A Behaviour-based Approach to Position Selection for Simulated Soccer Agents

Matthew Hunter, Kostas Kostiadis, Huosheng Hu

Department of Computer Science, University of Essex,
Wivenhoe Park, Colchester CO4 3SQ, , UK,
Email: mchunt@essex.ac.uk, kkostis@essex.ac.uk, hhu@essex.ac.uk

Abstract

Selecting an optimal position for each soccer robot to move to in a robot football game is a challenging and complex task since the environment and robot motion are so dynamic and unpredictable. This paper provides an overview of behaviour-based position selection schemes used by Essex Wizards'99 simulated soccer team, a third place in RoboCup'99 simulator league at Stockholm. The focus concentrates on how each position selection behaviour is selected for individual robot agents. The factors that need to be considered and the architecture used to implement such position selection are also described. Finally the team performance at RoboCup'99 is examined, and future extensions are proposed.

1. Introduction

The Robot World Cup initiative (RoboCup) is an attempt to foster AI and intelligent robotics research by providing a uniform task, the game of soccer [4]. Some of the software technologies include design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning and planning, intelligent robotics and sensor fusion and so forth [5]. In addition a number of hardware advances are required to achieve the stated goal of developing a robot soccer team capable of beating the human world cup champion team in the middle of this century. While the challenge of achieving the task is clear, the aim is to offer a goal that requires significant advance in the field, namely *Landmark Project*. So far three RoboCup Games and Conferences have been held [1]. The first RoboCup was held during the International Joint Conference on Artificial Intelligence (IJCAI-97) at Nagoya, Japan with 37 teams participating. RoboCup-98 was held at La Cite des Sciences et de I'Industrie (La Cite) in Paris to coincide with the human world cup competition, attracted 61 teams. The 3rd competition, RoboCup'99, took place in Stockholm in conjunction with IJCAI-99, with over 90 teams participated [9].

In the RoboCup Simulator League, the Soccer Server [8] provides a virtual pitch, and simulates all movements of ball and players. Players connect to the server via UDP/IP sockets over a local area network, and all communication must be carried out via the server. It is against the rules for players to establish additional connections, either with other players or some external program. Communication via the server is heavily restricted, only one player may speak at a time, and the range of the communication is limited. The player may only send one action command to the server in a cycle of 100ms. If no command is sent the player will be idle for that period, potentially giving the opposing team an advantage. If more than one command is sent one is picked at random. Sensor information is received from the server asynchronously, approximately every 150ms, although this depends on the quantity and quality of the information requested. A narrower or wider field of view can be requested, as well as a lower than normal quality setting that returns angle information for seen objects, but no distance information.

The server is responsible for generating 'visual' and 'auditory' information for the players, actually text strings that give a symbolic higher level indication of the agents perceptions. It also takes the action commands and parameters from the player, such as power and direction for kicks. Information passed between server and client is susceptible to noise, a player may not kick the ball exactly as intended, and the view of the pitch will also contain errors.

In this paper the focus is on the Position Selection Behaviour (PSB) for the Essex Wizards Simulated soccer team. More specifically, a brief description of the agent architecture for Essex Wizards is presented in section 2. The main problems in agent position selection, and ways to resolve them, are discussed in section 3. Section 4 details the position selection behaviours being developed. Some experimental results are given in section 5 to show the proposed Position Selection behaviours work well, achieving good performance in RoboCup'99 [6]. Finally conclusions and future work are presented in section 6.

2. Agent Architecture of Essex Wizards

The main objective for the Essex Wizards team is to build a firm research platform on which future work on multi-agent systems can be done. Currently, a multi-threaded approach is adopted to achieve real-time performance [6].

2.1 Modular Architecture

The modular architecture is adopted in overall agent implementation as shown in Figure 1. More specifically:

- ***Sensors*** **--** The Sensor module executes in a separate sensor thread [6]. Incoming visual and auditory information transmitted by the server is received and interpreted. The sensors only detect information from a limited area of the pitch, depending on the position and orientation of the player. This is unlikely to give sufficient information for

selecting a good action; therefore old information is passed to the Memory module. This allows the agent to use both current and past information to make decision.

