2016 IEEE-RAS 16th International Conference on
Humanoid Robots (Humanoids)
Cancun, Mexico, Nov 15-17, 2016

Jointly Learning Trajectory Generation and Hitting Point Prediction in Robot Table Tennis

Yanlong Huang[1], Dieter Büchler[1], Okan Koç[1], Bernhard Schölkopf[1], and Jan Peters[1,2]

***Abstract*— This paper proposes a combined learning framework for a table tennis robot. In a typical robot table tennis setup, a single striking point is predicted for the robot on the basis of the ball's initial state. Subsequently, the desired Cartesian racket state and the desired joint states at the striking time are determined. Finally, robot joint trajectories are generated. Instead of predicting a single striking point, we propose to construct a ball trajectory prediction map, which predicts the ball's entire rebound trajectory using the ball's initial state. We construct as well a robot trajectory generation map, which predicts the robot joint movement pattern and the movement duration using the Cartesian racket trajectories without the need of inverse kinematics, where a correlation function is used to adapt these joint movement parameters according to the ball flight trajectory. With joint movement parameters, we can directly generate joint trajectories. Additionally, we introduce a reinforcement learning approach to modify robot joint trajectories such that the robot can return balls well. We validate this new framework in both the simulated and the real robotic systems and illustrate that a seven degree-of-freedom Barrett WAM robot performs well.**

I. Introduction

The main goal of a robot table tennis is to generate proper robot trajectories, so that a racket attached to the end-effector of the robot can return balls. A table tennis task can be decomposed into two sub-tasks: the ball trajectory prediction and the robot trajectory generation. In the previous publications for the ball trajectory prediction, a predefined virtual striking plane is introduced. For the physical model methods [1], [2], [3], [4], [5], the intersection point between the predicted ball rebound trajectory and the predefined virtual plane is considered as the striking point. For the regression methods [6], [7], the striking point at the predefined virtual plane is predicted directly. One disadvantage of the virtual plane method is that the predefined plane is fixed, which might lead to an inappropriate striking point for the robot. Instead of using a virtual plane, some works [8], [9] propose to evaluate perspective striking points, then select an optimal point from them. Since all these methods focus on the prediction of a single striking point, the precision of the single striking point influences the robot performance greatly. However, it is difficult to predict striking points precisely due to uncertainties in the visual ball measurement and modelling errors in the ball trajectory prediction.

[1]Yanlong Huang, Dieter Büchler, Okan Koç, 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

Fig. 1: The 7-DoF robot table tennis setup.

Another issue in a traditional robot table tennis is that inverse kinematics and the robot trajectory generation are dealt with separately. Given a predicted striking point, the desired joint states at the striking time are first determined using inverse kinematics, so that the racket can strike the ball with the right velocity as well as the right orientation. Subsequently, joint trajectories going through their corresponding desired points are generated. The separation of inverse kinematics and the robot trajectory generation might lead to improper robot trajectories. For example, if the desired joint states in a redundant system result in an extremely difficult robot posture, the corresponding joint trajectories can be undesired or even lead to dangerous configuration. At this point, a natural question is: can we combine both inverse kinematics and trajectory generation into a single procedure rather than the separate treatments?

In this paper, we attempt to formulate the robot table tennis task from a different perspective. We consider the robot movement trajectory as a response to the ball flight trajectory, and try to generate the robot trajectory directly according to the ball trajectory. On the basis of the ball's initial state, we predict the entire ball rebound trajectory rather than a single striking point. Furthermore, we propose a function that measures the correlation between a ball flight trajectory and a racket movement trajectory. With this correlation function, we directly predict the robot joint movement pattern and movement duration, and subsequently, generate the robot joint trajectories with these movement parameters. Besides, we introduce a reinforcement learning (RL) method to modify the racket orientation such that the robot can return balls well. The advantage of our approach is that we can learn robot table tennis task from a data-driven perspective without inverse kinematics and physical models.

This paper is organized as follows. In Section II, an overview of our learning framework is illustrated. In Sec-

978-1-5090-4718-5/16/$31.00 ©2016 IEEE

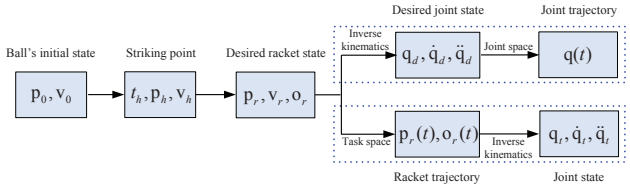


Fig. 2: A typical framework for a robot table tennis. A striking point is first predicted based on the initial ball state. Then, the desired Cartesian racket state at the striking time is determined. The robot trajectory is finally generated either in joint space or in task space.

