Sensors and Materials, Vol. 37, No. 1 (2025) 75–91
MYU Tokyo

S & M 3892

Binocular-vision-based Trajectory Prediction of Spinning Ball for Table Tennis Robot

Chung-Hsun Sun,[1] Ying-Shu Chuang,[1] Chun-Ting Liu,[1] and Hsiang-Chieh Chen[2*]

[1]Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung City 807618, Taiwan
[2]Department of Mechanical Engineering, National Central University, Taoyuan City 302317, Taiwan

(Received August 22, 2024; accepted December 27, 2024)

Keywords: binocular vision system, table tennis robot, trajectory prediction, deep learning

Establishing a reliable vision system is essential to developing table tennis robots as it provides vital information for accurately controlling the robot's movement to hit a ping-pong ball. In this study, we present a vision system and its main algorithm, which can detect and track a ball, classify its spin, and estimate the contact point and hitting time on a predefined hitting region. To achieve this, we used two well-calibrated cameras to construct a binocular vision system for detecting and positioning the ball in 3D space. Then, two deep-learning models were separately deployed to classify the ball spin type and predict the ball's trajectory. The contact point on the hitting plane and the hitting time were finally estimated. Experimental results showed that the proposed vision system performed well, with acceptable estimation errors for the contact point and hitting time, which are feasible for controlling our designed robot to strike a table tennis ball.

1. Introduction

Robotic systems have become increasingly common in various forms alongside humans, including service robots, unmanned vehicles, and industrial robots. However, some robot types, including sports-specific robots, still have not yet been widely produced or utilized in everyday life. A table tennis robot is an excellent platform for sports robotic research because it requires less operating space than other sports robots. Such a robot is also a testbed for computer vision, control practice, electromechanical integration, and intelligent theories. We aim to create table tennis robots that can play intelligently with humans or other robots. Several technologies such as vision system construction, object detection and tracking, robot motion planning, and real-time control are essential in developing a table tennis robot. Establishing the robot's vision is particularly crucial in this research field. Therefore, in this study, we focus on introducing the vision system and its integral algorithms for our proposed table tennis robot.

In the 2000s, a low-cost robot prototype was developed to play ping-pong against a human opponent.(1) It used only a single camera to detect the ball utilizing the shadow of the ball cast on

*Corresponding author: e-mail: hcchen@cc.ncu.edu.tw
https://doi.org/10.18494/SAM5322

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

the table. Matsushima *et al.*[2] suggested a learning-based approach for a robotic system to perform the table tennis task, which involved obtaining stroke movement and paddle condition at the impact point. Yu *et al.* implemented a humanoid table tennis robot with a 7-degree-of-freedom arm.[3] They used distributed controllers and a real-time binocular vision system, which enabled ball identification, trajectory prediction, and racket motion planning.

The visual system plays a crucial role in the functioning of a table tennis robot because it acts as the robot's eyes and provides vital information about the ball. Developing such a vision system involves solving several issues, such as ball tracking, trajectory prediction, and striking control. Trajectory prediction is critical as it enables the robot to strike the ball accurately. There are two main categories of trajectory prediction methods: model-based and learning-based. Model-based methods use force analysis to derive the ball's flying and rebound models relative to the table. A nonlinear method based on dynamic and bouncing models was proposed to predict trajectory when several initial trajectory points are known.[4] In addition, a binocular vision system was developed to predict ball trajectory using aerodynamics and rebound models with six consecutive coordinates of the detected ball.[5] Several trajectory prediction methods using dynamic/flying and rebound/bouncing models have been proposed,[6–9] with similar method of derivation. Learning-based methods provided new opportunities for time-series prediction with real-time constraints. Neural networks were used to estimate the object's position, velocity, and acceleration in a robotic environment using the last six measurements as inputs.[10]

The uncertainty of a ball's flight caused by its rotation (or spin) is a significant limitation when predicting its trajectory. Although a real-time prediction model based on binocular vision was designed to construct 3D information on table tennis,[11] accurately detecting ball rotation remained impossible. Tebbe *et al.*[12] evaluated three methods for spin detection using the movement of the ball's logo or its trajectory. The first two approaches used high-speed cameras to capture the logo printed on the ball in flight. Then, image processing and deep-learning techniques were used to predict the logo orientation. Meanwhile, the third method considered the effect of the Magnus force and attained a relatively high spin detection accuracy. Zhang *et al.* used the brand logo to estimate the ball's spinning state.[13] In addition, Zhang *et al.* also derived a continuous motion model of a spinning ball based on force analysis and proposed a model-based optimal algorithm for the ball's motion state.[14] The ball's rotational velocity after rebound can be estimated from its positions.[15] Wang *et al.* used a back-propagation neural network to identify the pattern in accordance with the predicted flight trajectory.[16]

