Motion Prediction in a high-speed, dynamic environment

Yu Sheng
College of Computer Science and Technology, Zhejiang University
Hangzhou, Zhejiang Province, P.R. China
mintbaggio@hotmail.com

Yonghai Wu
College of Mechanical and Energy Engineering, Zhejiang University
Hangzhou, Zhejiang Province, P.R. China
liunian@zju.edu.cn

Abstract

The immanent existence of system latency greatly affects the control behavior of a closed-loop system. In order to reduce the influence induced by latency, this paper proposes a systematic method based on neural network to predict the motion of objects in a high-speed, dynamic, and competitive environment. We apply this method to the competition of RoboCup Small Size League, which greatly improves the performance of our control system.

1 Introduction

The time elapsed between deciding to take an action and perceiving its consequences is called control latency, or delay [1]. The latency of a system is always unavoidable, and determined by its inherent attribute. To a closed-loop control system, the control precision is influenced by the latency because the decision is made based on the outdated perception.

In the RoboCup Small Size league (also known as F180 League), there is an inherent latency in the control system, mostly more than $100ms$. When the robots moving at a high speed, about $2m/s$ for our robots, the error between actual position and visual position might be more than $20cm$. Consequently, we have to eliminate the error to obtain a favorable control effect.

The concept of motion prediction was first introduced by Helmholtz when trying to understand how humans localize visual objects [2]. There has been many methods for solving this problem, such as Kalman filter, and Extended Kalman-Bucy Filter [4]. Kalman filter can predict well when the system is linear [3]. In the F180 Competition, however, even the motion of ball is nonlinear. Therefore, Kalman filter is not competent for this job. Some teams use Extended Kalman-Bucy Filter to predict nonlinear system. Unfortunately, a good model of the object is a prerequisite for this approach.

According to the disadvantages of the classical methods, we propose a method that based on neural network, together with some other complementary approaches to predict the motion of our robot, the ball and the opponent.

2 Measurement of Latency

The following approach helps us measure the accurate value of the system latency.

Because of the omni-wheel structure, we send to the robot a translation command that will make it travel along x axis, and a rotation command that will make it rotate around itself. Ideally, the track will be along the x axis. However, in fact, there will be an angle between the real track and the x axis because of the existence of latency.

The difference between the actual orientation and ideal orientation of the robot is constant due to the constant value of the rotation command. We can consider the actual track and ideal track as tracks of two different robots. The angle between them should be the orientation difference between the two robots, which is induced by the latency, so the quotient of the angle and the rotation speed is the value of latency. Applying this method, we calculate the value of latency, with a result of about $163ms$, about 4.9 frames.

3 Prediction of Our Robot

Not similar to most of other teams, our method not only makes a good prediction, but also improves the precision of motion control.

The commands sent to our robots are logged each cycle. If the robots always executed just as what the commands tell them to do, the consequences could be predicted easily. Our method is just based on this idea. Unfortunately, because of the inherent mechanical limitation, there are always some variations between what we tell our robot to do and the result gotten from the execution. We train a three layer feed-forward neural network to learn the variations,

Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)
1082-3409/05 $20.00 © 2005 IEEE

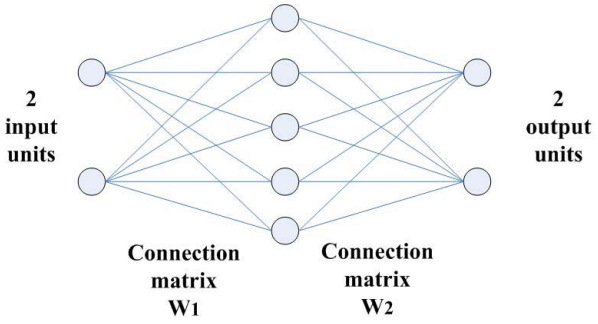

Figure 1. Architecture of the three-layer neural network

and make the corresponding modifications to the commands sent to the robots. Then, we can easily obtain the positions and behaviors of the robots using a linear prediction.

3.1 Neural Network

The neural network is trained to gain the relationship between the expecting and the actual consequences of executing. It has 2 input units, 5 hidden units, and 2 output units. The hidden units have a sigmoidal transfer function, while the transfer function of the output units is linear. We train the network with data using the standard back-propagation algorithm [5] [6]. The process of training can be easily repeated if something in the system changes.

The input vector is the command we send to the robot, given as a format of (v_x, v_y), and the target vector is the velocity it actually travels in the field, with the same format as the input. The structure of the network is depicted in figure 1 .

3.2 Training and Result

Because the symmetry of the hardware structure of our robots, we train the network with $9 * 10 = 90$ values, in an angle interval between $10\,^\circ$ and $170\,^\circ$, with a step of $20\,^\circ$, and a velocity interval between 200 and $2000mm/s$, with a step of $200mm/s$. We also measure a group of $9 * 10$ data for checking the accuracy of the network like the data above, except an angle interval between $20\,^\circ$ and $180\,^\circ$. The result is favorable.

As we have known the relationship between the expecting and the actual consequences, we can modify the actual command sent to the robots, making the robots behave just as what we expect.

3.3 Linear Prediction

Consequently, there are two types of commands, one are the commands generated by the path planner, which are used for prediction, the other are the ones actually sent to the robots. When the robot executes commands, its actual position and orientation should be the consequences of the information from vision and the execution of commands during the period of latency. Take position for example, if the robot position from vision is (x_0, y_0), and the commands sent during the last 5 frames (the value of system latency) are (v_{x1}, v_{y1}), (v_{x2}, v_{y2}), $\cdots$, (v_{x5}, v_{y5}), then the actual x coordinate should be

$$x = x_0 + (v_{x1} + v_{x2} + \cdots + v_{x5}) \cdot t \qquad (1)$$

t in the equation above is the time of one frame, which is $33ms$ for our system. The equation of y coordinate is the same as x.

