MarliK 2012 Soccer 2D Simulation Team Description Paper

Amir Tavafi[1], Nima Nozari [1], Reza Vatani[1], Mani Rad Yousefi[1], Sepideh Rahmatinia[1] and Pouyan Pirdir[1]

[1]University Of Guilan – UGM-LaB, Rasht, Iran
MarliK2D@gmail.com

Abstract. MarliK is a 2D soccer simulation team which has been participating in RoboCup competitions since 2008. In this paper we briefly describe our latest developments which are mainly focused on optimizing our algorithms and making our dribble skills more dynamic. Also although still in the preliminary stages, we are looking forward to develop some proper level of learning in our team which will be used to improve the recently optimized dribble algorithm and also our existing block algorithm in next stages in order to improve the goal of the more realistic simulated soccer agents.

Keywords: 2D Soccer Simulation, dribble, HELIOS Base, multi-agent systems, RoboCup

1 Introduction

We first started our work in the soccer simulation field back in 2006. Our first team, MarliK, was based on the UvA Trilearn source code and we won many national competitions with it. Later on we started another team which was based on Mersad 2005 released source code, completing many incomplete modules implemented in last stages of the Mersad project and adding new ones to it. This new team's name was *LEAKIN'DROPS* and with it we started participating in RoboCup competitions. *LEAKIN'DROPS* was present in RC 2008 at Suzhou and RC 2009 at Graz competitions. Afterwards we changed our base code and began working on the Helios base code, also changing our name to MarliK once again in the process [1].

Our works in the past was mostly focused on positioning systems for offensive and defensive situations. This year we tried optimizing our code a bit and making our algorithms more flexible and dynamic in order to adapt to any possible situation that might arise during the matches.

In this paper, we'll explain some of the algorithms that we are working on, and the dribble direction finding skill we developed in our team up until now.

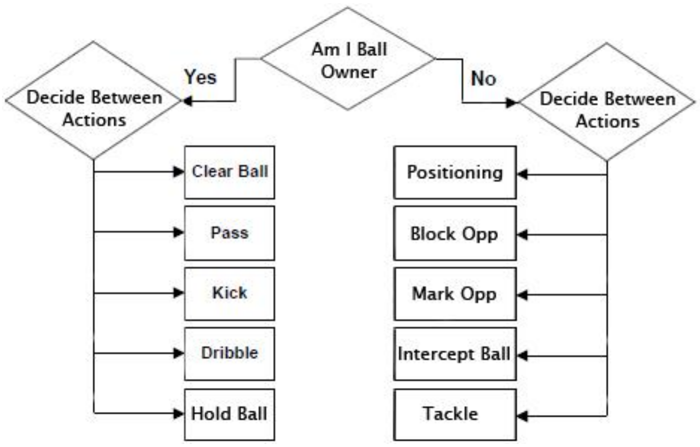

Fig. 1. A simple model of the decision tree we are using for our agents.

2 The Optimized Dynamic Path Finding Algorithm for Ball Owner

This algorithm is planned for the ball owner to find an optimum path to dribble with ball and create a better situation to pass the ball or create a dangerous situation in front of opponent's goal to score a goal. In this algorithm, the agent is making a decision based on the current situation of the environment without any further knowledge about its last experiences [2].

In every different part of the field the agent must have a default dribble target which is not affected by opponent agents yet and then the algorithm will change the path in order to achieve the best effective movement. A simple default target for each part of the field is shown in Fig. 2.

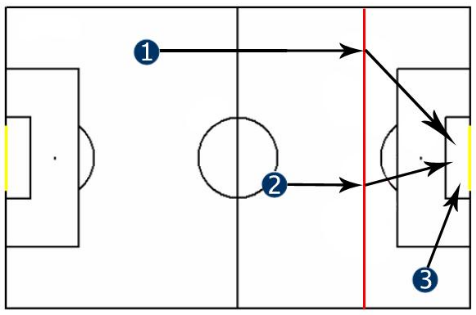

Fig. 2. Default dribble path for an agent in the soccer field.

If the ball owner is behind the red line, dribble target will be set to a point in front of the agent toward the X dimension only. But ahead of the red line, when the agent is near the opponent's goal, dribble target will be set to the center of the goal line in order to carry the ball to a position that a shoot may be viable. In the dribble skill, the ball owner agent performs a dribble to a particular direction which is calculated by the algorithm.