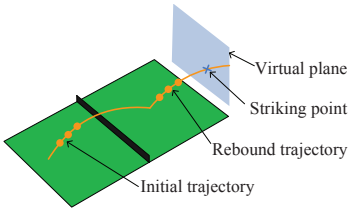


Fig. 3: Ball trajectory prediction with a virtual striking plane. On the basis of the ball's initial trajectory, the ball rebound trajectory can be predicted. Subsequently, the intersection point of the predicted rebound trajectory and a predefined virtual plane is determined as the striking point.

tion III and Section IV, the ball rebound trajectory prediction and the robot joint trajectory generation are discussed, respectively. The adjustment of robot trajectories is explained in Section V. Evaluation results in both the simulated and the real robotic systems are given in Section VI. Finally, concluding remarks are made in Section VII.

II. An Overview of the Combined Learning Framwork

In a traditional framework for a robot table tennis (Fig. 2), the ball's initial state (position and velocity) $\mathbf{s}_0 = (\mathbf{p}_0, \mathbf{v}_0)$ is first estimated from its initial flight trajectory. Then, a striking point (striking time, ball's position and velocity) $\mathbf{s}_h = (t_h, \mathbf{p}_h, \mathbf{v}_h)$ at the virtual striking plane is predicted so that the robot is provided with enough reaction time (Fig. 3). Subsequently, the desired racket state (position, velocity and orientation) $(\mathbf{p}_r, \mathbf{v}_r, \mathbf{o}_r)$ is determined. The robot trajectory can be generated either in joint space or in task space. In joint space, inverse kinematics is used to determine the desired joint state $(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)$. Finally, the entire joint trajectory $\mathbf{q}(t)$ is generated. In task space, the racket trajectory $\mathbf{p}_r(t)$ and the orientation curve $\mathbf{o}_r(t)$ are generated. Then, the corresponding joint states $(\mathbf{q}_t, \dot{\mathbf{q}}_t, \ddot{\mathbf{q}}_t)$ are obtained with inverse kinematics.

From the perspective of spatio-temporal trajectories, a robot table tennis task can be formulated as the generation of proper robot joint trajectories along with ball flight trajectories. Thus, we attempt to transform the traditional framework into a combined learning framework (Fig. 4), which consists of a regression part and a RL part. In the regression part, the *ball map* represents the relationship between the ball's initial state $\mathbf{s}_0$ and the ball's entire rebound trajectory $\mathbf{p}_b(t)$, i.e., a map from $\mathbf{s}_0$ to $\mathbf{p}_b(t)$; The *robot map* represents the relationship between the Cartesian racket trajectory $\mathbf{p}_r(t)$ and the robot joint trajectory $\mathbf{q}(t)$, i.e., a map from $\mathbf{p}_r(t)$ to $\mathbf{q}(t)$.

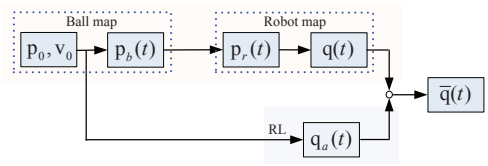


Fig. 4: The combined learning framework for a robot table tennis. The *ball map* predicts the ball rebound trajectory using the ball's initial state. The *robot map* predicts the initial joint trajectories on the basis of the Cartesian racket trajectories. The RL part determines the joint adjustment trajectories. The final robot joint trajectories are the sum of the initial joint trajectories and the adjustment trajectories.

Given a collection of measured ball positions, the ball's initial state $\mathbf{s}_0$ can be first estimated and subsequently the ball rebound trajectory $\mathbf{p}_b(t)$ can be predicted via the *ball map*. On the basis of a correlation measurement between the ball rebound trajectory $\mathbf{p}_b(t)$ and the racket trajectory $\mathbf{p}_r(t)$, the robot joint trajectory $\mathbf{q}(t)$ can be generated via the *robot map*, so that with joint trajectory $\mathbf{q}(t)$ the racket (attached to the end-effector of the robot) can strike incoming balls.

