Learning Optimal Striking Points for A Ping-Pong Playing Robot

Yanlong Huang[1], Bernhard Schölkopf[1], Jan Peters[1,2]

***Abstract*— In this paper, an approach for learning optimal striking points is proposed. Based on a ball-flight model and a rebound model, a set of reachable striking points within the robot's workspace can be obtained. However, while these striking points are geometrically reachable, their success probability differs substantially due to the robot's nonlinear dynamics, the distance to the ball, the need to reach sufficient velocity as well as the right angle at interception and non-uniform sensitivity to errors. Thus, it is crucial for a ping-pong robotic system to select striking points well. As a successful ball interception is the result of various factors that cannot be modeled straightforwardly, we suggest determining optimal striking points based on a reward function that measures how well the ping-pong ball's trajectory and the racket's movement coincidence. In this approach, we propose to learn a stochastic policy over the reward given the prospective striking point in order to facilitate exploration of a wide range of prospective striking points. The resulting learning method takes both the amount of experience data and its confidence into account to reach optimal solutions reliably. Evaluation with a real robotic system demonstrates the applicability of the proposed method.**

I. INTRODUCTION

Most typical ping-pong robotic systems [1], [2], [3], [4] are composed of visual ball position estimation, ball trajectory prediction, interception point determination, inverse kinematics and robot trajectory generation. Ball position estimation detects the ping-pong ball's 3-D position in the reference coordinate [5], [6], [7], and, subsequently, an accurate estimation of the ball's state (position and velocity) is obtained through polynomial fitting [4] or employing the Kalman filter [7]. Based on the current state, the remainder of the ball trajectory can be predicted and prospective striking points (i.e., position, velocity and time when the robot can reach the ball) can be determined.

Two kinds of methods for ball trajectory prediction are common: physical models [2], [4], [5], [8], [9], [10], [11] consist of several hybrid phrases of free ball flight and ball rebound during contact. Based on the current state, the ball flight model predicts the ball's landing position on the table as well as the ball's velocity just before the rebound. Subsequently, the rebound model predicts the ball's velocity just after the rebound. Finally, based on the landing position and the rebound velocity, the ball flight trajectory can be completed. In contrast, data-driven approaches [7], [12], [13] view ball trajectory prediction as a regression problem where a mapping from current ball state (i.e., ball position and velocity) to the futures ones is obtained through machine learning methods, even with off-the-shelf ones such as neural networks [7], locally weighted regression (LWR) [12], support vector regression (SVR), and Gaussian process regression (GPR) [13]. Such approaches have the advantage that they often can deal with substantially less pre-processed data, require no idealized physics models (often violated in real table tennis, e.g., due to non-ideal contacts during spin) and can even be used in conjunction with additional signals (e.g., Wang et al. [14] used opponent behavior in ball prediction, but also strong clues such as sound can be used straightforwardly). The accuracy of the ball prediction model is determined by the amount of sampled training data, which may often not suffice. Furthermore, if the amount of training data becomes sufficiently large, the matrix inversions in LWR or GPR may become computationally too expensive and only approximate versions of these methods (such as locally weighted projection regression for LWR [15] or sparse GPR [16] or local GPR [17]) may be applicable. Similarly, neural networks are often problematic as they frequently require extensive manual tuning of open parameters (such as the number of hidden units, learning and moment rates, etc) which may not always be possible in an online setting.

Just reaching the striking point with the racket does not suffice for a successful table tennis return. Instead, the success of a table tennis stroke is usually determined by the velocity and orientation of the racket. Returning an

