# Essex Wizards 2000 Team Description

H. Hu, K. Kostiadis, M. Hunter, N. Kalyviotis

Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, United Kingdom
Email: {hhu,kkosti,mchunt,nkalyv}@essex.ac.uk

**Abstract:** This article gives an overview of the Essex Wizards 2000 team participated in the RoboCup 2000 simulator league. A brief description of the agent architecture for the team is introduced. Both low-level behaviours and high-level behaviours are presented. The design issues regarding fixed planning and reinforcement learning are briefly outlined.

## 1. Introduction

In the RoboCup [6] Simulator League, the Soccer Server [11] provides a virtual pitch, and simulates all movements of the 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. Communication via the server is heavily restricted, only one teammate can be heard at a time, and the range of the communication is limited. The player may only send one exclusive command to the server in a cycle of 100ms. If no command is sent the player will be idle for that period, and if more than one exclusive command is sent one is picked at random. Visual information is received from the server asynchronously, approximately every 150ms, although this depends on the quantity and quality of the information requested.

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

In this article the focus is on the behaviour-based approach for the Essex Wizards team. More specifically, a brief description of the agent architecture is presented in section 2. Section 3 outlines both low-level behaviours and high-level behaviours being adopted. The framework for the fixed plan and reinforcement learning is briefly introduced in section 4. Finally a brief summary is presented in section 5.

## 2. Essex Wizards Agent Architecture

The main objective for the Essex Wizards team is to build a firm research platform for multi-agent systems research. Currently, a multi-threaded approach is adopted to achieve real-time performance [3][7], and a modular approach [4] is adopted in the overall agent implementation as shown in Figure 1. More specifically:

- Sensors -- It is executed in a separate thread [2]. Incoming visual and auditory information from 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. The old information is passed to the Memory module. This allows the agent to use both current and past information to make decisions.

- Actuators -- This is a separate thread. It is responsible for sending the actions to the server at regular intervals to ensure that the player has an opportunity to act. When the sensor thread detects entry to a new cycle, the actuators start sending the queued actions.
- Play Mode -- This module keeps track of the current server play mode to ensure that player behaviour is appropriate to the current state of the game.

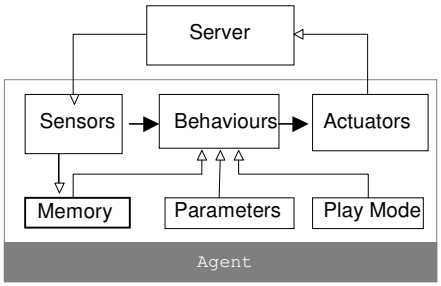



Figure 1 Proposed Agent Architecture

- Parameters -- This module holds game constants for both server variables, such as the width of the goal, and player options such as the passing power.
- 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 based on the current state. 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 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 the best course of action 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.

## 3. Agent Behaviours

The Behaviour module in each agent needs to carry out a number of tasks, including both the low-level behaviours such as moving and kicking, and the high-level behaviours such as selecting where to move to and which teammate to pass to [1], as shown in Figure 2.

### 3.1 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*. In order to provide the options for high-level behaviours, extended primitives have been implemented, including

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

### 3.2 High-level Behaviours

The high-level tactical behaviours are build on top of low-level primitive behaviours, and are currently implemented as a hybrid of Q-learning and rule based decision-making [9].

- 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.

