Motion Prediction in a high-speed, dynamic environment

Yu Sheng[1], Yonghai Wu[2], Wenfei Wang, Chengguo Guo

[1] College of Computer Science and Technology, Zhejiang University
mintbaggio@hotmail.com
[2] College of Mechanical and Energy Engineering, Zhejiang University
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, and implement different approaches for different types of objects. The predictor improves the performance of our control system, and it has been successfully tested at several RoboCup competitions with our ZJUNlict team.

1 Introduction

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control system, the control precision is influenced by the latency because the decision is made based on the outdated perception.

For a system with a demand of reacting as precisely as possible, its past information is not suitable for control planning any more. We should predict the future information, at the time when the control command arrives at the plant and is executed.

The concept of motion prediction was first introduced by Helmholtz when trying to understand how humans localize visual objects [2]. In this paper, we will propose our method to predict on the testing field of RoboCup Small Size League Competition.

1.1 Testing domain

We use ZJUNlict, a team competing in the RoboCup Small Size League, as the research platform. RoboCup is an international joint project to

promote AI, robotics, and related fields [3]. Small Size League, also known as F180 league, is one of its league divisions. In a Small Size soccer match, both teams have five robots, each of which must physically fit inside a cylinder with a diameter of $180mm$ and a height of $150mm$. Devices are permitted to dribble or kick the ball as long as they conform to the F180 rules. The competition field is approximately $4.9 \times 3.4m$ rectangular field, with an orange golf ball acting as the soccer ball. Teams use the global overhead vision as their primary sensors. Figure 1 shows a common competing environment in a F180 game.

For our team, there is a permanent latency of about 5 frames, that is $165ms$. When moving fast, our robot runs at a maximum speed of up to $2m/s$. Consequently, the difference between the actual position and perceiving position grows up to $33cm$.

Fig. 1. A picture from RoboCup 2004, Lisbon

In order to gain a good control effect, a method using neural network to predict the motion of the object is designed and implemented in our system. Tested on the research platform, our method successfully eliminate the effect of system latency. The excellent control effect helps us to win the runner-up of China RoboCup 2004 Small Size League competition, and be qualified for RoboCup 2004 and 2005 successively.

1.2 Paper Overview

The next section describes the system architecture of ZJUNlict to better understand the robotic soccer environment. Section 3 explains the for-

mation of latency, and how to measure it. Then we illustrate the method of motion prediction we design to correct the errors induced by latency, and present the experimental result. Finally, the work of this paper is summarized and the future work is highlighted.

2 System Architecture

2.1 Total System Introduction

Our system is closed-loop, and mainly contains 3 parts. Firstly, the vision subsystem. It is the primary sensor of our system. It consists of two video cameras fixed above the field, each of which runs at a rate of $30HZ$, and an off-the-field computer to process the data received from the cameras. The output of this subsystem is the positions of robots and the ball in the field. Secondly, AI subsystem retrieves the data generated by vision subsystem through network, and generates and sends the command executed by the robots through wireless communication. Thirdly, motion control subsystem interprets and execute the commands it receives from the AI one, making the robot behaves such as traveling, dribbling, or kicking. Similarly, the consequence of executing these commands is captured by the vision module, so begins the new cycle. Figure 2 shows the overall system components of ZJUNlict.

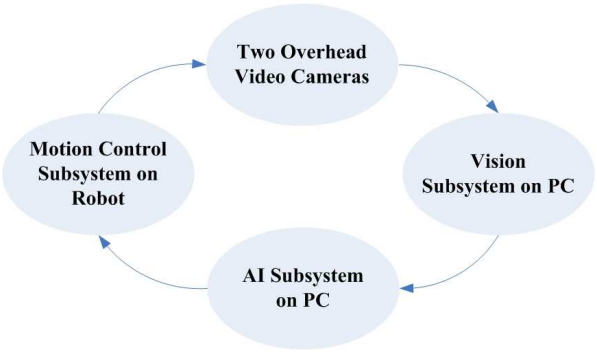

Fig. 2. System architecture of our team ZJUNlict

What this paper occupies is located in AI subsystem, so we will detail it in the next section.

2.2 AI Subsystem

AI subsystem makes use of the visual information in the field, and generate the commands for the robots to execute. The data from vision subsystem is just the positions of each object, which is called raw data. It has to be filtered, predicted, and recalculated for the use of AI subsystem.

Based on the information collected above, the situation in the field is analyzed, certain predesigned tactics is chosen to produce the behavior of each robot, and finally the path planner generates the command, which consists of the x, y axis speed, rotation speed, as well as the possible activation of dribbler, kicker.

3 Measurement of Latency

Before designing the predictor, it is important to measure the accurate value of our system.

3.1 Formation of latency

There are latencies in almost each part of this system. The following factors contribute to the formation of our system latency.

