Team Description of CZU2006_2D

Haix_Zhang Quan_Li Changchun_Zhu Yanbin Zhuang Chunguang Li
Institute of Artificial Intelligence and Robotics
School of Computer Information and Engineering
Changzhou Institute of Technology
Changzhou 213002, Jiangsu Province, P. R. China
z_hx25@126.com
leechunguang@sina.com

Abstract: This paper describes the main features of the CZU2005 soccer simulation team (2D), which got the 6th place in RoboCup2005 competition of China. After the brief introduction of CZU2005 soccer simulation team, the characteristics of CZU2006 are described. Now much improvement has been made in update layer, tackle model design based on BP_ANN, dribble behavior, and the assignment of heterogeneous agent by an online coach. Finally, we show our future research directions.

1. Introduction

Our team is base on the released TsinghuAeolus2002 source code [1] [2], with our great efforts, it shows amazing progress, and achieved the 6th in China RoboCup 2005 competition.

After the competition, we went on improving our code. We rewrite some code to make basic_action_layer fitter for the current soccer server, use more efficiency arithmetic in mid_atcion_layer selecting a suitability inner_action, and improve evaluator in decision layer. We also build up communication system, tackle system, and the assignment of heterogeneous agent by an online coach.

2. Improving Update Layer

During the match, there are two states of clock, clock running and clock stop. An Agent can do action such as “run”, “turn” etc when the clock is stopped. In TS02 code, there doesn’t exist the update function when clock is topped. So we build up the update system when clock stopped.

In normal state, the clock is running and time of current cycle is the only index of information. No matter the visual or sound information arrives before or after the body sense information, we can identify the information by body sense information witch include the current time label easily. However in clock stop state the time stay for 32 cycles, but information updating is still running. In other word, the time label of all sensor information is the same, but the data in information has changed as the mobile object moved in the field. In this state the time is no longer the only index of information and the original estimate system is no longer useful. So a data structure which is called “time3d”array is used and a new evaluation system is build up to solve this problem. The “time3d” array contains *three segments* which are *time segment*, *count segment*, and *data segment*. *Time segment* stores the time of current cycle, in fact it is useless. *Count segment*: the counter is touched off by new arriving oneself perception signal, and the number of the counter increases by 1 when

new oneself perception signal is received by the counter. So the value of the counter is unique. *Data segment* stores the data from the information. Because the number of the counter is unique, it can be used as the only index of a new “cycle”. Thus our solving scheme would be simplified. In this paper, we have defined three Boolean variables: *new_sight, new_sound, and new_bodysense* as the new information signs. So, in update function, check the new_bodysense sign before update seen and heard. When new_bodysense is true, but new_sight or new_sound is false, this means the new cycles begin but sight information or sound information haven’t come, sight or sound information is old , so estimate data should be subjoined to the old information which get from the “time3d” array. The estimate data depended on the effect of last action. While sight information or sound information arrives early than the body sense information, it is no use to estimate, because the data is new.

At last we test it for 30000 cycles, in clock stop state all agents can go to right position as well as clock running state, this proved our method is fit for match.

3. Tackle Model Design Based on BP _ ANN [7]

The tackle model [8] is one action model that server offered, and there is a conflict of interests in tackle action, namely, one agent can not move for 10 cycles after he do tackle action. On other hand, once tackle is successful, the ball will be kicked into the direction of the player’s body with highest speed. In this section, we discuss how to use BP neural network to improve the success rate of *Tackle*.

Server manual shows that the tackle success is determined randomly with the probability based on the ball’s position relative to the tackling player using the following formula:

$$fail\_prob = (player\_2\_ball.x / tackle\_dist)^6 + (|player\_2\_ball.y| / 1.25)^6 \quad (1)$$

Where ***tackle_dist*** is 2.0 if the ball is in front of the player and 0.5 otherwise. ***player_2_ball*** is a vector from the player to the ball, relative to the player's body direction.

Of course, we can use this formula to calculate the success rate directly. But in RoboCup such a highly dynamic and adversarial multiagent environment. Success rate not only depends on the parameters about the ball’s relatively position to body, but also the ball’s velocity, and in such a noise environment, it is possible that the ball’s real position is not the position an agent sees. So the actual success rate of (1) is so poor that it can be not able to match stably.

As we know, BP Artificial Neural Network [5] is provided with widely adaptability to a specifically environment. As a carrier of success examples, ANN can simulate all scenes and predict result. Building a BP_ANN needs many examples to cover the status uniformity. We collected enough examples to cover the status.

We use *offline coach* to Move Player and Move Ball, first move an Agent to a proper position, Vector(0.0 , 0.0) is the best , then move ball to a given point with a given velocity, let the Agent do tackle action. If tackle is success, then record the relative position and relative velocity. Take a test 10 times at one point, record the success rate. Thus our example is complete. Fig. 1 shows the creation of ball’s given point.