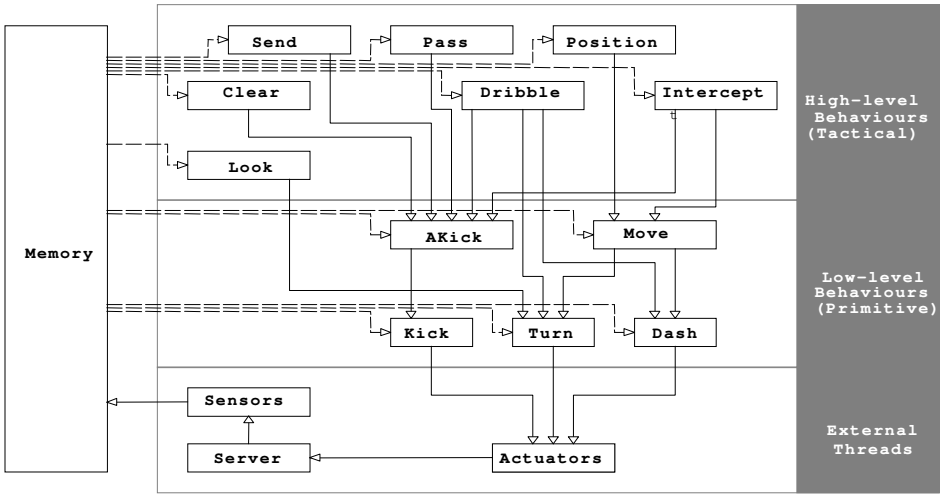



Figure 2 Agent Behaviour Architecture

- Look/Scan: If the ball has not been seen recently, the agent must scan for it by executing a number of turns or turn-neck commands. In some cases the agent might decide to improve its perception and execute a scan.
- Intercept: This involves predicting the trajectory of the ball and a suitable interception point to move to. Currently there are two modes of interception. The choice between them is based on the agent's role, position on the pitch and velocity of the ball.
- 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 to safety. This involves kicking the ball, using the *AKick* behaviour.
- Send Ball: This occurs when the agent is close to the opposing goal. In this situation it attempts to beat offside traps and advance into the enemy half of the pitch.
- Pass Ball: Pass Evaluation analyses the locations of teammates and opponents on the field in order to decide if a good pass is possible. It returns which teammate should pass to, and an estimate of the success.
- 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. More details about PSB can be found in [5].

## 4. Fixed Plans and Reinforcement Learning

Since having a fixed plan for a situation that happens frequently is a good strategy, the Essex Wizards 2000 team has implemented several fixed plans in RoboCup2000. The framework for our fixed plans consists of three major components, namely *Triggers*, *Actions* and *Abort conditions*. The *Triggers* is used as signals to allow or forbid the actions or plans that are predefined. The *Actions* is a combination of low-level and high-level behaviours that are executed sequentially. The *Abort condition* is a safe guard to ensure the conditions of the environment are suitable for the fixed plan being executed.

The reinforcement learning has been adopted in our behaviour-based decision making process since it provides a way to programming agents by reward and punishment without needing to specify how the task is to be achieved [1][7]. Each time the agent receives inputs, it then chooses an action to send. The action changes the state of the environment

and also provides the agent with either a reward if it does well or a punishment if it does badly. The agent should choose actions that maximise the long-term sum of rewards.

It should be noticed that the agents in our implementation not only have different roles and responsibilities, but also have different sub-goals in the team. Hence, every individual agent tries to reach its own goal state, and cooperation emerges when the goals of all agents are linked together. The ultimate goal, scoring against the opposition, becomes a joint effort that is distributed among team members, see [7] for more details.

## 5. Summary

In order to satisfy all the necessary timing constraints for real-time simulated football-playing agents, a multi-threaded implementation has been adopted in the Essex Wizards team so that the agents can perform various computations concurrently and avoid waiting for the slow I/O operations [3][8]. Moreover the behaviour-based approach plays a key role in building the Essex Wizards 2000 team. A decision-making mechanism based on reinforcement learning enables co-operation among multiple agents by distributing the responsibilities within the team [7].

The current focus for our Essex Wizards team is on adaptive position selection, flexible fixed planning, optimal decision making, and multi-agent learning.

**Acknowledgements**: We would like to thanks the CMU'99 team who contribute their source code to anyone who works on the RoboCup simulation league. We have benefited from this in our implementation. This research was financially supported by the university RPF award DDP940.

## References

1. Balch T.R., Integrating RL and Behaviour-based Control for Soccer, Proc. IJCAI Workshop on RoboCup, 1997.
2. Burns A. and Wellings A., Real-time Systems and Programming Languages, Addison-Wesley, 1997.
3. Butenhof R.D., Programming with POSIX Threads, Harlow, Addison-Wesley, 1997.
4. Hu H., Gu D., Brady M., A Modular Computing Architecture for Autonomous Robots, Int. Journal of Microprocessors & Microsystems, Vol. 21, No. 6, pages 349-362, 1998.
5. Hunter M., Kostiadis K., Hu H., "A Behaviour-based Approach to Position Selection for Simulated Soccer Agents", 1st European Workshop on RoboCup, Amsterdam, 28 May - 2 June 2000
6. Kitano H., RoboCup: The Robot World Cup Initiative, Proceedings of the 1st Int. Conference on Autonomous Agent, Marina del Ray, The ACM Press, 1997.
7. Kostiadis K. and Hu H., Reinforcement Learning and Co-operation in a Simulated Multi-agent System, Proc. of IEEE/RJS IROS'99, Korea, Oct. 1999.
8. Kostiadis K. and Hu H., A multi-threaded approach to simulated soccer agents for the RoboCup competition, IJCAI'99 workshop on RoboCup, 1999.
9. Mataric, J. M., Interaction and Intelligent Behaviour, PhD Thesis, MIT, 1994.
10. Nissanke N., Real-time Systems, London, Prentice Hall, 1997.
11. Noda I., Soccer Server: A Simulator for RoboCup, JSAI AI-Symposium 95: Special Session on RoboCup, 1995.
12. Stone P. and Veloso M., Team-Partitioned, and Opaque-Transition Reinforcement Learning, Proc. 15th International Conference on Machine Learning, 1998.