An Adaptive Trajectory Prediction Method for Ping-Pong Robots

Yifeng Zhang, Rong Xiong*, Yongsheng Zhao, and Jian Chu

State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou, China
rxiong@iipc.zju.edu.cn

Abstract. Trajectory prediction is a key issue to a ping-pong robot, many algorithms have been developed. To get more robust and accuracy prediction under different serving conditions, this paper presents a new prediction method. The proposed method establishes two equivalent forms of the dynamic model of flying ball, where the discrete form for state estimation and the continuous form for trajectory prediction. The two forms share the same parameters' value. According to force analysis, It is found that the model parameters are deeply related to ball's state (position, velocity) . So we train the model parameters offline respect to ball's state, instead of setting them to a constant value. This enables the model to be adapted accordingly online. Experimental results show the effectiveness and accuracy of the proposed method for the ball with different velocities.

Keywords: ping-pong robot, trajectory prediction, adaptive.

1 Introduction

Ping-Pong robot as a classic real-time 'eye-hand' platform is now attracting more and more researchers. Compared with other sports, the ping-pong ball is moving fast and the distance/time of flying is very short. To leave enough time for the robot to act, it is very essential to predict the ball's trajectory effectively and accurately. Trajectory prediction also has very important significance in many other applications such as target tracking, space intercept.

Many algorithms have been proposed to predict flight trajectory. R.L. Andersson[2] gave the moving equation based on three significant forces in ball's flight: gravity, air drag, and the Magnus Effect. But the spin is hardly perceptible and the author hasn't given the resolution. Z. Zhang et al[14] and Y. Wang et al[12] gave a more detailed analysis on the aerodynamic model of ball's flight, and gave the estimated values of forces impacted on. But to simplify the calculation, only gravity and air drag were applied in their predicting algorithm. Y. Zhang et al[13] used a model that also considers only gravity and air drag, and the air drag was simplified to be one-order respected to velocity. They built a linear equation to

* Corresponding author.

C.-Y. Su, S. Rakheja, H. Liu (Eds.): ICIRA 2012, Part III, LNAI 7508, pp. 448–459, 2012.

describe flight trajectory, then a kalman filter with constant parameters was used to estimate ball's state. F. Miyazaki et al[7] [8] built three input-output maps to determine the flight trajectory and paddle hitting command to return the ball to a desired point, the input value is the ball's state, which was calculated by fitting a first-order polynomial in horizontal direction and a second-order polynomial in vertical direction, thus no physical model was used. The spinning influence of objects with sphere shape has also been discussed. A. Nakashima et al[10] and R. Cross[4] [5] analyzed the spinning influence of a tennis ball when flying and bouncing. Y. Huang et al[6] analyzed the ping-pong ball's flight model with spinning, and deduced the spinning velocity by observing the offset of the ball's flight trajectory, but it depends on very accurate perceptions and works only when the ball is spinning heavily. By now it is still a very challenge task to predict ball's trajectory accurately under different serving conditions.

This paper focuses on online prediction of ball trajectory. There are two key processes have to be dealt with: estimate current ball's state and predict following flight trajectory. Most of existing implements estimated ball's state by fitting polynomial[7] [8] [12] [14] or by a kalman filter[9] [13], and predicted trajectory with discrete model[3] [6] [12] [14] or trained results[7] [8] [13]. We handle these two processes using a same dynamic model. The model is built through force analysis and have discrete and continuous forms. The discrete form is for state estimation based on discrete position caught by stereo-vision system, and the continuous form is for trajectory prediction. The two forms are equivalent and share the same parameters. Instead of simplifying model parameters to be a constant value, we train the relationships between parameters and ball's state offline, then the parameters can adapt with ball's state online. The vision system is working in 120 fps (frame per second) and the experimental results show the validity and effectiveness of the method proposed.

In the following of this paper, Section 2 gives the frame of adaptive prediction process and defines some importance events in ball's flight. Section 3 describes the two forms of dynamic models based on force analysis. Section 4 describes the whole process of filter and trajectory prediction. In Section 5, the experiments are provided to verify the effectiveness of the proposed method. Finally, a conclusion is given in Section 6.