- ***Actuators --*** This module is also implemented as a separate thread. It is responsible for sending the actions to the server at regular intervals to ensure that the player never misses an opportunity to act. The module simply waits for the next cycle to start, a time based delay as there is no message from the server to state that the cycle has changed. It then sends the last action received. This allows the agent to 'change its mind' about the best course of action in light of new information received by the sensors during a cycle. In addition the module receives messages from the agent regarding the next action to send along with unrestricted actions.

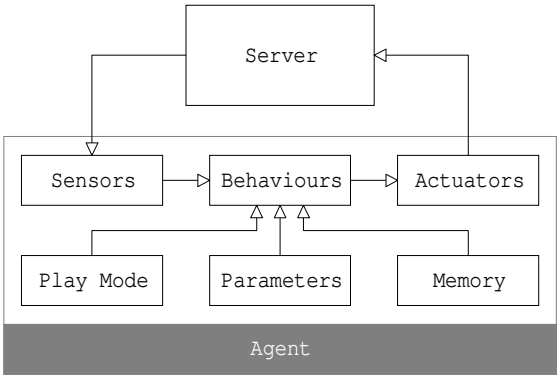


Figure 1 Essex Wizards Agent Architecture

- ***Play Mode --*** This module keeps tracking the current server play mode to ensure that player behaviours are appropriate to the current state of the game, i.e. not trying to kick the ball during the opposing team's kick off.
- ***Parameters --*** The Parameter module holds game constants for both server variables, such as the width of the goal, and player options such as the current formation.
- ***Memory --*** This module is responsible for keeping track of the current state of the pitch, and making that information available to other modules. The memory is updated explicitly when new information becomes available, based on new sensor information. In addition at the end of each cycle the state of the pitch for the following cycle is predicted by from the current state. This allows the agent to maintain more accurate world knowledge even when sensor data is slow or unreliable. Direct agent action information can also be used, if the agent kicks the ball for example the new speed and direction can be calculated. The memory does not perform any opponent modelling, it is not able to predict that an opponent player is kicking the ball for example. In such a situation out of date information would be retained until the ball is seen again. The predicted information can quickly become out of date, particularly for moving objects, so items in memory have a confidence value associated with them. If an object has just been seen the confidence will be high, as more predictions are made about the current position the confidence drops. Below a certain confidence value the object is ignored.
- ***Behaviours*** -- Based on information from the Sensors, Play Mode, Parameters and Memory modules the Behaviours module selects an action to perform and sends the result to the Actuators module for dispatch to the server. This is the most complex module since it does the "thinking" for the agent.

2.2 Robot Behaviours

The Behaviour module in each agent needs to carry out a number of tasks, including the agent's "thought processes". It must provide not only the low-level behaviours such as moving, dribbling and kicking, but also the high-level behaviours such as selecting where to move to, which teammate to pass to, and selecting between the behaviours, as shown in Figure 2.

Low-level Behaviours

At the lowest level any decisions made by the agent must be reduced to the core primitive actions provided by the server, i.e. "Kick", "Turn" and "Dash". These have been extended to extract useful information from the Memory to centralise common tasks. For example the "Turn" action turns the player by a certain degree, and is often useful to turn to face a particular direction, e.g. to face where the ball is expected to be. This is still not sufficient however as some tasks require sequences of actions. In order to provide the options that are required by the high-level behaviour, extended primitives have been implemented. However, the extended primitives must be implemented in terms of the core primitives, they cannot add anything to the server, but can extend over time, allowing sequences of actions.

- ***AKick***: Akick (Advanced Kick) can deal with more complex situations by moving the ball in stages to a position where the desired kick can be made.
- ***Move***: If the agent needs to move it must mix turns and dashes to reach the desired location.

High-level Behaviours

The high-level tactical behaviours build on the top of low-level primitive behaviours. These provide the options available to each agent during the game, which is currently implemented as a rule based system.

