Robotica (2024), **42**, pp. 3867–3887
doi:10.1017/S0263574724001693

RESEARCH ARTICLE

Table tennis ball landing control in a robotic system by cameras

Hsien-I Lin[1] and Cyuan-Fan Syu[2]

[1]Institute of Electrical and Control Engineering, National Yang Ming Chiao Tung University, Hsinchu, Taiwan
[2]Graduate Institute of Automation Technology, National Taipei University of Technology, Taipei, Taiwan
Corresponding author: Hsien-I Lin; Email: sofin@nycu.edu.tw

Received: 22 January 2024; **Revised:** 31 July 2024; **Accepted:** 4 September 2024; **First published online:** 14 October 2024

Keywords: table tennis robot; landing position control; cascade neural network; locally weighted regression

Abstract
Controlling the landing position of a spinning ball is difficult when using a table tennis robot. A complete physical model requires the factoring in of aerodynamic elements and object collisions, and inaccurate environmental coefficients would increase the landing position error. This study proposed a landing position control method based on a cascade neural network (CNN) that consists of forward and recurrent neural networks (RNNs). The forward NNs are used to estimate the velocity of the outgoing ball according to the velocity and acceleration of the incoming ball captured by cameras and the desired velocity of the outgoing ball. The RNN is employed to reverse-predict ball displacement based on the state of the incoming ball, desired landing point, and ball flight duration. The experiments verified that the method proposed in this study achieved control of differently spinning balls more effectively than the locally weighted regression (LWR)-based model did. The success rate of the CNN at two of six desired landing points was 25.9% and 32.9% higher, respectively, compared with use of the LWR-based model.

1. Introduction

The table tennis robot is a type of entertainment robot that has sparked the interest of numerous scholars worldwide. These robots are applied to machine vision, trajectory planning, and machine learning, all of which must be integrated into a set of systems, making these robots of considerable value in academic research. Acosta *et al.* [1] developed a complete robotic table tennis system that uses a simple robotic arm to achieve easy stroke movements. However, because of its small size, the robotic arm can hit table tennis balls to only a limited distance. Mulling *et al.* [11] used an anthropomorphic robotic arm and a machine learning method to imitate the arm swinging movement. Miyazaki *et al.* [9] and Yang *et al.* [19] proposed using a slide rail machine to control the stroke movement and angle of a table tennis paddle; the proposed method could return the ball within a limited timeframe. Regarding the method of predicting whether the robot can hit a ball, scholars have employed various physical models for estimating the force direction and state of the ball. Physical models involve gravity, air resistance, the Magnus effect, and elastic collision, all of which must be accurately calculated to achieve the desired outcome. The velocity of a moving ball approximately ranges from 4 to 20 m/s [18]; therefore, a high-performance computer vision system [4, 17] and camera array [6] are required to quickly calculate the high-sample rate of a ball. Subsequently, various equations for physical objects are used to deduce the required hitting point. Such vision systems are, however, expensive and impractical.

Ball landing positions are calculated using either physical models or machine learning. Early studies typically employed physical models of table tennis robots. Because of technical difficulties, a ball's rotation velocity is difficult to measure and therefore ignored in most studies, resulting in models that do not truly reflect the actual movement of a table tennis ball. Consequently, the trajectory of a spinning ball cannot be effectively predicted, and the landing point cannot be properly controlled. Following the recent

© The Author(s), 2024. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

development of image sensors with greater sampling frequency and novel operation, the spin of the ball has been considered in the physical models of table tennis ball movements, which has substantially improved trajectory prediction. Commonly, trajectory prediction employs image processing or tracking techniques on an object moving, which can process data efficiently and in real-time, as demonstrated by Lin *et al*. [8] and Zhi *et al*. [21]. These methods utilize advanced algorithms to analyze and predict the ball's path based on its observed motion, ensuring accurate and timely predictions. Considering ball spin situations in the tennis ball game, refs. [2, 14, 16] analyzed the phenomenon of a table tennis ball hitting the table and then a racket and established the table rebound model (TRM) and racket rebound model (RRM). In ref. [13], the control of landing position for a spinning ball was investigated. The table on the serving side was divided into nine grid squares to measure the success rate of the control of ball landing position; next, the simple aerodynamic model (ADM) and optimization approach were employed to calculate the initial ball velocity and rotational velocity when the ball landed in the various grid squares; finally, based on the ball velocity and rotational velocity at the landing point, the TRM and RRM were used to inversely calculate the parameters of the racket hitting the ball.

Nakashima *et al.* [12] proposed a method of controlling the ball landing position based on physical models, including the ADM and RRM, and reported the equations representing the movement of a ball during flight and when the ball hits the racket. Subsequently, they used the finite difference method to estimate the racket orientation and velocity corresponding to the landing point. However, they only used simulation methods to verify the landing position error of a single incoming ball at a single desired landing point. Li *et al.* [7] proposed a learning-based approach to controlling ball landing position. This learning system combines physical models with compensation mechanisms to slowly minimize the landing position error by adjusting the speed of the next hit according to the error in the landing position after each strike.

In previous studies, control of landing position by using a physical model has been applied to only balls with backspin. In physical models of landing position control, the parameters governing how the racket hits the ball and the velocity and rotational velocity of the outgoing ball are all unknown, which renders inverse operation extremely difficult. To mitigate this problem, multiple assumptions must be made regarding the specific rotational status of the incoming ball to simplify its physical model; hence, controlling the landing positions of different spinning balls is impossible.

Control of ball landing position by using machine learning does not require complex physical models; instead, it employs large volumes of training data to learn the association between the input and output data and construct a corresponding model. Matsushima *et al.* [10] described the process of playing table tennis by using three input–output maps: (1) the hitting point corresponding to the state of the incoming ball; (2) racket orientation and velocity corresponding to the change in ball velocity before and after impact; and (3) ball velocity after impact corresponding to the landing position of the returned ball and flight duration. Matsushima *et al.* used locally weighted regression (LWR) to learn the relations among these input–output maps. To equip a table tennis robot with learning capability, Huang *et al.* [5] combined LWR with fuzzy cerebellar model articulation control (FCMAC). When the robot hits a ball, LWR is performed to determine the required orientation and velocity of the racket upon impact. Subsequently, the FCMAC parameters are updated according to the errors between landing positions to adjust the racket parameters and minimize the error in the landing position. LWR is a nonparametric learning method; thus, when prediction is conducted using new input data, the coefficients must be recalculated. As the volume of training data increases, the calculation duration lengthens. To address this problem, Zhang *et al.* [20] used a neural network (NN) to learn the ball velocity after impact and the relationships between racket parameters during impact.

In control of ball landing position, the trajectory of the incoming ball, which differs from ball to ball, must be predicted. Solving this problem requires inverse operations. Inverse operation using physical models involves a highly complex calculation process and the use of other equipment and convoluted experiments to measure each environmental coefficient. Inaccurate environmental coefficients also influence model accuracy. Hence, this study separated the process of ball landing position control into multiple subtasks and employed NNs to construct a model for each subtask. Subsequently, the results were combined to form a complete model for control of landing position, with subtasks completed one after another.

This study is innovative in that multiple cascade neural networks (CNNs) were used to control the landing position of balls hit by table tennis robots, and the control was modeled by using large volumes of training data to learn the association between input and output data. The robot must be able to hit an incoming ball in such a way that the ball lands in the desired location. The method employed in this study consists of two parts. First, to control the robotic arm so that it successfully hits an incoming ball, the arm's hitting movement is planned, and the time of impact is controlled. Second, landing position is controlled, and the problem of this control is divided into four subtasks. In Subtask 1, in the absence of a high-flying ball, the appropriate flight duration is estimated using the incoming ball velocity and ball acceleration captured by cameras and ball's displacement from its actual hitting point to desired landing point. In Subtask 2, the trajectory of the ball from the desired landing point to the hitting point is predicted using the desired landing point, flight duration, and acceleration of the incoming ball. In Subtask 3, the desired velocity of the ball after it has been struck by the robotic arm is determined using the predictions from Subtask 1 (i.e., the distance and flight duration between the last point and hitting point along the trajectory). Finally, in Subtask 4, the required orientation of the racket surface is determined using the change in ball velocity after impact.

Compared with the LWR-based model for control of landing position [5], the CNN method proposed in this study resulted in a 233-mm smaller error in the average landing distance in the simulation experiment, with the standard deviation (SD) 201.3 mm lower. The success rate of the CNN method at landing points 1 and 2 was 25.9% and 32.9% higher, respectively, than when the LWR-based model was employed. For the landing point 5, the success rate was 2.6% lower. Regarding other landing positions, the success rate was slightly higher. Overall, the proposed NN exerted more favorable ball landing control than did the LWR-based model.

The rest of this paper is organized as follows: The next subsection provides an overview of the hardware system architecture and the environment for the table tennis setup. Section 2 details the proposed method for controlling the landing position, including the specifics of the table tennis robot and the motion control system for the robotic arm. Section 3 covers the experimental simulations, the process of generating training and testing data, and the methods used for evaluating landing position control. Finally, Section 4 presents the conclusions of the research and outlines potential directions for future work.

