SEU-3D 2006 Soccer Simulation Team Description

Yuan XU, Chunlu JIANG and Yingzi TAN

Southeast University, Nanjing 210096, China
xuyuan.cn@gmail.com, JamAceWatermelon@gmail.com, tanyz@seu.edu.cn

Abstract. This paper shortly describes the main features of the SEU-3D soccer simulation team, which participates in the RoboCup competition since 2005. In the last year we mostly concentrated on the base code and the lower layers skills of agents. After a few months research and development, SEU-3D has made great improvement.

1 Introduction

SEU-3D was started by two students from Southeast University for their BS thesis. 3D soccer simulation is a new and more realistic part of RoboCup simulation league in comparison with 2D soccer simulation. As it's the first experience of our team in 3D league, most of the team's work has been spent on developing a powerful and developmental base code and primary skills.

The SEU-3D simulation soccer team has successfully attended two competitions: ranked 7^{th} in the China RoboCup 2005 held in Changzhou, China; and ranked 4^{th} in AI Games 2005 held in University of Isfahan, Iran.

The reminder of this paper is organized as follows. Section 2 briefly describes our agent architecture. Section 3 illustrates the world model. Section 4 shows the individual skills and strategy. Section 5 enucleates our development tool. Section 6 is the conclusion and future work.

2 Architecture

SEU-3D layered agent architecture[1] consists of three layers including iteration layer,skills layer, and decision making layer and it can be shown as a whole in Figure 1.

Nevertheless, this is not strict layered agent architecture, it has been added new ideas to design SEU-3D's architecture. In this architecture, it is also possible for a low-level behavior to call a more abstract one, and a high-level behavior to call a low-level behavior directly without calling a mid-level behavior, etc. Consequently, most components of agent have only one instance, and to provide a global point of access to it. In other words, a agent must have a single world model, a single skill model and a decision model, etc. Furthermore, all models should call each other easily.

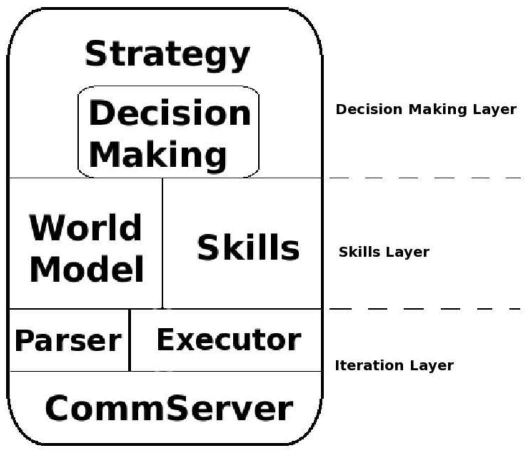

Fig. 1. SEU-3D Architecture. The components which binded together can call and impact each other.

Singleton[2] is widely used design pattern, and suitable for components of agent. Using a global object ensures that the instance is easily accessible but it doesn't keep you from instantiating multiple objects - you can still create a local instance of the same class in addition to the global one. Singleton pattern provides an elegant solution to this problem by making the class itself responsible for managing its sole instance. SEU-3D uses a template singleton basic class[3] to implement this architecture.

3 World Model

In order for an agent to behave intelligently it is important that he keeps a world model that describes the current state of the environment. The agent can then use this world model to reason about the best possible action in a given situation[1].

3.1 Self Localization

The basic function of world model is self-localization, and estimating self velocity. In current rcssserver3d, the agent has a 360 degrees view of the field and no direction, the agents location can be computed using relative position of only one flag. However, in order to minimize the noise, all 8 flags' information are used.

Particle filtering[1] and Kalman filtering[4] are both used in SEU-3D.

Step 1 estimates intermediate position P_1 and velocity V_1 using information of current world model.

Step 2 Particle filtering generates another intermediate position P_2 and velocity V_2 using information of all flags.
Step 3 Kalman filtering generates the preciser position, using P_1, V_1, P_2 and V_2.
Step 4 considering some special states that the agent's position be reset(ie. beaming and flicking).