In order to return balls well, we need to take the racket orientation $\mathbf{o}_r(t)$ into account. Since the racket orientation is also determined by the robot joint trajectories, we can directly modify joint trajectories (e.g. the last two joints in our robotic system). Here, the explicit form of the racket orientation is not involved. In the RL part, we use a RL method to learn joint adjustment trajectory $\mathbf{q}_a(t)$ on the basis of the evaluation of landing points. The final robot joint trajectory $\bar{\mathbf{q}}(t)$ is the sum of the robot trajectory $\mathbf{q}(t)$ and the adjustment trajectory $\mathbf{q}_a(t)$, so that the appropriate racket orientation can be achieved. With the final robot trajectory $\bar{\mathbf{q}}(t)$, the racket is not only capable of striking balls but also capable of returning balls well.

III. Ball Trajectory Prediction

Due to physical limitations of the robot, sufficient reaction time is necessary to strike balls. We discuss the ball trajectory prediction in this section, where the ball's entire rebound trajectory is predicted on the basis of the ball's initial state. In Section III-A, we show the experience data in the *ball map* followed by the ball trajectory prediction in Section III-B.

A. Experience Data in Ball Trajectory Prediction

As discussed in Section II, the ball trajectory prediction can be realized by constructing the *ball map*: from the ball's initial state $\mathbf{s}_0$ to the ball rebound trajectory $\mathbf{p}_b(t)$. With a stereo vision system, we can measure the initial flight trajectory of the ball. By applying polynomial fitting [2] to this initial trajectory, we can estimate the ball's initial state $\mathbf{s}_0$. Since the ball rebound trajectory $\mathbf{p}_b(t)$ can be

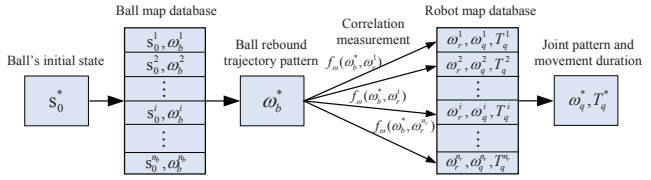


Fig. 5: The diagram of the regression learning. The ball rebound trajectory pattern is predicted via the *ball map*. Subsequently, with the correlation measurements, the robot joint movement pattern and movement duration are predicted via the *robot map*.

approximated by a parabolic curve, we fit it with basis functions $\mathbf{\Phi}_b(t) = [t^2 \ t \ 1]$, i.e.,

$$\mathbf{p}_b(t) = \mathbf{\Phi}_b(t)\boldsymbol{\omega}_b, \tag{1}$$

and then extract its pattern $\boldsymbol{\omega}_b$ using least squares method

$$\boldsymbol{\omega}_b = (\mathbf{\Phi}_b^T\mathbf{\Phi}_b)^{-1}\mathbf{\Phi}_b^T\mathbf{p}_b. \tag{2}$$

Thus, given an entire ball flight trajectory that consists of an initial trajectory and a rebound trajectory, we can extract both the ball's initial state $\mathbf{s}_0$ and the rebound trajectory pattern $\boldsymbol{\omega}_b$, and subsequently update the *ball map* with the experience data point $(\mathbf{s}_0, \boldsymbol{\omega}_b)$.

B. Ball Trajectory Prediction

With the experience data $(\mathbf{s}_0, \boldsymbol{\omega}_b)$, we can construct the *ball map* with a regression method, such as locally weighted regression (LWR) [10] and Gaussian process regression (GPR) [11]. An illustration of the ball map is given in Fig. 5, where we denote the i-th experience data in the prior database $\mathbf{D}_b$ as $\mathbf{D}_b^i = (\mathbf{s}_0^i, \boldsymbol{\omega}_b^i)$.

Assuming that the ball trajectory pattern $\boldsymbol{\omega}_b^*$ for a new state $\mathbf{s}_0^*$ is predicted, we can continue to predict the ball rebound trajectory $\mathbf{p}_b(t)$ using (1), i.e.,

$$\mathbf{p}_b^*(t) = \mathbf{\Phi}_b(t)\boldsymbol{\omega}_b^*. \tag{3}$$

Thus, the entire ball rebound trajectory $\mathbf{p}_b^*(t)$ for the new state $\mathbf{s}_0^*$ is predicted. After the ball rebounds on the table, we can fit the ball rebound trajectory and extract its pattern. Accordingly, we can update the database $\mathbf{D}_b$ and its size n_b. As the size of the database $\mathbf{D}_b$ increases, the ball trajectory prediction with a regression method will be improved.

IV. Robot Joint Trajectory Generation

To strike an incoming ball, we need to generate proper robot joint trajectories, so that the Cartesian racket trajectory intersects with the ball rebound trajectory. In this section, we discuss the robot trajectory generation on the basis of the Cartesian racket trajectory. First, the experience data in the *robot map* is analysed in Section IV-A. Then, in Section IV-B, a correlation measurement between the ball trajectory pattern and the racket trajectory pattern is formulated, that is henceforth used for the robot trajectory generation.

