Team Description RoboLog Koblenz 2D [⋆]

Heni Ben Amor, Joschka Bödecker, Anita Maas, Irene Markelic, Jan Murray,
Oliver Obst, Achim Rettinger, Christoph Ringelstein, and Markus Rollmann

Universität Koblenz-Landau, AI Research Group, D-56070 Koblenz
{amor,jboedeck,amaas,markelic,murray,fruit,achim,cringel,rollmark}@uni-koblenz.de

Abstract. RoboLog Koblenz is participating in the 2D Simulated Soccer competition for the 6th time. In this paper we focus on two of the features we improved from last year to this year. To improve the goal shot behavior of the RoboLog agents we included a scoring policy learned from observing games from RoboCup03 and the SSIL. In addition a smart dribbling algorithm based on *Inverse Steering Behaviors*–an extension to steering behaviors–has been added to the agents.

1 Introduction

Our team RoboLog Koblenz is participating in the 2D Simulated Soccer competition for the 6th time. As in previous years we are using our C++ kernel as interface to the soccer simulator. Our flexible layered architecture allows for realizing the high level behavior of an agent in both Prolog and C++. Consequently, different players of our team are implemented in Prolog or in C++.

The low-level C++ part handles the parsing of messages, maintains a consistent world model, and implements the basic skills of our players. We do not further describe the architecture here, as it has been described in previous papers [5–7]. Instead, in the subsequent sections, we focus on describing two of the features we improved from last year to this year.

2 Learning from Recorded Games

As in previous years goal shots seemed to be a problem for our team, we spent some time investigating how to improve our hand-coded scoring policy. The Optimal Scoring Problem is stated as follows (see [3]):

> Find the point in the goal where the probability of scoring is the highest when the ball is shot to this point in a given situation.

When observed in more detail another side of this problem appears to be essential for finding an optimal scoring policy:

[⋆] This research has been partially supported the special research program DFG-SPP 1125 funded by the German Research Council (DFG).

Given the point to shoot, determine the probability of scoring if the ball is shot to this point in a given situation.

This heuristic is especially interesting for deciding whether to shoot or not. Both problems are correlated to each other. If you can solve the first problem you know which point to test for the second problem. But if you can solve the second problem you can also find a good solution to the first problem by comparing numerous different points and taking the one with the highest probability of scoring. Thus, the second problem appears like an intermediate step to solve the first statement of the Optimal Scoring Problem.

2.1 Related Work

A detailed implementation of the scoring policy used by the UvA Trilearn 2001 team is described in [3]. Data was generated from repeated experiments where a striker was placed somewhere in front of the goal, an opponent goalie was placed somewhere in the goal and the ball was shot to some position in the goal. The outcome of this shot was evaluated statistically and in the end a function is presented that can calculate the probability of scoring if shot to a given point in the goal. Finally the best point to shoot is determined by computing the probability for some discretized scoring points on the goal line and choosing the global maximum of the results.

There are three major differences to our approach. On the one hand the training data is not generated by simulation of situations but already existing data is extracted from pre-recorded soccer games (logfiles). On the other hand far more influencing factors of a goalshot situation are taken into account compared to only regarding one player and one goalie. Furthermore two separate modules are developed to solve both problems stated above independently from each other. Thus, there is no need for discretized testing of shooting points.

In other approaches to learning from opponent behavior (see for instance [2, 9]), learning focused on a single opponent team, either by generating the matches dynamically or by using single logfiles from a given team; instead with using several logfiles from different teams we are aiming at learning a more general scoring policy.

2.2 Goalshot Situation Extraction

To obtain training samples, goalshot situations must be identified in logfiles. It is not enough to find successful scoring attempts because positive and negative training samples are needed for classifying the success rate. The characteristics of a potential goalshot are:

– A striker has kicked the ball.
– The striker is in a reasonable distance to the opponent goal.
– The shot has the potential to reach the opponent goal (reasonable power and direction).

Even if all those conditions apply, further tests need to be made to make sure that it is a valid goalshot and to obtain information about the outcome of this scoring attempt. To determine that, the successive cycles are scanned and checked individually:

- Can the situation be classified as goal, out, goalie catch or offside? Then it is a valid shot and the outcome is known.
- But if the ball was kicked by a player it could also be classified as passing (if kicked by a player from the own team) or dribbling (if kicked by the same striker again) and not as a scoring attempt. Unlike if kicked by an opponent defender or opponent goalie it is interpreted as a valid but unsuccessful goalshot.