Recently, machine learning techniques have been rapidly developed, allowing for several studies based on learning methodologies to predict a spinning ball's trajectory, including direction and speed.[17,18] For example, Liu and Ding used artificial intelligence algorithms to calculate and predict a table tennis ball's rotation, trajectory, and velocity.[17] They also validated their results by conducting various experiments and observed a close correlation between speed and spin. Gomez–Gonzalez *et al.* introduced a prediction method combining deep learning and conditional generative models,[18] in which the method represented the entire trajectory using on autoencoders. Aside from physical models, Achterhold *et al.* used a neural network to improve long-time prediction performance.[9] Li used two deep learning algorithms, spatial upsampling and reconstruction encoder and long short-term memory (LSTM), to analyze table tennis ball

trajectories.[19] The accuracy rate was 96.5%, but the prediction speed averaged 15 s, which is relatively slow for real-time applications. The trajectory prediction model for the table tennis ball in Ref. 20 is also based on LSTM. In another study, Li *et al.* proposed an algorithm that combines structured-output convolution neural network and LSTM.[21] LSTM is commonly used to predict the trajectory of a ball. Song and Lu briefly discussed the use and application of LSTM in a table tennis robot.[22] The method in Ref. 23 was inspired by the gated recurrent unit, which performs similarly to the LSTM network. It also predicted the ball's position (p), velocity (v), and acceleration (a). However, collecting ground truth data for p, v, and a is difficult; hence, the training data in Ref. 23 were obtained through simulation.

From the summary of the papers mentioned above, spin information, including direction and speed, is challenging to estimate but can significantly improve long-term prediction accuracy. Hence, in this study, we focus on trajectory prediction under different rotation directions and velocities. Specifically, we designed and implemented a vision system and its algorithms that can predict the entire trajectory of a table tennis ball. Moreover, we estimated the contact point position and hitting time on a predefined hitting plane, which can be used to control a robot to strike a ball.

2. Proposed Binocular Vision System

In this section, we introduce the proposed binocular vision system comprising two high-speed cameras to locate and track a ping-pong ball in 3D space. Figure 1 shows the main components of our proposed system, including two cameras, a table-tennis table, a computer, and a ball launcher. Here, the two cameras are well-calibrated in advance and placed about 2.4 m above the ground. The computer has a graphics processing unit to accelerate image processing, vision analyses, and deep-learning model inference. Additionally, the ball launcher serves the ball to collect training data and tests. On the basis of the above configurations, we proposed algorithms for ball detection, positioning and tracking, and trajectory prediction for spinning balls with different rotation directions. Accordingly, the contact point position and hitting time are finally obtained and can be used to control a robot to strike the ball. We focused on the algorithm of the proposed vision system but do not discuss the control of the robot.

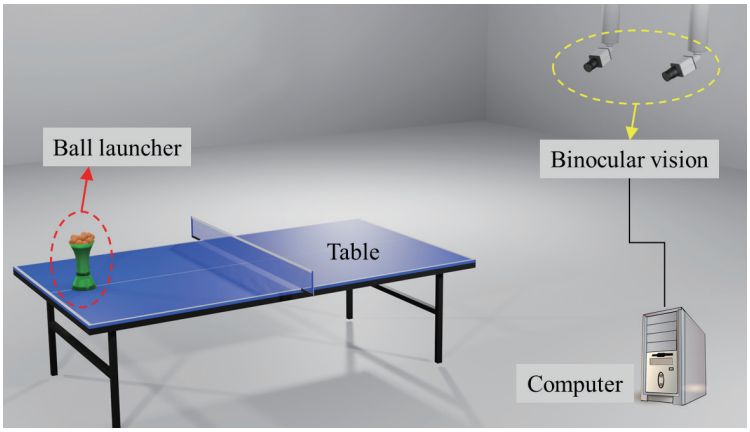


Fig. 1. (Color online) Overview of the proposed vision system.

