Team Description Mainz Rolling Brains 2001

A. Arnold, F. Flentge, Ch. Schneider, G. Schwandtner, Th. Uthmann, and M. Wache

Department of Computer Science
Johannes Gutenberg-University Mainz
D-55099 Mainz, Germany

1 Introduction

The Mainz Rolling Brains 2001 team is based on our last year's team. Our modular design as described in [1] has proved to be efficient and flexible. Thus the team could easily be adopted to the soccerserver's new features and some of the weak spots of our team could be eliminated.

We use a three-layers concept for our agents: a technical layer (purely technical matters like server communication), a transformation layer (the tools and skills a player might use) and the decision layer. The decision layer consists of various modules for different tasks (goal shot, pass, ballhandling, positioning and standard situation). These modules rate the adequateness of their respective actions and compete for the control over the player. More detailed information about our team architecture can be found in [1] and [2].

One of this year's main focuses was positioning which has been completely revised. We will describe our new position module below. We made extensive use of our new tool "FUNSSEL" (a debugging tool including an extended soccer monitor) which made it possible to further improve our technical skills and some of our modules as well. Due to the lack of space we will concentrate in the following on the new positioning module, FUNSSEL and our method of self-localisation.

2 Self-Localisation

As the agents in the simulation league do not know their absolute position on the field, they have to calculate it from the visual data they get. This data contains information about distance and angle (relative to the agent's body orientation) of the objects in the agent's field of vision. There are several flags in and around the field (of which the exact positions are known) we can use to calculate the position of the agent. Visual data is not exact, i.e. the soccerserver provides distances and angles only with an intentionally added error. Therefore, our position is in an area with its borders given by these errors. To calculate the position of our agents we create polygons with five corners approximating this

A. Birk, S. Coradeschi, and S. Tadokoro (Eds.): RoboCup 2001, LNAI 2377, pp. 531–534, 2002.

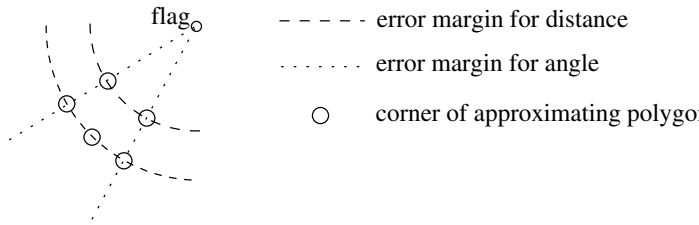

Fig. 1. A polygon approximating the area of the player's position

area as shown in Figure 1 for each visible flag. Since the error depends on the distance these polygons get the smaller the closer the flag is.

We intersect each two polygons and intersect the resulting polygon with the next. Polygons are intersected by iterating over the edges of the second polygon and keeping that part of the first polygon being inside the half-plain which is generated by the current edge and contains the second polygon. Finally, we calculate the agent's position inside the resulting polygon as the average of the polygon's corners. If there is no intersection of all polygons we use the weighted sum of the positions derived from all visible flags.

3 Positioning

We introduced a new positioning module this year giving each agent a very well defined role during a game. This should ensure that at least one agent feels responsible for an area of the field and that an attacking opponent is always attacked by at least one agent. The different roles of the agents are divided into four major groups: the defense, the defensive midfield, the offensive midfield and the attack. Currently we play with 3 defenders, 2 agents belong to each defensive and offensive midfield and 3 attackers.

Each agent has two standard positions, one in a more defensive, the other in a more offensive context. Depending on the ball position he places himself somewhere between these two positions. Positions are described as coordinates in a virtual 100 x 100 area. To get the actual position the player should take on the field these position values are projected onto a certain area of the soccer field. This area is determined by ball position, the opponent's offside line and a given minimum and maximum value for the area. This positioning system allows to generate certain tactical behaviours. For example, in defensive situations defensive midfield players are always placed in one line with the ball and therefore these players will almost always feel responsible for attacking the opponent with the ball while the whole defense stays covering the area between ball and own goal. In specific situations additional rules are triggered, e.g. if the defensive midfield agents fail to get the ball, the positioning behaviour is switched, i.e. one of the defenders attacks the opponent, and a defensive midfield tries to get behind the attacking defender to back him up. These rules have to be chosen very