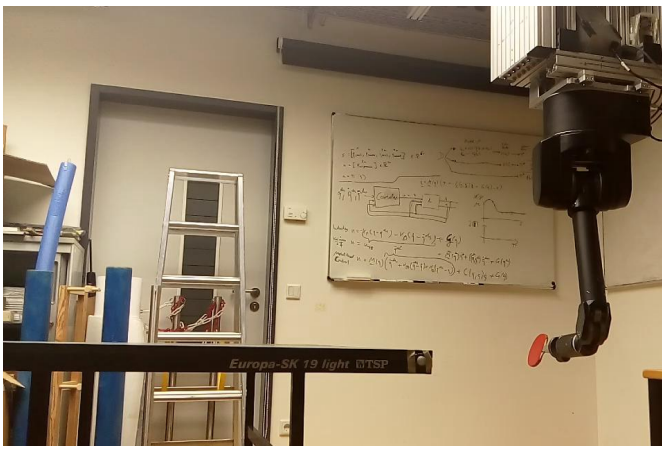

Fig. 1: Our robot table tennis setup consisting of a Barret WAM robot arm with seven degrees of freedom and of four high-speed cameras that track the table tennis ball. The employed robot arm is a custom-made and unique high-speed version of this arm.

[1]Yanlong Huang, Bernhard Schölkopf, and Jan Peters are with the Max-Planck Institute for Intelligent Systems, Spemannstr. 38, 72076 Tübingen, Germany. firstname.lastname@tuebingen.mpg.de

[2]Jan Peters is with Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany peters@informatik.tu-darmstadt.de

incoming ball to a desired position [11], [12], [18] can be decomposed into two sub-problems: the desired outgoing velocity of the ball after the impact on the racket needs to be predicted and the racket's velocity and orientation need to be adjusted such that the desired outgoing velocity of the ball can be achieved for the predicted incoming velocity. Simplified physical models [11] and LWR [12] were used to solve these two sub-problems, respectively. A combination of a fuzzy cerebellar model articulation controller (FCMAC) and LWR was suggested in [18] such that the overall learning efficiency was improved.

Another important issue is the determination of the desired joint states at the striking time based on the desired robot's Cartesian racket. The complexity of this problem depends on the ping-pong robot's mechanics. For both the four degrees of freedom (DoF) robot [12], [19] and the 5-DoF robot [18], [20], from the literature the mapping from racket to joint states is determined by the geometry of the robot (except for special cases) and joints can frequently be assigned predefined functions. For the 5-DoF robot [18], [20], three joints control the horizontal and vertical movements while the other two joints control the racket orientation; thus, the joint states can be directly obtained from the Cartesian states. However, when a redundant 7-DoF robot arm [1], [2], [3], [11], [13], [14], [21] has to strike a ball, infinitely many solutions exist. In this case, the geometric interception of the ball does not fully determine the solution and analytic decompositions become highly problematic. Instead, the desired joint states are found as an optimization where additional objectives (such as manipulability, proximity to a comfort posture, travel distance) are optimized, see e.g., [11]. From a machine learning perspective, the problem can be treated quite similarly. E.g., Kober et al. [22] proposed a reinforcement learning approach that predicted the desired joint hitting states using a cost-regularized Kernel regression.

To generate an entire robot arm trajectory, movement planning becomes an essential step due to the short reaction time and the high speed at interception. Such trajectories can either be planned in joint space [11], [12], [13], [20] or end-effector space [2], [3], where Cartesian trajectories require the additional solution of the inverse kinematics problem. Planning in joint space often creates more agile fast movements while planning in end-effector is often easier to comprehend. For planning the movement trajectory in the joint space, the fifth-order polynomial spline interpolation [11], [12] and plans consisting of standardized arc and line movements [20] have been proposed as classical robotics approaches. As an approach to generalizing of plans from demonstrations, the dynamic motor primitive (DMP) [23], [24] trained by kinesthetic teach-in generates the joint movement trajectory [13]. Movement planning in the end-effector space was studied in [2], [3], where the racket's position and posture (represented by Euler angles) were planned by interpolating with fifth-order polynomial splines, and, subsequently, inverse kinematics determined the corresponding joint space trajectory.