2.1 Camera calibration

The camera calibration process, which is designed to determine the intrinsic and extrinsic parameters of our proposed visual system's cameras, is essential to obtaining accurate binocular vision. In particular, several open-source toolkits have been widely used, and their effectiveness in camera calibration has been verified. In this section, we selected the Matlab camera calibration toolbox to estimate intrinsic, extrinsic, and lens distortion parameters per camera, as well as the stereo camera calibrator for their geometric positions and orientations in a binocular camera pair. The calibration results will be given in the experiment section.

2.2 Ball detection in images

After camera calibration, the camera position and orientation settings can be obtained. Consequently, we first set a region of interest (ROI) in every camera's captured image, through which the ball always flies. In this work, the ball detection is only performed within the ROI; thus, the computation can be significantly reduced. Next, image processing techniques are used sequentially to detect the ping-pong ball, including color model transformation, target object extraction, noise removal and hole filling, and ball centroid calculation.

The image frame captured from cameras is originally encoded in the RGB color model. However, this model has the disadvantage of producing inconsistent detection results owing to varying lighting conditions. To solve this problem, the RGB model is first transformed into the hue-saturation-value (HSV) color model, which is then used for subsequent analytics. The color of a table tennis ball is always a single color, such as orange or white, which can be distinguished from the background by setting specific ranges of H, S, and V values. When adjusting HSV intervals to detect the ping-pong ball, there is a possibility of detecting other undesired objects that share a similar color to the ball. The well-known median filter is used for noise removal because of its computational efficiency, and it also helps to fill the holes inside an extracted ball object.

Even after noise removal, some undesired noise or tiny objects may still exist. To address this issue, each detected object is extracted using connected-component labeling and then contoured by a minimum-enclosing circle. The object with the most pixels inside the circle is considered to be the ping-pong ball. The centroid of the object is calculated using the same labeling approach. The results of ball detection using these steps within the two-view image frames are shown in Fig. 2. To reduce the computation time of ball detection in every image, a tracking window is used to replace the ROI after the ball is detected. According to our experiments, this tracking window significantly reduces computation time and improves tracking robustness.

2.3 Ball positioning in 3D space

After ball detection in images, the ball positioning in 3D space is determined on the basis of nonstandard stereo geometry (nonSSG), as shown in Fig. 3. Let $\boldsymbol{p}^l = \left[p_u^l, p_v^l \right]^T$ and $\boldsymbol{p}^r = \left[p_u^r, p_v^r \right]^T$ be the image center coordinates of the detected ball in the left and right views,

(a) (b)

Fig. 2. (Color online) Ball detection results in (a) left-view and (b) right-view images.

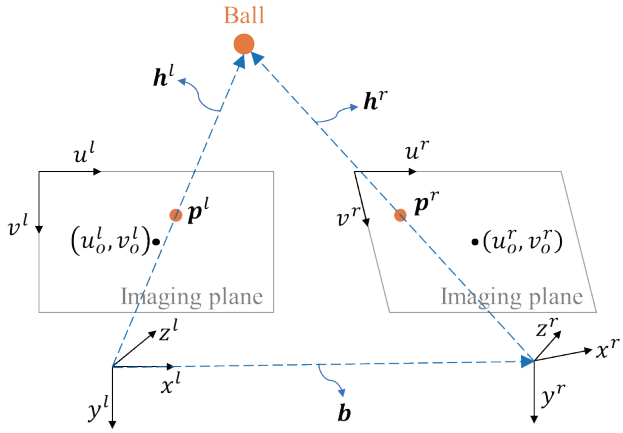


Fig. 3. (Color online) Geometric relationship in nonSSG concept.

respectively. Here, the subscripts u and v are the horizontal and vertical axes in the image frame, and the superscripts l and r denote the left and right views, respectively. After the camera calibration, Eq. (1) can be derived on the basis of image formation principles.

$$\begin{cases} \boldsymbol{p}^l = \left[p_u^l,\ p_v^l \right]^T = \left[u_o^l + f_u^l \left(h_x^l / h_z^l \right),\quad v_o^l + f_v^l \left(h_y^l / h_z^l \right) \right]^T \\ \boldsymbol{p}^r = \left[p_u^r,\ p_v^r \right]^T = \left[u_o^r + f_u^r \left(h_x^r / h_z^r \right),\quad v_o^r + f_v^r \left(h_y^r / h_z^r \right) \right]^T \end{cases} \tag{1}$$