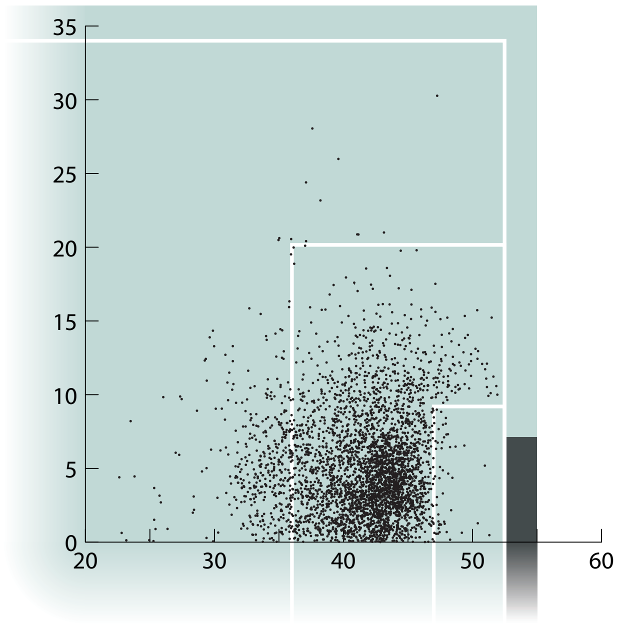

Fig. 1. Position of striker while kicking; successful shots

We have implemented a procedure that extracted automatically goal shots from Logfiles from RoboCup 2003 and from Simulated Soccer Internet League. For details of this procedure, please see [10].

A quick summary on some interesting statistical data from almost 1000 analyzed matches is that we identified 9315 goal shot attempts. Around 40% of these attempts were successful, and 77% of the unsuccessful ones were caught by the goalie. For an overview of the positions from where successful goal shots were attempted see Fig. 1, for the positions where successful shots crossed the goal line see Fig. 2.

2.3 Learning from Successful Examples

For the learning task, we used two basic 3-layered backpropagation neural networks. The first network is needed for the prediction of the point to shoot that maximizes the likelihood of scoring given a specific situation. The second network should be able to classify the success rate of scoring given a specific situation and the point to shoot. Diverse issues needed to be addressed concerning the pragmatics of neural learning.

After tweaking the various parameters involved in neural network learning, the prediction accuracy of both networks on the evaluation set (a third data set) showed promising results.

The *best-scoring-point network* gave a mean average error of 1.4 units which is reasonable if considered that the goal is more than 14 units wide. The *success-rate network* classified 85.4% of the goal/no-goal patterns correctly. Calculating the contribution factor for each input variable is another way to get information about the network performance[1]. Shortly summerized, we identified the angle between ball and goalie as the most important factor, followed by the distance to the goalie. For defenders the distance is more important than the angle and the closest defender has the biggest influence.

2.4 Experimental Results

For the purpose of finetuning and evaluating the performance 400 test matches were played. To have a direct comparison this was done by letting the conventional RoboLog team play against a modified RoboLog team with the new neural modules. Again, the goalshot extraction heuristic mentioned above was used to automatically evaluate all games played. Thus, the ratio of scored goals to scoring attempts could be calculated and the exact performance was determined.

After some finetuning, we compared both approaches in 60 final matches. As Table 1 shows, the ratio of successful shots to scoring attempts is significantly better for the team with the neural networks. In the end the new goalshot module could clearly outperform the conventional module by winning twice as often.

Another proof for the potentials of this approach is the performance of the RoboLog team at the RoboCup German Open 2004 where the new module was used for the first time in a competition. The bad results are due to the general team play. After having a closer look it becomes obvious that there were hardly any chances to score for the RoboLog team because the RoboLog strikers rarely got close to the goal. So once more

[1] Contribution factors are a rough measure of the importance of a variable in predicting the network's output, relative to the other input variables in the same network.

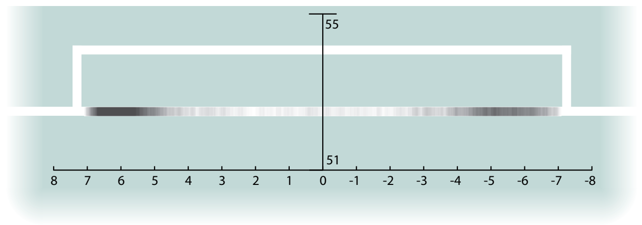

Fig. 2. Goal line crossings: dark areas denote more crossings

Table 1. Test results for the final 60 matches

	average final	
	conventional	ANN
games total	60	
games won	11	21
ratio won/total	0.183	0.35
shots total	75	83
shots goals	23	33
ratio goals/shots	0.307	0.398

the ratio of goals to scoring attempts is significant and it turns out to be exactly 50%. For the first time a RoboLog agent managed to score against the Brainstormers04 team. Brainstormers04 ranked third in the end and conceded only two more goals in the whole competition.

3 Dribbling with Inverse Steering Behaviors

Besides our goal shot policy, we also improved our dribbling behavior. To implement a new dribbling behavior, we made use of steering behaviors. *Steering* is the reactive, non-deliberative movement of physical agents [8]. A basic ingredient to steering are *steering behaviors*, reactive procedures that take local information about the environment as input, producing a steering vector (or steering goal) as output. A set of basic steering behaviors presented in [12] includes *seek, flee, pursuit, evasion, containment*, and *obstacle avoidance* for static obstacles.

To produce complex behaviors, e.g. flocking or queuing at a doorway, several steering behaviors can be combined with each other at a time. Each of the basic steering behaviors is executed separately or in parallel, and the different steering goals have to be combined to one result. Reynolds proposes different ways of "blending" steering behaviors, e.g. using their weighted average or choosing only one behavior at a time according to given priorities (see [11, 12]).