A. Experience Data in Robot Joint Trajectory Generation

The robot joint trajectory generation is essentially the *robot map*: from the Cartesian racket trajectory $\mathbf{p}_r(t)$ to the robot joint trajectory $\mathbf{q}(t)$. Denote the number of joints as N, for the k-th joint, $k \in \{1, 2, \ldots, N\}$, we can fit its trajectory $q_k(t)$ using a dynamical movement primitive (DMP) since the DMP can model human demonstrations well [12]. A DMP consists of a canonical system

$$\tau\dot{x}(t) = -\alpha_x x(t), \tag{4}$$

and a transform system

$$\begin{aligned} \tau\dot{z}_k(t) &= \alpha_q\left(\beta_q\left(g_k - q_k(t)\right) - z_k(t)\right) + f_k(x) \\ \tau\dot{q}_k(t) &= z_k(t) \end{aligned} \tag{5}$$

with a forcing term

$$f_k(x) = \frac{\sum_{i=1}^n \Phi_{qi}(x)\omega_{ki}}{\sum_{i=1}^n \Phi_{qi}(x)}x, \tag{6}$$

where α_x, α_q and β_q are positive constants; g_k is the goal position of the k-th joint; τ is the joint movement duration; $\Phi_{qi}(x) = \exp(-h_i(x-c_i)^2)$ are basis functions with $h_i > 0$ and $c_i \in [0, 1]$; $\boldsymbol{\omega}_k = [\omega_{k1}, \omega_{k2}, \cdots, \omega_{kn}]^T$ is a movement pattern vector that can be estimated using LWR or least squares method. By sequencing movement patterns of all the joints, we can obtain an extended movement pattern vector $\boldsymbol{\omega}_q = [\boldsymbol{\omega}_1^T, \boldsymbol{\omega}_2^T, \cdots, \boldsymbol{\omega}_N^T]^T$.

For the robot joint trajectory $\mathbf{q}(t)$, with forward kinematics we can compute the Cartesian racket trajectory $\mathbf{p}_r(t)$. Through fitting the racket trajectory with

$$\mathbf{p}_r(t) = \mathbf{\Phi}_r(t)\boldsymbol{\omega}_r, \tag{7}$$

we can extract a racket trajectory pattern

$$\boldsymbol{\omega}_r = (\mathbf{\Phi}_r^T\mathbf{\Phi}_r)^{-1}\mathbf{\Phi}_r^T\mathbf{p}_r, \tag{8}$$

where $\mathbf{\Phi}_r(t) = [t^5 \ t^4 \ t^3 \ t^2 \ t \ 1]$ are basis functions. With the combination of the racket trajectory pattern $\boldsymbol{\omega}_r$, the joint movement pattern $\boldsymbol{\omega}_q$ and joint movement duration T_q, an experience data $(\boldsymbol{\omega}_r, \boldsymbol{\omega}_q, T_q)$ is obtained for the *robot map*.

B. Robot Joint Trajectory Generation

Given an initial ball state $\mathbf{s}_0^*$, we can predict the ball rebound trajectory pattern $\boldsymbol{\omega}_b^*$ via the *ball map*. Notice that the input of the *robot map* is the racket movement pattern $\boldsymbol{\omega}_r$ as shown in Fig. 5. Hence, we need to find a bridge between $\boldsymbol{\omega}_b^*$ and $\boldsymbol{\omega}_r$, so that the joint trajectory pattern $\boldsymbol{\omega}_q$ and the movement duration T_q can be predicted, and subsequently the robot joint trajectories are generated with DMP.

Denote the prior database in the *robot map* as $\mathbf{D}_r$ and its i-th data point as $\mathbf{D}_r^i = (\boldsymbol{\omega}_r^i, \boldsymbol{\omega}_q^i, T_q^i)$. Given the predicted ball trajectory pattern $\boldsymbol{\omega}_b^*$, we can determine the ball rebound trajectory $\mathbf{p}_b^*(t)$ using (3). Similarly, for each prior data $\boldsymbol{\omega}_r^i \in \mathbf{D}_r^i$, we can compute the racket trajectory $\mathbf{p}_r^i(t)$ with (7). Naturally, a necessary condition of striking an incoming ball is that the racket trajectory intersects with the ball flight trajectory. However, it is possible that many racket trajectories intersect with the predicted ball trajectory.