- **Dribble**: Dribbling involves intermixing kicks, dashes, and possibly turns to move the ball across the pitch, while keeping the ball under the agent's control.
- **Look**: It is important to keep track of the current ball location. If the ball has not been seen recently, the agent must scan for it by executing a number of turns. This also updates the agent's picture of other objects on the pitch.
- **Intercept**: If the agent is to gain control of the ball it must move to the ball. This involves predicting the location of the ball at the time of intercept and moving to that location. If the ball is moving quickly towards the player it may be better to wait until the ball comes close to avoid overshooting. This uses moves and kicks.

- **Clear Ball**: If the agent is near the home goal and in danger of losing the ball to an opponent, and no teammates are in a position to receive a pass it may be necessary to clear the ball. This involves kicking the ball, using the Advanced Kick behaviour, away from the goal, towards the side of the pitch and the opponent's end.

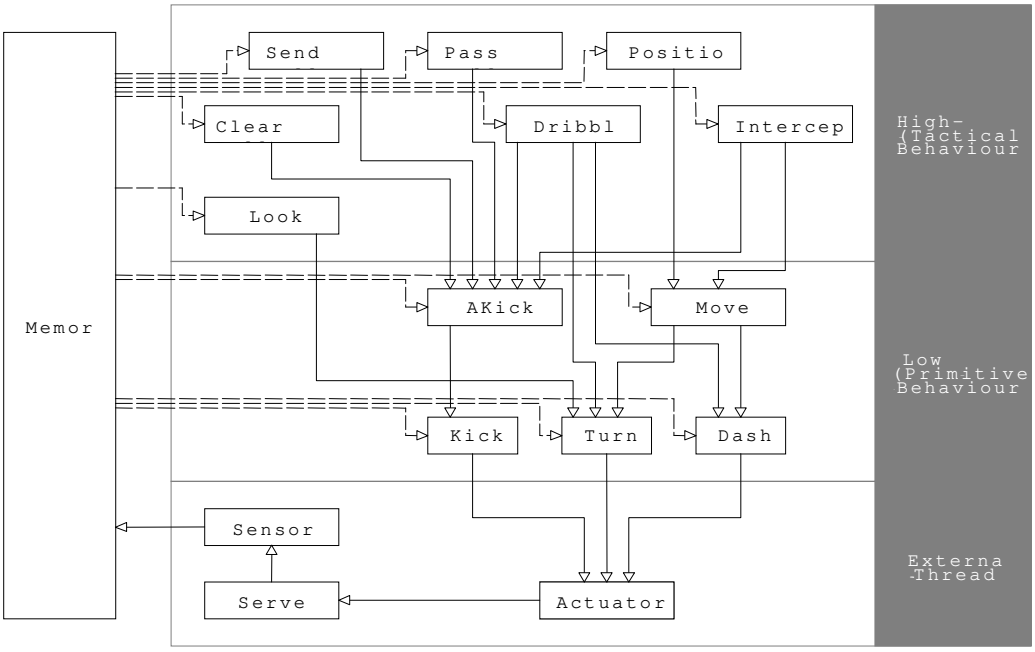


Figure 2 Agent Behaviour Architecture

- **Send Ball**: This is similar to clearing the ball but occurs when the agent is close to the opposing goal. In this situation it attempts to get the ball to a position from which a teammate can score.
- **Pass Ball**: The key decision in ball passing is to select a good position to pass. Pass Evaluation analyses the locations of teammates and opponents on the field in order to decide if a good pass, i.e. one with a high chance of success, is possible. This uses a decision tree for evaluation. It returns information about the most suitable teammate to pass to, along with an estimate of the success. The high-level behaviour can then decide if the pass is a good option, and if so uses the Advanced Kick behaviour to pass the ball.
- **Position Selection**: The Position Selection behaviour (PSB) examines the current view of the pitch and uses it to suggest a good place to move to. Selection of such a position is a non-trivial task, requiring information about the current role of the agent and the state of the pitch. For example, it is not desirable for the goalkeeper to move to a good position to make a shot on the opponent's goal, and a forward should avoid being so forward as to be caught in an offside trap.

The remainder of this paper concentrates on the different PSBs for selecting a good target position and how they can be combined for more complex behaviour.