The combination process of these behaviors poses several difficulties, as each of them makes its decision independently of the others. This can result in conflicting commands, and depending on the arbitration method single behaviors can possibly cancel the effect of each other out or result in suboptimal paths. The arbitration method of building a weighted average of steering vectors shares its greatest disadvantages with potential field methods when used for robot navigation [4]:

- Trap situations due to local minima (cyclic behavior).
- No passage between closely spaced obstacles.
- Oscillations in narrow passages or in the presence of obstacles.

3.1 Inverse Steering Behaviors

To overcome the drawbacks mentioned in the last section, we extended the concept of steering behaviors. In the original approach, steering behaviors only take facts about the environment as input and produce a single steering vector as output. Our approach additionally takes a number of different steering vectors as input denoting possible solutions for the steering task. Based on a given criterion, each direction produces a certain cost. In turn, from the calculated cost we produce a rating for each steering vector as output. In principle we inverted the process of each single steering behavior, which is why we call our approach *Inverse Steering Behaviors*. To execute a single behavior, the steering vector producing the smallest cost is selected.

To achieve complex tasks, several Inverse Steering Behaviors can be combined similar to the original approach. In our case the problem is not to combine different steering vectors or actions, but to merge the ratings for each given direction. All relevant behaviors for a task have to be executed, producing a rating for each given steering vector. Merging the ratings is achieved with a heuristics combining ratings of all involved behaviors for each direction separately. The result of applying the heuristics for each given steering vector is a list of ratings like the one produced by a single Inverse Steering Behaviors. Like in the case of single steering behaviors the “best” action can be chosen by simply selecting the steering direction minimizing the cost.

The heuristics employed in the process of combining several behaviors is dependent on the number of behaviors involved and specific to the complex task that has to be achieved. For a detailed description on how we used Inverse Steering Behaviors for dribbling while avoiding obstacles, see [1].

With an appropriate heuristics, merging the ratings for each single behavior can produce better results than the original approach simply adding up steering vectors or selecting one. In cases where two behaviors have conflicting desires, Inverse Steering Behaviors will select a steering vector obeying each of the behaviors to a degree, and thus make a reasonable compromise for all involved behaviors.

4 Ongoing and Future Work

Although we are focusing on our 3D team, there is still work on our 2D team, but only regarding singular research aspects and not on the team behavior as a whole.

For future work we are planning to investigate on using machine learning techniques to simplify building heuristics and to combine our hand crafted behaviors with machine learned ones.

References

1. Heni Ben Amor, Oliver Obst, and Jan Murray. Fast, neat and under control: Inverse steering behaviors for physical autonomous agents. Fachberichte Informatik 12–2003, Universität Koblenz-Landau, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 2003.
2. Sebastian Buck and Martin Riedmiller. Learning situation dependent success rates of actions in a robocup scenario. In R. Mizoguchi and J. Slaney, editors, *PRICAI 2000 Topics in Artificial Intelligence*, number 1886 in Lecture Notes in Artificial Intelligence, page 809. Springer Verlag, 2000.

3. Jelle Kok, Remco de Boer, and Nikos Vlassis. Towards an optimal scoring policy for simulated soccer agents. In C. Torras M. Gini, W. Shen and H. Yuasa, editors, *Proceedings of the 7th International Conference on Intelligent Autonomous Systems*, pages 195–198. IOS Press, California, March 2002.
4. Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, pages 1398–1404, 1991.
5. Jan Murray, Oliver Obst, and Frieder Stolzenburg. Towards a logical approach for soccer agents engineering. In Tucker Balch, Peter Stone, and Gerhard Kraetzschmar, editors, *Proceedings of the 4th International Workshop on RoboCup*, pages 90–99, Melbourne, 2000.
6. Jan Murray, Oliver Obst, and Frieder Stolzenburg. RoboLog Koblenz 2001. In Andreas Birk, Silvia Coradeschi, and Satoshi Tadokoro, editors, *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Artificial Intelligence*, pages 526–530. Springer, Berlin, Heidelberg, New York, 2002. Team description.
7. Jan Murray, Oliver Obst, and Frieder Stolzenburg. RoboLog Koblenz 2002 – short team description. In Gal A. Kaminka, Pedro U. Lima, and Raul Rojas, editors, *RoboCup 2002: Robot Soccer World Cup VI*, Berlin, Heidelberg, New York, 2003. Springer.
8. Alexander Nareyek, Nick Porcino, and Mark Kolenski. AI interface standards: The road ahead. A Roundtable Discussion of the 2003 Game Developers Conference, March 2003. `http://www.ai-center.com/events/gdc-2003-roundtable/`.
9. Tayler Raines, Milind Tambe, and Stacy Marsella. Towards automated team analysis: A machine learning approach. In *Third international RoboCup competitions and workshop*, 1999.
10. Achim Rettinger. Learning a scoring policy for simulated soccer agents. Studienarbeit, Fachbereich Informatik, Universität Koblenz-Landau, May 2004.
11. Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.
12. Craig W. Reynolds. Steering behaviors for autonomous characters, 1999.