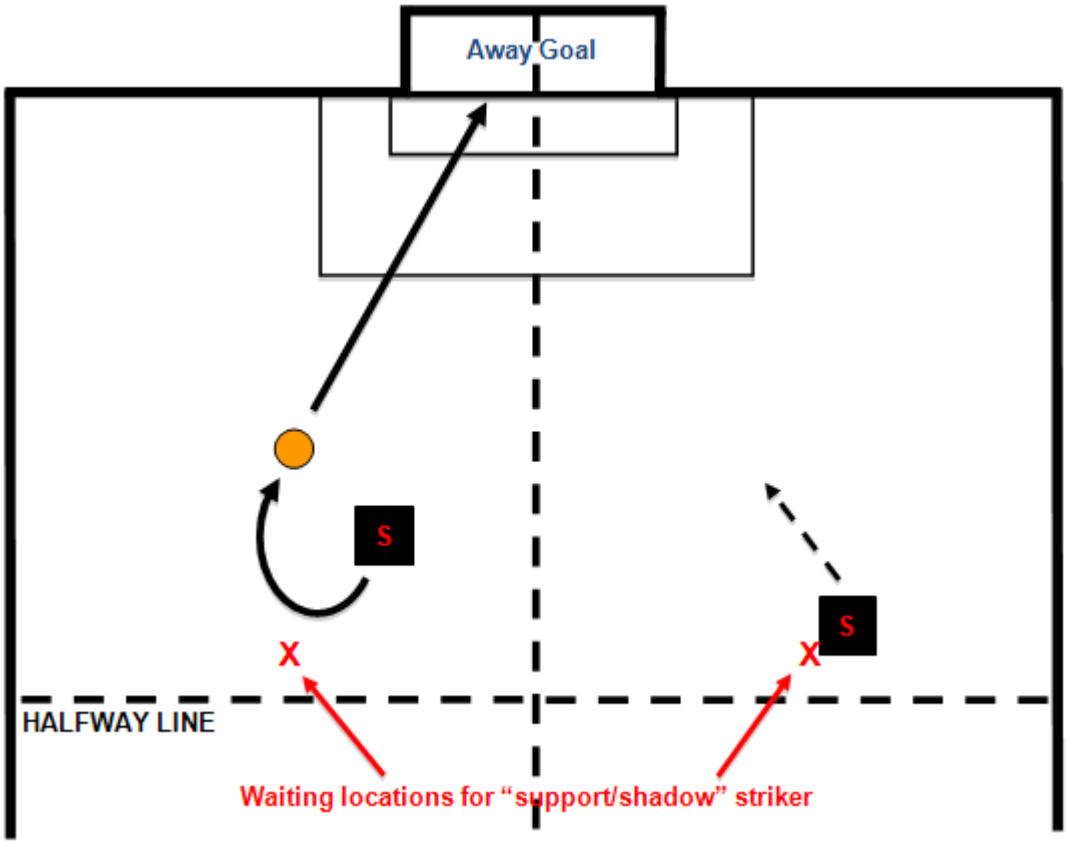

Figure 25 – Striker 1/2 Locations

10.4 DECIDING THE "ROBOT IN CONTROL"

The following couple of pages go into depth the method used to deduce which robot is "in control" of the ball and has the permission to shoot.

The following series of tests are undertaken to ensure that control does not keep flicking back and forth and therefore act "indecisive" and as a result no robot taking a shot. If stage three is reached, the chance algorithm will be called to deduce if a switch is required or not.

1. If the current control robot has a "strike lock" and is therefore preparing to shoot it will stay in control of the ball for as long as it maintains a lock.

2. If the current control robot has just been allocated control, it will remain in control for a minimum of 2 seconds before conflicts can be addressed.

3. The robot with the highest "chance" score will take control of the ball.

10.4.1 CHANCE SCORING:

If stage 3 is reached, this algorithm will determine which robot is in the best position to shoot at the goal and therefore if a switch in control is required.

The algorithm is based on the principle of an expert system. Various attributes are weighted to produce an overall score that represents the chance that robot will be able to make a shot at goal. The robot with the highest chance score will assume control of the ball.

All scores are initially judged between the limits of 0.0 → 1.0 where the highest score is awarded to the best possible outcome, and the lowest the worst.