2 Adaptive Prediction Policies

2.1 The Frame of Adaptive Prediction Process

The frame of adaptive prediction process is shown in Fig. 1. The force impacted on a flying ball and the dynamic model of ball's flight is analyzed first. Then two equivalent forms of flight model are established accordingly, the discrete one is used to estimate current ball's state and the continuous one is used to predict flight trajectory. These two form models share the same model parameters. According to force analysis, the model parameters are deeply related to ball's state. A neural network is applied to train the relationships between model

parameters and ball's state. Thus the model parameters can be adapted online, both for estimation and prediction.

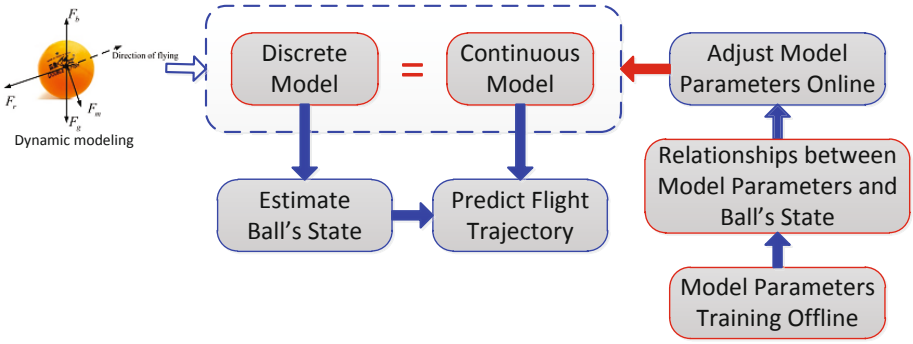

Fig. 1. The frame of model adaption process

2.2 Definition of Ball Events in One Stroke

To make the following explanation clear, we define some "ball events", as shown in Fig. 2. Ball is flying from opponent's court to robot's court. Virtual plane is a standard measurement plane for parameters offline training and online adapting. That means wherever ball is, it have to trace back/forward to the point when it passes through virtual plane to adapt the model. Hit plane is where the robot hits the ball.

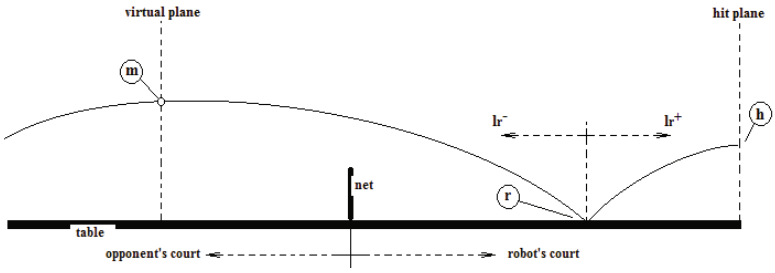

Fig. 2. Definition of ball events

There are three important events in one stroke, and the ball' state on these events is also recorded as following:

- Event (v): ball is passing through the virtual plane with state (S_v, V_v).
- Event (r): ball is bouncing on the table with state (S_r, V_r^-, V_r^+).
- Event (h): ball is hit back by robot with state (S_h, V_h).

S is ball's position and V is velocity, V_r^-, V_r^+ are velocities before and after bounce. The flight trajectory can be split into two parts based on event (r). Donate l_r^- is trajectory before bouncing and l_r^+ is trajectory after bouncing.

3 Models of Ball's Flight

3.1 Dynamic Model of Flying Ball

The ball flying in the air is a typical aerodynamic problem, based on the analysis of fluid mechanics. There are four main forces impacted on the ball: gravity F_g, buoyancy F_b, air drag F_r, and Magnus force F_m. Fig. 3 shows the direction of each force, and the detailed calculation can be refer to literature[14].

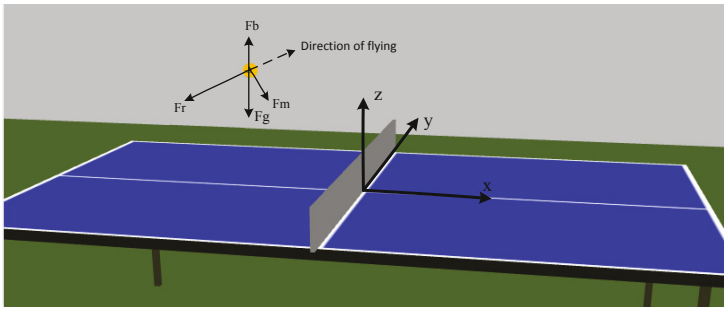

Fig. 3. Force analysis of flying ball

The world coordinate is also defined in Fig. 3. It is reasonable to assume that the movements along x y z are independent with each other, and the dynamics of the movements along x and y are same. So only the dynamics of movements along x and z are discussed here.