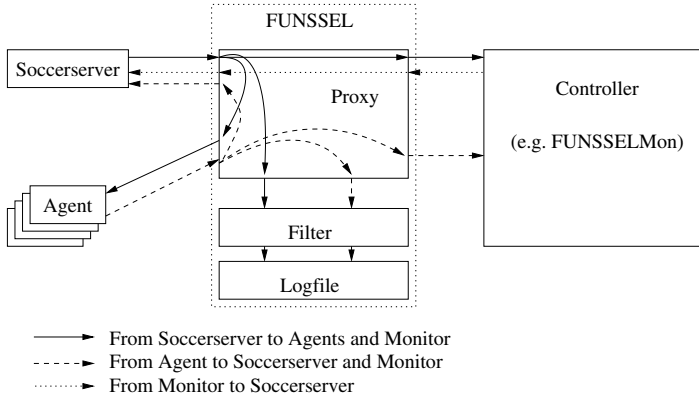

Fig. 2. Scheme of FUNSSEL functionality

carefully, so that they altogether lead to a complex behavior with well organized interactions between the players.

4 FUNSSEL - A Visualisation and Debugging Tool

FUNSSEL ("Flexible Utility for Network based Soccer Simulation using Extended Language") developed by the Mainz Rolling Brains is a powerful utility to debug simulation league agents. It consists of two single programs, *FUNSSEL* and *FUNSSELMon*.

FUNSSEL is a kind of proxy server that is placed between the soccerserver and both the agents and the monitor. FUNSSEL passes standard soccerserver commands in the usual way, while special commands allow redirection of commands directly to the monitor (e.g. FUNSSELMon) without sending anything to the soccerserver. All this is implemented using an extended version of the standard soccerserver protocol to make it easy to use the special monitor commands. It enables the players to report information about their current worldmodel and tactics or even to control the game. The whole communication is logged through a filter for a later replay. The logging of the players additional output may increase the logfile size dramatically. That means filtering out unneeded information from the soccermonitor protocol and compressing the data is necessary to reduces the logfile to a normal size. For a structural overview of FUNSSEL and the related processes, have a look at Figure 2 showing the data flow of different command types.

To use all the features described above, a special monitor is needed - the FUNSSELMon. Acting like a normal Soccermonitor it also offers extended functionality to control FUNSSEL and to gain more information about the game. Using special player commands FUNSSELMon can display some agent's worldmodel and special tactical information as graphics on the playing field as well

as simple logfile entries. While watching a game or logfile different colors can be assigned to players to display data of several agents simultanously. Other commands allow players to pause the game, do a drop ball or a freekick or to use the Soccermonitor set player command, all being useful for training or debugging certain game situations. To have a closer look at some technical issues during the game, a zoom function is available which allows to open separate windows for different regions, each at its own scale. FUNSSELMon can display technical information like the heterogenous player types, stamina values for all players, the kickrange and a viewcone showing the current vision of one or more players.

The main advantage of this concept is that it is nearly independent of the soccer server version used. FUNSSEL is not dependent on FUNSSELMon, therefore other monitor programs could be designed to provide for example a batch mode to train the agents and run games automatically.

5 Conclusion and Outlook

The newly developed debugging tool FUNSSEL offers a lot of assistance in monitoring and analysing soccer simulation league agent's behaviour. It clearly has improved our understanding of what the agents "see", on how they decide what to do and last but not least why they do not act like they are supposed to. Thus it helps to increase the strength of our agents' soccer team. For the next future we are planning to make use of machine learning techniques (e.g. Kohonen Feature Maps) for analysing games and changing tactical behaviour.

Acknowledgements: We wish to mention all current members of the Mainz Rolling Brains: Axel Arnold, Jochen Ditsche, Felix Flentge, Manuel Gauer, Marc Hoeber, Christian Knittel, Claudia Lautensack, Christian Meyer, Birgit Schappel, Christoph Schneider, Goetz Schwandtner, Thomas Uthmann, Martin Wache. We also would like to thank our former team leader Daniel Polani and our sponsors SerCon and F^3G.

References

1. Schappel, B., Schulz, F.: Mainz Rolling Brains 2000. In: Stone, P., Balch, T., Kraetzschmar, G. (ed.): RoboCup 2000: Robot Soccer. World Cup IV. Lecture Notes in Computer Science, Vol. 2019. Springer-Verlag, Berlin Heidelberg New York (2001)
2. Uthmann, Th., Meyer, C., Schappel, B., Schulz, F.: Description of the Team Mainz Rolling Brains for the Simulation League of RoboCup 2000. `http://www.rollingbrains.de/mrb2000/MainzRollingBrains2000.ps`