While there exists this myriad of approaches to all the components of robot table tennis as described above and most of the approaches have reached relatively high reliability, a crucial component has yet to catch up with them: how can a good striking point be chosen? In some previous publications [2], [4], [8], [11], [20], the striking point is commonly defined as the intersection point between the predicted ball rebound trajectory and a virtual striking plane, which is obviously a heuristic albeit that it can be motivated from human subject studies [11]. The virtual plane has usually been chosen either as parallel continuation of the table's top [4], [8], [20] or as a perpendicular plane between table and robot base [2], [11]. A similar simplification of the choice of the considered striking points can result from geometry, e.g., a linear axes-based robot where the racket is fixed at a specific height [12] can be seen as a virtual striking plane parallel to the table plane. Besides the virtual striking plane, alternative simplifications have been suggested such as finding the nearest point [3], predetermining the strike duration [7], constructing the fuzzy decision system based on human insights (e.g., Huang et al. [25] analyzed the simplified joint acceleration, Su et al. [26] limited the set of considered striking points). In contrast to the heuristics and simplifications used in the past, this article proposes an approach for learning striking points without an explicit acceleration analysis and no limitation onto a specific search area.

Such an approach can result in crucial difference. For example, human players frequently move the racket along the ball flight trajectory but in the opposite direction when returning the incoming ball – a stark contrast to just intercepting a point. Such a scheme could reduce the effects of the prediction errors as well as of accumulated execution errors and thereby substantially increase the success rate. Motivated by this observation, a reward function that measures the coincidence between the ping-pong ball flight trajectory and the racket's movement trajectory is defined in this paper. A stochastic policy over the reward given the striking point is derived for evaluating prospective striking points sampled from the predicted rebound trajectory. After the optimal striking point is provided for the robot, the incoming ball is returned, and, subsequently, the actual reward is recorded. Based on the optimal striking point and the actual reward, the policy over the reward given the striking point is subsequently updated.

The paper is organized as follows. In Section II, our method is described in detail. Evaluations on a real robotic system are given in Section III. Finally, we summarize our contributions and discuss our findings in Section IV.

II. An Approach for Non-Parametric Learning of Optimal Striking Points

The quality of the striking point is the key to success in most common approaches for robot table tennis [1], [2], [3], [4]. In this section, we propose a policy for the determination of striking points based on a database with prior striking points and their reward (Section II-A). After trying a new

striking point, the learning system updates its striking point database (Section II-B).

To accomplish this goal, our system relies on an existing robot table tennis setup (described in Section III-A and shown in Fig. 1) that can generate good striking movements based on a given striking point.

A. Determining the Optimal Striking Point

For determining our optimal striking point, we assume that we have access to a database D with N prior striking points $\mathbf{h}^i$ and associated accumulated rewards R^i. The system also has access to the predicted ball trajectory and can determine a set of n prospective striking points

$$H = \{\mathbf{h}_j = (\mathbf{p}_j, \mathbf{v}_j, t_j) | j = 1, 2, \ldots, n\} \tag{1}$$

by selecting the part of the ball trajectory that lies within the robot's workspace. Here, $\mathbf{p}_j$ and $\mathbf{v}_j$ represent the ball's position and velocity at the striking time t_j.

The database D represents the relationship between the striking point $\mathbf{h}^i$ and the reward R^i. Usually, if the size N of the database D is sufficiently large, we can predict a reward for a given prospective striking point precisely. However, for the high-dimensional problem, the prior data is often not enough at the beginning of an experiment; thus, we follow a stochastic policy approach to predict the reward, where the variance facilitates the exploration of a wide range of prospective striking points.

For $\forall \mathbf{h}_j \in H$, its reward R_j is subject to the stochastic policy

$$\pi(R_j|\mathbf{h}_j) = \mathcal{N}(R_j|\mu(\mathbf{h}_j), \sigma^2(\mathbf{h}_j)), \tag{2}$$

where both the mean $\mu(\cdot)$ and the variance $\sigma^2(\cdot)$ depend on the striking point $\mathbf{h}_j$. By integrating all the data in the prior database D with the weighted average technique, we can obtain the mean

$$\mu(\mathbf{h}_j) = \frac{\sum_{i=1}^{N} f_h(\mathbf{h}_j, \mathbf{h}^i) R^i}{\sum_{i=1}^{N} f_h(\mathbf{h}_j, \mathbf{h}^i)}, \tag{3}$$

where $(\mathbf{h}^i, R^i)$ represents the i-th data in the database D, $f_h(\cdot)$ is defined as

$$f_h(\mathbf{h}_j, \mathbf{h}^i) = \exp\left\{-\frac{1}{2}(\mathbf{h}_j - \mathbf{h}^i)^T \Sigma_h (\mathbf{h}_j - \mathbf{h}^i)\right\} \tag{4}$$

with the weighted diagonal matrix Σ_h.