3. Position Selection Behaviour

3.1 Objectives

The PSB is for the robot agent to select the best position on the pitch to move to. A good position is defined as one that will result in the good team strategy, perhaps being in a position to receive a pass, or preventing an opponent from doing so. In general the factors being considered in a game include the position of teammates and opponents on the pitch, the location of the ball, and the abilities of the players. Allowing the opponents to have a clear shot of goal is bad, while having a clear shot at the opponent's goal is good. It is also important to be in a good position to receive a pass from a teammate. Additionally it is important for the player to fulfil its role in the team. Players are assigned a role, based on the shirt number allocated by the server during initialisation. This role defines a home position, and often a home area, which represents the sphere of influence of the player in that role. The role, and therefore the home position and area, can change under certain circumstances.

There are two important issues that are not the responsibility of a PSB. Firstly it only decides where to move, the low-level behaviours are responsible for getting there safely. It is likely that a new position may be chosen before the old one is reached. In this situation the old target is ignored, and the player starts moving directly to the new position. Secondly each PSB does not operate globally for all players since the communication cost is very high for the simulator league. Each player therefore finds a target position independently during the game. It is also worth noting that Position Selection is one of several tactical options available to the player. It is only used when the player has no ball or specific task. If the player is the closest to the ball the player will just dash to the ball without using PSBs.

3.2 Definitions

This section describes each of the PSBs currently implemented. Figure 3 presents a typical example, where several PSBs position the player defensively to find a location that prevents the opponent team scoring. More specifically,

Fixed & Fixed Relative PSB

The Fixed PSB is extremely simple. The selected target point is the player's home location. The Fixed Relative PSB is similar; it returns a point at a fixed offset from the location of the ball.

Tracker PSB

The Tracker PSB protects the goal by selecting a position on the line between an opponent and the home goal. This is performed in two steps, selecting which opponent should be tracked and selecting an appropriate point on the line.

Selecting an opponent to track involves ranking them by the opponent's distance from either the home goal or the player's home location. The list of opponents is then checked in ranking order to see if a teammate is already nearby. The first opponent without a teammate nearby is the one chosen. If all the opponents within the home area have a teammate nearby, and there are opponents outside the home area then the last one inside is tracked, in the hope that the teammate, having a different home area, will be able to track a different opponent.

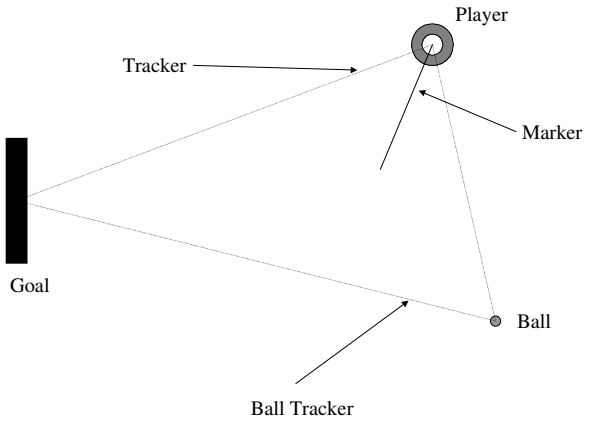


Figure 3 Tracker PSB

Once an opponent has been selected the distance between the opponent and the goal is measured and used to choose the target point. Two main values affect this choice, the minimum distance, and the relative distance. The final distance is given by $d = \max(\text{dist}(G,O)\times r, m)$ where d is the distance from the opponent being tracked, r is the relative distance, m is the minimum distance and G and O are the locations of the goal and opponent respectively. The point chosen is distance d along the line from O to G. If relative distance and minimum distance are both 0 then the chosen point will be the location of the selected opponent ($d = 0$). If relative distance is 0 and minimum distance is greater than 0 then the chosen point will always be minimum distance away from the selected opponent ($d = m$). Finally the point is adjusted to ensure it is inside the player's home area.