After setting the default target, a comparative factor *d* is defined. This factor is the main part of the algorithm which affects the final direction of dribbling and is associated with the distance of the opponent to the ball owner agent. It is calculated with a comparative ratio. This *d* should be converted from distance to an angle with a high sensitivity for small distances and low sensitivity for long distances so the agent can have a good reflex while the opponent moves near him and to have less consideration while the opponent moves in far distances e.g. when opponent's distance from agent changes from 4 meters to 2 meters, it should affect more on the dribble direction than a change from 15 meters to 13 meters. The inverse tangent function is appropriate for this situation. As shown is Fig. 3, the inverse tangent function has more sensitivity for low values and low sensitivity for high values in both negative and positive values [2].

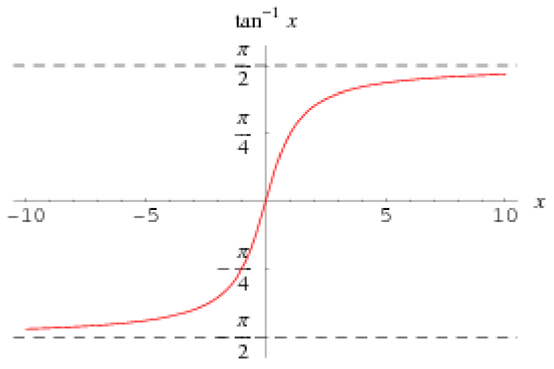

Fig. 3. The graph of inverse tangent function.

After *d* is converted to the desired angle with the suitable function, the range of *d* should be changed to match with all possible movements for agent.

The agent should do this process for every single opponent to find one effective direction for each opponent and in the end, add the average of these calculated directions to the default dribble direction which was created from the defined dribble targets. The algorithm for this process is shown below, in Table 1.

Table 1. The optimized path finding algorithm.

```
Initialize the environment's current situation.
Define a modifier with the value 0.
  Repeat for every opponent:
    Declare difference with value of the difference of angle between these two
    vectors:
        - current opponent's position to ball
        - agent's dribble target to ball
    Define $d$, the distance between the opponent and the ball owner.
    Normalize the value of $d$.
    Convert the type of $d$ with inverse tangent function.
    Normalize the range of $d$.
    Update the value of modifier according to:
                  $modifier \leftarrow modifier + difference*d$.
  Until next seen opponent is not checked.
Update the final dribble direction according to:
  final direction ← default direction + (modifier / number of checked opponents)
```

3 Conclusions and Future Work

In this paper, we have briefly described our latest works in the field of multi-agent systems and artificial intelligence. Despite our efforts, we are yet to develop a proper learning system for our agents. We hope to achieve this goal in the near future and manage to develop techniques allowing our agents to learn and adapt to any situation like humans do.

References

1. Tavafi, A., Nozari, N., Vatani, R., Rad Yousefi, M., Rahmatinia, S., Piredeyr, P.: MarliK 2011 Team Description Paper. In: RoboCup 2011 Symposium and Competitions, Turkey (2011)
2. Tavafi, A., Majidi, N., Shaghelani, M., Seyed Danesh, A.: Optimization for Agent Path Finding In Soccer 2D Simulation, In proceeding of Second International Conference on Advances in Information Technology and Mobile Communication – AIM 2012 (2012)
3. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
4. de Boer, R., Kok, J.: The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team. Master's thesis, University of Amsterdam, The Netherlands (2002)
5. Marian, S., Luca, D., Sarac, B., Cotarlea, O.: Oxsy 2011 Team Description Paper. In: RoboCup 2011 Symposium and Competitions, Turkey (2011)
6. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Upper Saddle River, NJ (1995)

7. Akiyama, H., Noda, I.: Multi-agent positioning mechanism in the dynamic environment. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F., eds.: RoboCup 2007: Robot Soccer World Cup XI, Lecture Notes in Artificial Intelligence. Volume5001., Springer (2008) 377-C384.
8. Ko, J., Klein, D. J., Fox, D., & Hähnel, D. (2007b). Gaussian processes and reinforcement learning for identification and control of an autonomous blimp IEEE international conference on robotics & automation (ICRA).
9. Kitano, H., Minoro, A., Kuniyoshi, Y., Noda, I., Osawa, E.: Robocup: A challenge problem for ai. AI Magazine 18(1), 73–85 (1997)