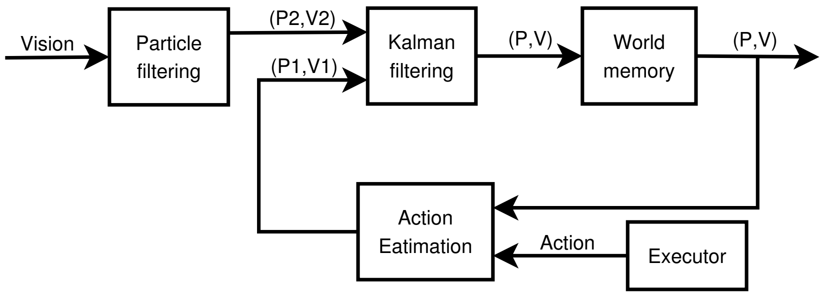

Fig. 2. SEU-3D Self-localization filtering.

The whole process can be seen in Figure 2. Using this method, the average error of position and velocity are both decreased(See Table 1).

Table 1. The error of self-localization and estimating self velocity

Error type	Position(m)	Velocity(m/s)
Average error	0.03	0.10
Max error	0.10	0.50

3.2 Update Ball's Information

Another basic function of world model is calculating the global position and velocity of dynamic objects. In SEU-3D, the ball's information is updated by Kalman filtering[4].

Step 1 estimates intermediate position P_1 and velocity V_1 using information of current world model.
Step 2 calculate the ball's information P_2, V_2 using visual message. Note, here V_2 is equal to the sum of agent's velocity V_a and the relative velocity V_r. Using the distance variance Δr and direction variance $\Delta\phi$ and $\Delta\theta$ which are

included in visual messages to estimate the ball velocity $V_r(v_{rx}, v_{ry}, v_{rz})$, according to the following equations:

$$v_{rx} = (\varDelta r\cos\phi - \varDelta\phi r\sin\phi)\cos\theta - \varDelta\theta r\sin\theta \tag{1}$$

$$v_{ry} = (\varDelta r cos\phi - \varDelta\phi r\sin\phi)\sin\theta - \varDelta\theta r\cos\theta \tag{2}$$

$$v_{rz} = [\varDelta\phi r + (v_x\cos\theta + v_y\sin\theta)\sin\phi]/\cos\phi \tag{3}$$

Step 3 Kalman filtering generates the preciser position, using P_1, V_1, P_2 and V_2.
Step 4 considering some special states.

Using this method, the average error of ball's position and velocity are both decreased(See Table 2).

Table 2. The error of ball position and velocity.Note,here d means the distance between the agent and ball.

Error type	Position(m)	Velocity(m/s)	P(m)($d < 5m$)	V(m/s)($d < 5m$)
Average error	0.05	2.50	0.03	1.50
Max error	0.25	5.00	0.10	2.50

4 Skills and Strategy

4.1 Skills

Currently, Individual skills is the focus of our research , for it is important to ensure the performance of the whole team.

Ball interception is one of the most important available player skills that is frequently used by agent. SEU-3D ueses an analytical method with the help of neural network.

In order to implement this hybrid method, some basic functions are needed:

- a high-performance drive skill, which returns both drive force and drive time
- a precise estimation of the free motional ball's information in the next simulation step
- the judgement of successful intercepting(ie. if the ball is ready to be kicked)
- a well debuged iteration loop

Dribble is an useful skill when agent possessing the ball. In current server, agent max speed is about $1.6m/s$, the simulation step interval time is $0.01s$, and the kickable margin is about $(0.331m, 0.4m)$. The ball and the agent have the same motion direction in dribble mode. That means there is at least 4 simulation

