Team Description
Mainz Rolling Brains 2006

Robert Deußer, Christoph Schneider, Peter Sinzig, Sascha Metz

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

1 Introduction

Last year we introduced a whole new decision layer design [1] based on general principles for coordinated multiagent decision making. The new design worked out well concerning the coordinated decision making. But since we used the world model and basic skills of our 2004 agent [2] the overall performance of last years MRB agent was not as good as we hoped for.

This year we will introduce a new approach to extract more reliable data from the sensed information. This approach is based on applying a Kalman-filter and is described in section 4. We plan to improve our agents basic skills by adjusting them to the more reliable information provided by the Kalman-filter. Furthermore we will improve our agents positioning behaviour to better exploit the possibilities supplied by our decision layer design. We hope to incorporate the new offside rules of the 3D-Soccerserver into the positioning behaviour. Soccer concepts like an offside-trap fit nicely into our coordinated decision making process and we will experiment with them to see if we are able to successfully apply them to the 3D-Soccerserver domain.

In the following section we give an overview on our decision layer design. In section 3 we will present a short overview of the overall agent design and give some hints on how the general approach is realised in the decision layer. Thereafter we describe the new Kalman-filter we use. We conclude with section 5 where we sum up the current status.

2 Coordinated Multiagent Decision Making

Our approach on coordinated multiagent decision making is based on our last years agent [1]. Even though explicit communication is possible with the new 3D-Soccerserver, we do not use communication to achieve a coordinated team behaviour. The basic idea of our approach is still to choose some *team action* based on a clearly structured situation evaluation first and to deduce the appropriate *individual actions* after that. In the following paragraphs we will first describe the different types of actions and situations we use. Then we will show how we arrive at the individual action to be executed by the agent.

2.1 Actions

In order to achieve a coordinated team behaviour, we introduce *team actions* a^T which are chosen from a small set of predefined team actions A^T. Each team action a^T consists of eleven *individual actions* $a^P \in A^P$, i.e. one for each player. For example, the team action could be "pass from player 5 to player 7". It consists of the following individual actions:

- player 5: kick the ball towards player 7
- player 7: prepare to receive the ball from player 5
- every other player: choose an appropriate position

The individual actions contain generic instructions on how to achieve the goal associated with the action. If necessary they store information on the action like involved players or the progress of the action. The individual actions use the agent's behavioural *skills* to interact with the simulation. Following a least commitment strategy the *skills* use the most recent information available to the agent. This means that a *team action* and its associated *individual actions* can be executed successfully, even if the state of the simulation has changed with respect to the state it had when the *team action* was chosen. But if the state of the simulation has changed to a degree, that a different *team action* becomes sufficiently more promising than the currently selected, this *team action* is selected.

2.2 Situations

Ideally at each time step all players in the team should participate in one single team action and follow the inferred individual actions to accomplish this team action. To increase the probability that all agents choose the same team action, we use a clearly structured and systematic decision process. Therefore we differentiate between three different kinds of situation descriptions:

- The *world situation* s^W is a complete state description of the full world (estimated by the world model), i.e. a large vector consisting of positions and velocities of all players and the ball.
- The *tactical situation* $s^T \in S^T$ describes the situation the whole team is in on an abstract level. Examples are: "ball possession and ball in front of the opponent's goal" or "not in ball possession and ball somewhere in central midfield". Thus tactical situations are from a finite (and very small) set. A hand-coded decision tree classifies world situations into tactical situations (see section 3).
- The *succession state* s^S is a vector consisting of N_o objectives which describe the outcome of a team action on an abstract level. Examples for such objectives are "chance to score a goal" and "safety of ball possesion". Every team action has a corresponding succession state to describe what could be achieved by applying the team action.

2.3 Action Selection

The heart of our action selection algorithm is a function that calculates the utility value $u\left(a^T\right)$ of a team action a^T as

$$u\left(a^T\right) = p\left(a^T, s^W\right) \sum_{i=1}^{N_o} w_{i,s^T} s_i^S\left(a^T\right) \tag{1}$$

where $p(a^T, s^W)$ is the probability of successfully applying a^T in the current world situation s^W. The weights w_{i,s^T} depend on the tactical situation s^T and the specific objective. They describe the importance of the specific objective in the current tactical situation. For example, if the ball is near the opponent's goal, it may not be important to draw nearer to the goal, but to shoot towards the goal. The degree to which the team action will accomplish a specific objective $s_i^S\left(a^T\right)$ is derived from the succession state of the team action.

To select an action we first determine the tactical situation s^T and the applicable team actions a^T with their corresponding succession state $s^S\left(a^T\right)$. These are evaluated using equation (1) and we choose the team action a^T with the highest utility. Finally we deduce the individual action a^P from the chosen team action. A more detailed view on our decision layer design can be found in [3].

3 Architecture

Our agent is split into three layers. The basic layer, which handles the communication with the server, the transformation layer, which acts as an abstraction between the basic layer and the third layer, the exchangeable decision layer (see Figure 1).