1.1. Hardware system architecture

Figure 1 illustrates the hardware system architecture of the table tennis robot used in this study. The architecture consists of an imaging system and a robotic system. Unlike the research conducted by Cohen et al. [3], which used a single camera positioned above the environment, we employ three 1.31 megapixel Imaging Development Systems (IDS) color industrial cameras (model UI-3140 CP-C-HQ) equipped with H0514-MP2 lenses to capture images and send them to a computer. These cameras offer a resolution of 1280x1024 pixels and a frame rate of 224 FPS, providing high-quality image data crucial for accurate tracking and prediction. Building on our previous study [8], we implemented ball tracking and trajectory prediction techniques to obtain the pixel coordinates of the spherical body (ball) and the three-dimensional (3D) coordinates of the table tennis ball. The system then calculates the velocity and acceleration of the ball by processing these sequential frames, ensuring precise tracking. To mitigate the effects of visual noise, we used a noise-reduction filter and a region-of-interest mechanism, focusing processing on the relevant areas. The computer records the trajectory of the flying ball for a period of time, predicts the hitting point, and calculates the desired orientation of the racket according to the hitting point and desired landing point. Additionally, we set up the environment to remain consistent, avoiding any changes in lighting that could affect the images. We used the same lighting throughout the experiments to ensure stable conditions. Finally, the results are translated into the Cartesian coordinates of a

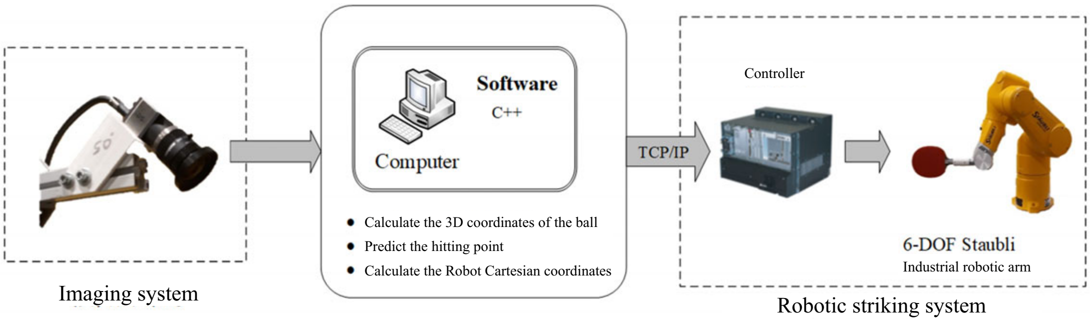


Figure 1. *Hardware system architecture.*

Stäubli TX60 industrial robotic arm, and the movement command is sent to the robotic arm controller, completing the hitting movement.

The CRACK V-989E table tennis ball machine was used in this study. This table tennis robot can hit balls at various velocities and rotational strengths through adjustment of the rotational velocity of the top and bottom wheels. It can hit nine types of spin through modification of the angle of the rotational head. The pitch angle of the machine head can be adjusted to hit balls that bounce either once or twice. The robot's controller can be used to adjust the left- or right-tilted deflection angle of the machine head and thus control the landing point of the outgoing ball.

1.2. Hitting movement of the robotic arm

The range over which the robotic arm hits balls and the arm's hitting movements in this study were designed with due consideration of the arm's working range. First, given the default range of the hitting movement, the angles must be arbitrarily adjustable within the default range of two degrees of freedom for controlling the racket's surface orientation. In this context, the default range refers to the initial constraints on the robotic arm's movements, ensuring the angles can be adjusted safely and effectively. This limitation is necessary because the base of the robot is fixed, and extending the range beyond these constraints could lead to dangerous situations, as the robot's movements could become hazardous. Second, the safe range over which the robotic arm can hit a ball must be considered to prevent the arm or racket from hitting the table during the hitting process, which could damage the equipment. Third, limited by the rotational angle of the arm's joints, the arm's terminal point affects the hitting movement and orientation that can be adopted.

Based on the three aforementioned conditions, the range of the arm's hitting movement was planned as illustrated in Figure 2. Because this study employed a robotic arm, for which the rotational velocity of each joint cannot be controlled, complex movements could impede the control of the time point at which the ball was struck. Additionally, only limited movements could be achieved at different locations within the working range of the arm because of structural factors. To ensure the ball clears the net, which was a standard constraint in our setup, this study defined the hitting movement of the robotic arm as illustrated in Figure 3: the center of the racket at the terminal end of the robotic arm moved from initial position P_0 to P_A and then from P_A to P_B; the Y and Z axes are determined by the hitting points predicted using the visual system of the table tennis robot.

Figure 4 presents a schematic of how the orientation of the table tennis racket at the terminal end of the robotic arm was adjusted. In the diagram, α is the Y-axis rotation of the racket at the base coordinates of the robotic arm, which adjusts the pitch angle of the racket, and β is the Z-axis rotation of the racket at the base coordinates of the robotic arm, which adjusts the deflection angle of the racket. Because the robotic arm's movement command controls the center of flange at the terminal end, to achieve the desired surface orientation of the racket so that the center of the racket coincides with the hitting point,

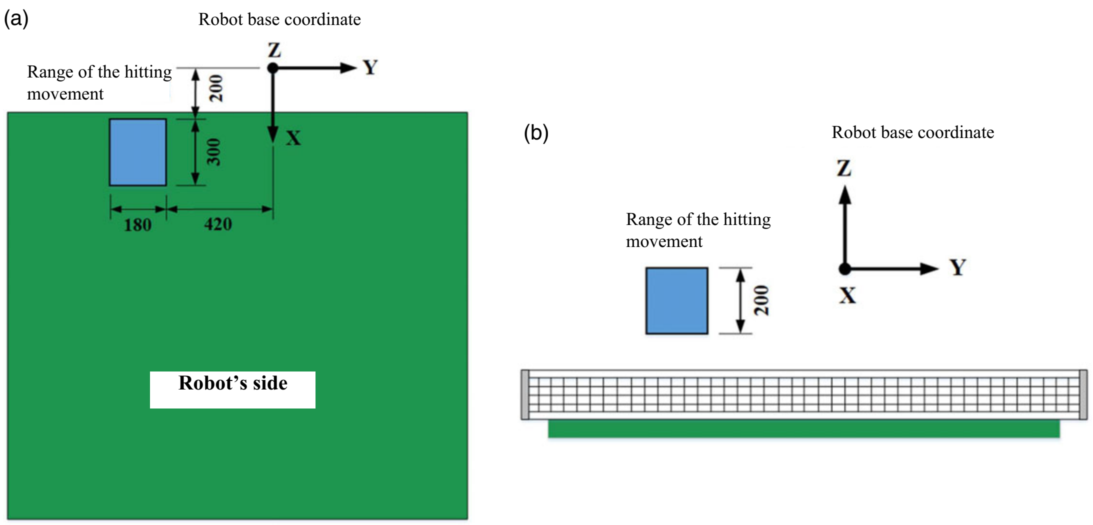


Figure 2. *Range of the robot hitting movement.*

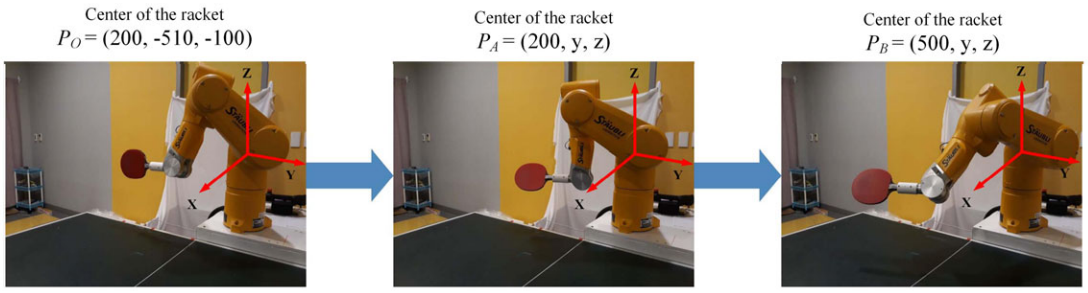


Figure 3. *Definition of hitting movement.*

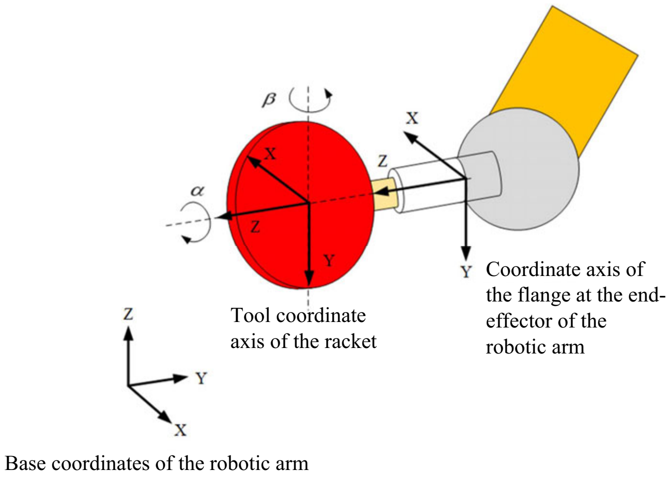

β
α
X
Z
Y
X
Z
Y
Coordinate axis of the flange at the end-effector of the robotic arm
Tool coordinate axis of the racket
Z
Y
X
Base coordinates of the robotic arm

Figure 4. *Schematic showing adjustment of racket orientation.*

the tool coordinate axis of the racket must be rotated to the desired racket surface orientation to push back the position and orientation of the flange.

Within the default range of hitting movement in this study, the robotic arm adopts similar orientations to hit a ball; hence, using Cartesian coordinates during movement control prevents discrepancies between the joint's rotational direction and user's expectation. The X, Y, and Z parameters of the Cartesian coordinates in movement control represent the center of the flange at the terminal end of the robotic arm. RX, RY, and RZ represent the angle of X-, Y-, and Z-axis rotation along the coordinate axis, and these three rotational angles can be used to determine the arm's orientation at its terminal end.