Donate a_x and a_z are respectively the accelerations of ping-pong ball along axis x and z. According to the force analysis above, the dynamic model along axis x and axis z are

$$ma_x = -F_{rx} + F_{mx} = -\frac{1}{2}k\rho_a A{v_x}^2 + \frac{1}{2}k\rho_a Al\omega v_x \tag{1}$$

$$ma_z = -F_{rz} - F_{mz} + F_b - F_g = -\frac{1}{2}k\rho_a A{v_z}^2 - \frac{1}{2}k\rho_a Al\omega v_z + \rho_a Vg - mg \tag{2}$$

v_x and v_z are respectively the velocities of ball along axis x and z, m is the quality of ball, g is acceleration due to gravity, k is the resistance coefficient, ρ_a is the air density, l is the lift coefficient, ω is the angular velocity, A is the cross-sectional area, V is the volume of ball. Denote

$$K_x = f(v_x) = \frac{1}{2}k\rho_a Av_x - \frac{1}{2}k\rho_a Al\omega \tag{3}$$

$$K_z = f(v_z) = \frac{1}{2}k\rho_a A v_z + \frac{1}{2}k\rho_a A l\omega \tag{4}$$

$$F_v = constant = -\rho_a V g + mg \tag{5}$$

Thus (1) and (2) can be simplified as follows.

$$ma_x = -K_x v_x \tag{6}$$

$$ma_z = -K_z v_z - F_v \tag{7}$$

K_x and K_z are variables strongly related to ball's state. And though the parameters m, g, ρ_a, k, l are constant and can be known by data-sheet in advance, but their values are varied in some range and are inaccurate. For example, the value range of ρ_a is usually $1.21 \sim 1.27(kg/m^3)$. So it will significantly influence the precision of prediction if K_x and K_z were set to a constant value. In this paper, we will try to train the relationships between these parameters and ball's state later.

3.2 Kinematical Model of Flying Ball

With the dynamics model shown in equations (6) and (7), we can get the kinematical model of flying ping-pong ball. Denote

$$p_x = -\frac{K_x}{m}, \quad p_z = -\frac{K_z}{m}, \quad q_z = -\frac{F_v}{m}, \tag{8}$$

The continuous kinematical model can be presented as

$$v_x(t) = v_{x_0} e^{p_x t} \tag{9}$$

$$v_z(t) = v_{z_0} e^{p_z t} + \frac{q_z}{p_z}(e^{p_x t} - 1) \tag{10}$$

$$s_x(t) = s_{x_0} + \frac{v_{x_0}}{p_x}(e^{p_x t} - 1) \tag{11}$$

$$s_z(t) = s_{z_0} + \frac{v_{z_0} + \frac{q_z}{p_z}}{p_z}(e^{p_z t} - 1) - \frac{q_z}{p_z}t \tag{12}$$

(s_x, s_y, s_z) is the position of the flying ball, $(v_{x_0}, v_{y_0}, v_{z_0})$ and $(s_{x_0}, s_{y_0}, s_{z_0})$ are the initial velocities and positions of the trace.

Considering the perception of vision system is periodical and discrete, there should also be a discrete model to estimate ball's state. With the assuming that the acceleration and the velocity between two sample times is constant, and let the discrete velocity be the position distance.

$$s(i+1) = s(i) + v(i) \tag{13}$$

The discrete kinematical model can be presented as

$$v_x(i+1) = v_x(i) p'_x \tag{14}$$

$$v_z(i+1) = v_z(i) p'_z + \frac{q_z}{p_z}(p'_z - 1) \tag{15}$$

$$s_x(i+1) = s_x(i) + v_x(i) \tag{16}$$

$$s_z(i+1) = s_z(i) + v_{zx}(i) \tag{17}$$

where

$$p'_x = e^{p_x \Delta t}, \quad p'_z = e^{p_z \Delta t} \tag{18}$$

Δt is the sample period and is known in advance.

These two forms of kinematical model are equivalent and their parameters can be transformed to each other easily. Once we have trained the set of parameters of one model form, the parameters of the other model form are also trained spontaneously.