As suggested in [9], if a racket moves along the ball flight trajectory but in the opposite direction, the robot performance will be improved. Hence, we measure the distance between the ball trajectory and the racket trajectory as well as the angle between the ball velocity and the racket velocity. The correlation function between $\boldsymbol{\omega}_b^*$ and $\boldsymbol{\omega}_r^i$ is defined as

$$\begin{aligned} f_\omega(\boldsymbol{\omega}_b^*, \boldsymbol{\omega}_r^i) = & \max_t f_p\left(\mathbf{p}_b^*(t), \mathbf{p}_r^i(t)\right) f_v\left(\mathbf{v}_b^*(t), \mathbf{v}_r^i(t)\right) \\ \text{s.t.} \quad & \mathbf{p}_b^*(t) = \boldsymbol{\Phi}_b(t)\boldsymbol{\omega}_b^* \\ & \mathbf{p}_r^i(t) = \boldsymbol{\Phi}_r(t)\boldsymbol{\omega}_r^i \end{aligned} \tag{9}$$

with a position function

$$f_p\left(\mathbf{p}_b^*, \mathbf{p}_r^i\right) = \exp\left(-\alpha_p\left((\mathbf{p}_b^* - \mathbf{p}_r^i)^T(\mathbf{p}_b^* - \mathbf{p}_r^i)\right)^{\frac{1}{2}}\right), \tag{10}$$

and a velocity function

$$f_v\left(\mathbf{v}_b^*, \mathbf{v}_r^i\right) = -\frac{< \mathbf{v}_b^*, \mathbf{v}_r^i >}{\sqrt{< \mathbf{v}_b^*, \mathbf{v}_b^* >< \mathbf{v}_r^i, \mathbf{v}_r^i >}}, \tag{11}$$

where $f_p\left(\mathbf{p}_b^*, \mathbf{p}_r^i\right)$ is a function of the distance between $\mathbf{p}_b^*$ and $\mathbf{p}_r^i$; α_p is a positive constant; $< \cdot, \cdot >$ is the inner product; $f_v\left(\mathbf{v}_b^*, \mathbf{v}_r^i\right)$ represent the angle between $\mathbf{v}_b^*$ and $\mathbf{v}_r^i$.

It can be seen from (10) that, if the distance between $\mathbf{p}_b^*$ and $\mathbf{p}_r^i$ is small, the position function will be large. In (11), if the direction of the racket velocity is near to the opposite direction of the ball velocity, the velocity function will be close to 1. With the function (9), we can determine correlations between $\boldsymbol{\omega}_b^*$ and $\boldsymbol{\omega}_r^i, i \in \{1, 2, \ldots, n_r\}$. Here, n_r is the size of the database $\mathbf{D}_r$. Through a linear combination of the prior data in $\mathbf{D}_r$, the robot movement pattern is predicted to be

$$\boldsymbol{\omega}_q^* = \frac{\sum_i f_\omega(\boldsymbol{\omega}_b^*, \boldsymbol{\omega}_r^i))\boldsymbol{\omega}_q^i}{\sum_i f_\omega(\boldsymbol{\omega}_b^*, \boldsymbol{\omega}_r^i))}, \tag{12}$$

and the robot movement duration is

$$T_q^* = \frac{\sum_i f_\omega(\boldsymbol{\omega}_b^*, \boldsymbol{\omega}_r^i))T_q^i}{\sum_i f_\omega(\boldsymbol{\omega}_b^*, \boldsymbol{\omega}_r^i))}. \tag{13}$$

According to the joint movement parameters $\boldsymbol{\omega}_q^*$ and T_q^*, we can generate the robot joint trajectories separately using the DMP as given in (4)–(6).

V. Reinforcment Learning of the Joint Adjustment Trajectory

The regression part focuses on striking balls without considering actual landing points after strikes of the racket. In order to return balls well, the appropriate racket orientations are needed. Since the racket orientation is determined by the joint trajectories, we attempt to directly modify joint trajectories such that balls can not only be struck but also be returned well. We enable that by adding the adjustment trajectories to the initial trajectories generated by the regression part (Section V-A) and subsequently learning the optimal parameters of the adjustment trajectories with a RL method (Section V-B).

A. Joint Adjustment Trajectory

Given an initial state $\mathbf{s}_0$ of the ball, we can generate robot joint trajectories $\mathbf{q}(t)$ via the regression part, so that the ball is struck by the racket attached to the end-effector of the robot. In order to return the incoming ball well, an intuitive way is to add minor adjustments to $\mathbf{q}(t)$.