1. Vision latency. It includes camera integration time (about $10ms$), time for transmission to frame buffer (about $33ms$) and main memory of the off-the-field computer (about $40ms$), and time for vision subsystem (about $10ms$). Thereby, the total amount of this part is approximately $93ms$.
2. Latency of AI subsystem. It takes about $20ms$ to complete the task.
3. Wireless latency. Due to buffering commands and sending data to robots serially, the time spent on it is longer than we expect, which is about $20ms$.
4. Execution latency. It is the sum of time to interpret the commands, which is about $10ms$, plus robot reaction time.

Among the factors above, the robot reaction time is variable, and depends on the hardware inertia, which is hard to model and calculate. Moreover, it is the total system latency that influences the control system, so we should find a way to measure it.

3.2 Measurement

The analysis above is just an estimate. For practical application, we have found the following approach to measure the latency and confirm the rightness of the estimate.

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. Figure 3 is the two curves drawn by Matlab.

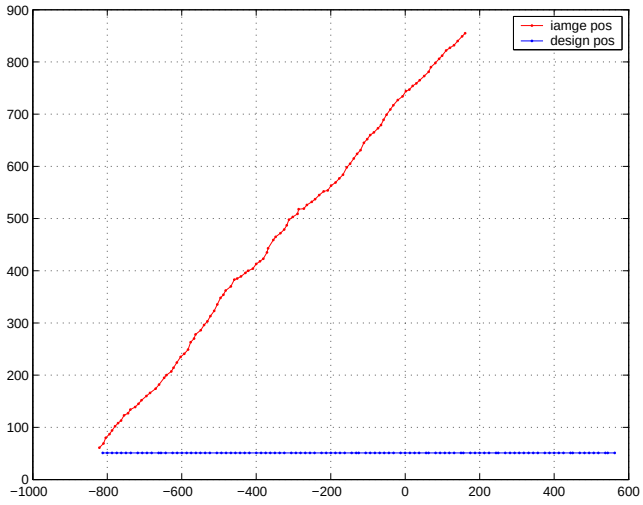

Fig. 3. Curves for measuring latency: The blue curve is the track if the system has no latency, the red one is the actual track from cameras

From the result above, we calculate the value of latency, with a result of about $163ms$.

4 Using Neural Network to predict

Kalman filter can predict well when the system is linear [4]. In F180 competition, however, even the motion of ball is nonlinear. We find that the field's resistance to the ball is correlated to the ball speed. When the speed is high, the resistance is big too, and so is the acceleration. Therefore, Kalman filter is not competent for this job.

Some teams use Extended Kalman-Bucy Filter [5] to predict nonlinear system. However, a good model of the object is a prerequisite for this approach. As it is difficult to obtain the precise motion models of robots and ball, we propose a method that basically uses neural network, together with some other complementary approaches to predict the motion of our robot, the ball and the opponent.

5 Prediction of Our Robot

Not similar to most 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 for each cycle. So if the robots always executed just as what the command 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, 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.

5.1 Neural Network

The neural network is trained to gain the relationship between the expectation and the actual consequence 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 [6] [7]. 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 4 .

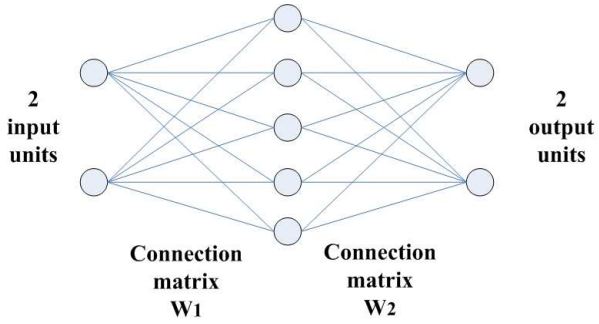

Fig. 4. Architecture of the three-layer neural network

5.2 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. Figure 5 show the comparision between the output of network and the actual consequences of execution.

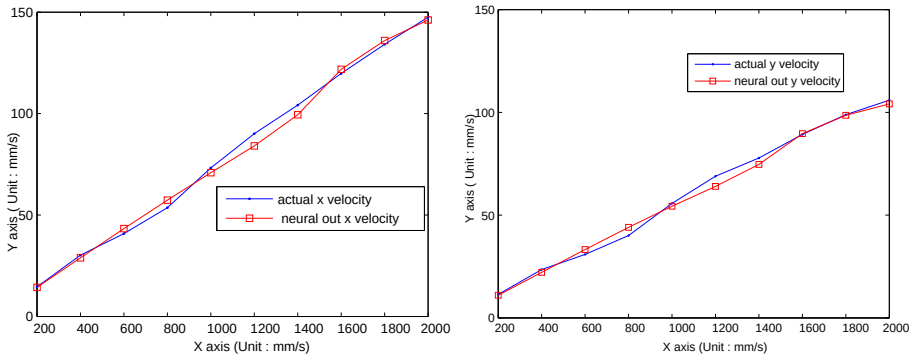

Fig. 5. x and y velocity from output of neural network and actual vision data. The red curve is the output of neural network, and the blue curve is the actual vision data. The left figure depicts the curve of x velocity in different total velocities of robot, from $200mm/s$ to $2000mm/s$, and the right describes the y one.