3.3 Bouncing Model

A significant feature of ping-pong game is that the ball will collide with the table and bounce up before the player hits it. Such a collision involves the degradation of energy which results in the change of velocity, and it will change the flight trajectory of ball dramatically. According to the law of conservation of energy

$$\frac{1}{2}mv^{+2} = \eta^2 \frac{1}{2}mv^{-2} \tag{19}$$

Here η is the coefficient of energy degradation, v^- and v^+ are respectively the velocities before and after collision. We assume that the values of η are varied in x and z axis, thus

$$v_x^+ = \eta_x v_x^-, v_z^+ = \eta_z v_z^-, \tag{20}$$

η_x and η_z are universal in both continuous and discrete model forms. Their values are variable and will be trained later.

4 Training Model Parameters

As discussed in Section 2.2, the entire flight process can be split into three parts based on event(r): flying before bouncing, bouncing, and flying after bouncing. Assuming that parameters of all models are the same in axis x, y, and trajectory l_r^- and l_r^+ share the same flying model but with different parameters. There are eight parameters in all:

- (p_x, p_z, q_z): flying model parameters of l_r^-.
- $(p_{x_2}, p_{z_2}, q_{z_2})$: flying model parameters of l_r^+.
- (η_x, η_z): bouncing model parameters.

To train the model parameters, we have collected 1107 trajectories recorded by high-speed vision system, with various initial positions and velocities served by human and a server machine. For each trajectory, ball's states on event points (S_v, V_v, S_r, V_r^-, V_r^+, S_h, V_h) were fitted through a kalman smoother. Then all the model parameters for this trajectory can be calculated based on these key values using (9) to (12).

Based on (3) and (4), model parameters K_x and K_z are functions of ball velocity and spinning velocity. Since the spinning velocity is hardly perceptible, only relationships between parameters and velocity will be trained. We assume

that the ball flight is a Markov process, the model parameters are constant in different points of the same flight trajectory, and are determined by ball's state when it pass through the virtual plane. We collect a large number of parameter pairs: $(V_m$ - $p_x, p_z, q_z)$, $(V_r^- - \eta_x, \eta_z)$, $(V_r^+ - p_{x_2}, p_{z_2}, q_{z_2})$, then all the parameters can be trained by AI learning method. The result of neural network training was shown in Fig. 4.

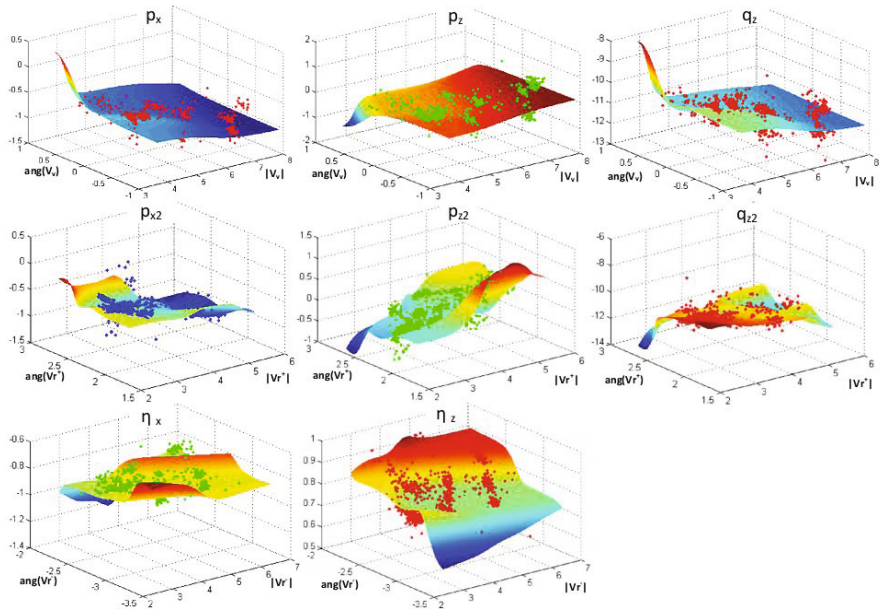

Fig. 4. Model parameters training results by neural network

Fig. 4 illustrates the relationships between model parameters and the input values, and all the model parameters can be determined by (S_v, V_v) through following steps:

1. Get (p_x, p_z, q_z) based on (S_v, V_v) and training result.
2. Predict V_r^- based on $(S_v, V_v, p_x, p_z, q_z)$.
3. Get (η_x, η_z) based on V_r^- and training result.
4. Calculate V_r^+ based on (V_r^-, η_x, η_z).
5. Get $(p_{x_2}, p_{z_2}, q_{z_2})$ based on V_r^+ and training result.