Having performed these calculations, various weightings are applied to elevate various attributes over others based on testing and performance.

The four factors that are judged:

- **Strike cone proximity**
- **Being behind the ball**
- **Distance to the ball**
- **Angle needed to turn to face the ball**

The scoring of these attributes can be found below:

10.4.1.1 STRIKE FACTOR

- If the robot is not in any of the strike cones, **a score of 0.0** will be added.
- If the robot is within the outer strike cone, **a score of 0.5** will be added.
- If the robot is within the inner strike cone, **a score of 1.0** will be added.

10.4.1.2 BEHIND THE BALL FACTOR

- If the robot is in front of the ball, **a score of 0.0** will be added.
- If the robot is behind the ball, **a score of 1.0** will be added.

10.4.1.3 DISTANCE FROM BALL FACTOR

- If the robot is 90” + from the ball, **a score of 0.0** will be added.
- If the robot is next to the ball, **a score of 1.0** will be added.
- For distances between these extremes, the score is adjusted on a linear basis.

10.4.1.4 ANGLE FROM BALL FACTOR

- If the robot is 90^{o} - from an attacking position, **a score of 0.0** will be added.

- If the robot is 0° - from an attacking position, **a score of 1.0** will be added.
- For distances between these extremes, the score is adjusted on a linear basis.

These principles are visualised in a diagram:

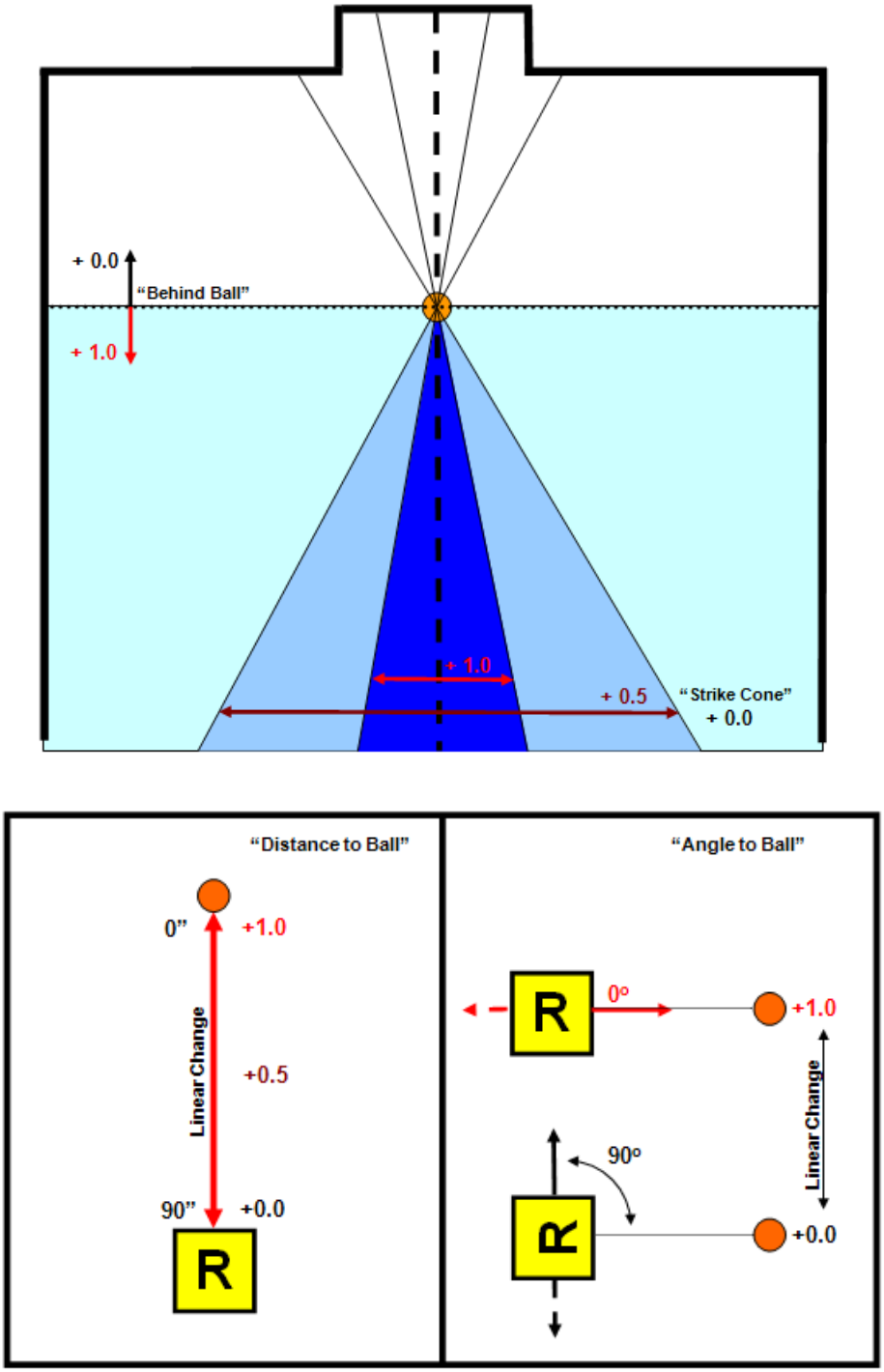

Figure 26 – "Expert System" Rules Overview

10.4.2 WEIGHTING THE ATTRIBUTES

It was found through testing that the current attributes were not producing the desired response and often switched when it was not appropriate to do so. As a result all the attributes were subjected to a weighting system that elevated some above the others in terms of significance, and demoted others.

Following a significant period of testing the following series of weightings produced the optimum response:

Attribute	Max score without weighting	Weighting Factor	Max score with weighting
Strike	1.0	x2.0	**2.0**
Behind	1.0	x1.5	**1.5**
Distance	1.0	x1.0	**1.0**
Angle	1.0	x0.5	**0.5**

As the table demonstrates, it was found that the most important aspects of determining which robot is in control of the ball are whether it is located within the strike cone and whether it is behind the ball. Fulfilling these criteria now automatically gives a score of at least 3.5.

To understand these concepts further, a worked example can be found overleaf.