1.3. Hitting time of robotic arm

The industrial robotic arm used in this study does not provide functions for controlling joint rotational velocity; only the percentage of movement velocity can be adjusted. Thus, measuring the time required for the arm to move from its initial position to the hitting point is difficult. Nevertheless, trial and error were used in this study to solve this problem and ensure that the arm could hit the ball. The purpose of this method is to measure the time required for the robotic arm to move from its initial position to the hitting point as it performs the default hitting movement at a constant velocity and acceleration and with different racket surface orientations. This approach allows us to gather accurate data on the time required for various movements, despite the lack of precise control over individual joint speeds. As illustrated in Figure 5, when an incoming ball passes $X = 1650$, the robotic table tennis system starts recording the trajectory of the moving ball. The choice of $X = 1650$ was based on the specific dimensions of our table tennis setup, standardizing the position from which measurements and analyses were conducted. This value ensures consistency across all experiments, providing a controlled environment for accurate data collection. When 10 trajectory points have been recorded, these points are used to predict the position of the ball at $X = 400$. This predicted position, along with a random racket surface orientation, serves as the basis for generating the robotic arm's movement command. The value $X = 400$ is used specifically for measurements and observations to determine the optimal striking point. When the ball is struck by the racket held by the robotic arm, the trajectory of the moving ball from $X = 1650$ to the point of impact is recorded, including the time intervals between each recorded trajectory point.

The position of the table tennis ball at $X = 400$ can be used to determine the position of the racket's center in the directions Y and Z as the robotic arm performs a hitting movement. The RX, RY, and RZ angles of the robotic arm can also be used to calculate the orientation of the racket surface. Because racket orientation is randomly decided, when the racket is tilted, the point of contact between the racket and ball is difficult to determine. Thus, the position of the racket's center in the X direction during impact is difficult to calculate. To address this problem, the center of the ball as it hits the racket is adopted as the basis for the hitting point, and when the surface of the racket coincides with plane F' in Figure 6, the racket's center is used as the estimated hitting point, where plane F' represents the center of the spherical body of the ball when it hits the racket.

Finally, the first predicted point of the trajectory of the moving ball to the point of impact is extracted, and the time intervals between the predicted points in the trajectory are summed, obtaining t_2 in Figure 7. t_2 represents the total time taken from the last recorded position of the ball to the point of impact. This includes the time needed to predict the hitting point, calculate the Cartesian coordinates for the robotic arm's movement, and execute the command. Because it takes less than a millisecond to predict the hitting point and calculate the Cartesian coordinates of the robotic arm's flange corresponding to the hitting point and racket orientation, the total calculation time is negligible in this study. Therefore, t_2 is considered the time required to send a movement command to the robotic arm's controller and move the arm from its initial position to the hitting point. Using the aforementioned method of measuring the time taken to perform a hitting movement, this study collected 200 sets of data regarding the time required for the robotic arm to move from its initial position to the joint angle corresponding to the hitting position.

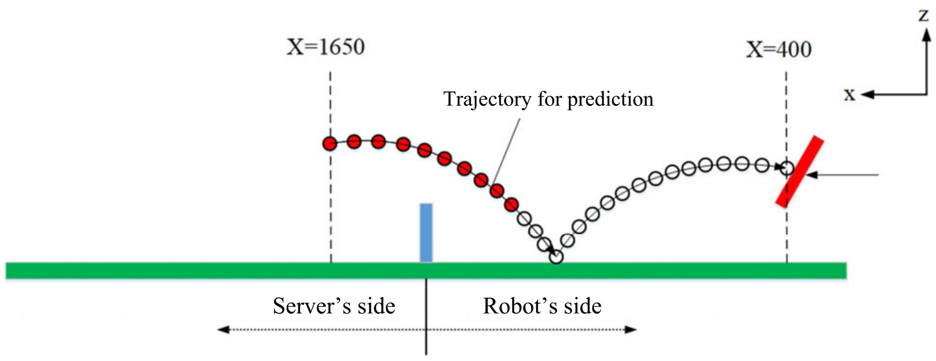


Figure 5. *Prediction of hitting point.*

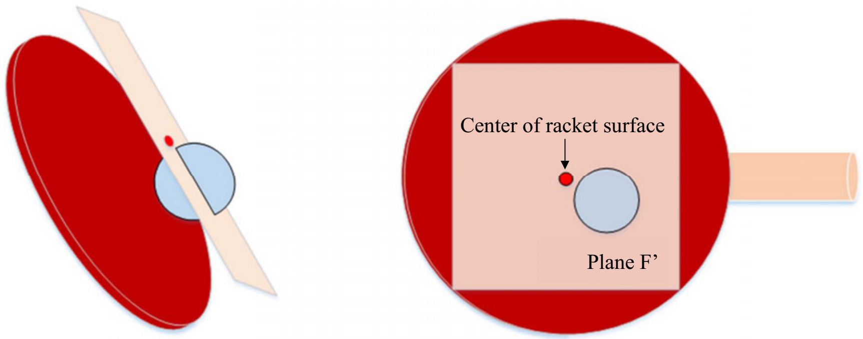


Figure 6. *Plane parallel to the surface of the racket and passing through the center of the ball when the ball strikes the racket.*

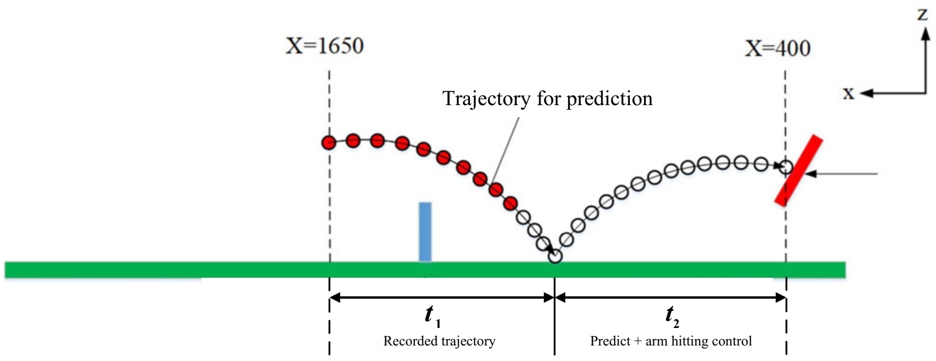


Figure 7. *Allocation of time by the robotic table tennis system.*

The 200 sets of data comprised 160 sets of training data and 40 sets of test data. These data were used to construct a model for estimating the time required to perform a hitting movement.

This study used multivariate linear regression (MLR) and a NN estimation model. Table I shows the estimation errors in the training and testing data for both methods. According to Table I, the average absolute error, maximum absolute error, and SD based on the training and test data were more favorable when neural-network-based prediction was employed than when MLR was used. Therefore, this study used a NN as the model for estimating the time required to perform a hitting movement.

Table I. *Estimation error of the time required to perform a hitting movement.*

	Training data		**Test data**	
Prediction Models	**MLR**	**NN**	**MLR**	**NN**
Ave. Abs. Err (s)	0.016	0.012	0.022	0.018
Max. Abs. Err (s)	0.074	0.046	0.102	0.067
Min. Abs. Err (s)	0.000	0.000	0.001	0.001
SD (s)	0.020	0.016	0.028	0.020

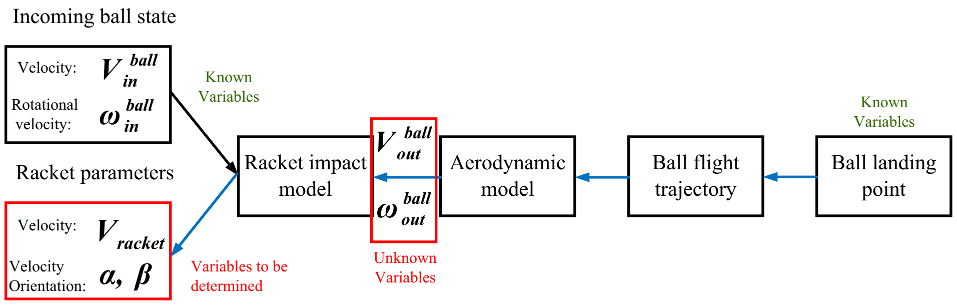


Figure 8. *Inverse operations using physical models to control the landing point of a returned ball.*

2. Controlling the landing position of the returned ball

2.1. Planning of landing position control

The problem of ball landing position control can be considered an inverse operation problem. As illustrated in Figure 8, the ball velocity after impact, angular velocity, racket orientation during impact, and racket velocity are all unknown in this problem. These unknowns can only be determined using the incoming ball velocity, angular velocity, desired landing point, and desired ball flight duration. The physical models, consisting of the aerodynamic and bouncing models as referenced by Nakashima *et al*. [15], are essential to this approach. The ADM accounts for drag and lift forces, which are crucial for understanding the ball's flight path, especially under varying spin conditions. The bouncing model encompasses the dynamics of the ball's interactions with both the table and racket, providing a detailed understanding of the ball's behavior upon impact. Because a table tennis ball movement model can be viewed as a quadratic nonlinear formula, using these physical models to perform inverse operations for controlling the landing point of different spinning balls is, in reality, both complex and difficult.

The machine learning methods proposed in previous studies are not applicable to the problem of landing position control for various spinning balls because they were typically designed for specific conditions like topspin or backspin and relied on assumptions about the ball's rotational state. These assumptions limited their ability to accurately control the landing position under varied spin conditions. In this study, we modified these methods to handle the complexities of different spins. The problem of return ball landing position control is associated with four sequential subtasks, as shown in Figure 9. Our approach utilizes a CNN model that divides the task into these subtasks, allowing for more precise and adaptable control over the ball's trajectory and landing position, regardless of the spin type. This modification enhances the generalizability and effectiveness of the control system, providing a more comprehensive solution than previous methods. Additionally, we integrated a physical model of the table tennis robot, considering factors such as aerodynamics and the interactions between the ball and the racket. This integration enables the system to account for real-world factors like air resistance and spin effects, which were not adequately addressed in previous methods. This combination of machine learning and physical modeling offers a more robust solution for predicting and controlling the landing position of various spinning balls.