5 Trajectory Prediction

5.1 Estimation of Ball's State

As discussed in section2, the perception is discrete and periodical. A kalman filter is used to estimate the ball state on every sample time. Based on the discrete kinematical model, the state function is presented as follows:

$$\begin{bmatrix} s_x(i+1) \\ v_x(i+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & p'_x \end{bmatrix} \begin{bmatrix} s_x(i) \\ v_x(i) \end{bmatrix} + W_x(i) \tag{21}$$

$$\begin{bmatrix} s_z(i+1) \\ v_z(i+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & p'_z \end{bmatrix} \begin{bmatrix} s_z(i) \\ v_z(i) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{q_z}{p_z}(p'_z - 1) \end{bmatrix} + W_z(i) \tag{22}$$

The observation functions are same in different axis and can be presented as:

$$\breve{s}(i) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s(i) \\ v(i) \end{bmatrix} + R(i) \tag{23}$$

$\breve{s}$ is the raw ball position caught by vision system on each sample time, W_x, W_z are the process noise and R is the observation noise, their values are set in advance based on the analysis of kinematical model process and perception accuracy of vision system.

5.2 Prediction of Ball Flight Trajectory

Once the ball's state of current ball is obtained, the following trajectory can be calculated easily by (9) to (12). Compared with discrete model, the continuous model can provide the entire information of following flight and more accurate bouncing process. There are two key steps in the predict process: adapting model parameters and modifying ball's state when bouncing. The model parameters are updated each time when vision system catches new ball position. To adapt new model parameters, the ball's state should be predicted to or backtracked to the measurement virtual plane to get (S_v, V_v). The steps to calculate model parameters based on (S_v, V_v) are discussed in Section 3.4 and the adapting process is an iterative process to get a set of optimized parameters for current flight.

The collision position on axis z is known as the radius of ping-pong ball. Thus according to

$$s_z(t_{col}) = r = s_0 + \frac{v_0 + \frac{q_z}{p_z}}{p_z}(e^{p_z t_{col}} - 1) - \frac{q_z}{p_z} t_{col} \tag{24}$$

The time of collision t_{col} can be calculated. With the flying model and bouncing model, the bouncing process can be resolved accordingly and the flight after bouncing can also be calculated with new flying model parameters (p_{x_2}, p_{z_2}, q_{z_2}).

6 Experimental Results

Experiments are conducted to verify the models and prediction method proposed. Two PointGrey Grasshopper cameras working in 120fps and 640*480 pixels are used to catch 3D ball position, the maximal error of static localization is 0.9cm. The method proposed works effectively and the time cost is only related to bouncing times, normally less than 0.002ms on PC.

Table 1. Model parameters of two methods used in Fig. 5

Model parameters of method in [14]							
K_m	K_{r_x}	K_{r_y}	K_{r_z}	b_x	b_y	b_z	g
0.160	0.503	0.752	-0.813	0.107	-0.011	0.322	11.200
Model parameters of method proposed							
p_x	p_z	q_z	p_{x_2}	p_{z_2}	q_{z_2}	η_x	η_z
0.657	0.218	10.540	0.789	0.311	10.535	0.736	-0.912

Table 2. Error analysis in Fig. 5

		Method in [14]		Method proposed without adapting parameters	
prediction error on bouncing point	/mm	x	y	x	y
	average	4.26	10.73	4.29	11.81
	variance	6.11	13.83	6.21	16.39
prediction error on hitting point	/mm	x	y	x	y
	average	48.00	19.90	9.61	13.07
	variance	52.97	27.42	12.00	18.51

Fig. 5 compares the prediction errors on bouncing point and hitting point with the method in [14], which estimates ball's state by fitting polynomial and predicts trajectory using a two-order air drag discrete model. Ball is served with similar initial positions and velocities, the parameters of both methods are fixed to a constant value optimized to this trajectory set, which is given in Table 1. The prediction is executed when ball passing through the virtual plane. These two methods both work well on prediction of bouncing point. But on hitting point, the method proposed works much better because it considers the model change caused by bouncing process. The error analysis is shown in Table 2.