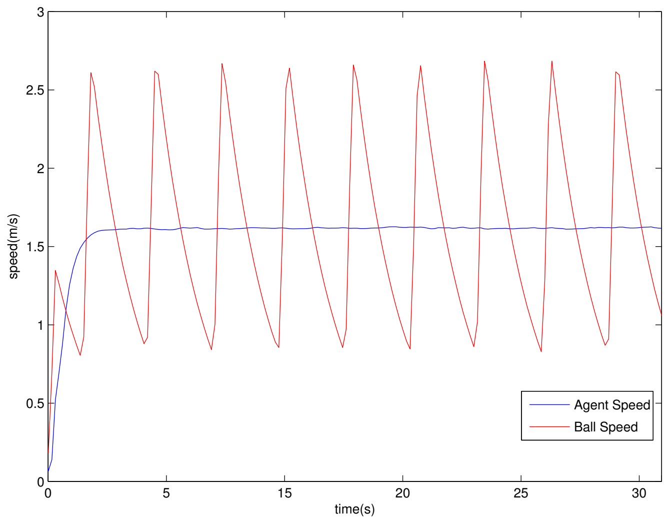

Fig. 3. The agent speed and ball speed while dribbling.

steps before agent collides with the ball when agent is in full speed. So, a full speed agent have enough time to kick the ball before he collides with the ball. SEU-3D has implemented the dribble without deceleration(See Figure 3).

Contemporary, SEU-3D agent also can pass, shoot, block, clear ball, etc. And we note that taking advantage of body is very important, this would be our future work. All kinds of AI algorithm will be imported to enhance SEU-3D's skills.

4.2 Strategy and Decision Making

Currently, we do not have much new ideas about this problem, we are planning to use the MDP decision method. There is a simple model of the decision for our qulifacation team. The method in references like[1] and has been added our own ideas to design this simple strategy in order to fit the 3D enviroment.

The strategy is based on the following principles:

- keeping the distance between ball and our goal as far as possible
- if possible, and in shoot area, shoot as soon as possible
- if possible, possession ball (ie. pass or dribble)

The agent can only accelerate the ball radially away from the body. This means that agent have to move to special position, so that it can kick the ball to the desired direction. But in some cases, opponents don't allow the agent to spend much time. Agent has to kick before opponents break it. The idea, freedom which indicates how much time and space the agent has, has been added. Agents make decision depending on the freedom value. If the freedom value is high, agent will possession ball. Otherwise, agent will clear the ball.

In our implement, freedom is calculate according to the following factors:

- the position relationship of ball, agent and the closest opponent
- the distance between agent and ball
- the distance between the closest opponent and ball

Integrates all the factors, and then normalizes the value by a sigmoid function.

5 Development Tool

Besides using the defined team developing rules and standards, a time saving and high performance utility, named trainer, is used to test server enviroment, train the agents and calculate statistics of matches. MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. We analyse the simulation data both from the trainer and agent, with the help of MATLAB.

The main parts of the utility are:

1. Trainer
   - connect to the rcssserver3d or load a logfile
   - send commands to server to set the training state
   - log every objects' global position and velocity
   - calculate statistics of matches
2. MATLAB
   - m-files which analyse the simulation data
   - use kinds of Toolboxes, for example, training the Neural Network

6 Conclusion and Future Work

In this paper, we described a few aspects of our current team. We addressed the way we implemented the world model, skills, and simple strategy. In addition, the development tools are described.

The first results achieved by SEU-3D team are very hopeful. But there is much work to do in the future.

In our project, the most two important objectives are improving the individual skills and implementing a good decision algorithm and planning. At the same time, we are going to develop the biped simulation robot, which will be used in the furture.

References

1. Remco de Boer, Jelle Kok: The Incremental Development of a Sythetic Mu,The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team. Faculty of Science,University of Amsterdam (2002) 43-56 159-168
2. Danny Kalev: Implementing the Singleton Design Pattern. http://gethelp.devx.com/techtips/cpp_pro/10min/10min0200.asp
3. Scott Bilas: Game Programming Gems. Charles River Media (2001) 36-40
4. Rudolf Emil Kalman: A New Approach to Linear Filtering and Prediction Problems (1960)