AllemaniACs3D Team Description

Philipp Vorst, Alexander Ferrein, and Gerhard Lakemeyer

Knowledge-Based Systems Group
RWTH Aachen,
Aachen, Germany
{vorst, ferrein, gerhard}@i5.informatik.rwth-aachen.de

Abstract. For more than three years our AllemaniACs RoboCup team has participated in the Middle-Size league and used the 2D soccer simulation as a test environment for our deliberative approach to team coordination. In order to tackle the trade-off between reactivity and planning intelligent courses of actions, the DR-Architecture was developed. This year we realize our concepts in the RoboCup 3D soccer simulation – a league which combines realistic features known from the Middle-Size league and coordination challenges arising in the 2D simulation. In this paper we sketch our approach, present the agent architecture, and show first results.

1 Introduction

In robotic soccer [1] one of the main challenges is the intelligent decision making of the soccer agents or robots. While with real hardware the focus also lies on issues like sensors or actuators, building stable hardware, designing robust highly integrated embedded systems, one can abstract from these issues in the Simulation league. This means that the designer is able to concentrate on the decision making of the agents. This does, though, not mean that problems like localization and perception do not have to be solved.

For the decision making of the soccer agent we proposed to follow a hybrid approach trying to combine deliberation with reactive behavior [2]. We believe that we need some form of deliberation or planning in order to be able to yield "intelligent" decisions. While it is advantageous to be able to make plans for future courses of actions it comes with the disadvantage of being computationally expensive. The fast-paced soccer does not allow to think too long for the next action to be executed. This is the reason why deliberation alone might not be successful in the soccer domain. Thus, we integrate a reactive component in the system architecture which is able to provide a next action to be performed immediately.

For deliberation we use the logic-based agent programming language Readylog [3] which is based on Golog [4]. It combines explicit agent programming with planning. For the specification of the soccer agents we want to adapt soccer moves from human soccer theory as described in [5, 6]. For reactive behavior we want (among other things) to apply our work described in [7] where we successfully applied decision tree learning for the 2D simulation environment.

So, the aim of the AllemaniACs3D soccer team are two-fold. First, we want to use our experience from the Middle-size league wrt. to tasks like localization and sensor fusion and our previous work on decision making in the robotic soccer context. Second, we want to gather new experience how our approach scales in a more complex, more realistic simulated soccer environments with 11 players. In this paper we describe our approach in some detail and present the current state of our work for the 3D Simulation league.

In the following section we describe our software architecture in greater detail before we turn to the world model of the 3D soccer agent in Section 3. In Section 4 we describe some of our basis skills. We give an outlook of the future work in Section 5.

2 Agent Architecture

While deliberation has many advantages for the decision making of an agent, it has the disadvantage of being slow compared to generating actions in a reactive fashion. In [2] we proposed a hybrid architecture which allows the combination of deliberation with reactivity. Here, we give a brief overview of our architecture. For more information we refer to [2].

Figure 1 shows the DR-Architecture. Based on the SPADES agent library [8], the *CommServer Interface* is directly connected to the simulation and is responsible for the coding and decoding of messages. Parsed data is used to update the *World Model*, which provides a representation of the world as well as strategic information. From the sensory input we build our world model. We discuss the world model in detail in the next section.

The decision module in the DR-Architecture is divided into three sub-modules. To be able to settle an action immediately the *Reactive Component* computes the next action to be executed based on the current game situation. In [7] we propose to formulate the reactive action choice as a classification problem and apply Quinlan's C4.5 [9] to learn the assignment between world situations and actions. The reported results are from the Simulation league. In the future we will apply these results to the Middle-size league. The main problem lies in the relatively many examples needed in order to learn an appropriate situation-action mapping.

The *Deliberative Component* calculates a plan projecting into the future choosing among the possible action sequences. The choice for the best action sequence is supported by an optimization theory. We apply decision-theoretic planning to calculate an optimal policy in the Readylog framework. We refer to [3, 10] for more details about the language Readylog.

Having an immediate action and a plan concurring for executing one needs to decide which of both to use. Currently, we are developing the *Action Selection* module based on a Reinforcement Learning approach. The idea is that the action selection module uses its observation which of the both components yields better results in terms of a global reward function in the respective world situation. First experiments in the 2D Simulation league are promising.