Worked algorithm example

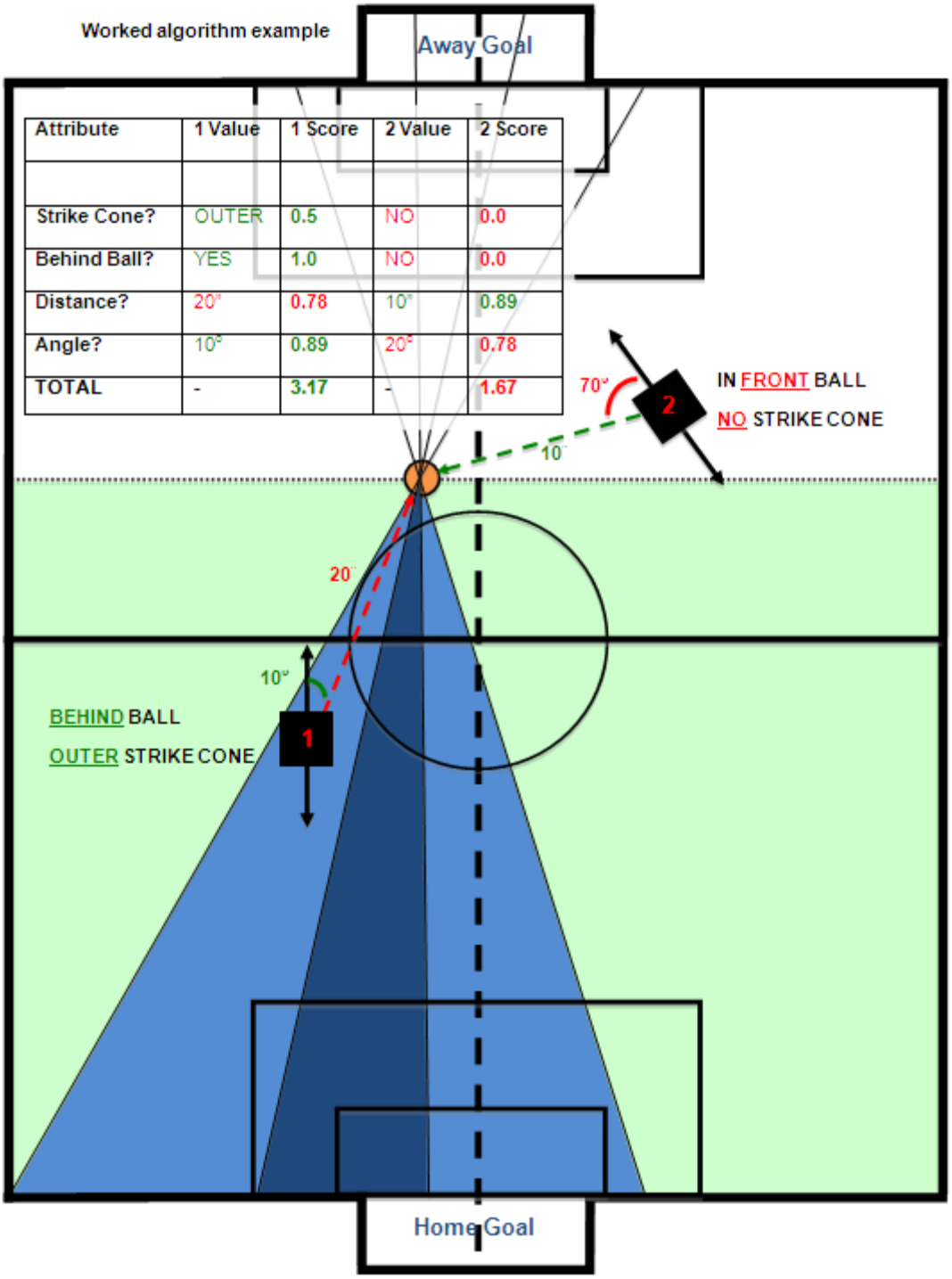

Figure 27 – "Expert System" Example

The outcome is that robot 1 has the highest score and is in control of the ball.

11.0 TESTING PHASE 1

At this stage, all the individual elements had come together to form a complete strategy that could now be tested as a whole in real game situations. While testing was a major concern throughout the entirety of the development cycle, it was only at this stage could it be seen as more than just the sum of the individual parts.

A number of observations were noted of which a brief summary is provided:

- Defence performed as planned and provided excellent blocking capabilities. The goalkeeper and defender were very quick getting to the intercept location using straight-line movement. Having acquired the line, both robots stuck linearly to it, needing re-orientation at very rare intervals.

- The operation of the multitasker in defensive capabilities exceeded expectations and proved highly successful. Whenever the ball was in a position where line of sight through the side of the penalty box could be achieved, the robot was quick to rush to a linear blocking position. Of course this approach would not be as successful as permanently assigning a blocker to each of the sides, but this would prove highly inefficient from a tactical point of view and lead to less strikers, thus achieving less goals. After all, goals win matches.

- The multitasker also performed goal clearance duties exceptionally well. On no occasion of testing to date has the robot in this role scored an own goal as a result of a bad clearance attempt. The horizontal sweeping motion is highly noticeable in game situations and can usually clear the ball within 5 seconds of it entering the box.

- The offensive strategy however was not as successful. The expert system deciding control of the ball worked incredibly well preventing clashes, constant switching and almost always chose the right robot that was in the best position to make a shot.

- The problem lied with the attacking motion. Quite simply the algorithm used to attack the goal appeared to be optimised for slow moving balls as opposed to the high speed and pace of a real game scenario. The robots would take too long preparing the shot and tracking behind the ball, as opposed to striking at the goal.

- Using the straight-line motion for shooting was found to be deeply inefficient.

- A successful attack requires keeping the ball in the opposite side of the pitch and making strike attempts at every possible opportunity. Another benefit of keeping the ball in the opposition half is that most strategies reduce their level of attack and focus on defence in these situations, therefore keeping the risk of a goal scored against us, low.

As a result of these observations, it was decided to keep the defensive elements as they are and redesign the attack strategy from the ground up. The outcome of the offensive Phase 2 can be found overleaf.