Unlike the Fixed and Fixed Relative PSBs, the *Tracker* PSB is able to use in the failsafe mechanism under the conditions: (i) No opponents are visible. (ii) All visible opponents are outside the home area. (iii) All visible opponents are inside the home area and already being tracked. (iv) The distance between the opponent and the goal is smaller than m (optional). In each of these situations it is either impossible to pick a suitable target (cases 1 and 2) or a better option than tracking may exist (cases 3 and 4).

Marker PSB

The *Marker* PSB is very similar to the *Tracker*. An opponent is selected for marking in exactly the same way as for tracking. Unlike tracking however the target position is chosen based not only on the position of the selected opponent, but also the position of the ball. The aim is to position the player so that it can intercept a pass to the opponent, and intercepting a shot on goal. As before the line from the selected opponent to the goal is considered. These lines cross at the selected opponent, and the angle between them is bisected. A mark line is created extending from the selected opponent, in the direction of the bisector, for a distance equal to the shorter of the two base lines. Once the mark line is obtained a point is selected along it in the same way as for the *Tracker* PSB.

Ball Tracker PSB

The *Ball Tracker* PSB does not perform the opponent selection phase since it chooses a target location on the line between the ball and the goal. It can optionally fail if the ball is not in the player's home area, or if the ball to goal distance is less than the minimum distance.

Offside Trap PSB

An offside trap consists of moving the defenders towards the opponent's end so that the opposing teams attackers cannot get past them without being offside. The Offside Trap PSB aims to improve on this by selecting a point on the offside line that shadows an opponent, preventing the opponent having a direct line across the Offside Trap line. The Offside Trap PSB will fail if the ball is too close to the offside line.

General Purpose PSB

The PSBs described above all have specific aims and uses. In addition a general-purpose scheme is needed for situations where it is less clear what action should be carried out. The *strategic position by attraction and repulsion* (SPAR) method [9] was used as the base of the *General Purpose* PSB for the Essex Wizards. SPAR works by maximising the distance between the target point and other players on the pitch (both opponents and teammates), and minimising the distance between the target point and the opponent's goal, and between the target point and the ball. Four additional constraints are also applied: (i) Stay in an area near the player's home location. (ii) Stay within the pitch boundaries. (iii) Avoid being in an offside position. (iv) Stay in a position in which it is possible to receive a pass.

4. Implementation Consideration

Initially a single monolithic method for position selection had been adopted, based on the *strategic position by attraction and repulsion* (SPAR) method [9]. SPAR is only one method of selecting a position however, and it is not always suitable, for example it cannot be used to mark an opponent. Another possibility was to implement SPAR and any other necessary methods independently. The player can then be responsible for selecting which method should be used. This has the disadvantage that it complicates the main body of the player, especially if the interface to each method is different. It also complicates the overall interface between player and Position behaviour.

For example, a simple rule is for a defender to track an opponent if the ball is in the defenders half of the pitch, or set up an offside trap if the ball is in the opponents half. If the formation changes however, the player may no longer be a defender, requiring a completely different set of rules. The number of possibilities rapidly escalates making the selection of a method a complex task. It becomes unclear which method is used in a given situation, and there is no easy way to logically group all the situations in which a specific method may be used.

In order to present a simple interface to the core player a more complex architecture has been selected. A library of PSBs has been produced, representing the different methods that can be used to select a position, for example tracking and setting an offside trap. Positioned objects are created as instances of PSBs for certain circumstances. The core player then activates a specific PSB based on the current situation to find the position to move to. This brings consistency to the core player, in order to find the desired target location it simply activates the PSB for the current tactical situation. There is no need for the player to know which PSB underlies the PSB, the interface to each is identical.

Multiple Players

Co-operation between players is implemented through the design of the positioning architecture within specific PSBs.

- Firstly, basic indirect communication is used by certain PSBs. For some schemes the observed positions of teammates is taken into account when selecting a suitable position. If all the potential good positions are already taken by teammates a player can deliberately move close to a teammate to try and get the teammate, which may have more options, to select a different position.
- Secondly although each player uses identical code, the configuration of each, described in the next section, is different. This leads to different behaviours of individual players, ensuring that not all players decide to be defenders for example. Once again this is currently designed into the player explicitly.