In fact, if the weighted coefficient $f_h(\mathbf{h}_j, \mathbf{h}^i)$ for $\forall i \in \{1, 2, \ldots, N\}$ in (3) is small, the confidence of the mean $\mu(\cdot)$ is low. Otherwise, the confidence is high. The variance $\sigma^2(\cdot)$ should depend on this confidence: if the confidence is low, the exploration (variance) should be large; otherwise, the exploration should be small. The confidence $c(\cdot)$ of the mean in (3) is defined as

$$c(\mathbf{h}_j) = \max_i f_h(\mathbf{h}_j, \mathbf{h}^i), \forall i \in \{1, 2, \ldots, N\}. \tag{5}$$

Besides this confidence in (5), we also need to consider the size N of the database. When N is small, the variance $\sigma^2(\cdot)$ should be large to ensure that a wide range of experience data is generated. When N is large, the variance $\sigma^2(\cdot)$ should be small since enough experience data ensures a high confidence in (5). Assuming that the size limit of the database D is U_N, the storage ratio will be N/U_N. The variance $\sigma^2(\cdot)$ should satisfy the following two conditions

1) $\sigma^2(\mathbf{h}_j) \propto 1 - c(\mathbf{h}_j)$,
2) $\sigma^2(\mathbf{h}_j) \propto 1 - \frac{N}{U_N}$,

where $N/U_N \in (0, 1]$, $c(\mathbf{h}_j) \in (0, 1]$. A simple choice of the variance $\sigma^2(\cdot)$ is

$$\sigma^2(\mathbf{h}_j) = \gamma \left(1 - \frac{N}{U_N}\right)^2 (1 - c(\mathbf{h}_j))^2, \tag{6}$$

where $\gamma > 0$ is a scalar.

For $\forall \mathbf{h}_j \in H$, we can firstly calculate the mean $\mu(\mathbf{h}_j)$ and the variance $\sigma^2(\mathbf{h}_j)$ based on (3) and (6), and then we can obtain a sample from the Gaussian distribution (2) as the reward R_j. The striking point $\mathbf{h}_k \in H$ satisfying

$$R_k(\mathbf{h}_k) \geq R_m(\mathbf{h}_m), \forall m \in \{1, 2, \ldots, n\} \tag{7}$$

is the optimal one in the set H.

B. Striking Point Database Update

As soon as the optimal striking point $\mathbf{h}^*$ is predicted, the robot will return the incoming ball. We assume that we can receive the actual reward R^* for this optimal striking point $\mathbf{h}^*$ (a reward function will be suggested in Section III-B.1). As the number of trails increases, the size N of the prior database D will continuously increase accordingly. To keep the database's size N reasonable and reduce the data storage burden, we need to update the database D especially when its size N reaches the upper limit U_N. The update mechanism is given below.

1) If $N < U_N$, the new data $(\mathbf{h}^*, R^*)$ is added to the end of the database and becomes $(\mathbf{h}^{(N+1)}, R^{(N+1)})$.

2) If $N = U_N$, we firstly need to search the database D and find the nearest data $(\mathbf{h}^i, R^i)$ to the new data $(\mathbf{h}^*, R^*)$ using

$$f_h(\mathbf{h}^i, \mathbf{h}^*) \geq f_h(\mathbf{h}^j, \mathbf{h}^*), \forall j \in \{1, 2, \ldots, N\}. \tag{8}$$

Then, if

$$\sum_{k=1}^{N} f_h(\mathbf{h}^k, \mathbf{h}^i) - f_h(\mathbf{h}^i, \mathbf{h}^i) \geq \sum_{k=1}^{N} f_h(\mathbf{h}^k, \mathbf{h}^*) - f_h(\mathbf{h}^i, \mathbf{h}^*), \tag{9}$$

the new data $(\mathbf{h}^*, R^*)$ will replace the nearest data $(\mathbf{h}^i, R^i)$; otherwise, this data will not be stored.