11.1 OFFENSIVE: PHASE 2

From the previous findings, it was clear that major modifications needed to be made to the offensive side of the strategy. Looking back at past strategies and the video footage of previous games, the following modifications were decided upon:

11.1.1 PROBLEMS

- Poor attacking algorithm.
- Too many robots attacking the ball at the same time leads to clashes and uncontrollability. This is undesirable in a game as potential strikes are ruined by interference from other attacking robots and time can be wasted trying to separate “stuck” robots due to collision.

11.1.2 SOLUTIONS

- A total of 3 attacking robots will be used, the striker, a support striker and the multitasker.
- The striker and the shadow will be coordinated to work together. The striker will always be trying to find the best location to shoot from, whilst the shadow meanwhile will be following slightly behind and offset, ready to take over and switch roles if the striker loses the ball.
- These two robots will always only be active when the ball is away from the home goal and within 10” of the half way line. This is to keep these robots away from our goal area as it has been found that trying to use attacking robots in a clearance role, not only gives away penalties by having too many units in the box, it also leads to clashes and robots getting “stuck” to one another.
- The multitasker will be added to the attacking line up to keep the ball moving and to keep it within the opposition side. The multitasker is the only robot that works in both the home and away halves, performing clearance duties before moving it up the pitch to the strikers. Once the ball is in the opposite side, it will support the strikers by attacking the ball whenever an opportunity presents itself. However the multitasker is linked to the striker and will hold off an attack if there is a signal that it is about to/currently striking.

11.2 STRATEGY TACTICS OVERVIEW: PHASE 2

Strategy name: PID++REDUX

The final strategy developed for the year 2007/2008 utilises all 5 robots in the following configuration:

The numbers in brackets refer to the robot ID number that the role is assigned to.

Defensive	Misc	Offensive
Goalkeeper **(0)**	Multi-tasker **(2)**	Striker **(3)**
Defender **(1)**		Shadow Striker **(4)**

A simple overview of the details of the various roles follows.

11.2.1 GOALKEEPER

The goalkeeper maintains a fixed defence in front of the goal line and uses prediction and straight-line movement to prevent the scoring of goals.

11.2.2 DEFENDER

The defender maintains a fixed defence at the top of the penalty box and uses prediction and straight-line movement to prevent the ball entering the home goal area.

11.2.3 STRIKER

The striker's objective is to score goals. It does this through angle prediction to get behind the ball and shoot in the direction of the opposing goal. The striker is only ever in use for offensive attack, never in the home goal area.

SHADOW STRIKER

The shadow striker's objective is to support the striker by maintaining a fixed distance behind. In the event that the shadow has a goal opportunity over the striker, the striker will stand down and let the shadow take the shot. The shadow is only ever in use for offensive attack, never in the home goal area.

11.2.4 MULTI-TASKER

The multi-tasker has the most varied role and takes on a variety of different tactics depending on the current zone the ball is currently in. In various situations it either takes a defensive role, a ball-clearing role or as an additional attacker.

Note the main change between Phase 1 and Phase 2, is the redefinition from having two strikers, of equal importance, to two entirely separate roles. The striker is now the most important goal-scoring role, whilst the shadow takes a support job, looking for opportunities when the striker loses the ball.

In addition, the multitasker now plays a lot more aggressively up front as well as being the only robot to handle the ball in both the home and away sections. By increasing the role up front, the aim isn't to take importance away from the jobs of the strikers, it was purely found that by making these adjustments, more goals were scored, possession significantly increased and general performance was up.

11.2.5 STRIKER/SUPPORT STRIKER

The striker uses "PID" control to quickly get behind the ball and utilises a wide strike cone to hit the ball in the rough direction of the goal. The approach the second time around was that the precise mathematics of phase 1 often led to too much tracking and not enough scoring, as a result a more imprecise method was adopted. The ethos of this strategy was that more attempts at goal, even if some where not exactly on target, would lead to more chances at scoring.

Meanwhile, the shadow would adopt a position offset behind the striker, waiting for an opportunity for when the striker loses control of the ball. In these instances, the striker would temporarily step down and allow the shadow to strike before maintaining their usual roles.

The striker is now the primary goal scorer. It was considered adding the same expert control algorithm used in phase 1, however due to the timing and the approaching of the national championships, a simpler tactic was chosen with less chance of observing the same problems that plagued the first iteration.

