UvA Trilearn 2001 Team Description

Remco de Boer, Jelle Kok, and Frans Groen

Faculty WINS, Department of Computer Science
Intelligent Autonomous Systems Group, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{remdboer,jellekok,groen}@science.uva.nl

Abstract. This paper describes the main features of the *UvA Trilearn 2001* soccer simulation team. *UvA Trilearn 2001* is a new team that participated for the first time in 2001. It has been built from scratch and does not contain any code copied from other RoboCup teams. Topics that will be discussed include our architecture, world model and synchronisation method. On a higher level we will talk about an optimal scoring policy and about our fast-play strategy which makes use of heterogeneous players. *UvA Trilearn 2001* finished 5th in the German Open 2001 and reached 4th place at the RoboCup 2001 World Cup.

1 Introduction

The *UvA Trilearn 2001* soccer simulation team was built by two masters students from the Intelligent Autonomous Systems Group at the University of Amsterdam for their graduation project. For two reasons we decided not to copy any code from other RoboCup teams. Firstly, we found that the released source codes did not conform to regular software standards, i.e. they were not well structured and scarcely documented. We therefore thought that it would be more time-consuming to reuse the code than it would be to write it all ourselves. Secondly, we also felt that the lower levels used by the top teams from recent years could be improved in several ways. We thus decided to build a new team from scratch. Much of the initial effort has therefore gone into getting the lower levels to work. We have spent a lot of time on this particular task since we felt that the lower levels in the architecture would be the most crucial for the success of the team. Furthermore low-level imperfections can not be compensated for by high-level behaviour added later on. We had several ideas for improvements in the low-level methods used by top teams from the past, which we have successfully implemented. This has among other things led to an advanced synchronisation method and accurate estimation techniques for positions and velocities. The higher levels were added later on. This resulted in a fast-play strategy and an optimal scoring policy. During the project we have also focused much attention on software engineering issues [2, 6] to facilitate future use. This has led to highly modular object oriented code and to a multi-level log system for quick debugging (similar to [8]). Much effort has also gone into documenting our code for future release using the documentation system Doxygen [4]. The main features of *UvA Trilearn 2001* will be discussed in the remainder of this paper.

2 Architecture

Agents are capable of perception, reasoning and acting. We use a multi-threaded architecture which allows our agents to use a separate thread for each of these three tasks. In this way we minimize the delay caused by I/O to and from the server so that agents can spend the majority of their time thinking about their next action [5]. We use a 3-layer architecture for each agent. The threads used for perception and acting can be seen as the bottom layer which hides the soccer server details from the other layers. On top of this we have the *Skills Layer* which uses the functionality offered by the layer below to implement the different skills of each player (e.g. marking). The *Control Layer* then selects the best possible action from the *Skills Layer* depending on the current world state.

3 World Model

The world model of each agent is a probabilistic representation of the world state based on past perceptions. For each object an estimation of its position and velocity is stored (among other things) together with a confidence value which indicates the accuracy of the estimation. We have experimentally compared different methods for estimating positions and velocities of the dynamic objects in the simulation and used the best of these methods based on statistical results. The world model is updated when new information about the world is received by the agent and thus always contains the last known information. Objects which are not seen are also updated based on their last observed velocity. The confidence in the estimate decreases however for each cycle in which the object is not seen. Our agents also use communication to improve the accuracy of their world model. Depending on the position of the ball, the agent that has the best view of the field communicates his world model to nearby teammates. The teammates that hear this message then update their world model using the communicated information about the parts of the field that are currently not visible to them. Experiments have shown that on average the world model of an *UvA Trilearn 2001* agent contains up-to-date information about 17 players on the field. In addition, the world model also contains various methods which use the low-level world state information to derive higher-level conclusions.

4 Synchronisation

Synchronisation is an important issue in the soccer server, since actions that must be executed in a given simulator cycle must arrive at the server during the right interval [3]. We use an advanced synchronisation method which in each cycle determines the optimal moment to send an action to the server. This moment depends on the arrival times of the different server messages and thus changes from each cycle to the next. Furthermore our method guarantees that the action sent is always based on the latest information from the server. After sending an action it is checked whether the server has actually performed it. When this

is not the case (i.e. the action did not arrive in time) no action is sent in the next cycle to avoid a *clash*[1]. Experiments have shown that our synchronisation method clearly outperforms several others used in the past. During a full-length match consisting of 6000 cycles the number of *holes*[2] for each agent is less than 0.2% of the total number of cycles and the number of *clashes* is less than 0.01%.