C. Complete Algorithm

Assuming that we have predicted the prospective striking points and saved them in the set H. The striking point learning algorithm summarized in Algorithm 1 can determine the optimal striking point $\mathbf{h}^*$. Then, we need to determine the desired Cartesian racket states and the desired joint states of the robot at the striking time. Subsequently, we can generate the robot's movement trajectory in the joint space or Cartesian space. When the robot is moving toward the

Algorithm 1 Learning optimal striking points for the robot

Input: prospective striking points $H = \{\mathbf{h}_j | j = 1, 2, \ldots, n\}$

For $j = 1$ to n

Determine the mean $\mu(\mathbf{h}_j)$

$$\mu(\mathbf{h}_j) = \frac{\sum_{i=1}^{N} f_h(\mathbf{h}_j, \mathbf{h}^i) R^i}{\sum_{i=1}^{N} f_h(\mathbf{h}_j, \mathbf{h}^i)}.$$

Determine the variance $\sigma^2(\mathbf{h}_j)$

$$\sigma^2(\mathbf{h}_j) = \gamma \left(1 - \frac{N}{U_N}\right)^2 (1 - c(\mathbf{h}_j))^2.$$

Draw the reward $R_j(\mathbf{h}_j)$ from a Gaussian distribution

$$R_j(\mathbf{h}_j) \sim \mathcal{N}(R_j | \mu(\mathbf{h}_j), \sigma^2(\mathbf{h}_j)).$$

end for

Output: optimal striking point $\mathbf{h}_k \in H$ satisfying $R_k(\mathbf{h}_k) \geq R_m(\mathbf{h}_m), \forall \mathbf{h}_m \in H$.

incoming ball, we can determine the racket's position and velocity based on the forward kinematics and subsequently calculate the actual reward R^*(see Section III-B.1). After the striking movement is finished, we can update the experience database D following the method in Section II-B.

III. Experimental Setup, Evaluations & Results

In this section, we first describe embedding of the striking point learning algorithm with a robot table tennis player and subsequently discuss its results.

A. Experimental Setup

The experimental setup consists of a robot arm performing the movement, the cameras tracking the ball, a computer processing the images from the cameras, a table tennis trajectory generator that yields the arm movement for a given striking point, and a table with the standard size.

1) Physical Setup: The real robotic system consists of the vision system, the 7-DoF Barrett WAM robot, a trajectory generator and a table. The vision system consists of four Procsilica Gigabit GE640 cameras (200fps) and a computer for image-processing. Barrett WAM arm is a high-speed version of this arm, which can accomplish the fast striking movement. Besides, a standard racket is attached to the end-effector of the robot arm. The trajectory generator yields the arm movement trajectory such that an entire striking movement is finished. The table is a standard one with the length $2.740m$, width $1.525m$ and height $0.760m$.

2) Low-Level Table Tennis Player: We have decomposed the problem of playing robot table tennis into three steps. First, we predict an optimal striking point based on the incoming ball trajectory. Second, we generate a robot arm trajectory which is being executed using an inverse dynamics controller. Third, we calculate the actual reward and update the prior database. The more detailed explanation is given as follows. The ball's current state is estimated by the second-order polynomial fitting method [4]. The iterative ball-flight model [4] and the linear rebound model [11] predict the prospective striking points within a proper domain. The learning algorithm described in Algorithm 1 selects the optimal striking point. The inverse kinematics method [11] determines the desired joint states (the striking states) at the striking time, and, subsequently, the fifth-order polynomial spline interpolation generates the joint movement trajectories, where all the joints move from the initial states to the striking states and then go back to the initial states. The desired robot arm trajectory is executed using an inverse dynamics controller. After the striking movement is finished, the actual reward is determined by the method in Section III-B.1. Finally, the learning algorithm updates the prior database with the optimal striking point and the actual reward (Section II-B).