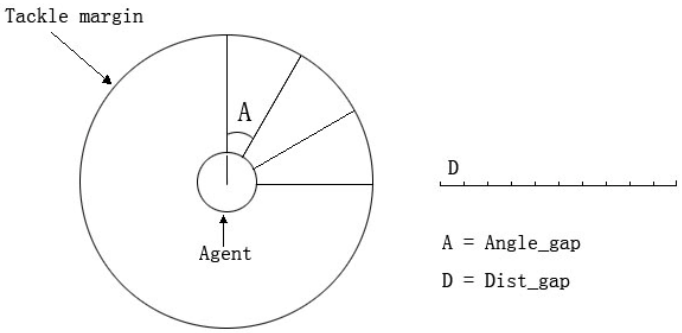

Fig. 1: Creation of ball's point

This test takes us 4 hours to collect 720 examples. We use MATLAB[6] to train the network. First we normalize examples into [0, 1], then choose function "newff" to create a forwardback BP network with 5 layers (4 input layers and 1 output layer) in MATLAB, and third select a proper number of hidden layers, use 16 hidden nodes, and use LM as training algorithm and tansig as *activating function*, Finally begin training, the training result is shown in Fig.2:

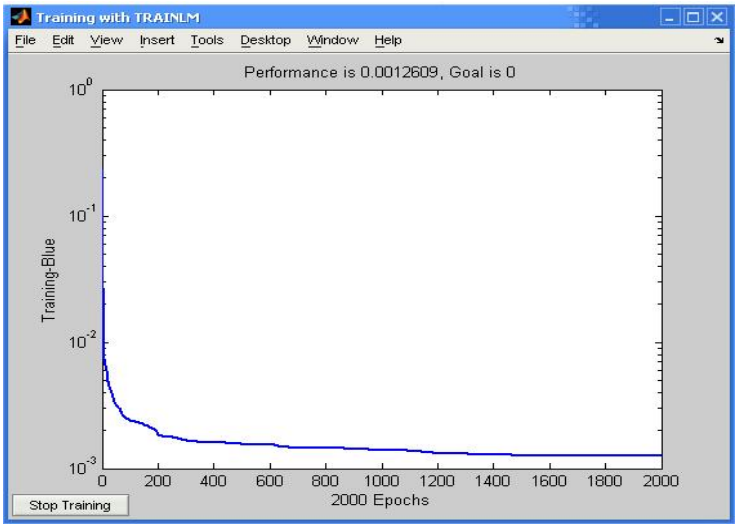

Fig. 2: Matlab train result

We used the released TsinghuAeolus2002 source code, studied his format of net storage.

Both calculation of geometric and BP Artificial Neural Network is used in our code. This improved greatly our tackle success rate, after 20 matches the fail rate was just 0.08, and this rate is acceptable.

4. Local Path Programming in Dribble Action

Dribble path programming [3] [4] is a basic problem. The direction of the opponent's goal will be the macroscopically direction, and it is the best direction while there does not exist an opponent. But RoboCup is an environment which is full of antagonize. Opponent will intercept optimal dribble routes for defense. The local path programming becomes more and more important. Potential Field Method is one of the famous programming algorithms. But to build up an artificial potential field is too complex, and it has a disadvantage that is the local minimum problem. In this section, we will introduce a new

method of local path programming [3] used in our code that is tangent method. This method is simple and easy to carry out, and the most important is without the local minimum problem.

The algorithm is:

First judge if there is s bar between dribble path and self dribble range (kickable_margin), if true then confirm the bar's position, make agent as a circle with his position as the centre and his kickable_margin + self_body_radius as the radius, so it is possible to calculate two tangents of the circle. Then, make a line from self position and target position, judge the bar position, if in right of the line, select the left tangent as dribble direction, else select the right tangent. See Fig. 3

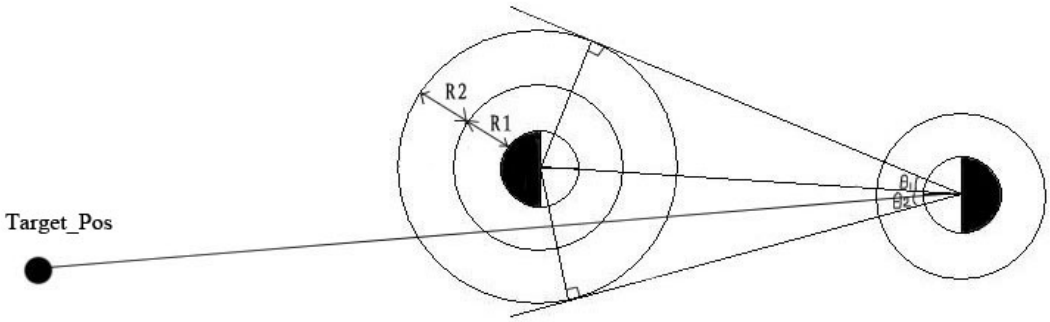

Fig.3: Dribble avoid enemy path programming

This method is used in our code and shows amazing *dribble* skill in China RoboCup 2005 competition.