5 Optimal Scoring

Our agents use an optimal scoring policy [1] which determines the optimal shooting point in the goal together with an associated probability of scoring when the ball is shot to this point. This probability depends on the position from which the ball is shot and on the position of the goalkeeper. The optimal scoring problem was solved by decomposing the problem into two subproblems. The first subproblem is to determine the probability that the ball will enter the goal when shot to a specific point in the goal from a given position. This probability was estimated by performing an experiment in which a player was placed straight in front of the goal and the ball was shot to the middle of the goal from different distances. From the resulting data set it was possible to determine the average deviation of the ball given the distance that the ball had travelled. Using this function we could calculate the probability that the ball would enter the goal when shot at a certain point from a given position. The second subproblem was to determine the probability of passing the goalkeeper in a given situation. This was estimated by performing an experiment in which a player shot the ball in a straight line while a goalkeeper would be placed in different positions relative to this player. A data set was formed by recording several values for each situation together with a boolean indicating whether the goalkeeper had intercepted the ball or not. From these data it was possible to extract a function which determined the probability of passing the goalkeeper. Combining the solutions to both subproblems led to the optimal scoring policy used by the agents.

6 Team Strategy

The main philosophy of *UvA Trilearn 2001* is to keep the ball moving quickly from each player to the next and preferably in a forward direction. In addition, our agents often try to pass the ball into the depth in front of the wing attackers at the side of the field thereby cutting through the opponents' defense and disorganizing their team. The effectiveness of this strategy is greatly enhanced by the use of heterogeneous players on the wings. Although these players become tired more quickly, it is their faster speed which makes the difference. We have developed a formula for determining the quality of a heterogeneous player. For

[1] A *clash* occurs when two actions are sent during one cycle. This situation is undesirable since the server then randomly chooses one for execution.

[2] A *hole* occurs when no action is sent during a cycle. This is also undesirable since missing an action opportunity may lead to the opponents gaining an advantage.

each player type offered by the server, this formula returns a utility value based on the player parameters. In this way we can determine which player types are best suited for which positions on the field. The standard formation used by the team is a 4-3-3 formation. Inside this formation each player chooses a strategic position determined by a weighted sum of his home position in the formation and the current position of the ball which serves as an attraction factor [7, 9]. In general, the behaviour of each agent depends on where the ball is located on the field. We have divided the field into different areas and the action that an agent chooses depends on the area in which the ball is located.

7 Results

UvA Trilearn 2001 has participated in two international robotic soccer competitions. The first one was the German Open held in Paderborn in June of this year. In this competition we used a very simple high-level strategy due to the fact that up to that point we had spent most of our time on the lower levels. Despite this, we still managed to reach 5th place in the tournament. In the next two months we worked on our high-level strategy and on the optimal scoring policy. In the beginning of August we then participated in the RoboCup World Championship in Seattle. During this competition it was clear that the team had improved a lot since Paderborn. *UvA Trilearn 2001* finished in 4th place.

References

1. Remco de Boer, Jelle Kok, Nikos Vlassis: An Optimal Scoring Policy for Simulated Soccer Agents in the RoboCup Simulation League. To appear. Currently at http://gene.wins.uva.nl/~jellekok/robocup (2001).
2. Frederick P. Brooks Jr.: The Mythical Man-Month: Essays on Software Engineering. 20th anniversary edition, Addison-Wesley (1995).
3. Marc Butler, Mikhail Prokopenko, Thomas Howard: Flexible Synchronisation within the RoboCup Environment: a Comparative Analysis. In Proceedings of the RoboCup-2000 Workshop (2000).
4. Dimitri van Heesch: Documentation System Doxygen. At http://www.doxygen.org (1997).
5. Kostas Kostiadis, Huosheng Hu: A Multi-Threaded Approach to Simulated Soccer Agents for the RoboCup Competition. In Proceedings of the 16th IJCAI RoboCup Workshop (1999), 137-142.
6. Roger S. Pressman: Software Engineering: a Practitioner's Approach. 4th edition, McGraw-Hill (1997).
7. Luís Paulo Reis, Jośe Nuno Lau, Luís Seabra Lopes: FC Portugal Team Description Paper. At http://www.ieeta.pt/robocup/archive.htm (2000).
8. Patrick Riley, Peter Stone, Manuela Veloso: Layered Disclosure: Revealing Agents' Internals. In Proceedings of the Seventh International Workshop on Agent Theories, Architectures and Languages (ATAL-2000).
9. Peter Stone: Layered Learning in Multi-Agent Systems. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA (1998). Available as technical report CMU-CS-98-187.