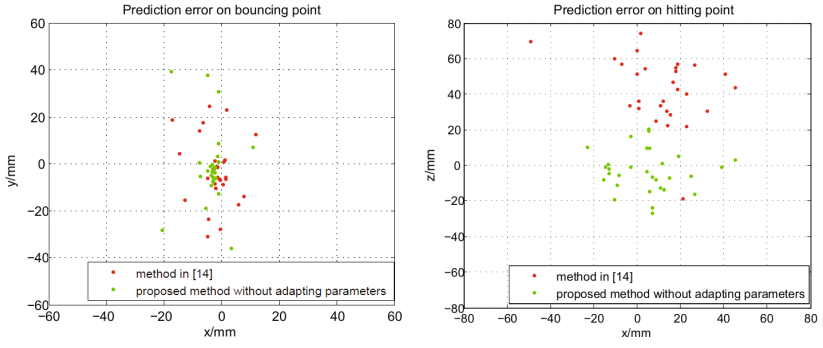

Fig. 5. Prediction errors on bouncing point and hitting point with similar (s_0, v_0)

Table 3. Model parameters of each sample period in Fig. 6

sample period	3	4	5	6
(p_x, p_z, q_z)	0.819,0.169,11.123	0.664,0.245,10.559	0.669,0.260,10.595	0.670,0.262,10.604
$(p_{x_2}, p_{z_2}, q_{z_2})$	0.665,0.330,10.400	0.507,0.149,10.465	0.513,0.144,10.506	0.514,0.143,10.505
η_x, η_z	0.764,-0.903	0.746,-0.920	0.746,-0.919	0.746,-0.918

Table 4. Error analysis in Fig. 7

	Method in [14]		Method proposed without adapting parameters		Method proposed with adapting parameters	
/mm	x	y	x	y	x	y
average	8.72	39.23	6.46	46.82	4.49	9.91
variance	10.34	56.01	11.17	71.78	5.71	12.79

Fig. 6 demonstrates another important feature of the method proposed, parameters are adapted during prediction. The green trajectory is the predicted results of each future sample time, compared with the white trajectory observed and fitted posterior. The prediction process is enabled since the 3rd sample period, using model parameters given in advance. Table 3 list the model parameters of each sample period. Model parameters are adapted since the 4th sample period and dramatically improve the precision of prediction.

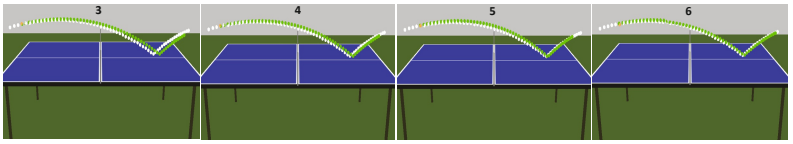

Fig. 6. Prediction results on different sample periods

Fig. 7 compares the prediction errors on bouncing point under different serving conditions, of method proposed with and without adapting model parameters, and the method in [14]. Ball is severed with various initial velocities ($4.37 - 8.32 m/s$), and various initial angles ($3.24 - 20.92\,^\circ$). The prediction is executed when ball passing through the virtual plane. Table 4 shows the error analysis of these three methods. While initial velocity changes, both the method in [14] and the method proposed without adapting parameters works unsatisfactorily. That means fixed parameters would not meet this situation, especially for the model proposed which greatly simplifies the flying model. But while model parameters adapted, the prediction accuracy improved dramatically and fully meets the rally task.

We applied the proposed method to two humanoid ping-pong robots Wu and Kong, and the results suggest that when the hitting point is in robot's valid

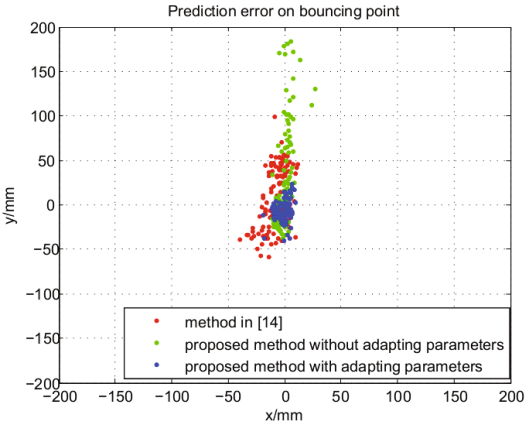

Fig. 7. Prediction errors on bouncing point with varied (s_0, v_0)

movement area, the probability of successfully returning the ball is about 100%. The best record is 88 turns (176 strokes) rally with each other and 145 turns with a human. Fig. 8 shows a scene of robots playing with each other and with human, full video are reachable on the website `http:www.youtube.com/watch?v=t_qN3dgYGqE` .