4 Prediction of the Ball

We predict the motion of ball by dividing its behaviors in the field. When it is rolling freely and can be captured directly by cameras, we use neural network to predict. When it cannot be seen from cameras, we will check its status of colliding with robots, and predict it by collision model.

4.1 Neural Network

When the ball is rolling freely, we train a three-layer neural network to learn the motion of the ball just like the one mentioned in the last section, by changing the number of input units to 6 , target units to 5, and hidden units to 7.

The input data only includes the vision data from the last six frames for the position of the ball. In order to reduce the training work, we set the input and output as distance. Thereby, the input data consists of six values, which are the distance between the current frame and the other six frames in the past, such as $\sqrt{(x_0 - x_{-6})^2 + (y_0 - y_{-6})^2}$, thus the input has $6 \times 1 = 6$ values. The output is the difference of distance between the current position and five frames forward in the future, that is $5 \times 1 = 5$ values.

From the output, we can use Least Square Method to generate the motion direction of ball,and can calculate the position easily with the two values, distance and direction.

4.2 Collision Model

The method above is valid only when the ball is rolling freely. In fact, when competing, the ball might always be in the situation of be prevent by robots from captured by cameras. Then, the method above is not suitable any more,

so we apply a collision model [7]to predict the motion of ball.

The collision between robots and ball is checked for the time length of system latency, and we use the prediction positions of robots and ball as vision input. We divide one frame into several time steps, and check whether the collision will happen during one step.

The condition for confirming the occurrence of collision is different from types of robots due to various understanding of robots. For our robot, the collision is dependent on whether the ball is in the valid dribbling area because the information needed is available. However, for the opponent, the information of dribbling area is unavailable, so we consider the occurrence of collision when the track of ball intersects that of the opponent.

We record the occurrence of collision, and the information of robot colliding with the ball. When the ball disappears, we refer to the collision record and predict the position of ball until it reappears.

4.3 Prediction of the Opponent

The prediction of opponent robots is more similar to the one of ball, rather than our robots, because we do not have any information about both orientation and commands. Therefore, we make the prediction like what we have done to the ball except the collision model.

5 Experimental Results

We integrate our prediction method of our robots, ball, opponent into the ZJUNlict system. It functions well and almost eliminates the negative effect of latency. Figure 2 shows the result of ball prediction as an example.

In our system, the average error of predicting our robot is $2.25cm$, and that of predicting ball is $1.58cm$. All the results are much better than the results by applying Kalman Filter to our system.

6 Conclusion and future Work

The negative effect brought by system latency makes the prediction indispensable. We have successfully designed, implemented a predictor and made use of it in the testing field of RoboCup Small Size League. The predictor implements various rules to different types of objects. As a result, it not only compensates the system latency and improves the precision of motion control, but also enhances the quickness of system response to situation changes in the field.

However, there is also much work to do. One example is to identify and predict the actions of the opponents when competing, putting forward a high requirement of online study for opponents' behavior. We can also apply the predictor to higher levels of our system, where the latency might be also long. With the completion of these work, our control system will be more accurate and effective.

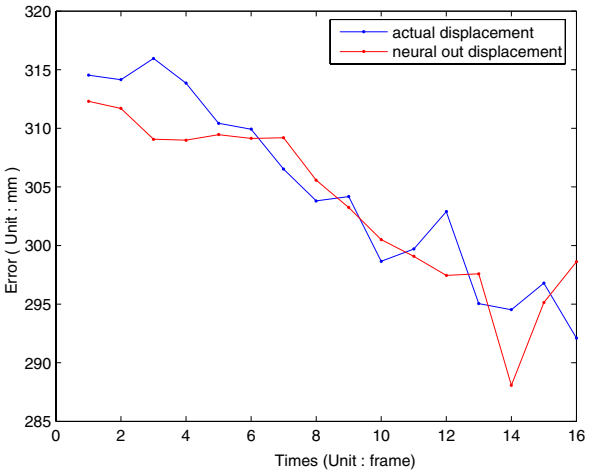

Figure 2. Ball prediction: A sample of displacements between predicted position and actual visual position in successive 16 frames.

References

[1] S. Behnke, A. Egorova, et al., "Predicting away Robot Control Latency", Lecture Notes in Computer Science, **3020** (2004) pp. 712 - 719, 2004

[2] Wolpert Daniel M., Flanagan J. Randall, "Motor Prediction" Current Biology Magazine, vol. 11, no. 18

[3] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems", Transaction of the ASMEłJournal of Basic Engineering, pp. 35-45, 1960

[4] Browning B., Bowling M., Veloso M.M. "Improbability Filtering for Rejecting False Positives", Proceedings of ICRA-02, the 2002 IEEE International Conference on Robotics and Automation, 2002

[5] Martin Hagan, H. Demtuh, M. Beale, "Neural Network Design", PWS Publishing, Boston, 1996

[6] Rojas, Raul, "Neural Networks – A Systematic Introduction", Springer Verlag, Heidelberg, 1996

[7] Wenfei Wang, "Research on fundamental strategy of RoboCup Soccer of Small Size League", bachelor thesis, Hangzhou, 2005 (in Chinese)