Here, $\left(u_o^l, v_o^l\right)$ and $\left(u_o^r, v_o^r\right)$ are optical geocenters in their corresponding images. $\left(f_u^l, f_v^l\right)$ and $\left(f_u^r, f_v^r\right)$ are focal lengths of the left and right cameras, respectively. $\boldsymbol{h}^l = \left[h_x^l, h_y^l, h_z^l \right]^T$ is the position vector based on the left-view coordinate system. Equation (1) is then reformulated as follows.

$$\begin{cases} \left(h_x^l,\ h_y^l \right) = \left(\left(p_u^l - u_o^l \right) h_z^l / f_u^l,\quad \left(p_v^l - v_o^l \right) h_z^l / f_v^l \right) \\ \left(h_x^r,\ h_y^r \right) = \left(\left(p_u^r - u_o^r \right) h_z^r / f_u^r,\quad \left(p_v^r - v_o^r \right) h_z^r / f_v^r \right) \end{cases} \tag{2}$$

As plotted in Fig. 3, Eq. (2) is obtained by vector addition and coordinate transformation theory:

$$\boldsymbol{h}^l = \boldsymbol{R}_r^l \boldsymbol{h}^r + \boldsymbol{b}, \tag{3}$$

where $\boldsymbol{R}_r^l$ is the rotation matrix from the right view to the left view, and b is the translation vector from the center of the left camera to the right camera. Also, we have

$$\begin{aligned} \boldsymbol{h}^r &= \begin{bmatrix} h_x^r & h_y^r & h_z^r \end{bmatrix}^T = \left(\boldsymbol{R}_r^l\right)^{-1}\left(\boldsymbol{h}^l - \boldsymbol{b}\right) = \boldsymbol{R}_l^r \boldsymbol{h}^l - \boldsymbol{R}_l^r \boldsymbol{b} \\ &= \boldsymbol{R}_l^r \begin{bmatrix} h_x^l \\ h_y^l \\ h_z^l \end{bmatrix} - \boldsymbol{R}_l^r \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} h_x^l \\ h_y^l \\ h_z^l \end{bmatrix} - \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}. \end{aligned} \tag{4}$$

By using Eqs. (2) and (4), we have

$$\begin{aligned} & a_{11}\frac{\left(p_u^l - u_o^l\right)}{f_u^l} h_z^l + a_{12}\frac{\left(p_v^l - v_o^l\right)}{f_v^l} h_z^l + a_{13} h_z^l - d_1 \\ &= \frac{\left(p_u^r - u_o^r\right)}{f_u^r}\left[a_{31}\frac{\left(p_u^l - u_o^l\right)}{f_u^l} h_z^l + a_{32}\frac{\left(p_v^l - v_o^l\right)}{f_v^l} h_z^l + a_{33} h_z^l - d_3 \right]. \end{aligned} \tag{5}$$

Therefore,

$$\boldsymbol{h}^l = \begin{bmatrix} h_x^l \\ h_y^l \\ h_z^l \end{bmatrix} = \begin{bmatrix} \frac{1}{k}\frac{\left(p_u^l - u_o^l\right)}{f_u^l}\left[d_1 - \frac{\left(p_u^r - u_o^r\right)}{f_u^r} d_3 \right] \\ \frac{1}{k}\frac{\left(p_v^l - v_o^l\right)}{f_v^l}\left[d_1 - \frac{\left(p_u^r - u_o^r\right)}{f_u^r} d_3 \right] \\ \frac{1}{k}\left[d_1 - \frac{\left(p_u^r - u_o^r\right)}{f_u^r} d_3 \right] \end{bmatrix}, \tag{6}$$

where

$$k = \left[a_{11}\frac{\left(p_u^l - u_o^l\right)}{f_u^l} + a_{12}\frac{\left(p_v^l - v_o^l\right)}{f_v^l} + a_{13} \right] - \frac{\left(p_u^r - u_o^r\right)}{f_u^r}\left[a_{31}\frac{\left(p_u^l - u_o^l\right)}{f_u^l} + a_{32}\frac{\left(p_v^l - v_o^l\right)}{f_v^l} + a_{33} \right]. \tag{7}$$

Now, a world coordinate system is indispensable to provide a unified reference; thus, integrating vision and robot systems becomes possible. Here, we set the origin of the world coordinate system at the corner closest to the left camera. Figure 4(a) illustrates the left-camera, right-camera, and world coordinate systems used in this study. Consequently, as shown in Fig. 4(b), the position vector with reference to the world coordinate system is calculated using

$$\boldsymbol{h}^w = \boldsymbol{R}_l^w \boldsymbol{h}^l + \boldsymbol{b}_0^w, \tag{8}$$