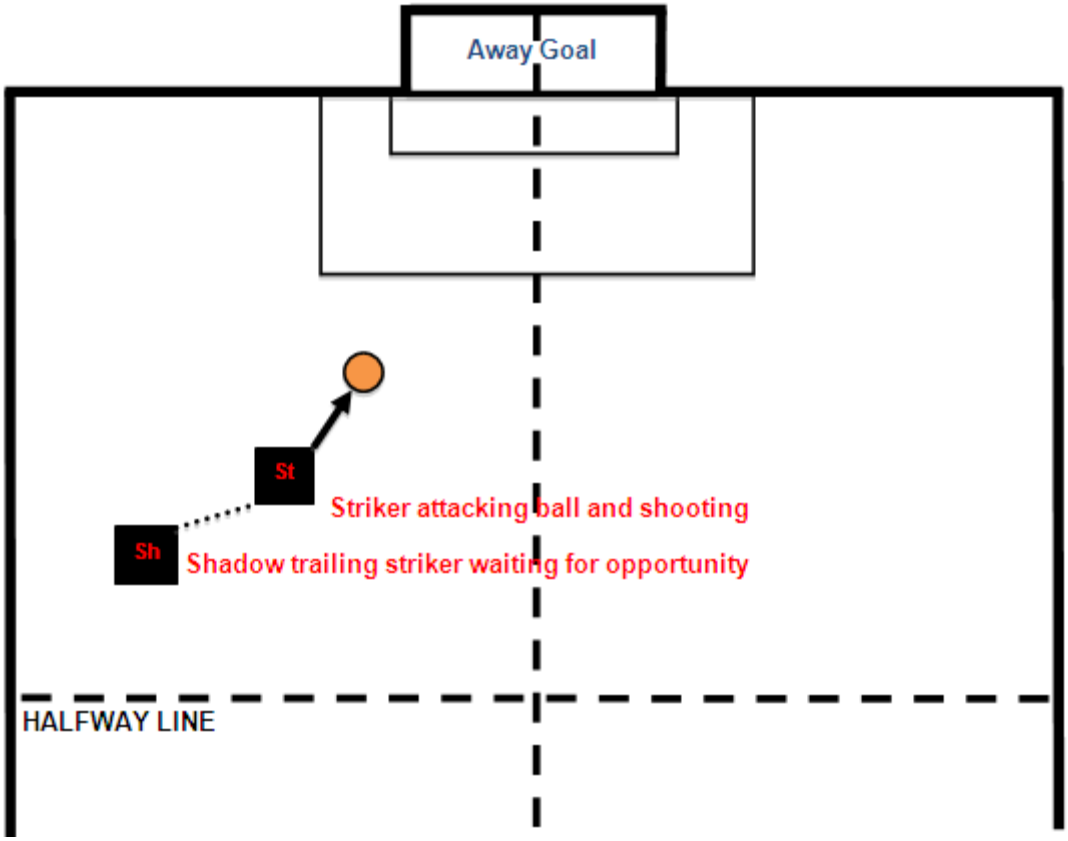

Figure 28 – Striker/Support Striker Overview (Offensive)

Noting the observations with the problems within phase 1 development, a brave move was taken to restrict the offensive robots to the half way line and the away side of the pitch. The striker and the shadow will not attempt ball clearance or aid the defence in any manner.

The decision to essentially exclude 2 players from the home goal area was taken as multiple penalties were being given away as a result of too many robots within the penalty box. Not only this but having a high concentration of robots in the same area leads to collisions and wasted time “unsticking” them.

As a result, when the ball is in our half, the offensive robots take up position just over the half way line, as pictured in the following diagram:

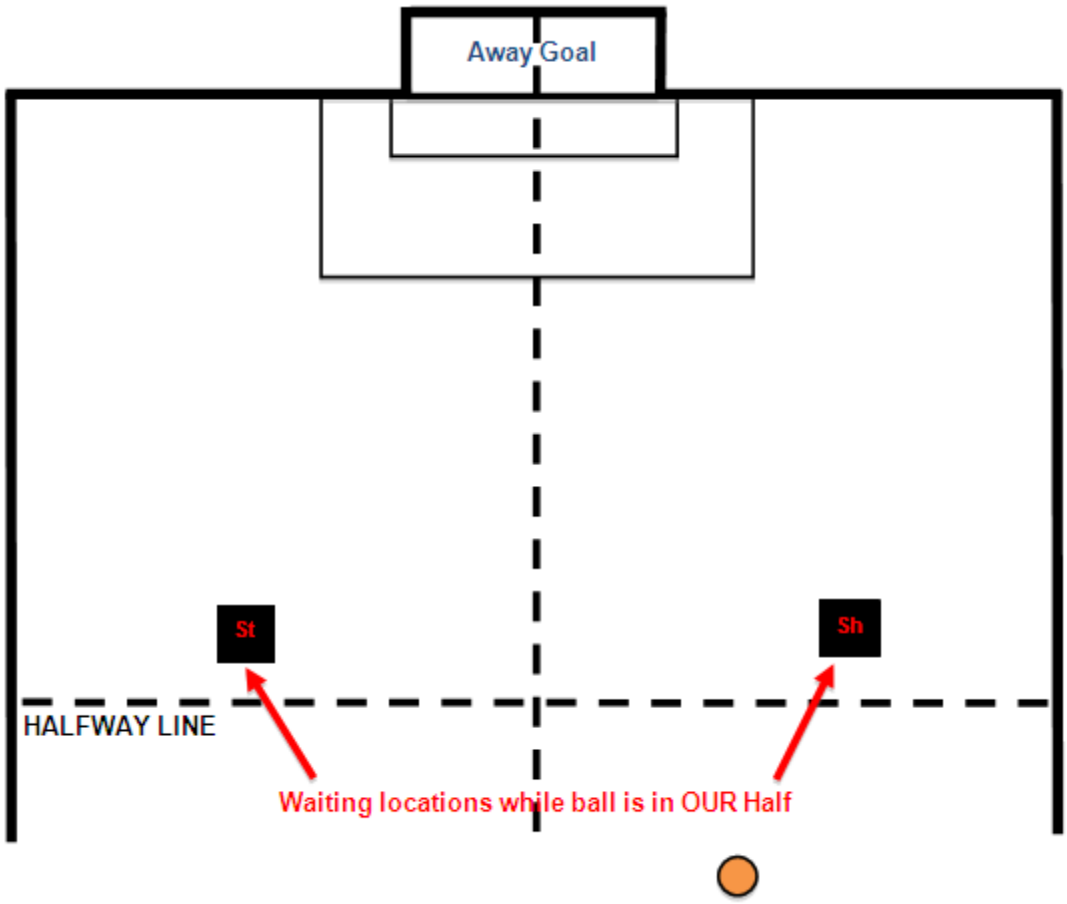

Figure 29 – Striker/Support Striker Overview (Defensive)

11.2.6 MULTITASKER

The multitasker maintains all of the defensive properties that have been previously described; therefore they will not be repeated again. When not performing defensive duties, the multitasker will work the ball up the pitch towards the strikers and the goal.

If the robot gets a chance to shoot it will take it, if not however and the striker indicates that it has the potential to shoot, the multitasker will back off and move to the designated wait position in the centre of the pitch.