As we have known the relationship between the expectation and the actual consequence, we can modify the actual command sent to the robots, making the robots behaves just as what we expect.

5.3 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 execute 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

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

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

6 Ball Prediction

We predict the motion of ball by dividing its behaviors in the field. When it is rolling freely and can be captured 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.

6.1 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.

6.2 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 [8]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. Figure 6 shows the collision check of our robots and opponent.

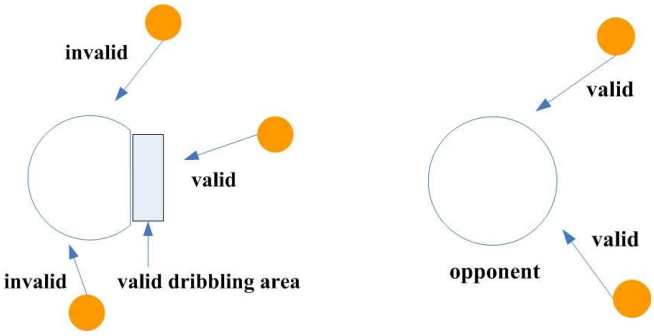

Fig. 6. Collision check for our robot and opponent. In the left figure, the collision is valid only when the ball is moving towards the dribbling area of our robot. In the right figure, the intersection of the track of opponent and ball means the occurrence of collision. Opponent is described as a circle because we do not know its orientation.

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.

6.3 Opponent Prediction

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 command. Therefore, we make the prediction like what we have done to the ball except the collision model.

7 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 7 and figure 8 shows the result of our robot prediction and ball prediction.

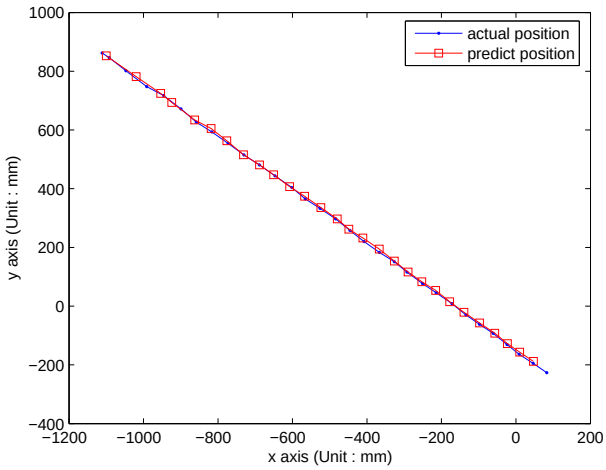

Fig. 7. Our robot prediction: Positions of our robot prediction and actual visual ones. The value of x axis is the x coordinate, and so is y.

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 favorable.

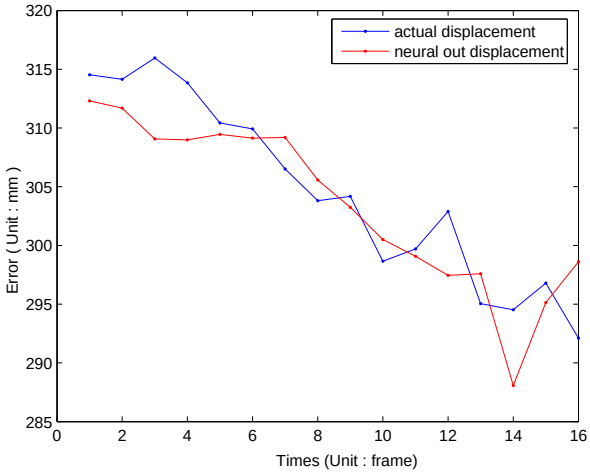

Fig. 8. Ball prediction: A sample of displacements between predicted position and actual visual position in successive 16 frames.

8 Conclusion and future Work

The negative effect brought by system latency makes the prediction indispensable. We have successfully designed, implemented a predictor and make 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.

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. RoboCup, http://www.robocup.org, 1998

4. Kalman, R. E., “A New Approach to Linear Filtering and Prediction Problems”, Transaction of the ASMEJournal of Basic Engineering, pp. 35-45, 1960
5. 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
6. Martin Hagan, H. Demtuh, M. Beale, “Neural Network Design”, PWS Publishing, Boston, 1996
7. Rojas, Raul, “Neural Networks C A Systematic Introduction”, Springer Verlag, Heidelberg, 1996
8. Wenfei Wang, “Research on fundamental strategy of RoboCup Soccer of Small Size League”, bachelor thesis, Hangzhou, 2005 (in Chinese)