Many functions can be used to generate the adjustment trajectories, such as Gaussian function or polynomials. We write the adjustment function of the k-th joint as $q_a\left(\boldsymbol{\gamma}_k, t\right)$, where $k \in \{1, 2, \ldots, N\}$, $\boldsymbol{\gamma}_k$ represents function parameters to be learned. For example, the parameters of a Gaussian function can be amplitude, center and bandwidth. If $\boldsymbol{\gamma}_k$ is known, the adjustment function $q_a\left(\boldsymbol{\gamma}_k, t\right)$ is obtained. With the integration of an initial joint trajectory $q_k(t)$ and a minor adjustment trajectory $q_a(\boldsymbol{\gamma}_k, t)$, a mixed trajectory $\bar{q}_k(t)$ for the k-th joint becomes

$$\bar{q}_k(t) = q_k(t) + q_a(\boldsymbol{\gamma}_k, t). \tag{14}$$

B. Parameter Learning of the Adjustment Trajectory

In order to generate an appropriate adjustment trajectory for the k-th joint, the parameters $\boldsymbol{\gamma}_k$ of an adjustment function $q_a\left(\boldsymbol{\gamma}_k, t\right)$ should take the initial ball state $\mathbf{s}_0$ as well as the evaluation of actual landing points into account. We can transform learning of parameters $\boldsymbol{\gamma}_k$ into a RL problem, where the *state* is the ball's initial state $\mathbf{s}_0$, the *action* is the parameter vector $\boldsymbol{\gamma}_k$, the *reward* is an evaluation r of the landing point. Assuming that a collection of prior data in the form of $(\mathbf{s}_0^m, \boldsymbol{\gamma}_k^m, r^m)$ is available, where m represents the m-th data point.

Policy search has been studied widely recently [13], and there exists the myriad of approaches to different applications. Unlike many policy search methods with explicit basis functions, the cost-regularized kernel regression (CrKR) [14] uses kernels instead, leading to a non-parametric approach. Thus, we use the CrKR to learn parameters $\boldsymbol{\gamma}_k$ in response to $\mathbf{s}_0$. An optimal parameter $\gamma_{ki} \in \boldsymbol{\gamma}_k$ can be sampled from a Gaussian distribution

$$\gamma_{ki} \sim \mathcal{N}\left(\gamma_{ki}(\mathbf{s}_0), \sigma^2(\mathbf{s}_0)\right) \tag{15}$$

with mean

$$\gamma_{ki}(\mathbf{s}_0) = \mathbf{k}(\mathbf{s}_0)^T(\mathbf{K} + \lambda\mathbf{C})^{-1}\mathbf{T}_{ki} \tag{16}$$

and variance

$$\sigma^2(\mathbf{s}_0) = k(\mathbf{s}_0, \mathbf{s}_0) + \lambda - \mathbf{k}(\mathbf{s}_0)^T(\mathbf{K} + \lambda\mathbf{C})^{-1}\mathbf{k}(\mathbf{s}_0), \tag{17}$$

where $k(\cdot, \cdot)$ is a kernel function; $\mathbf{k}_m = k(\mathbf{s}_0, \mathbf{s}_0^m)$; $\mathbf{K}_{mn} = k(\mathbf{s}_0^m, \mathbf{s}_0^n)$; $\mathbf{C}$ is a diagonal matrix with non-zero elements $\mathbf{C}_{mm} = \frac{1}{r^m}$; λ is a positive constant; $\mathbf{T}_{kim} = \gamma_{ki}^m$. After all parameters of $\boldsymbol{\gamma}_k$ are determined, the adjustment function $q_a(\boldsymbol{\gamma}_k, t)$ is known. Accordingly, the mixed joint trajectory for k-th joint can be generated using (14).

VI. Evaluation Results

In this section, the combined learning method is evaluated in a simulated robotic system (Section VI-A) and a real robotic system (Section VI-B). Both of the simulated and