B. Evaluations & Results

Based on the human insights, a reward function is defined. The algorithm of learning optimal striking points was evaluated in the described robotic system shown in Fig. 1.

1) Reward Function for Evaluations : The overall performance of the ping-pong playing robot not only depends on every single technique, such as vision measurement, inverse kinematics and so on, but also the coordination of these techniques. Learning optimal striking points can be seen as a kind of coordination of these existing methods with the purpose of improving the overall performance of the robot. When the robot prepares to return the incoming ball, it will have a higher probability of success if the racket's movement trajectory has the large coincidence with the ball flight trajectory around the striking time.

To measure the coincidence between the ball's trajectory and the racket's trajectory, both the ball's state (position and velocity) and the racket's state (position and velocity) are considered in the reward function. The reward function includes the position reward R_p

$$R_p = \frac{\sum_{v_{ry}(t)<0} f_p(\mathbf{p}_b(t), \mathbf{p}_r(t)) w(t, t_h)}{\sum_{v_{ry}(t)<0} w(t, t_h)} \tag{10}$$

and the velocity reward R_v

$$R_v = \frac{\sum_{v_{ry}(t)<0} f_v(\mathbf{v}_b(t), \mathbf{v}_r(t)) w(t, t_h)}{\sum_{v_{ry}(t)<0} w(t, t_h)} \tag{11}$$

with the predicted striking time t_h. The Y direction is parallel to the long side of the table. $f_p(\cdot)$ is defined as

$$f_p(\mathbf{p}_b, \mathbf{p}_r) = \exp\left\{-\frac{1}{2}(\mathbf{p}_b - \mathbf{p}_r)^T \Sigma_p (\mathbf{p}_b - \mathbf{p}_r)\right\} \tag{12}$$

with the weighted diagonal matrix Σ_p; it represents the distance measurement between the ball's position $\mathbf{p}_b = [p_{bx}, p_{by}, p_{bz}]^T$ and the racket's position $\mathbf{p}_r = [p_{rx}, p_{ry}, p_{rz}]^T$. $f_v(\cdot)$ is defined as

$$f_v(\mathbf{v}_b, \mathbf{v}_r) = -\frac{\mathbf{v}_b^T \mathbf{v}_r}{\sqrt{(\mathbf{v}_b^T \mathbf{v}_b)(\mathbf{v}_r^T \mathbf{v}_r)}}; \tag{13}$$

it reflects the angle between the ball's flight velocity $\mathbf{v}_b = [v_{bx}, v_{by}, v_{bz}]^T$ and the racket's movement velocity $\mathbf{v}_r = [v_{rx}, v_{ry}, v_{rz}]^T$. $w(\cdot)$ is defined as

$$w(t, t_h) = \exp\{-c_w |t - t_h|\} \tag{14}$$

with a scalar $c_w > 0$; it ensures that the immediate reward around the striking time t_h has more large contribution to the final accumulated reward. In both (10) and (11), R_p and R_v are subject to the constrain $v_{ry} < 0$. This constrain means that we only calculate the rewards when the racket is moving towards the incoming ball.

Combining the position reward R_p and the velocity reward R_v, the final reward R is

$$R = \alpha R_p + \beta R_v, \tag{15}$$

where $\alpha > 0$ and $\beta > 0$ are scalars. Assuming that the optimal striking point $\mathbf{h}^* = (\mathbf{p}^*, \mathbf{v}^*, t^*)$ is predicted, and, subsequently, the robot returns the incoming ball, the actual reward R^* can be determined by (15). Due to high-dimensional (7-D) parameters in the striking point $\mathbf{h}^*$, we update the prior database D with the striking position $\mathbf{p}^*$ and the reward R^* instead. In this case, the distance measurement function (4) will be identical as the function (12).