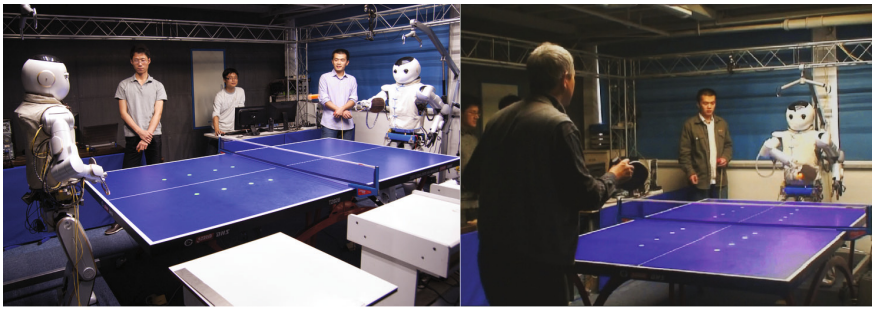

Fig. 8. The ping-pong robots Wu and Kong play with each other and with human

7 Conclusion

In this paper, we have discussed a new approach to describe and to predict ball's trajectory. The whole trajectory is split into three parts, and is modeled both in continuous and discrete form, which makes the parameters adapting realizable. The effectiveness and accuracy of this method under various serving conditions is validated by experiments. And it works very well on ping-pong robots to rally with each other and with human.

Acknowledgment. This work was supported by the National Nature Science Foundation of China (Grants No. NSFC: 61075078), and the Key Project of the National 863 plan (Grants No.2008AA042602).

References

1. Acosta, L., Rodrigo, J.J., Mendez, J.A., Marichal, G.N., Sigut, M.: Ping-Pong player prototype. IEEE Robotics & Automation Magazine 10, 44–52 (2003)
2. Andersson, R.L.: Understanding and applying a robot ping-pong player's expert controller. In: IEEE International Conference on Robotics and Automation, vol. 3, pp. 1284–1289 (1989)
3. Chen, X., Tian, Y., Huang, Q., Zhang, W., Yu, Z.: Dynamic model based ball trajectory prediction for a robot ping-pong player. In: IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 603–608 (2010)
4. Cross, R.: Bounce of a spinning ball near normal incidence. American Journal of Physics 73, 914–920 (2005)
5. Cross, R.: Measurement of horizontal coefficient of restitution for a superball and a tennis ball. American Journal of Physics 70, 482–489 (2001)
6. Huang, Y., Xu, D., Tan, M., Su, H.: Trajectory Prediction of Spinning Ball for Ping-Pong Player Robot. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3434–3439 (2011)
7. Matsushima, M., Hashimoto, T., Takeuchi, M., Miyazaki, F.: A learning approach to robotic table tennis. IEEE Transactions on Robotics 21, 767–771 (2005)
8. Matsushima, M., Hashimoto, T., Miyazaki, F.: Learning to the robot table tennis task - ball control & rally with a human. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 2962–2969 (2003)
9. Mulling, K., Kober, J., Peters, J.: A Biomimetic Approach to Robot Table Tennis. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1921–1926 (2010)
10. Nakashima, A., Ogawa, Y., Kobayashi, Y., Hayakawa, Y.: Modeling of Rebound Phenomenon of a Rigid Ball with Friction and Elastic Effects. In: American Control Conference (ACC), pp. 1410–1415 (2010)
11. Wang, Z., Lampert, M.K., Scholkopf, B., Peters, J.: Learning anticipation policies for robot table tennis. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 332–337 (2011)
12. Wang, Y., Sun, L., Liu, J., Yang, Q., Zhou, L., He, S.: A novel trajectory prediction approach for table-tennis robot based on nonlinear output feedback observer. In: IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1136–1141 (2010)
13. Zhang, Y., Wei, W., Yu, D., Zhong, C.: A tracking and predicting scheme for ping pong robot. Journal of Zhejiang University-SCIENCE C (Computer & Electronics) 12, 110–115 (2011)
14. Zhang, Z., Xu, D., Tan, M.: Visual Measurement and Prediction of Ball Trajectory for Table Tennis Robot. IEEE Transactions on Instrument and Meansurement 59, 3195–3205 (2010)