As the basic simulation engine is based on SPADES, we use the SPADES agent library from P. Riley [4] to handle the communication with the server. We use the same communication scheme which was used by our 2D agent, i.e. the agent waits for a message from the server and updates its world model according to it. Then the player selects an action based on the new information, sends it to the server and starts waiting for a new message from the simulator. As we are still using the same scheme as in 2D it will be easy to adapt our agent to the proposed new timing model for this years competition.

Our world model, which is part of the second layer, is still quite simple. It stores information about all world objects the agent has seen. Therefore it is updated every time the player gets new sensor information. Using this information the world model calculates the agent's position with a "weighted flag" algorithm [5]. To reduce the error on the position, the velocity of the player and its old position is also considered. The world model also tries to calculate the velocity of the ball by simulating it forward from the last time it was kicked using a movement model improved by a Kalman-filter. (c.f. section 4)

Another part of the second layer are our basic skills. These are an abstraction and extension of the commands provided by the server, as they contain routines for driving to a position or kicking the ball into a specified direction.

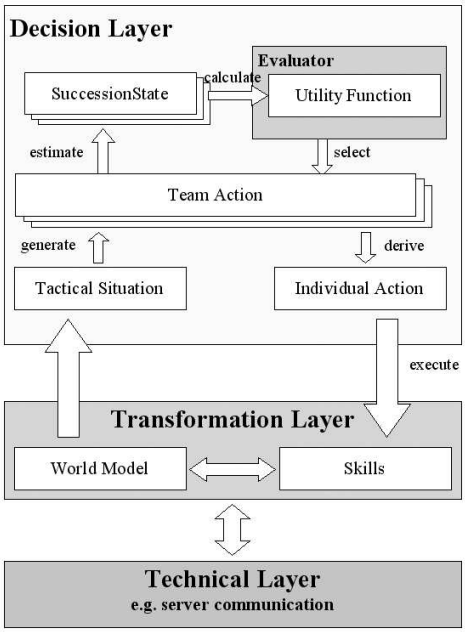

Fig. 1. Architecture of the MRB agent.

In the decision layer the concept of coordinated multiagent decision making as described in section 2 is realized (see Figure 1). We determine the tactical situation based on the information of the world model. Depending on this tactical situation possible team actions are generated. The *Evaluator* rates them and chooses the team action to be performed. The agent's individual action is then derived and carried out using the agent's basic skills.

4 Applying a Kalman-filter to approximate velocities

According to the paper of R. Kalman [6], we implemented a Kalman filtering algorithm to get a better approximation of noisy values sent by the server.

In the simulation environment an agent receives noisy coordinates of every object which is in its vision range. Therefore it is hard to calculate the velocity of an object because the position seems to change even if the object is not moving.

In [7] we give an implementation of a Kalman-filter to approximate the ball velocity. If ball movement changes due to an agent or a game situation like a kick in, the internal state x_k of the ball is initialized with the actual position. A ball movement model M uses x_k to calculate a prediction $\hat{x}_{k+1}$ of the following state x_{k+1}.

$$\hat{x}_{k+1} = M \cdot x_k + u_k, \tag{2}$$

where u_k represents external factors like gravity which influence the movement of the ball.

Finally our algorithm compares $\hat{x}_{k+1}$ with the sensed state x_{k+1} of the actual ball position. The difference is then used to create an error model for further predictions.

Using this technique our ball movement model gets more accurate. Our analysis shows, that this approach improves the process of ball velocity estimation. After a few steps the approximated ball velocity converges towards the real ball velocity.

5 Conclusion

We finished implementing the general approach for coordinated multiagent decision making introduced last year. We presented Kalman-filters as a way to improve the accuracy of the ball velocity. This year we will use this technique to improve the knowledge of the positions in the simulation environment and adapt the team to the new possibilities like communication and offside in the simulation environment.

References

1. Flentge, F., Schneider, C., Jung, T., Metz, S., Deußer, R.: Mainz Rolling Brains 2005. In: RoboCup 2005 Symposium papers and team description papers (CD-ROM). (2005)
2. Arnold, A., Flentge, F., Schneider, C.: Mainz Rolling Brains 2004 - 3D. In: RoboCup 2004 Symposium papers and team description papers (CD-ROM). (2004)
3. Deußer, R.: Flexible Agentenarchitektur zur Unterstützung kooperativer Entscheidungsfindung. B.sc. thesis, Johannes Gutenberg-Universität, Mainz (2005)
4. Riley, P.and Riley, G.: Spades – A distributed agent simulation environment with software-in-the-loop execution. In: Winter Simulation Conference Proceedings. (2003) 817–825
5. Arnold, A., Flentge, F., Schneider, C., Schwandtner, G., Uthmann, T., Wache, M.: Team description. Mainz Rolling Brains 2001. In: RoboCup-01. Robot Soccer World Cup V. (2002)
6. Kalman, Rudolph, E.: A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME–Journal of Basic Engineering **82** (1960) 35–45
7. Sinzig, P.: Weltmodellierung in der RoboCup Soccer Simulation League mit Hilfe von Kalmanfiltern. B.sc. thesis, Johannes Gutenberg-Universität, Mainz (2005)