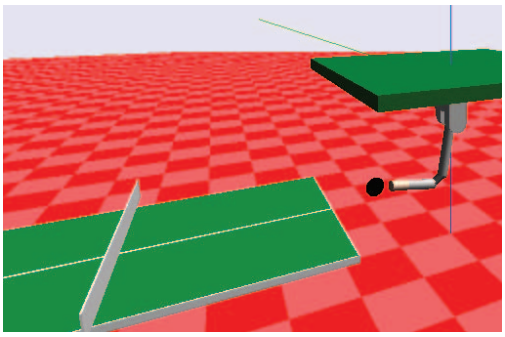
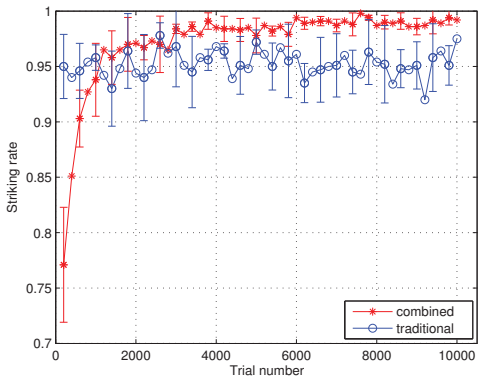
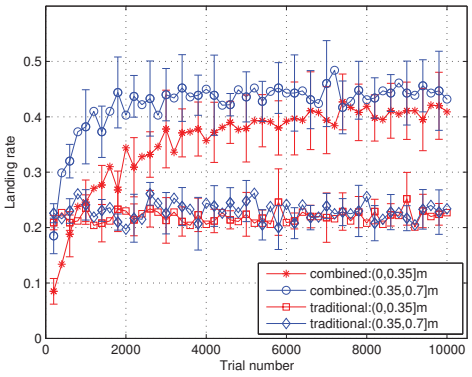


Fig. 6: Simulation environment in *SL*.



(a) Striking rates



(b) Landing rates

Fig. 7: Statistical results of the combined learning method and a traditional method in the simulated robotic system. The curves represent mean values while error bars represent standard deviations. (a): striking rates; (b): landing rates in different regions (defined with landing errors).

the real robotic systems consist of a seven degree-of-freedom (DoF) robot arm, a racket attached to the end-effector of the robot arm, a table and a ping-pong ball. The racket, the table and the ball have the standard sizes. The initial state of the robot is at the middle position of the table.

A. Evaluations on the Simulated Robotic System

The simulated robotic system (Fig. 6) is developed in the framework of *SL* [15]. The ball's start state is generated randomly towards the forehand side of the robot. The ball's subsequent state is obtained using an iterative flight model [2] and a linear rebound model [1]. Before running the combined learning method, we initialize the *ball map* and the *robot map* with 500 prior data separately, where the *robot map* is built with human demonstrations.

We run the combined learning method 5 times and for each run we have 10000 trials. Here, both the regression part (*ball map* and *robot map*) and the RL part will update online. The adjustment function in (14) is modelled by a fifth-order polynomial with amplitude and via point to be learned, where boundary conditions (position, velocity and acceleration) are zeros. The desired landing point is at the center of the human opponent side of the table. The reward is $r = \exp(-\alpha_r d^2)$, where $\alpha_r > 0$ is a scalar; d represents a landing error between an actual landing point and a desired landing point.

As a comparison, we also run a traditional method in Fig. 2, which predicts a strike point on the virtual striking plane using a flight model [2] and a linear rebound model [1], then determines the desired racket state with a simplified ball-racket contact model and the desired joint state with inverse kinematics, finally generates robot trajectory in joint space using fifth-order polynomials [3], [8]. We calculate striking rates every 200 trials; Meanwhile, we can statistically analyse actual landing points according to landing errors d. Evaluation results are shown in Fig. 7. Since the virtual striking plane is fixed in the traditional method, sometimes the intersection point of the predicted ball trajectory and the virtual striking plane is inappropriate for the robot (e.g. exceed the robot's workspace), which leads to a lower striking rate (95%). For the combined learning method, the striking performance is improved as both the *ball map* and the *robot map* are updated online. Besides, the combined method uses the correlation measurement to compare the ball flight trajectory and the racket trajectory so that the restriction of striking points is alleviated. Hence, a higher striking rate (99%) is achieved.

For the traditional method, both the landing rate-1 ($d \leq 0.35m$) and the landing rate-2 ($0.35m < d \leq 0.7m$) are around 22%. For the combined method, after around 6000 trials the landing rate-1 is nearly 41% while the landing rate-2 is nearly 44%. For the traditional method, it is difficult to determine a proper racket state at the striking time on the basis of an analytical model, since both the ball-racket contact model and the ball flight model are non-linear. Besides, the fixed virtual striking plane increases the difficulty of returning balls. For some inappropriate striking points, the robot can only strike balls, but cannot return them well. A benefit of the combined approach is that physical models are avoided. On the basis of the feedback evaluations of actual landing points, the RL method can modify joint trajectories such that balls are returned well.