2) Experimental Results: In the real robotic system, we used the ball launcher to send balls towards the forehand of the robot. The striking point learning algorithm was run 3 times, where each run had 120 trials. The size limit of the database D was set as $U_N = 100$ for each run. The prior database D was empty at the beginning of each run, i.e. $N = 0$, we simply set the estimated rewards for prospective strike points as 1. Subsequently, the optimal striking position and associated accumulated reward were saved, the database D was updated accordingly and its size became $N = 1$. From then on, the learning algorithm worked normally as described in Algorithm 1.

Initially, the learning algorithm explored the prospective striking points in a wide range. As more and more prior data was collected, the exploration decreased. We calculate the average values of every 6 rewards, thus each run has 20 average rewards. As shown in Fig. 2, the curve represents the mean values over 3 runs, the error-bars represent the standard deviations. It can be seen that the overall reward increases as the experiment progresses. Hence, the learning algorithm is able to select the striking points associated with higher rewards.

IV. Conclusion

To improve the overall performance of the ping-pong playing robot, an algorithm for learning optimal striking points is proposed. For evaluating the prospective striking points, a stochastic policy over the reward is formulated. This policy depends on the prior database that consists of the striking point and associated accumulated reward. Following the human insights, we suggest a reward function that measures the coincidence between the ball flight trajectory and the racket's movement trajectory.

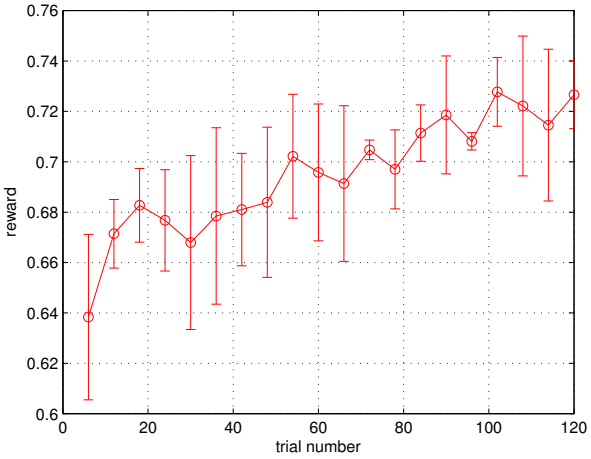

Fig. 2: The figure shows the performance of the striking point learning algorithm over 3 runs. It can be seen that the overall accumulated reward increases with the experiment proceeding.

The coordination of related methods in the ping-pong robotic system is important. Besides selecting the optimal striking point, some other topics need further study, such as determination of desired joint states at the striking time, and robot trajectory generation. If related methods not only perform well independently, but also coordinate well with each other, the overall performance of the ping-pong playing robot will have a significant improvement.

References

[1] K. Mülling, J. Kober, O. Krömer and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *International Journal of Robotics Research*, vol. 32, no. 3, pp. 263-279, 2013.

[2] Z. Yu, Y. Liu, Q. Huang, X. Chen, W. Zhang and J. Li, etc, "Design of a humanoid ping-pong player robot with redundant joints," in *Proc. International Conference on Robotics and Biomimetics,* Shenzhen, China, 2013, pp. 911-916.

[3] H. Li, H, Wu, L. Lou, K. Khnlenz and O. Ravn, "Ping-pong robotics with high-speed vision system," in *Proc. International Conference on Control, Automation, Robotics & Vision*, Guangzhou, China, 2012, pp. 106-111.

[4] Z. Zhang, D. Xu and M. Tan, "Visual measurement and prediction of ball trajectory for table tennis robot," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 12, pp. 3195-3205, 2010.

[5] R. L. Andersson, "A robot ping-pong player: experiment in real-time intelligent control," Cambridge, MA, USA: MIT Press, 1988.

[6] L. Acosta, J. J. Rodrigo, J. A. Mndez, G. N. Marichal and M. Sigut, "Ping-pong player prototype," *IEEE Robotics & Automation Magazine*, vol. 10, pp. 44-52, 2003.

[7] Y. Zhang, W. Wei, D. Yuan and C. Zhong, "A tracking and predicting scheme for ping pong robot," *Journal of Zhejiang University-Science C*, vol. 12, no. 2, pp. 110-115, 2011.

