ZJUBase 3D 2005 Team Description

Dijun Luo, Hao Jiang, Jun Guan, Yifeng Zhang, Yang Zhou

National Lab of Industrial Control Technology, Zhejiang University, China

djluo@iipc.zju.edu.cn

Abstract The introduction of RoboCup 3D soccer competition in 2004 was a significant step of the simulation league. In this paper, some technology of building our agents is discussed. Our research focused on the filtering of world model and some basic skills such as positioning at the beginning of the development of our team. The decision making architecture is also demonstrated briefly.

1 Introduction

In the last decade, multi agent systems (MAS) have emerged as an active filed of Artificial Intelligence. RoboCup competitions provide an excellent platform on which various ideas of AI algorithm have been implemented. RoboCup 3D soccer competition is introduced in 2004, with the idea of *The System for Parallel Agent Discrete Agent Simulation (SPADES)* [1].

ZJUBase 3D team started in 2004. After 3-month development, we won the champion of RoboCup 3D Soccer Competition 2004 of China. Optimization methods of building our basic player are of the most importance to ensure the performance of the whole team. In building a RoboCup 3D agent, filtering of world model and the positioning skill are the two major problems which are solved very well in our team.

The decision algorithm of our RoboCup 2D soccer team [2] which is based on the UvA Trilearn source code [3] is also implemented in our 3D soccer team.

2 Agent Architecture

Our research focused on the filtering of world model and some basic skills such as positioning at the beginning of the development of our team. Therefore the decision making architecture is simply the same as our 2D team. Figure 1 illustrates the architecture of our agents.

The *perception* and *command executor* module are the interface between the agent and the 3D soccer server. The *world model* module stores the information of everything on the field with which all the decisions are made. *Filter* module are used

to deal with the noise in *world model* module. The *reasoning module* is our main decision-making component.

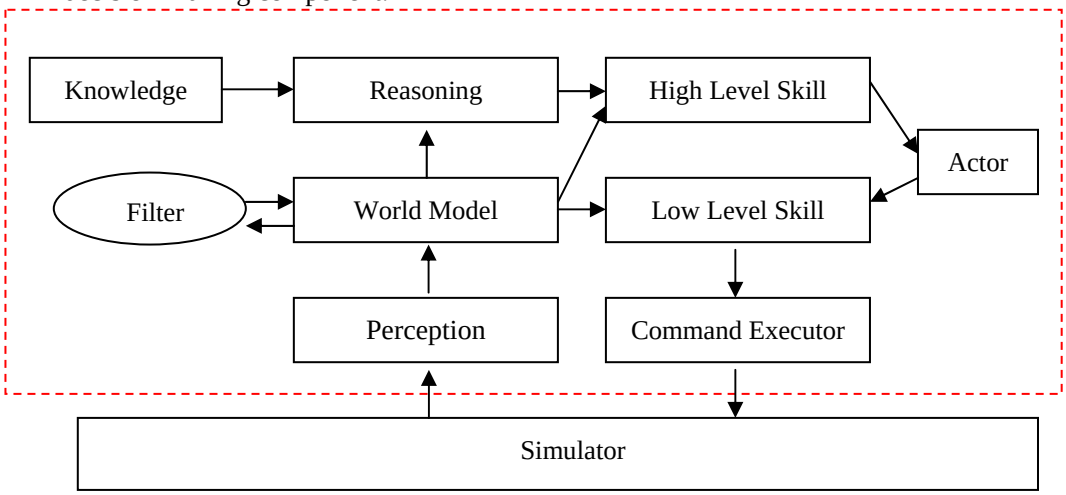

Figure 1 Agent Architecture of ZJUBase 3D

We use LSE method in the *Filter* module and *nonlinear programming method* to build the one of the most important skills, positioning, in the *low level skill* module.

3 Filtering of World Model

The statuses of the ball and the self-agent are two major components of the filtering of world model. We consider both the ball and the self-agent as a linear system:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$
$$z_k = Hx_k + v_k$$

x is status of the bal or agent, and z is the measure value of it.

Due to the independence of the coordinates of x, y and z in world model, let

$$x = \begin{bmatrix} p \\ v \end{bmatrix},\ A = \begin{bmatrix} 1 & 1 \\ 0 & k \end{bmatrix}, B = \begin{bmatrix} 0 \\ \alpha \end{bmatrix}$$

As a result,

$$x_n = A^n x_0 + A^{n-1}Bf_0 + A^{n-2}Bf_1 + \ldots + Bf_{n-1}$$

Let

$$E = (\hat{p}_0 - \tilde{p}_0)^2 + (\hat{p}_1 - \tilde{p}_1)^2 + \ldots + (\hat{p}_n - \tilde{p}_n)^2$$

Or

$$E = (\hat{p}_0 - \tilde{p}_0)^2 + (\hat{p}_0 + \hat{v}_0 - \tilde{p}_1)^2 + \ldots + (\hat{p}_0 + \sum_{i=0}^{n-1} k^i \hat{v}_0 + \alpha \sum_{j=0}^{n-2} \sum_{i=0}^{n-2-j} k^j f_j - \tilde{p}_n)^2$$

Where $\hat{p}_i$ and $\tilde{p}_i$ are the estimation and measured value of p_i.

To reach the minimal value of E, the following is necessary:

$$\frac{\partial E}{\partial \hat{p}_0} = 0 \tag{1}$$

$$\frac{\partial E}{\partial \hat{v}_0} = 0 \tag{2}$$

The solution of (1) and (2) is estimation of the position and velocity of the agent or the ball.