Similar reactive approaches have been used elsewhere, particularly in behaviour based robotics in [2]. Subsequent work in [3][7] used multiple robots capable of forming and maintaining formations, such as flocking. Such formations would be simple to implement using a suitable PSB.

External Configuration

Rather than passing a large number of mostly static arguments between the core player and Position Selection most of the configuration, specifically the Position Selection Scheme to use and the associated options, is extracted from an external configuration file at start up. All players share the same file, which is split into sections for the different roles. The player extracts the information for the role to be played in each formation.

This makes a clear distinction between tactical and implementation details, and hides many implementation details completely. For example each PSB has a different set of options. Accessing each PSB directly requires that the player know these options. In addition changing which PSB was used in a given situation may require a completely different set of options.

Failsafe Operation

In certain situations a single PSB may not be able to select a good position, due to the state of the pitch. For example an offside trap is no use if the ball is already beyond it. In this situation there are several options available. The PSB could return an error signal of some kind, but this would need checking by the high-level behaviour, which would then need to either chose a default position itself, or redirect the request to an alternative PSB. Alternatively other high-level behaviours could guarantee never to call a PSB unless it will succeed. These options increase the complexity of the high-level behaviour that in the latter case would need to know which underlying PSB was in use to ensure that it would not fail. To avoid this, a PSB guarantees it will always return a valid position to the high-level behaviour. By default if a PSB fails to generate a valid position internally, the home location of the player is returned. Constructing a stack of PSBs can change this situation. If the first one fails the next one is called, and so on down the stack until one succeeds.

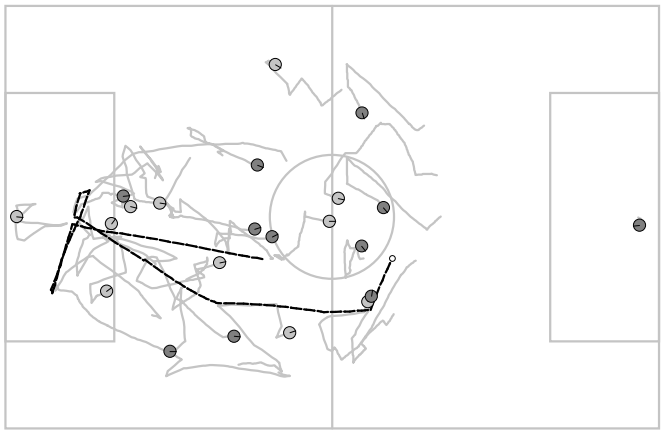


Figure 4 Offside Trap in Action

5. Implementation Results

Each PSB described in the previous chapter has been tested extensively in isolation to ensure that it works as designed. They must also perform well in a real game situation.

It is difficult to give a clear indication of a dynamic environment such as a RoboCup game on a static page. The diagrams in this section were produced from the logfile of the game between Essex Wizards and Rolling Brains at the RoboCup Competition in Sweden. Essex Wizards are dark grey players, and the lines represent the movement of players over time. The small white circle is the ball, and the black dashed line is the path taken by the ball. Figure 4 shows the pitch during 150 cycles near the middle of the second half of the game. Concentrating on the right end of the pitch, occupied by Essex Wizards several actions are clear. The goalkeeper has moved very little, conserving energy as it is a long way from the action. The four defenders are more interesting. At the start of the period shown the ball is moving towards the opponent's end of the pitch, and the four defenders move forwards to take position for an offside trap. As the ball comes back the offside trap breaks as the defenders try to improve their positions. This can be seen more clearly in Figure 5, which is taken from earlier in the same game.

