Team Description of 8MHM8 Soccer Simulation 3D

Hamide Vosoughpour, Fereshteh Abbasi,
Samira Abnar, and Mahsa YousefZadeh

Farzanegan High School, NODET
Tehran, Iran
{Hamide, Fereshteh.Abbasi,
Samira.Abnar, Mahsa.YousefZadeh}@gmail.com

Abstract. 8MHM8 is a team designed to compete in soccer simulation 3D field. Since the aim of Robocups genesis is not only to play a good football but taking a step in Artificial Intelligence, in designing 8MHM8 we tried to notice both of these subjects. We try to present a good football by writing strong basic skills and have clever agents by using learning algorithms.

1 Introduction

The aim of designing 8MHM8 is to have a football team which plays almost like a real team. We think we can achieve this aim by paying attention to two topics:

1. Having agents which can learn in addition to thinking.
2. Soccer Simulation 3d details.

Each of the topics above will be explained through out the paper. Actually, in the match against Caspian3d, some parts were not completed and our agents didnt have the ability to learn anything. It means there were no learning algorithms used in them. The basic skills too, are much weaker than what we want to have in the Robocup. But we more focused on having good basic skills till now. Its mentionable that because of it was our first experience in writing agents, we spent lots of time to get familiar with this field. We are going to complete both parts, before the competitions start (both local and international ones).

In the following parts youll read about our team architecture, strategies, the details we worked on, and things we are going to do in the future.

2 Team Artichecture

Maybe the most important part of our teams architecture is having different player types. Weve got four classes for our agents: Goalie, Defender, Midfielder and Attacker. This is so useful for us because our agents can act different from each other. For example maybe the limitation for an attacker to kick is having at

least one open angel in opponents goal but for defenders it is another condition. The limitation, here, is if the number of opponent players in a specified area around him was more than 3 (for example), it can do a strong kick to send the ball away. This kind of architecture helps us to have: 1.Less switch commands. 2. A more similar team to a real human team. Inside the act functions, it checks if the agent has got the ball or not. If yes, it will do actWithBall().Otherwise it will do the positioning. Positioning of our agents is really mixed with math rules. This helps it to have more exact positioning. Positioning functions, too, are different for each play mode and each player type. So weve got many functions for only positioning.

Decreasing the thinking time was the thing we tried to notice through out all our codes. Because it can affect the accuracy of things such as ball velocity which agents must measure by themselves with taking the ball position in different steps. So, if thinking takes us a lot, it may be too late to use this information.

3 Team Sterategies and Positioning

The formation we chose for 8MHM8 was 2-3-2-3. Its an innovative formation we chose because of its suitability with our strategies and its potentiality for having an offensive game. The main part of our strategy is the amount an agent should move in different team states such as attacking, defending or nothing. This amount is suitable with ball position and the agents distance from our goal. This helps the agents to be distributed well in the field (Fig.1.)

After the strategy is executed, the main positioning will work. It checks if the agent should block a pass, wait for a pass or etc. Then it chooses where to stand. Its mentionable that the agent cant move outside a specific area (because of the strategy).

When we are defending, the strategy works in a way similar to Fig.1.(c) which describes the position of players when attacking. The difference is in the place that this happens. While attacking, its near the opponents goal but when defending its near the our goal.

Theres also a function which chooses the team state. The team state will be specified due to balls position and the accumulation of team mates and opponents near the two goals.

4 Details We Worked On

The details usually exist in basic skills of the agent. For example when we want to go to a point, how we should get there, is something which has got a great effect on reaching the point faster or slower, because the way to a point is never empty of preventives. We have not yet worked on this completely. Our agent does reach the target by moving on line, drawn directly from its centre to the targets centre. If there was a preventive in its way, itll turn around it. Sure we are going to fix this.