B. Evaluations on the Real Robotic System

In the real robotic system (Fig. 1), a 7-DoF Barrett WAM robot is implemented. A launcher is used to send

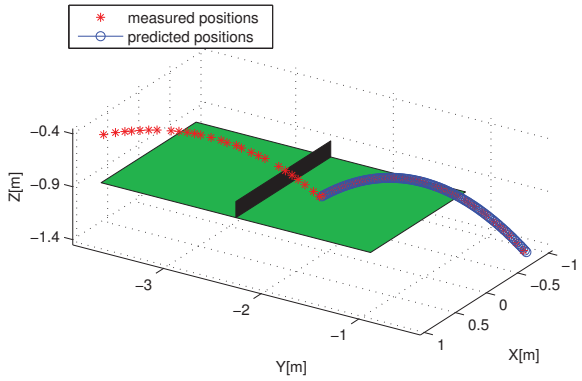


Fig. 8: The ball trajectory prediction via the ball map in a real robotic system is shown, where the entire ball rebound trajectory is predicted.

balls towards the forehand side of the robot. A vision system consisting of four Procsilica Gigabit GE640 cameras (200fps) is applied to the measurement of the ball's 3-D positions. The *ball map* and the *robot map* are initialized with 500 prior data, respectively. The ball trajectory prediction via the *ball map* is illustrated in Fig. 8, where the ball's initial state is estimated using the polynomial fitting method [2]. It can be seen that the entire ball rebound trajectory is predicted. Moreover, the predicted trajectory coincides well with the ball's actual trajectory. As more ball trajectories are observed, the prediction via the *ball map* will improve.

We train the combined method with 90 trials, then we evaluate the combined method with 25 trials: the striking rate is 92%. Subsequently, we continue to train the combined approach with 90 more trials and then evaluate it with another 25 trials: the striking rate is 96%. Here, the ball trajectories for evaluations have smaller variance compared with that in the simulated system. We also run the traditional method and evaluate it with 50 trials, the striking rate is 82%.

For the traditional method with physical models [1], [2], it is difficult to predict the single striking point precisely due to modelling errors in the ball flight model and the ball rebound model. Especially, the linear rebound model can not characterize the rebound phenomenon well. For some ball trajectories, the virtual striking plane [1], [3] leads to improper striking points for the robot, and it is very hard to hit balls at these points.

Instead of physical models, the combined learning method predicts the entire ball rebound trajectory from a data-driven perspective. With the correlation measurement, the combined method has a sufficient comparison between the entire ball rebound trajectory and the entire racket trajectory. Thus, more appropriate joint movement parameters are obtained.

VII. Conclusions

This paper contributes to providing a different perspective to treat robot table tennis, where a novel learning framework is illustrated. This framework consists of a regression part and a RL part. The regression part is able to generate robot joint trajectories so as to strike incoming balls. The RL part is able to modify joint trajectories so as to return balls well. Within this framework, the robot table tennis task is learned from a data-driven perspective without inverse kinematics and physical models.

Due to the difficulties in visual measurement of rotational velocities, only the case of non-spinning balls is considered in this paper. In the future, we will identify the rotational information and incorporate spinning balls into our framework. Besides, a possible extension of the current work is to take the ball's 2-D coordinates in images as the inputs and the motor commands for the robot as the outputs. The data-driven approach proposed in this paper can also be applied to other robotic systems, such as the badminton robots and ball-catching robots. Instead of explicit physical models, the data-driven approach provides an alternative way to the robotics systems with complicated dynamics.

References

[1] K. Mülling, J. Kober and J. Peters, "A biomimetic approach to robot table tennis," *Adaptive Behavior*, pp. 359-376, 2011.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Y. Huang, B. Schlkopf, J. Peters, "Learning optimal striking points for a ping-pong playing robot," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems,* Hamburg, Germany, 2015, pp. 4587-4592.

[10] C. G. Atkeson, A. W. Moore and S. Schaal, "Locally weighted learning," *Atificial Intelligence Review*, vol. 11, pp. 11-73, 1997.

[11] C. E. Rasmussen and C. K. I. Williams, "Gaussian processes for machine learning," The MIT Press, 2006.

[12] 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.

[13] M. P. Deisenroth, G. Neumann and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, nos. 1-2, pp. 1-142, 2011.

[14] J. Kober, E. Oztop and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *Proc. International Joint Conference on Artificial Intelligence*, Barcelona, Spain, 2011, pp. 2650-2655.

[15] S. Schaal, "The SL simulation and real-time control software package," University of Southern California, 2006.