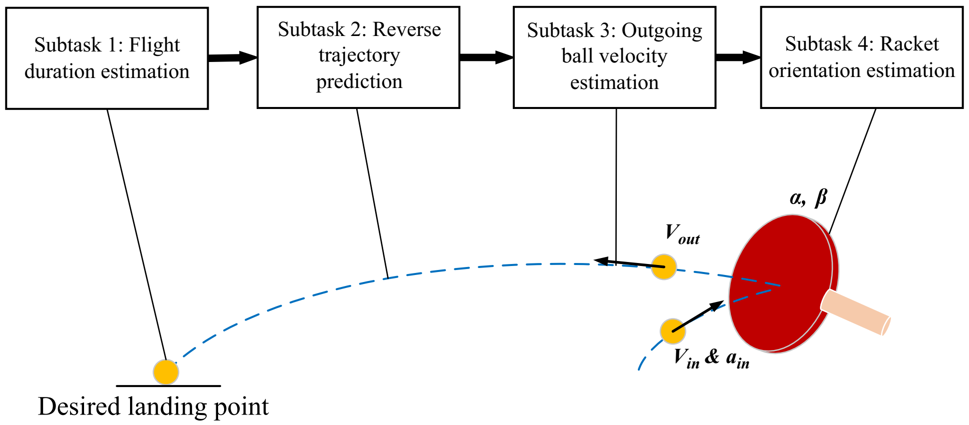

Subtask 1: Flight duration estimation
Subtask 2: Reverse trajectory prediction
Subtask 3: Outgoing ball velocity estimation
Subtask 4: Racket orientation estimation
α, β
V_{out}
V_{in} & a_{in}
Desired landing point

Figure 9. *Planning of ball landing position control subtasks.*

Subtask 1: In this subtask, the flight duration of a ball from the moment of impact to the desired landing point is determined. In studies regarding the hitting control of table tennis robots, the racket velocity and racket surface orientation are the parameters for implementing control. However, the true velocity at the terminal end of the arm used in this study cannot be controlled; hence, the velocity and acceleration percentages of the robotic arm are set as constant to ensure that the racket's hitting velocity approximates constant values and the ball landing point is controlled by the orientation of the racket. Assume that at constant racket velocity and incoming ball velocity, only the time a ball takes to reach a certain flight distance after impact is adjusted under the condition of racket orientation, then flight duration is primarily affected by the ball velocity after impact and flight distance. However, the ball velocity after impact is unknown, and other known conditions must be employed to determine the flight duration. At constant racket velocity, the ball velocity after impact is proportional to the velocity of the incoming ball, and in the absence of a high-flying ball, flight duration and flight distance can be assumed to be proportional. In [7], the states of topspin and backspin balls after impact were determined, and the flight trajectory of a topspin ball was found to be higher, whereas the backspin ball began falling after being hit by the racket. This reveals that the rotational state of the incoming ball considerably influences the state of a ball after impact. Because the rotational velocity of a spherical body cannot be determined for this study's robotic table tennis system and because the ball's acceleration varies during its process of rotation, this study used acceleration to represent the rotational state of the ball. According to the aforementioned factors that influence flight duration, the Subtask 1 model was defined to comprise the following inputs: distance between the hitting point and desired landing point, incoming ball velocity, and incoming ball acceleration. The model's output is flight duration.

Subtask 2: In this subtask, the ball's flight trajectory from the desired landing point to the hitting point is reverse-predicted from among the trajectory points segmented by the same time interval. The obtained point is used to increase the accuracy of the ball velocity estimate after impact. Unlike trajectory prediction of an incoming ball, forward trajectory prediction is not applicable in this case because the ball velocity at the initial position (hitting point) is unknown and the end point of the trajectory (desired landing point) is self-defined (i.e., a known term). In addition, the trajectory of a moving ball after impact is related to flight distance, flight duration, and the rotational state of the ball, whereas the displacement from the trajectory at each moment is related to that in the previous moment. Hence, in Subtask 2, the displacement from hitting point to desired landing point, flight duration, and acceleration of incoming ball are the initial inputs for estimating the displacement in the previous moment. Displacement in the next moment is predicted using the displacement in the previous moment, flight duration, and incoming ball acceleration, subsequently obtaining, from among the trajectory points that are segmented by the same time interval, the trajectory point that is closest to the hitting point.

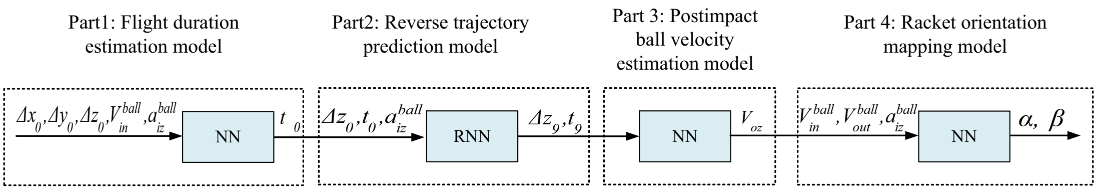

Part1: Flight duration estimation model
Part2: Reverse trajectory prediction model
Part 3: Postimpact ball velocity estimation model
Part 4: Racket orientation mapping model
$\Delta x_0, \Delta y_0, \Delta z_0, V_{in}^{ball}, a_{iz}^{ball}$
NN
t_0
$\Delta z_0, t_0, a_{iz}^{ball}$
RNN
$\Delta z_9, t_9$
NN
V_{oz}
$V_{in}^{ball}, V_{out}^{ball}, a_{iz}^{ball}$
NN
α, β

Figure 10. *Proposed cascade neural network.*

Subtask 3: In this subtask, the velocity of the outgoing ball after impact, V_{out}, is estimated at the last point of flight trajectory, which was predicted in Subtask 2. Because this position is closer to the hitting point, the state of the ball at this position is similar to that at the hitting point. The velocity of the ball after impact is more accurately estimated than if using the total displacement and total flight duration.

Subtask 4: According to the physical model employed in ref. [15] for illustrating the impact between a table tennis ball and racket, the velocity of an outgoing ball after impact is related to the ball velocity before impact, ball rotational velocity, racket orientation, and racket velocity. In the present study, the racket velocity is constant, and the racket surface orientation is used to control the landing position of the returned ball. Hence, ball velocity before and after impact and incoming ball acceleration are the input parameters for estimating racket orientation.

2.2. Cascade forward and recurrent neural networks

On the basis of the subtasks of ball landing position control planned in the previous section, a CNN based on a forward and recurrent neural network (RNN) was designed (Figure 10) to achieve control of ball landing position when using a robotic table tennis system. In Parts 1, 3, and 4 of the model, control is implemented using the forward NN, whereas in Part 2, the RNN is employed to exercise control. Compared with other regression algorithms, NNs do not require input of hypothesized relationships between input and output variables such as index, polynomial, and trigonometric functions. By adjusting the hidden layer and number of neurons, the network's ability to express models can be improved to learn the relationships of complex input–output maps. Therefore, Parts 1, 3, and 4 of the model in this study employ forward NNs as the model architecture. The forward NN handles the non-linearities associated with the ball's motion, such as drag and spin effects, by providing initial estimations of the ball's trajectory. Regarding Part 2, because a ball's displacement at every moment is related to the displacement at the next moment, a model based on the RNN, which contains state memory units, is used to predict trajectory points by referring to information on the displacement at the last moment and the history stored in the state memory unit. The RNNs refine these predictions by capturing temporal dependencies, essential for precise motion planning and control, especially in dynamic scenarios like table tennis. In the network architecture designed in this study, Part 2 is the reverse trajectory prediction model, which is only used to predict the Z-direction trajectory, and Part 3 is the postimpact ball velocity estimation model, which is only employed to estimate the Z-direction velocity of the ball because the movement trajectory of a topspin or backspin ball after impact is similar to that of an isokinetic movement.

In the flight duration estimation model (Part 1), the inputs are the displacement of the incoming ball from the hitting point to the desired landing point, denoted Δx_0, Δy_0, Δz_0; the acceleration of the incoming ball $V_{in}^{ball} = (V_{ix}, V_{iy}, V_{iz})$; and the Z-direction acceleration of the incoming ball a_{iz}^{ball}. The output is the flight duration t_0 from hitting point to desired landing point. In this study, topspin, backspin, and no-spin balls were considered. Topspin and backspin influence only the Z-direction acceleration of the ball; therefore, only the acceleration in this direction was considered.

In the reverse trajectory prediction model (Part 2), the flight trajectory of the ball from hitting point to landing point was divided into 10 segments with equal time intervals. As illustrated in Figure 11, P_0 is the ball landing point and P_{10} is the ball-hitting point. Based on Figure 11, the definition of the RNN architecture for reverse trajectory prediction is displayed in Figure 12, where the time step $T = 0, 1, \cdots, 9$; S_T is the state memory unit of the RNN; Δz_T is the Z-direction displacement of the ball

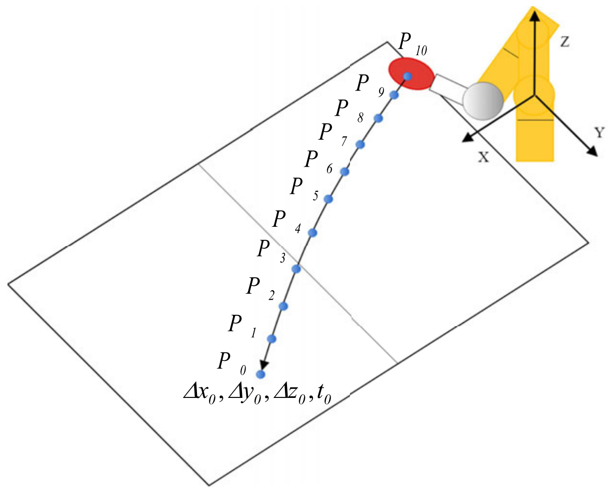


Figure 11. *Outgoing ball flight trajectory divided into 10 segments with equal time intervals.*

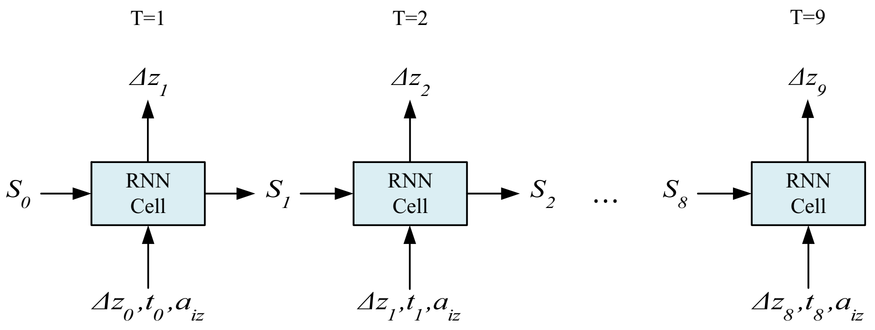


Figure 12. *Architecture of the recurrent neural network for reverse trajectory prediction.*

from hitting point P_{10} to P_T; t_T is the flight duration of the ball from hitting point P_{10} to P_T in Figure 11; and a_{iz}^{ball} is the Z-direction acceleration of the incoming ball once it has reached hitting point P_{10}. In the RNN architecture, when the time step $= T$, the network receives S_{T-1}, Δz_{T-1}, t_{T-1}, a_{iz}^{ball} as the inputs and outputs Δz_T.