[8] Y. Huang, D. Xu, M. Tan and H. Su, "Trajectory prediction of spinning ball for ping-pong player robot," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, USA, 2011, pp. 3434-3439.

[9] L. Sun, J. Liu, Y. Wang, L. Zhou, Q. Yang and S. He, "Ball's flight trajectory prediction for table tennis game by humanoid robot," in *Proc. International Conference on Robotics and Biomimetics*, Guilin, China, 2009, pp. 2379-2384.

[10] J. Nonomura, A. Nakashima and Y. Hayakawa, "Analysis of effects of rebounds and aerodynamics for trajectory of table tennis ball," in *Proc*. *SICE Annual Conference*, Taipei, Taiwan, 2010, pp.1567-1572.
[11] K. Mülling, J. Kober and J. Peters, "A biomimetic approach to robot table tennis," *Adaptive Behavior*, pp. 359-376, 2011.
[12] M. Matsushima, T. Hashimoto, M. Takeuchi and F. Miyazaki, "A learning approach to robotic table tennis," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 767-771, 2005.
[13] K. Mülling, J. Kober and J. Peters, "Learning table tennis with a mixture of motor primitives," in *Proc*. *IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, USA, 2010, pp. 411-416.
[14] Z. Wang, K. Mülling, M. P. Deisenroth, H. B. Amor, D. Vogt, B. Schölkopf and J. Peters, "Probabilistic movement modeling for intention inference in human-robot interaction," *International Journal of Robotics Research*, vol. 32, no. 7, pp. 841-858, 2013.
[15] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real-time robot learning," *Applied Intelligence*, pp. 49-60, 2002.
[16] C. E. Rasmussen and C. K. I. Williams, "Gaussian processes for machine learning," The MIT Press, 2006.
[17] D. N. Tuong, M. Seeger, and J. Peters, "Local Gaussian processes regression for real-time model-based robot control," in *Proc*. *International Conference on Intelligent Robots and Systems*, Nice, France, 2008, pp. 380-385.
[18] Y. Huang, D. Xu, M. Tan and H. Su, "Adding active learning to LWR for ping-pong playing robot," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1489-1494, 2013.
[19] F. Miyazaki, M. Takeuchi, M. Matsushima, T. Kusano and T. Hashimoto, "Realization of the table tennis task based on virtual targets," in *Proc*. *IEEE International Conference on Robotics & Automation*, Washington, DC, USA, 2002, pp. 3844-3849.
[20] P. Yang, D. Xu, H. Wang and Z. Zhang, "Control system design for a 5-DOF table tennis robot," in *Proc*. *International Conference on Control, Automation, Robotics and Vision*, Singapore, 2010, pp. 1731-1735.
[21] B. Zhang, R. Xiong and J. Wu, "Kinematics analysis of a novel 7-DOF humanoid manipulator for table tennis," in *Proc*. *International Conference on Electronics, Communications and Control*, Ningbo, China, 2011, pp.1524-1528.
[22] J. Kober, E. Oztop and J. Peters, "Reinforcement learning to adjust robot movements to new situations," *Robotics: Science and Systems*, pp. 33-40, 2010.
[23] J. Kober, K. Mülling, O. Kromer, C. H. Lampert, B. Schölkopf and J. Peters, "Movement templates for learning of hitting and batting," in *Proc*. *IEEE International Conference on Robotics and Automation,* Anchorage, USA, 2010, pp. 853-858.
[24] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no.2, pp. 328-373, 2013.
[25] Y. Huang, D. Xu and M. Tan, "A parallel fuzzy learning approach to determine the hitting point for ping-pong playing robot," *International Journal of Innovative Computing Information and Control*, vol. 9, no. 10, pp. 4181-4192, 2013.
[26] H. Su, D. Xu, G. Chen and Z. Fang, "Striking position selection based on two-step multi-purpose fuzzy decision method for robotic table tennis," *Control Theory & Applications*, vol. 30, no.5, pp. 597-603, 2013.
[27] S. Schaal, "The SL simulation and real-time control software package," University of Southern California, 2006.