5. Future Research Directions

In hope of making our agent smarter, an online study system is in planning, and the structure has been built up. We will do our best to improve our Agent skills.

Overall strategy of our CZU2006 term: when a ball approaches opposite penalty area and our term is in the attacking situation, there will be not less than 4 vanguards to participate in the attack, but when a ball approaches our forbidden area and our term is in the defense situation, there will be not less than 5 fullbacks to participate in the defense. The formation will be dynamically adjusted according to the field situation and the ball's position.

This strategy has tested severely the physical strength of players. Whether does choose the appropriate player both to be able to bring the special skills and talents of the isomerism players into full play and not to affect bringing the special skills and talents of the isomerism players into full play because of the physical strength. This will play a key role in the final success of the entire team. Therefore we give two targets of net consumption measurements of the physical strength at the full speed and the running and starting time into careful consideration [1].

5.1 The Net Consumption Measurements of The Physical Strength At The Full Speed [9]

The net consumption measurements of the physical strength at the full speed: player_sta_inc_lost. The player's physical strength decided its successor competition performance, and the abundant physical strength may guarantee player's runs at the maximum speed when necessity. Player_sta_lost is referred to physical strength value consumption of each cyclical when the player runs at full speed, therefore is

smaller is better, more saves the physical strength. Each cyclical player consumes the physical strength because of dash; also can automatically supplement physical strength stamina_inc_max. Unifies player's physical strength model, the player consumes the physical strength consumption at full speed each cycle as follows:

player_sta_lost = speed_max_ab*(1.0-player_decay)/(dash_power_rate * effort_max) - stamina_inc_max; (1)

Therefore full speed time physical strength only consumption

player_sta_inc_lost = player_sta_lost- stamina_inc_max (2)

5.2 The running and starting time player_start_cyc

Running and starting time is a number of the periodicities that is needed when the player achieved the highest speed. This represented player's speed ability. In this paper, smaller player_start_cyc is better running ability of a player. In the parameter list of the isomerism agent, the highest speed of the player player_speed_max is the fixed value 1.2. In fact, not all isomerism types all can achieve 1.2; player_speed_max is only upper limit of the speed. Moreover, the isomerism agents that can achieve the highest speed also have some differences of the starting speed. The running effect of a player is decided by the highest acceleration which can be obtained each period and the inherent attenuation of itself speed.

The highest acceleration means the acceleration that a player can achieve when he runs with all one's strength:

acc_max = max_power * dash_power_rate * effort_max ; (3)

Based on player's movement model, we can deduce the highest speed that the player can achieve is:

speed_max_ab = Min (player_speed_max, acc_max /(1- player_decay)); (4)

Running and starting time is calculated as follows:

Temp = 1.0- player_speed_max*(1.0- player_decay)/acc_max; (5)

If (temp>0.0001)

player_start_cyc = ceil (log10 (temp)/log10 (player_decay)); (6)

Else
{
player_start_cyc = 10+20*(player_speed_max -acc_max*(1.0-pow(player_decay,5)) /(1.0-player_decay)); (7)
}

Among them, if tmp>0.0001, this indicated this isomerism type can achieve player_speed_max; Otherwise, indicated this isomerism type can achieve maximum speed:

speed_max_ab< speed_max_ab<player_speed_max (8)

So in order to enable all isomerism type player_start_cyc to be allowed the mutual comparison, Equation 7 has made special processing, this is, the maximum speed could not achieve 1.2 isomerism types, the starting periodicity is additionally increased. This is the empirical formula.

Note：Min (a, b) return to smaller value between a and b, and pow (x, y) goes back to x^y.

Assignment algorithm

After processing two targets which is refined the synthetical income is obtained according to the income function below：

Reward=pow(m, player_start_cyc_uni) * pow (n, sta_inc_los_uni); (9)

Where *m* and *n* are the empirical values.

References

[1] Yao Jinyi, Lao Ni, Yang Fan, Cai Yunpeng, and Sun Zengqi: Technical solutions of TsinghuAeolus for Robotic Soccer.

[2] Cheng Jiang: Thesis Submitted to TsinghuAeolus for Bachelor Degree

[3] Huo Yinhui**,** Zhang Lianming: A method of local path programming in mobile robot.

[4] Peter Stone: Layered Learning in Multi-Agent Systems.

[5] Simon Haykin: Theory of Artificial Neural Network.

[6] Matlab toolbox.

[7] Haix_Zhang Quan_Li Changchun_Zhu Chunguang Li: A Method of Tackle Model Design Based on BP-ANN

[8] RoboCup Server Manual.

[9] Guo Ye-Jun: Thesis Submitted to Zhejiang University for Master Degree.