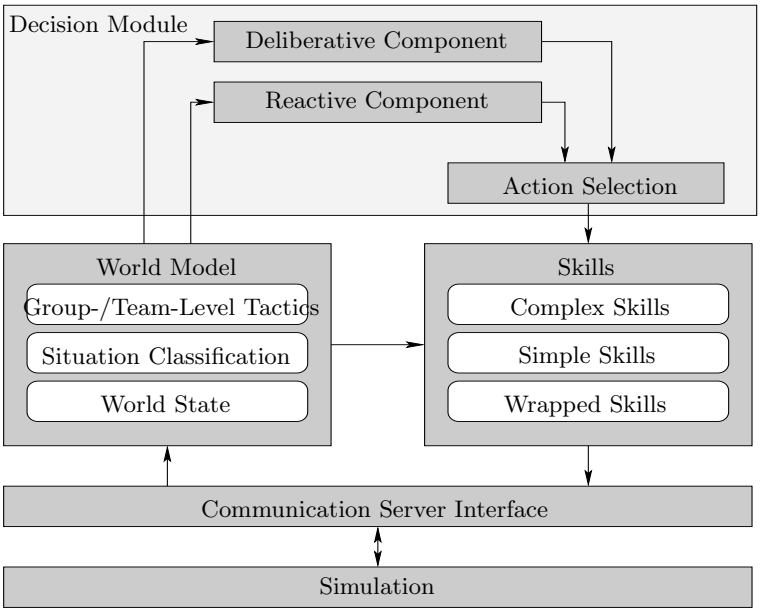

Fig. 1: DR-Architecture adapted to RoboCup 3D soccer simulation

3 World Model

The world model is layered according to the level of abstraction of the provided information. Each layer adds a compact portion of functionality based on the information of inherited layers.

The *world state* comprises a static model of the environment and current positions/ velocities of the ball and players. World model updates are only applied to this layer. Apart from the data which is required to estimate the velocities of dynamic elements of the playing field, this layer is memoryless.

World state is first of all subject to world model updates and filtering. For self-localization we employ a combination of dead reckoning and Kalman filter-based fusion of pose estimates (cf. [11] for the basic filtering technique). This method yields an average error of 0.055 meters in our experiments and outperformes the nearest-landmark heuristic.

The attributes which are provided by *world state* are available both in a quantitative and a qualitative fashion [12]. The ball position, for example, is characterized both by its numerical coordinates (x, y, z) and by a qualitative description such as (`zone_back`, `side_left`, `lifted`). Thereby this world model layer already prepares to be effectively accessible by the agent designer, who may want to specify the agent's behavior based on expressive descriptions of the world state.

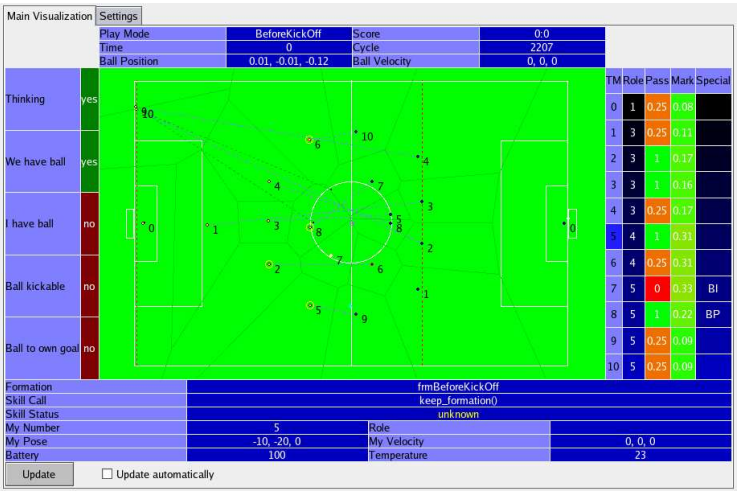

Fig. 2: World model visualization

The task of *Situation Classification* is to assess the current world state and to derive information on a higher level of abstraction. Here, for instance, the player who can intercept fastest and good pass partners are determined. With respect to tactical assessments, for instance, *Situation Classification* furthermore determines whether the current setting is offensive or defensive. Such an attribute is not only required for action selection, but also for the next higher level of the world model.

Group- and Team-Level Tactics is primarily concerned with the assignment of *roles* to players, the direct assignment of opponent players to be *marked* by a teammate, and the choice of an adequate situation-based *formation*. For the assignment of roles, the use of the so-called *Hungarian method* (e.g. see [13]) has proven to be adequate. Although the choice of simple cost functions such as the distance of a teammate from a formation-based strategic position can lead to unintended role switches in few cases, we consider the method to be preferable. It is more stable than greedy assignments, provably optimal with respect to the defined cost function, and computationally negligible.

4 Skills

The basic simulation commands `drive`, `kick`, `beam`, and `say` are wrapped as atomic skills within the *Skills* module (see Figure 1). Neither do they have a state of their own nor do they query any information from the world model in this kind of representation. Contrary to this nature, *simple skills* are those which can assume several internal states and which take world model information as additional input for their performance. They make use of exactly one of the atomic abilities. Examples of simple skills are `move_to` (although embodying

motion control and collision avoidance) or `block_position`. *Complex skills* such as `dribble_to` finally denote those ones which are composite of more than one lower-level abilities.

We decided to first implement them based on geometric models. In a later phase, we may replace them by machine-learned behaviors successively, exploring learning methods, which have successfully been employed by other teams. The advantage of our model-based approach, however, is that we can *reuse* the models for our high-level control.