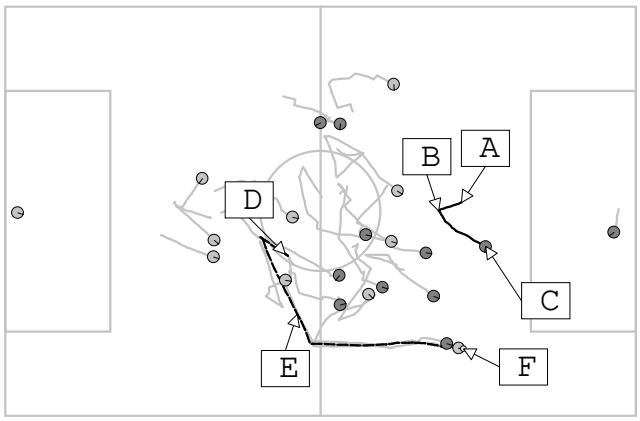


Figure 5 Ball Tracking in Action

The points A and D represent the positions of the Essex Wizards sweeper and the ball respectively at around cycle 3150. The ball is moving in to the other half of the pitch, and the sweeper moves forward to enforce the offside trap. Before the sweeper arrives the ball returns close to the centre line (E) and the sweeper (B) immediately begins ball tracking instead, attempting to reach a point between the ball and the goal. The sweeper currently moves more slowly than the opponent with the ball, as it needs to recheck the location of the ball periodically. Even at point C, the sweeper cannot see the ball at point F without stopping and turning.

6. Conclusions & Future Work

This paper has introduced the team architecture of Essex Wizards, focusing in particular on the Position Selection behaviours being implemented. Effective position selection requires that a player choose an optimal position beneficial to the team as a whole. The choice must be made based on locally available information only. Building and maintaining an up-to-date image of the pitch is difficult due to the slow rate of sensing and communication.

The proposed Position Selection behaviours have been designed to take advantage of general knowledge regarding the game, using formations and roles to restrict movement, while allowing sufficient freedom to avoid breaking down when unfamiliar situations are encountered. This will be expanded in the future to take account of more information that is not currently used, and to make the system more adaptable. The abilities of the Essex Wizards team was put to the test at RoboCup '99 where it achieved the third prize for its excellent performance.

The future work for the Essex Wizards simulator team will focus on the following issues: (i) *New PSBs* -- The need for attack strategies and defensive ones requires new general purpose PSBs. (ii) *Learning* -- Learning is required for players to automatically improve their positioning behaviours instead of the hard-coded approach currently used.

Acknowledgements

We would like to thank Michael Seabrook and Nikolaos Kalyviotis who made their contributions to the project, and the University of Essex for the financial support to the project by providing the Research Promotion Fund DDP940.

References

[1] Asada, M. and Christensen, H.I. *Robotics in home, office and playing field,* Proc. 16th Int. Joint Conf. on Artificial Intelligence, Vol.2, Stockholm, pp 1385-1392, 31 July - 6 Aug., 1999

[2] Brooks, R. *A robust layered control system for a mobile robot*. IEEE J. Robotics & Auto., RA-2, pp 14-23, 1986

[3] Hu H., Kelly I., Keating D., Vinagre D., Coordination of Multiple Mobile Robots via Communication, Proc. SPIE'98, Mobile Robots XIII Conference, Boston, pages 94-103, Nov. 1998

[4] Kitano, H. *RoboCup: The Robot World Cup Initiative* In proceedings of The 1st Int. Conf. on Autonomous Agents (Agents-97) Marina del Ray, The ACM Press, 1997

[5] Kitano, H., et. Al., *RoboCup: A Challenge Problem for AI and Robotics* RoboCup-97: Robot Soccer World Cup I, Kitano, H. (ed), Springer, Lecture Note in Artificial Intelligence 1395 pp 1-19 1998.

[6] Kostiadis, K. and Hu, H. *A Multi-threaded Approach to Simulated Soccer Agents for the RoboCup Competition*, Proceedings of 16th IJCAI RoboCup Workshop, pages 137-142, Stockholm, Sweden, 31 July - 6 August 1999

[7] Mataric, M.J. *Interaction and Intelligent Behaviour*, Technical Report AITR-1495, MIT AI Lab, 1994

[8] Noda, I. Soccer Server: *A Simulator for RoboCup*. JSAI AI-Symposium 95: Special Session on RoboCup, 1995

[9] RoboCup Official Site http://www.robocup.org/, 1999

[10] Stone, P., *Layered Learning in Multi-Agent Systems* PhD Thesis, Carnegie Mellon University 1998