The ball velocity estimation model (Part 3) receives the Z-direction displacement Δz_9 and flight duration t_9 of the ball from hitting point P_{10} to P_9 in Figure 11 as its input and outputs the Z-direction velocity of the ball at hitting point P_{10} after it has been hit. Because the movement of the ball in the X and Y directions is similar to an isokinetic movement, the X- and Y-direction velocity of the ball at hitting point P_{10} after it has been hit can be calculated as follows:

$$V_{ox} = \frac{\Delta x_0}{t_0}, V_{oy} = \frac{\Delta y_0}{t_0}, \tag{1}$$

Finally, the change in velocity and racket orientation mapping model (Part 4) receives the velocity of the ball at hitting point P_{10} before (V_{in}^{ball}) and after (V_{out}^{ball}) it hits the racket and the Z-direction acceleration of incoming ball (a_{iz}^{ball}) as the inputs and outputs the corresponding angles of racket orientation α and β.

2.3. Landing position control based on LWR

Unlike the systems used in previous studies, the robotic table tennis system proposed in this study does not use racket velocity as the parameter for controlling the ball's landing position. Hence, the landing

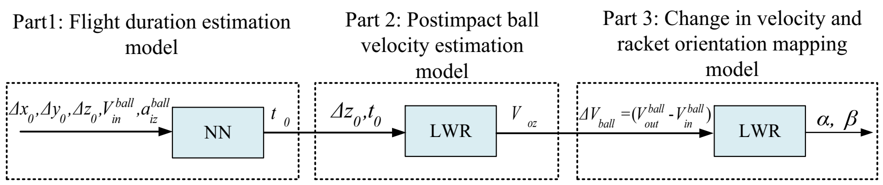

Part1: Flight duration estimation model
Part 2: Postimpact ball velocity estimation model
Part 3: Change in velocity and racket orientation mapping model
$\Delta x_0, \Delta y_0, \Delta z_0, V_{in}^{ball}, a_{iz}^{ball}$
NN
t_0
$\Delta z_0, t_0$
LWR
V_{oz}
$\Delta V_{ball} = (V_{out}^{ball} - V_{in}^{ball})$
LWR
α, β

Figure 13. *Modified locally weighted regression-based landing position control method.*

position control methods developed by past studies cannot be compared with the method proposed in this study. For this reason, the LWR-based landing position control method in ref. [5] was modified to act as a comparison method for the present study's system. Figure 13 illustrates the architecture of the modified LWR-based landing position control method. Part 1 of the method is the flight duration estimation model, which is identical to Part 1 of the multilayer NN architecture proposed by this study. Part 2 is the postimpact ball velocity estimation model. In ref. [5], only the displacement of the ball from the hitting point to the desired landing point was used to estimate the velocity of the ball at the hitting point. However, the same level of displacement may occur at different ball velocities. Hence, the ball velocity estimation model was modified to receive the Z-direction displacement Δz_0 and flight duration t_9 of the ball as inputs and to output the Z-direction velocity V_{oz} of the ball after the ball has been hit. Subsequently, the ball velocities in the X and Y directions can be calculated using (1). Part 3 is the change in velocity and racket orientation mapping model, which receives the change in ball velocity ΔV_{ball} at the hitting point and outputs the corresponding angles of racket orientation α and β.

3. Experimental results

3.1. Simulation

The ball landing position control results of a robotic table tennis system are affected by not only training errors in the landing position control model itself but also errors in the 3D calculation of imaging systems, hitting point prediction, estimation of incoming ball velocity and rotation, and hitting time point control. These factors cause greater landing position errors. Therefore, this study first used simulation to compare the performance of the proposed CNN and LWR models in landing position control when only errors in landing position control model exist. The physical model of the table tennis ball movement process in ref. [15] was employed to simulate the entire process of a ball flying, bouncing on the table, and being hit by a racket.

3.2. Generating training and test data

Incoming balls with different movement directions, velocities, and rotational states were generated by adjusting the initial position, serving angle, velocity, and rotational velocity of the incoming ball. The ball landing position control only considered topspin and backspin balls; hence, the initial rotational state of the incoming ball was configured to include topspin and backspin balls. To ensure that the landing points of the returned ball in the training data were evenly distributed among the nine grid squares on the server's side (Figure 14), 900 sets of data (consisting of the trajectories of the incoming ball and return ball) were obtained for model training. Among the 900 sets of data, 100 sets were data on the return ball's landing points in each grid square; 720 sets were training data, and 180 sets were test data. Next, another 900 sets of data were obtained, 100 of which were data on the landing points in each grid square. These datasets were used to evaluate the landing position control method and verify that landing position control was achieved. Half the training data and test data involved topspin balls, and the other half, backspin balls.

Table II. *Structural settings for cascade neural network in simulation experiment.*

Model	No. of Neurons	Activation function
Part 1	9	tanh
Part 2	12	ReLU
Part 3	12	tanh
Part 4	14	tanh

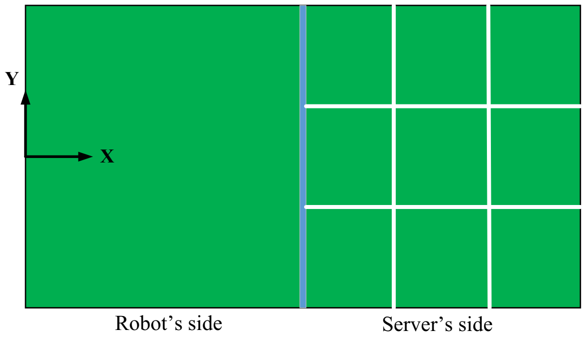


Figure 14. *Planning of the landing area of a return ball.*

3.3. Model evaluation

The previous section described the method by which the training data and test data in the landing position control model were obtained. This section presents an evaluation of the CNN proposed in this study and the LWR-based landing position control mode. The purpose was to evaluate each part of the model that causes errors in the landing position to accumulate. Table II shows the structural settings for each part of the proposed CNN shown in Figure 10 in the simulation experiment.

To assess the accumulation of errors in each part of the CNN, 4 bits were used to express whether each part was activated. Take 0001 as an example; 0001 means that Parts 1, 2, and 3 are unactivated, whereas Part 4 is activated. Similarly, the LWR-based landing position control model was planned using the same approach. Because the LWR-based model has only three parts, 3 bits are used to indicate whether each part is activated.

After the input data required by each model were incorporated into the model, the racket orientation parameters α and β could be obtained, which were then input to the physical model to obtain the return ball's landing point. Subsequently, the landing distance error was calculated on the basis of the desired and actual landing point. Figure 15 shows the errors in landing distance for the training data based on the four types of models planned by the CNN. Models 0001 and 0011 had extremely similar average errors and SDs, suggesting that adding a model to Part 3 did not significantly influence the errors. Next, the errors of Models 0011 and 0111 were compared, and if Part 2 was added, the average error increased substantially from 33.1 to 40.7 mm whereas the SD increased from 51.5 to 60.2 mm, both of which are a considerable increase compared with the two aforementioned comparisons. This difference was obtained because the error in flight duration t0 estimated by the Part 1 model simultaneously influenced the input parameter t_9 of the Part 3 model and the input parameter V_{out}^{ball} of the Part 4 model. Unlike the trend prior to the addition of Part 1, for every part added to the model, the errors accumulated gradually. In Model 0111, the errors were higher because Part 2 had been added, causing the output errors of Part 3 to be greater, which indirectly increased the output errors of Part 4. Once Part 1 had been added, the errors accumulated gradually, and the errors input to the Part 3 and Part 4 models increased, thereby further raising the errors in the model's final output, which was the orientation of the racket surface. According

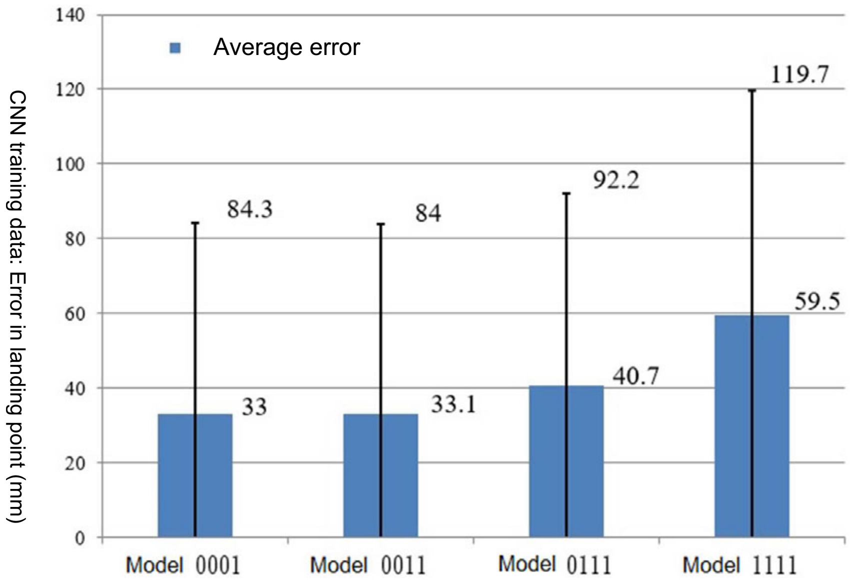


Figure 15. *Landing distance errors of four cascade neural network-based models for training data.*