4 Positioning skill

Positioning skill is an essential issue in basic skill in an agent system. We define the positioning skill as an optimization path planning problem: run to any specific position in a shortest time without colliding with any other objects in the filed. The problem can be written as follow,

$$\begin{aligned} &Min.\ \ n \\ &S.T.\ \ p_n = 0,\ v_n = 0 \\ &\quad |\ f_i\ | \le F_i\ (i = 1, 2, ..., n) \\ &\quad |\ p_i - p_{i,ball}\ | > R_{ball} + R_{me}\ (i = 1, 2, ..., n) \end{aligned}$$

And we then convert the problem to an easy-solved one,

$$\begin{aligned} &Min.\ \ f_i \\ &S.T.\ \ p_n = 0,\ v_n = 0 \\ &\quad |\ f_i\ | \le F_i\ (i = 1, 2, ..., n) \\ &\quad |\ p_i - p_{i,ball}\ | > R_{ball} + R_{me}\ (i = 1, 2, ..., n) \end{aligned}$$

This is a nonlinear programming problem which is somewhat different with the typical one. To us, an arbitrary feasible solution is what we need. A solution with less value of object function is meaningless, since the desired number of cycles is determined by enumeration. This important feature helps to deal with the problem above.

To solve the constrained nonlinear programming problem we employ the *penalty function* to convert it to non-constrained one, since we could not determine a feasible solution initially. The algorithm works as follows,

Step 1. Given a sequence $\{ r_k \}$, $r_k \ge r_{k+1} > 0$, $r_k \to 0$, *and an arbitrary point* x^0, *let* $k = 1$.

Step 2. With the initial point x^{k-1}, *solve a non-constraint problem,*

$$\min_{x \in E^n} G(x, r_k) = f(x) + \frac{1}{r_k} P(x)$$

where $f(x)$ *is the object function. Here we assume* x^k *is the solution of* (3.6).

Step 3. Take a judge whether $P(x^k) = 0$ *is true. If so, stop and* x^k *is the solution of the original constrained problem, otherwise update* k *with* $k+1$ and *return to Step 2.*

In our work, we set the sequence $\{r_k\}$ as an exponential sequence that $r_k = \alpha^k$ where α is a real number between (0, 1), and $P(x) = \sum_{i=1}^{m}(-\min\{0, g_i(x)\})$ where $\{g_i(x)\}$ is the set of constraints.

And then we use *conjugated gradient method* to solve this non-constrained optimization problem.

Figure 3.2 shows the result of several cases of path planning.

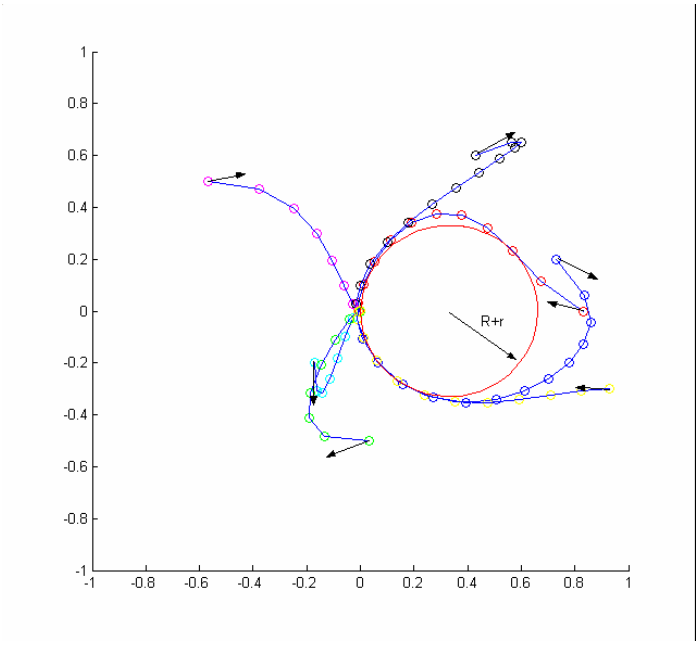

Figure 2 Some cases in path planning. In these cases agent is located in various positions with different velocity and the target is to run to a specific position relative to the ball without colliding with it. Here we assume that the ball is not moving. If the distance of positions in the path and ball is larger than the sum of the radius of ball r and the agent R this requirement is satisfied.

5 Conclusions and Future Direction

Filtering and path planning are two major problems in building a basic player in a RoboCup 3D soccer game. Our players in which the optimal filter and path planning

algorithm are implemented won the champion of RoboCup 3D Soccer Competition 2004 of China. The model based optimization methods are also applicable in some high level decision making. For example we can fetch the statistic characteristics from plenty of games with opponents or an agent can learn a model of the players of an opponent team. Once the mathematical model of a system is determined, a lot of optimization methods can be employed to the system.

Reference

[1] Proceedings of the 2003 Winter Simulation Conference S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

[2]Jun Guan, Dijun Luo and Daming Liang. The Architecture of Team ZJUBase: Implement A Cooperative Team. In Proceedings of Chinese Robot Soccer Competition 2004, Chinese Association of Automation and Institute of Automation, Guang Zhou, China, October 2004.

[3] 1. Remco de Boer, Jelle Kok. The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team[Master Thesis]. Artificial Intelligence and Computer Science, University of Amsterdam. February 2002.