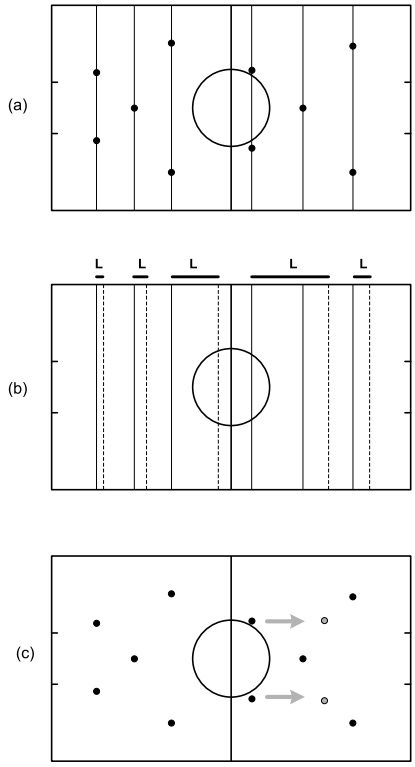

Fig. 1. (a) Default positions. (b) Attacking positions: The x coordinates of agents Changes suitably with the distance of the agent from our goal. This is not observed for agents 9, 10 and 11, because it helps the attacking strategy. L is the amount each agent moves. (c) Attacking positioning.

The main detail which we think can affect the game too much is reaching a moving object in the shortest possible time. For example ball. Actually we can write an algorithm which follows the ball every moment, but this way the agents way to the ball will be a curve line which is not the shortest way. For finding the shortest way to ball, we found the following formula based on math and physics rules:

$$\sin a = \frac{h}{\frac{1}{2}a \times \Delta t^2 - \frac{1}{2}a' \times \Delta t^2 - \Delta t \times \sqrt{2a' \times \frac{h}{a} + 4a'V_b \times \frac{h}{a} + (V_b + a' \times \sqrt{2 \times \frac{h}{a}})^2}}$$

in which,

a is the maximum acceleration of the agent,

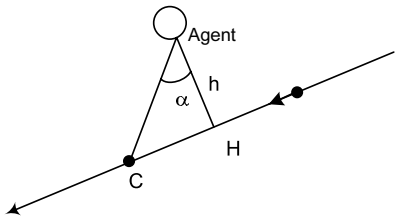

Fig. 2. Intercept

a' is the ball acceleration,
h is the distance of the agent and the ball motion line,
Δt is the time interval between events: The agent reaching H and the ball reaching H (Which may not happen), and
V_b is the velocity of ball.

So C will be the point we should move to, for catching the ball in the shortest time.

Another point we are now working on is how to reach the ball while facing the direction we are going to use the ball in, when we reach it. Weve seen some of the games of last year teams and we saw that sometimes agents lose the ball because they want to (for example) kick it in a specified direction so they have to avoid the ball and turn around it. While doing this, the opponents get the ball. We do not want this to happen for our agents.

Another thing we see it necessary to mention is that we used many different math and physics rules in different things our agents need to do. First we didnt want to do that because we thought a human never does this during a game but we thought it may help us in decreasing inexactnesses.

5 Conclusion and Future Works

Our team is not a perfect team now. As it was said before we are going to fix some parts of the basic skills and give our agents the ability to learn.

For learning we are going to use machine learning algorithms and parallel thinking methods. We have separated our agent intelligence to three parts: Prediction, Learning, and Choosing the best action to do.

1. Prediction: It is based on 4 concepts. The first one is physical rules. By using physical rules the agent can predict the world state in next steps. So it can decide what to do, according to what will happen in the future. The second is past experiences. Our agent works according to some special rules (e.g. physical rules). But they might not work because physical rules usually do not exactly tell us what is happening. If our agent percept such a thing, it

will make changes on the rules in a way to make them work properly. The third one is guessing. This is usually used about the agent who has got the ball. So our agent tries to decide what it would do, if it had the ball. This way, it can guess what the ball owner will do.

2. Learning: Of course no being can learn without a memory. So we have decided to put a memory for our agents. An 8MHM8 player has a memory for all the objects and actions it knows.
   (a) Action Memory: Once our agent does an action in a specified time it remembers that. The other time the occasion happens, our agent adds or minuses some points to that action points depending on that actions success or failure. This is exactly what you call rewarding and punishment.
   (b) Other players Memory: This is used when our agents wants to predict other agents' actions. For instance if a player does an action in a special situation more than one times, our agent will give more points to that action when that situation happens.
   (c) Static Objects like ball: We all know that for predicting objects' behaviours like ball, we need to have some valued parameters. But there is no guarantee that they work in the real world the same as we thought. So our agent can easily change them according to its experiences.

   Also our agent can use other players' memory to do their successful actions in some special situations.
3. Choosing the best action to do: Due to the points the agent has given to different actions in the steps before, it'll decide what to do. The most important part of this is that each point, dependent to the reason it was given, has got a factor. This factor shows the importance of that reason. The amount of this factor can change according to the results the agent gets from each action it does. In computer science, it is called supervised learning.