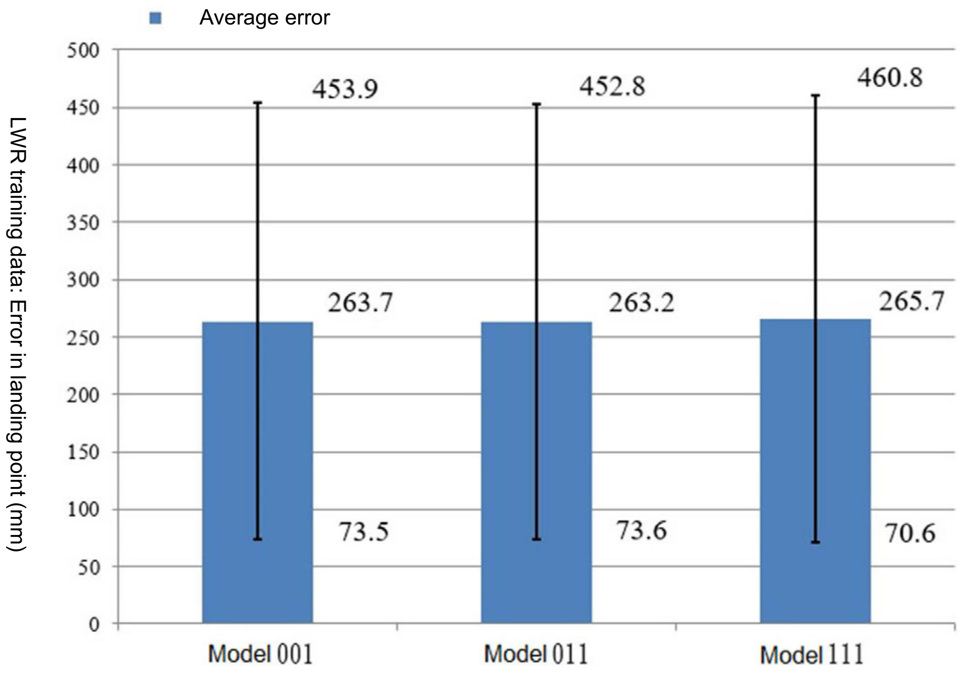


Figure 16. *Landing distance errors of three locally weighted regression-based models for training data.*

to Figure 16, the LWR-based landing position control produced greater errors in landing distance than the CNN for the training data based on three models. Figure 16 shows that the average errors of the three models were all greater than 260 mm (SD = 190 mm).

Figure 17 illustrates the errors in the landing distance for test data when using the four CNN models. The error trend was identical to that for the training data. Models 0001 and 0011 had similar errors.

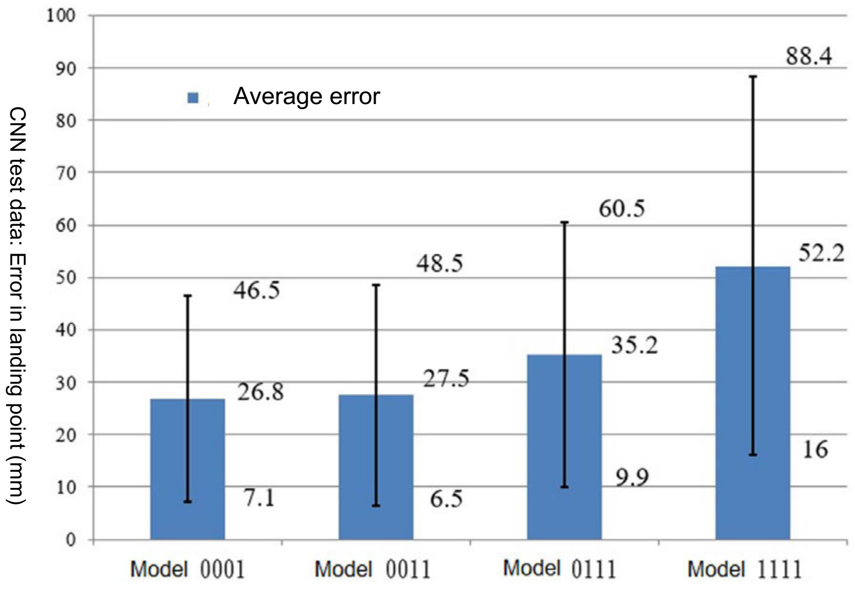


Figure 17. *Landing distance errors of four cascade neural network-based models for test data.*

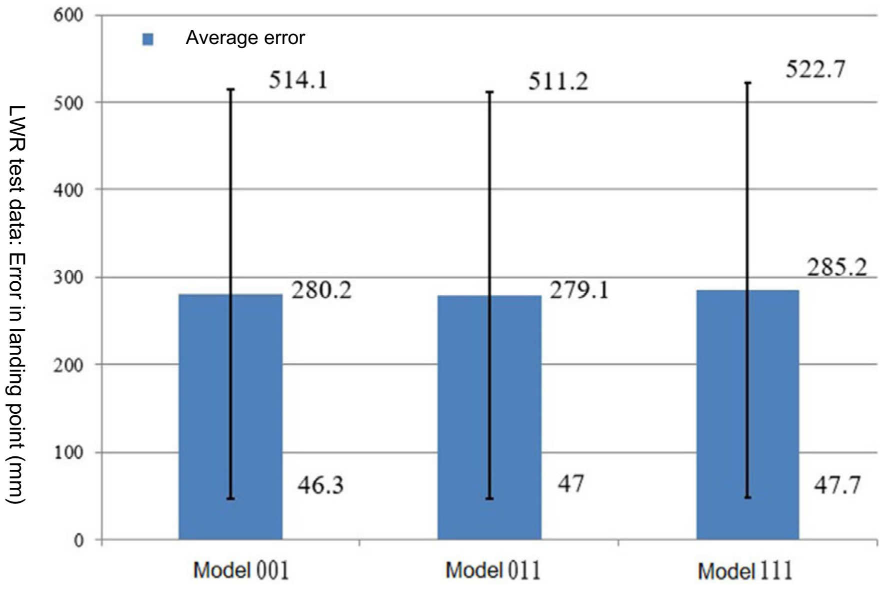


Figure 18. *Landing distance errors of three locally weighted regression-based models for test data.*

Model 0111 generated considerably greater errors, and the errors of Model 1111 were dramatically higher. Figure 18 shows the errors in the landing distance for test data when using the three LWR-based models. The error trend was again identical to that for the training data. The average errors and SDs of the three models were similar; however, the average error and SD for the test data were 20 and 40 mm greater than those for the training data, respectively. This result reveals that the LWR-based landing position control model performed poorly for the test data.

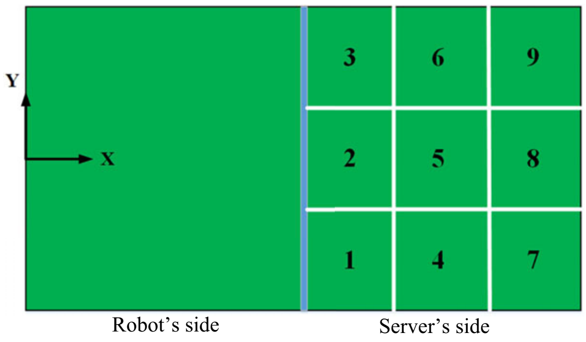


Figure 19. *Planning of desired landing points.*

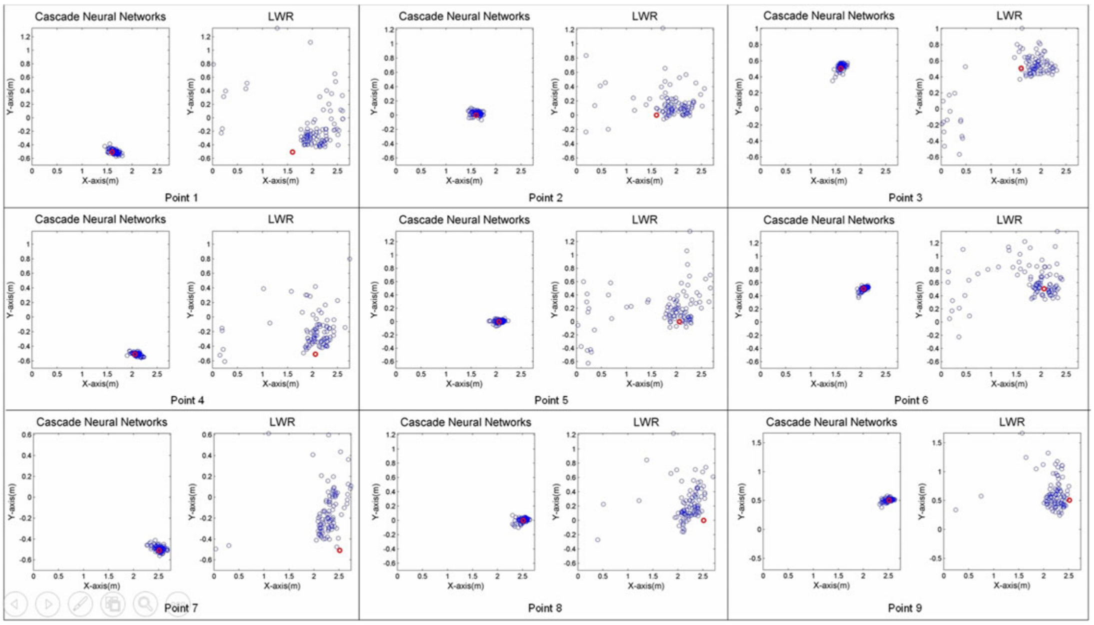


Figure 20. *Distribution of landing points for desired landing point 9.*

3.4. Landing position control

To compare the landing position control performance of the CNN proposed in this study and the LWR-based landing position control model, the desired landing points are displayed in Figure 19 were selected. The server's side was divided into three equal parts in the X and Y directions, forming nine grid squares of equal size. The center point of each square was designated as the desired landing point. In this experiment, 100 incoming ball trajectories (50 topspin and 50 backspin balls) were employed for each desired landing point to examine the distribution of return ball landing points generated using the two model types. Figure 20 presents the distributions of ball landing points for desired landing points 1–9, respectively. In the diagrams, red circles represent the desired landing point and blue circles represent the actual landing point of a returned ball.