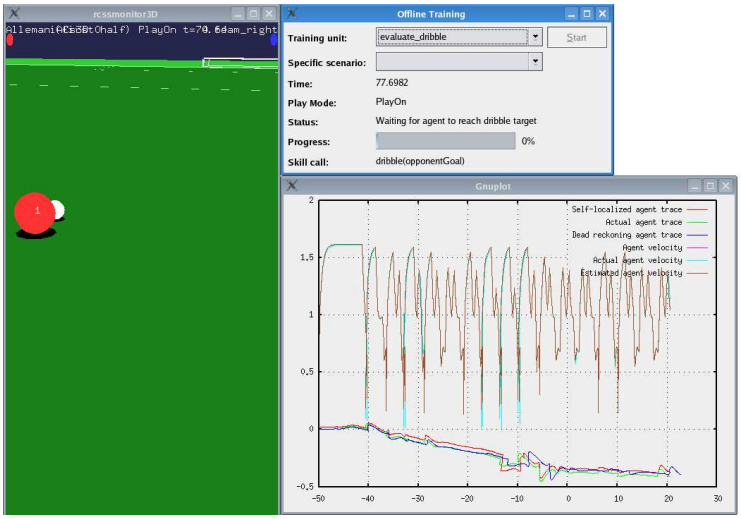

Fig. 3: Evaluation and training environment

In order to evaluate skills and world model data, we have established an experimental environment integrating various kinds of training modules. Each module comes up with an experiment setup and performance procedure, unified logging facilities, and live tracking of experimental results. This permits systematic analyses, debugging, and optimization of the agent's behavior. Figure 3 shows a screen shot of a recent evaluation session for the `dribble` skill, comprising the simulation window, a session control widget, and the data plot visualizing the agent trace and tracked velocities. That session, for instance, showed that the agent quite precisely followed the target direction, but which also indicated that the lower-bound velocity during dribbling had to be improved.

5 Current Work and Outlook

At the moment we are concerned with the low-level control of the 3D soccer agents, refining the skills and improving the world modeling. We especially focus on the extensibility of the low-level system. Currently, the environment data from

3D soccer server are very accurate and problems with noisy data are not severe. This is the reason why we yield good localization results with the Kalman filter. In the future, the simulation will become more realistic and other methods like Monte Carlo localization might be needed. At that point our experience with robotics might pay off.

Our decision making layer consists of a hand-coded reactive agent at the moment. Clearly, this is only the starting point in this project. In the near future we want to implement the deliberative component and the action selection as part of the DR-Architecture. In this context we are going to continue our work on adapting soccer theory in the RoboCup context (cf. [5, 6]). The 3D league is of special interest for the work on soccer theory as it is the first real 3D league. In the Middle-size league, for example, the representation of pass reachability (i.e. when a pass to a player can be played) in 2D is sufficient at the moment. More complex soccer moves can be undertaken and we expect interesting results about the scalability of the planning approach used in the agent programming framework Readylog.

References

1. http://www.robocup.org (2006)
2. Dylla, F., Ferrein, A., Lakemeyer, G.: Acting and Deliberating Using Golog in Robotic Soccer - A Hybrid Approach. In: 3rd International Cognitive Robotics Workshop (CogRob 2002), AAAI Press (2002)
3. Ferrein, A., Fritz, C., Lakemeyer, G.: On-line decision-theoretic golog for unpredictable domains. In: Proc. of 27th German Conference on AI. (2004)
4. Levesque, H.J., Reiter, R., Lesperance, Y., Lin, F., Scherl, R.B.: GOLOG: A logic programming language for dynamic domains. Journal of Logic Programming **31** (1997) 59–83
5. Dylla, F., Ferrein, A., Lakemeyer, G., Murray, J., Obst, O., Röfer, T., Stolzenburg, F., Visser, U., Wagner, T.: Towards a league-independent qualitative soccer theory for robocup. In: RoboCup 2004: Robot World Cup VIII, Springer (2005)
6. Schiffer, S., Ferrein, A.: A qualitative world model for soccer agents. submitted (2006)
7. Konur, S., Ferrein, A., Lakemeyer, G.: Learning decision trees for action selection in soccer agents. In: Proc. 18th BNAIC. (2004)
8. Riley, P.: User's Guide and Reference Manual for the Agent Library provided as part of SPADES. (2003)
9. Quinlan, J.: C4.5 Programs for Machine Learning. Morgan Kauffmann (1993)
10. Ferrein, A., Fritz, C., Lakemeyer, G.: Using golog for deliberation and team coordination in robotic soccer. KI (2005)
11. Kalman, R.: A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering (1960) 35–45
12. Schiffer, S.: ReadyWorld – A Qualitative World Model for Autonomous Soccer Agents in the ReadyLog Framework. Diploma Thesis, Knowledge-Based Systems Group, RWTH Aachen, Germany (2005)
13. Gerkey, B., Matari, M.: On role allocation in RoboCup. In: RoboCup 2003: Robot Soccer World Cup VII. Volume 3020 of LNCS. Springer-Verlag (2004)