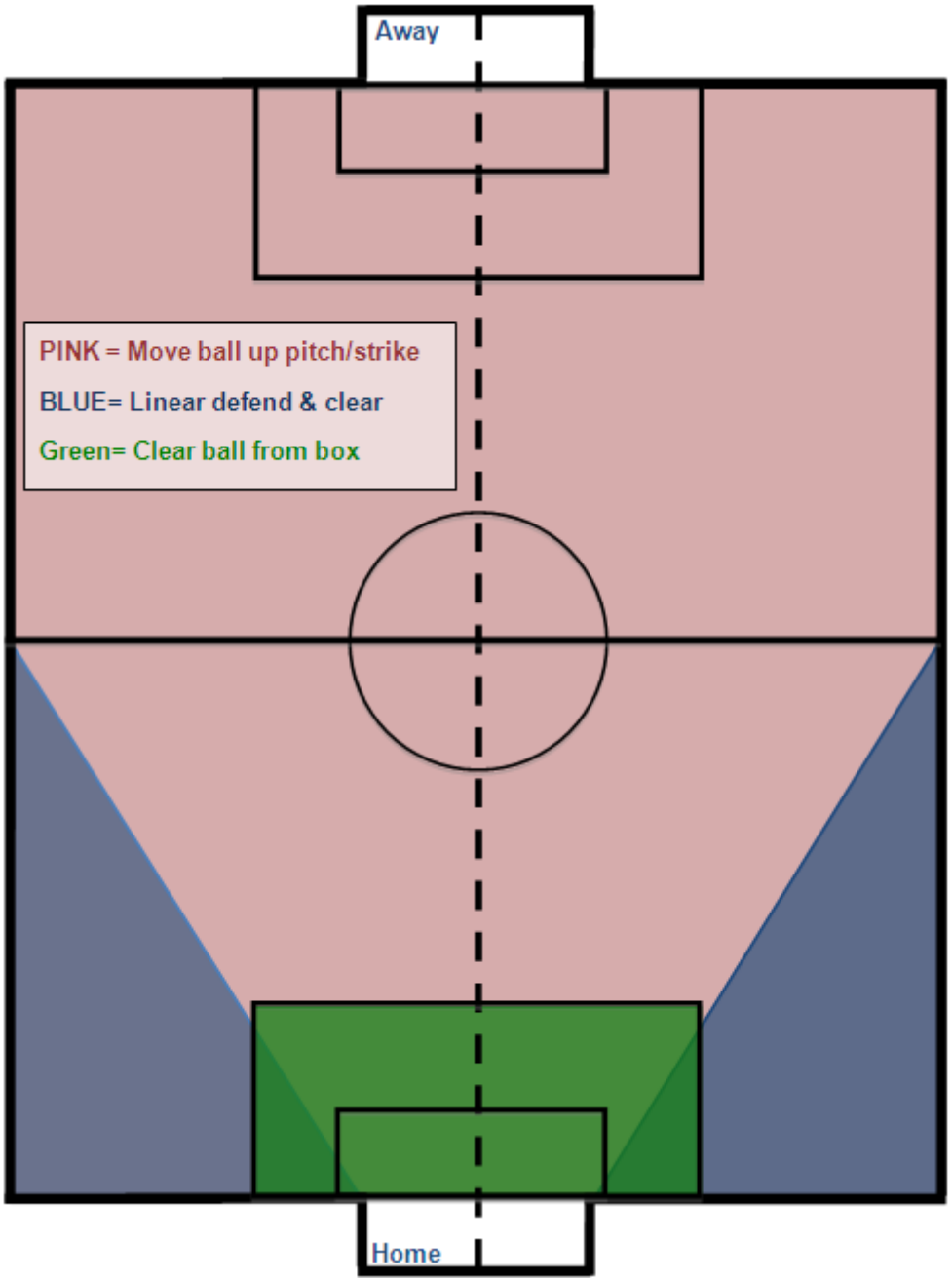

Figure 30 – Multitasker Overview

12.0 TESTING PHASE 2

The phase 2 strategy (PID++REDUX) was completed and ready approximately 3 weeks before the National Championships were to be held.

This allowed for 3 weeks of testing ensuring all the variables were optimised for game play. Testing consisted of monitoring tactics when an opposition side is not present as well as 10 full games, using the old Evo3 as the competition running last year's strategy.

The results of these test matches can be found below:

Date	Home Strategy	SCORE		Away Strategy	Result
15/02/08	PID++REDUX	8	2	PID	WIN
18/02/08	PID++REDUX	4	2	PID	WIN
20/02/08	PID++REDUX	8	2	PID	WIN
22/02/08	PID++REDUX	6	3	PID*	WIN
26/02/08	PID++REDUX	5	2	PID*	WIN
27/02/08	PID++REDUX	8	4	PID*	WIN
29/02/08	PID++REDUX	6	2	PID*	WIN

As can be seen from the tabulated results, the newly created PID++REDUX strategy typically beats the older strategy by a 2:1 margin. It may look as though improvements were not made over time and that even the strategy may have got progressively worse after the first four games, however notice the asterisks next to the opposition team name.

From game 5 onwards, instead of simply setting the robots off playing, games were then played against the strict rules against a competitor that knew how to control the opposition.

Playing matches and winning against some of the current PhD students who have previously worked on the project was inspiring and gave us every hope going into the National Competitions.