Examination of Figure 20 reveals that when the CNN was used for control, the landing points were relatively more centralized and closer to the desired landing points than when the LWR-based model was used. In this experiment, landing position control was more effective with the proposed CNN than with the LWR-based model.

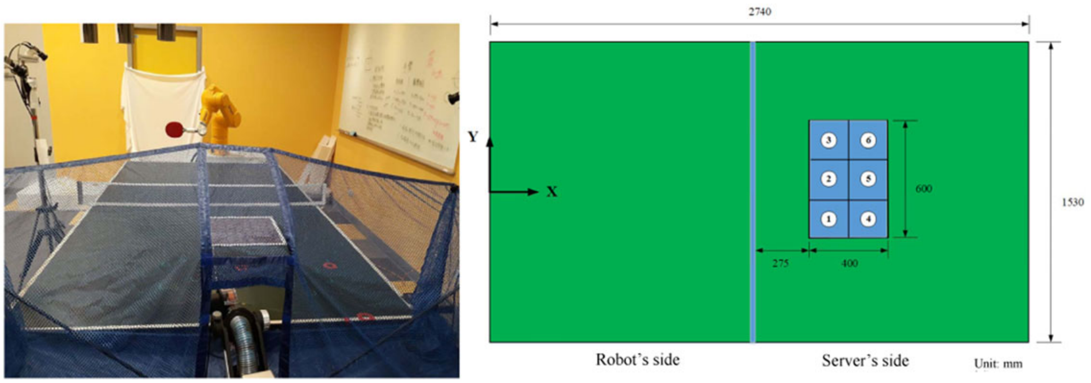


Figure 21. *Experimental environment and controllable area of return ball landing point and planning of desired landing points.*

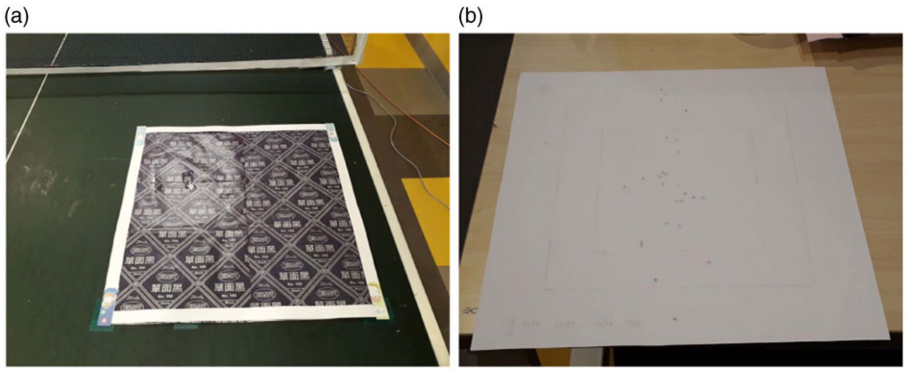


Figure 22. *The block (where the ball should land and bounce on the table) used in this experiment. (a) Placement of the block during the experiment. (b) Prints left on white paper when the block was hit by the ball.*

3.5. Experiment

The objective of this experiment was to compare the landing position control performance of the proposed CNN and LWR-based control models in a robotic table tennis system. Figure 21 illustrates the experimental environment and placement of relevant equipment in the robotic table tennis system. The visible range of the visual system in the robotic table tennis system could not include the whole of the server's side; therefore, the blue area (shown in Figure 21) was defined as the controllable range of the return ball landing point according to the visible range of the visual system and distribution of landing points of a ball returned by the robot when using a randomly oriented racket. This blue area was divided into six grid squares of equal size. The center of each grid was the desired landing point in this experiment.

Given the 3D calculation errors inherent in a visual system, for this experiment, a 40×40 cm^2 block (where the ball should land and bounce on the table) was designed as shown in Figure 22(a). The block consisted of four layers. The topmost layer was a 2-cm-wide white frame that facilitated positioning of the block when testing different desired landing points; the second layer was carbon paper so that when a ball hit the block, it would leave a print on the piece of white paper at the bottom [Figure 22(b)]; the third layer was a frame of the same size as the topmost layer to avoid leaving prints on the white paper at the bottom when the ball landed outside the block; and the bottommost layer was the aforementioned piece of white paper.

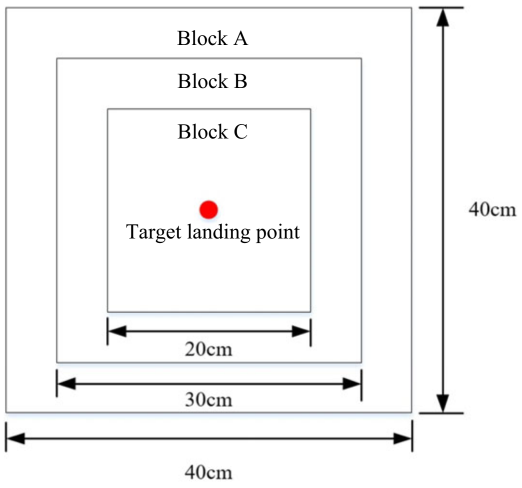


Figure 23. *Blocks of differing sizes.*

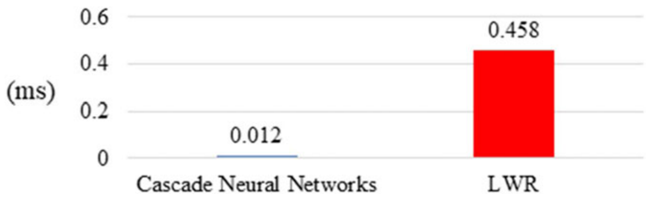


Figure 24. *Comparison of execution time of the cascade neural network and locally weighted regression-based models.*

In this experiment, the errors in the returned ball landing point were affected by numerous factors, including the errors in the CNN, control of hitting time and hitting point, incoming ball velocity, and acceleration prediction. Therefore, blocks of three sizes were planned, as illustrated in Figure 23. The success rates for these three blocks were averaged to represent the success rate of a desired landing point. In this experiment, the trajectories of topspin, backspin, and no-spin balls returned by a randomly oriented racket were collected for model training.

Once the robotic table tennis system completed estimation of the hitting point and incoming ball velocity and acceleration, the landing position control model was employed to calculate the racket orientation, and finally, the robotic arm was ordered to move and return the ball. To ensure that the robotic arm had sufficient time to complete a hitting movement once the calculation had been completed, the calculation time of the model had to be minimized as much as possible. Figure 24 shows the average amount of time spent by the CNN and LWR-based models to make 100 calculations. Because both models took less than 1 ms, the time required by robotic arm to execute the remaining movement was not adversely affected. The average execution time of the CNN and LWR-based models was 0.012 and 0.458 ms, respectively. Compared with the LWR-based model, the CNN model took 97.38% less time. Figure 25 shows the number of parameters associated with the CNN and LWR-based models. The LWR-based model had 3766 parameters, whereas the CNN model had 424 parameters, 88.74% fewer. When predicting every new input, the LWR-based model must recalculate the model coefficients by using the training data; therefore, a large volume of training data must be stored in the matrix, taking up a considerable amount of memory.

Table III summarizes the average success rates at the six desired landing points when the CNN and LWR-based models were used. The results indicate that the success rate at desired landing points 1 and 2 was considerably higher for the CNN model than for the LWR-based model. The success rates at desired

Table III. *Average total success rate at various desired landing points in blocks of varying sizes.*

Desired landing point	Cascade Neural Networks	LWR
1	37.8%	11.9%
2	58.5%	25.6%
3	73.0%	70.7%
4	45.9%	45.2%
5	68.9%	71.5%
6	68.9%	59.6%

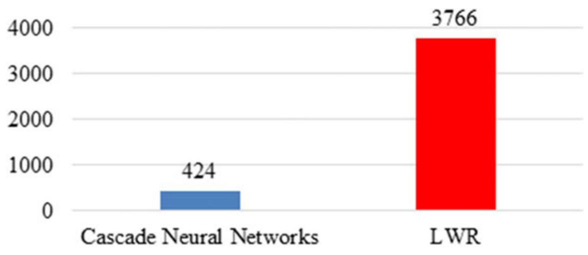
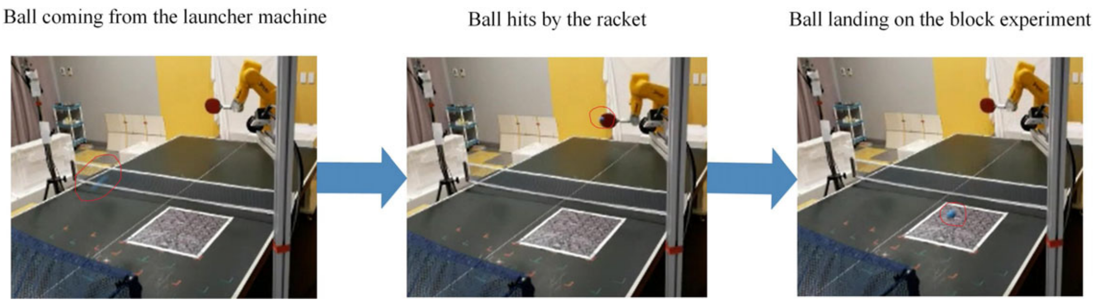


Figure 25. *Comparison of the number of parameters associated with the cascade neural network and locally weighted regression-based models.*



Ball coming from the launcher machine
Ball hits by the racket
Ball landing on the block experiment

Figure 26. *Sequential stages of table tennis ball interaction during experimental testing.*

landing points 3, 4, and 5 differed between methods by 3%, and this difference can be considered to be the result of measurement errors. The two methods had similar control effects at these three desired landing points. At desired landing point 6, the CNN model was somewhat superior to the LWR-based model. Overall, the CNN model achieved favorable landing position control. Figure 26 shows the sequential stages of the table tennis ball interaction during experimental testing. The first stage captures the ball coming from the launcher machine, the second stage shows the ball being hit by the robotic arm's racket, and the final stage illustrates the ball landing on the block used for measuring landing positions.

Next, the success rates for topspin, backspin, and no-spin balls were calculated. Tables IV and V show the success rates of controlling the landing positions for the three spin types using the CNN and LWR models, respectively. Based on the results, CNN model maintains a relatively stable control effect on landing points, with success rates not less than 50% for all six points.

Additionally, comparing the success rates across 18 sets of data (covering three spin types and six landing points), the success rates of topspin balls at desired landing points 3, 4, and 5 and no-spin balls at desired landing points 5 and 6 were lower than those obtained using the LWR-based model. Therefore, the CNN overall more effectively controlled the landing of differently spinning balls.

Table IV. *Average success rates of cascade neural networks model for different spin types.*

Desired landing point	Topspin	Backspin	No Spin
1	58.9%	36.7%	17.8%
2	57.8%	73.3%	44.4%
3	50.0%	85.6%	83.3%
4	67.8%	24.4%	45.6%
5	60.0%	80.0%	66.7%
6	70.0%	65.6%	71.1%
Average	**60.75%**	**60.93%**	**54.82%**

Table V. *Average success rates of locally weighted regression model for different spin types.*

Desired landing point	Topspin	Backspin	No Spin
1	20.0%	8.9%	6.7%
2	27.8%	46.7%	2.2%
3	75.6%	57.8%	78.9%
4	73.3%	23.3%	38.9%
5	70.0%	73.3%	71.1%
6	47.8%	50.0%	81.1%
Average	**52.42%**	**43.33%**	**46.48%**

4. Conclusion and future work

In the CNN proposed in this study, the task of controlling the return ball landing position by cameras was divided into four subtasks. A forward NN and RNN were used for model training and combined to form a landing position control model. A LWR-based landing position control model [5] was compared with the proposed CNN. The models were evaluated through a simulation experiment. For the LWR-based model, the average landing distance error was 285.2 mm (SD = 237.5 mm), whereas for the CNN, the average landing distance error based on the test data was 52.2 mm (SD = 36.2 mm). This was an average error reduction of 233 mm and SD reduction of 201.3 mm, indicating considerable improvement. In the physical experiment on the ball landing position control of the robotic table tennis system, the success rate of the LWR-based model at desired landing points 1 and 2 was 11.9% and 25.6%, respectively, whereas that of the CNN at the same landing points was considerably improved at 37.8% and 58.5%, respectively. Regarding the other four desired landing points, the success rates of the two methods differed by 3% for desired landing points 3, 4, and 5, which can be considered to be a result of measurement errors. The two methods thus exhibited similar control effects for these three points. At desired landing point 6, the CNN (68.9%) outperformed the LWR-based model (59.6%) by 9.3%. The experimental results verified that the proposed CNN generally performed more favorably than the LWR-based model in controlling the landing position of a table tennis ball.

Because the robotic arm used in this study does not provide functions for controlling joint velocity, the velocity of the racket could not be included in the model as a control variable. If the robotic arm were replaced with a self-made ball-hitting structure, the user would be able to control the racket velocity. Hence, introducing racket velocity to the model as a control variable can be considered to further enhance the effectiveness of landing position control.

Author contributions. H. I. Lin and C. F. Syu conceived and designed the study. H. I. Lin wrote the article. C. F. Syu conducted data gathering, experiments, and statistical analyses.

Financial support. The manuscript is supported by National Science and Technology Council for the funding NSTC 111-2221-E-A49-192-MY3.

Competing interests. The authors declare no conflict of interest.

Ethical standard. Not applicable.

Supplementary material. The supplementary material for this article can be found at https://doi.org/10.1017/S0263574724001693.

References

[1] L. Acosta, J. Rodrigo, J. A. Mendez, G. N. Marichal and M. Sigut, "Ping-pong player prototype," *IEEE Robot. Autom. Mag.* **10**(4), 44–52 (2003).
[2] H. Bao, X. Chen, Z. T. Wang, M. Pan and F. Meng. "Bouncing Model for the Table Tennis Trajectory Prediction and the Strategy of Hitting the Ball," **In:** *2012 IEEE International Conference on Mechatronics and Automation*, IEEE (2012) pp. 2002–2006.
[3] Y. Cohen, O. Bar-Shira and S. Berman, "Motion adaptation based on learning the manifold of task and dynamic movement primitive parameters," *Robotica* **39**(7), 1299–1315 (2021).
[4] Y. Huang, D. Büchler, O. Koç, B. Schölkopf and J. Peters. "Jointly Learning Trajectory Generation and Hitting Point Prediction in Robot Table Tennis," **In:** *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, IEEE (2016) pp. 650–655.
[5] Y. Huang, D. Xu, M. Tan and H. Su, "Adding active learning to LWR for ping-pong playing robot," *IEEE Trans. Contr. Syst. Technol.* **21**(4), 1489–1494 (2012).
[6] A. J. Ijspeert, J. Nakanishi and S. Schaal, "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots," **In:** *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH3729)*, IEEE, (2002) pp. 1398–1403.
[7] J. Li, X. Chen, Q. Huang, X. Chen, Z. Yu and Y. Duo, "Designation and control of landing points for competitive robotic table tennis," *Int. J. Adv. Robot. Syst.* **12**(7), 92 (2015).
[8] H.-I. Lin, Z. Yu and Y.-C. Huang, "Ball tracking and trajectory prediction for table-tennis robots," *Sensors* **20**(2), 333 (2020).
[9] M. Matsushima, T. Hashimoto and F. Miyazaki, "Learning to the Robot Table Tennis Task-Ball Control & Rally with a Human," **In:** *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, IEEE, (2003) pp. 2962–2969.
[10] M. Matsushima, T. Hashimoto, M. Takeuchi and F. Miyazaki, "A learning approach to robotic table tennis," *IEEE Trans. Robot.* **21**(4), 767–771 (2005).
[11] K. Muelling, J. Kober and J. Peters. "Learning Table Tennis with a Mixture of Motor Primitives," **In:** *2010 10th IEEE-RAS International Conference on Humanoid Robots*, IEEE (2010) pp. 411–416.
[12] A. Nakashima, D. Ito and Y. Hayakawa. "An Online Trajectory Planning of Struck Ball with Spin by Table Tennis Robot," **In:** *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, IEEE (2014) pp. 865–870.
[13] A. Nakashima, J. Nonomura, C. Liu and Y. Hayakawa, "Hitting back-spin balls by robotic table tennis system based on physical models of ball motion," *IFAC Proc. vol.* **45**(22), 834–841 (2012).
[14] A. Nakashima, Y. Ogawa, Y. Kobayashi and Y. Hayakawa. "Modeling of Rebound Phenomenon of a Rigid Ball with Friction and Elastic Effects," **In:** *Proceedings of the 2010 American Control Conference*, IEEE (2010) pp. 1410–1415.
[15] A. Nakashima, Y. Ogawa, C. Liu and Y. Hayakawa. "Robotic Table Tennis Based on Physical Models of Aerodynamics and Rebounds," **In:** *2011 IEEE International Conference on Robotics and Biomimetics*, IEEE (2011) pp. 2348–2354.
[16] J. Nonomura, A. Nakashima and Y. Hayakawa. "Analysis of Effects of Rebounds and Aerodynamics for Trajectory of Table Tennis Ball," **In:** *In, Proceedings of SICE Annual Conference.*, IEEE (2010) pp. 1567–1572.
[17] S. Schaal, J. Peters, J. Nakanishi and A. Ijspeert. "Learning Movement Primitives," **In:** *The eleventh International Symposium on Robotics Research*, Springer (2005) pp. 561–572.
[18] H. Yang, Y. Yi, G. Liu and X. Shi. "CS-BS-EF: An Integrated Method for Tracking Flying Ping-Pong with Motion Blur," **In:** *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, IEEE (2013) pp. 1220–1223.
[19] P. Yang, D. Xu, H. Wang and Z. Zhang. "Control System Design for a 5-DoF Table Tennis Robot," **In:** *2010 11th International Conference on Control Automation Robotics & Vision*, IEEE (2010) pp. 1731–1735.
[20] K. Zhang, Z. Fang, J. Liu and M. Tan. "Modeling the Stroke Process in Table Tennis Robot Using Neural Network," **In:** *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE (2015) pp. 243–248.
[21] J. Zhi, D. Luo, K. Li, Y. Liu and H. Liu, "A novel method of shuttlecock trajectory tracking and prediction for a badminton robot," *Robotica* **40**(6), 1682–1694 (2022).

Cite this article: H.-I. Lin and C.-F. Syu (2024). "Table tennis ball landing control in a robotic system by cameras", Robotica **42**, 3867–3887. https://doi.org/10.1017/S0263574724001693