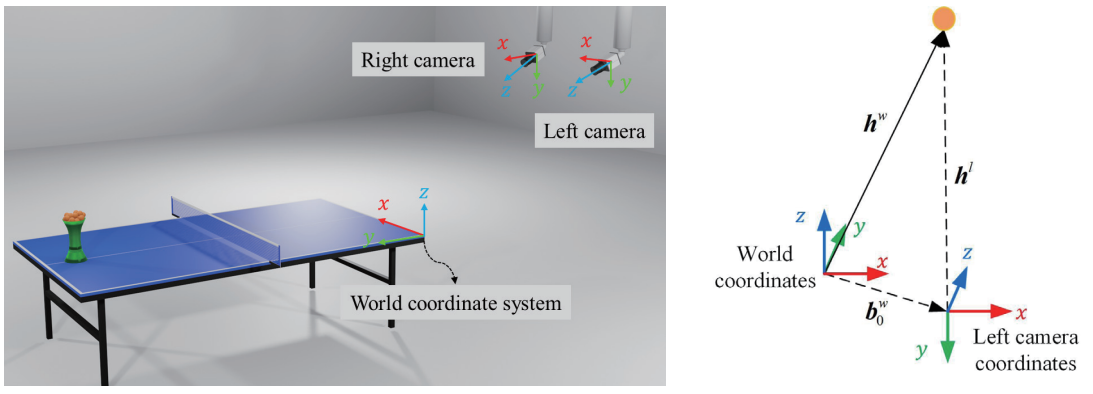
where $\boldsymbol{b}_0^w$ denotes the translation from the world coordinate to the left-camera coordinate system; $\boldsymbol{R}_l^w$ is the rotation matrix from the left camera to the world coordinate system's orientation.

3. Trajectory Prediction of Spinning Ball

Let $\boldsymbol{h}^w(t) = \left[h_x^w(t), h_y^w(t), h_z^w(t)\right]$ be the position of the ball with reference to the world coordinate system at time t, and is herein written as $\boldsymbol{h}(t) = \left[h_x(t), h_y(t), h_z(t)\right]$. When N measures of $\boldsymbol{h}(t)$ for $t = t_1, t_2, ..., t_N$ detected from N consecutive image frames form a trajectory segment of the flying ball, the entire trajectory can be estimated using projectile motion formulae. However, the rotation of the ping-pong ball itself always causes changes in the trajectory owing to its light weight and aerodynamic influence. In this section, we introduce a method that contains two deep-learning-based models to estimate the flight trajectory taking into consideration the ball's rotation. Precisely, one model is for ball spin classification, and the other is for trajectory prediction under a specified spin type.

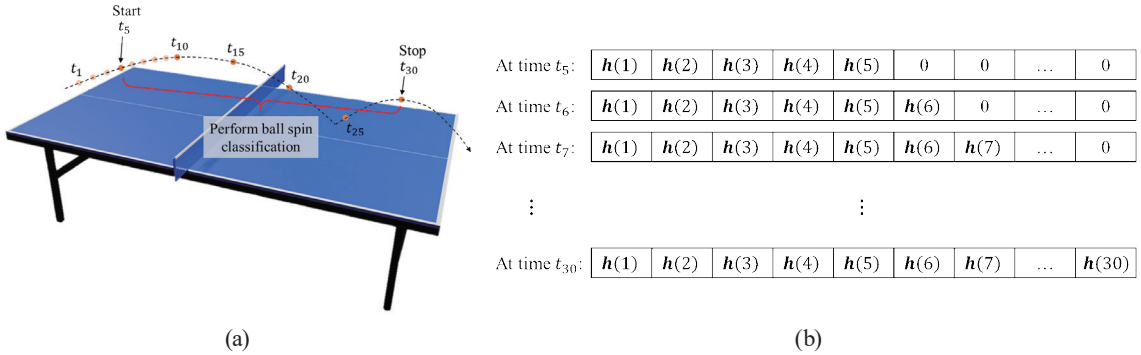
3.1 Deep-learning-based prediction models

We first used an LSTM network to classify the spin type of a flying ping-pong ball. This model will start after receiving five consecutive coordinates of the ball until 30 records are collected. Figure 5(a) shows that the classification model is enabled in $[t_5, t_{30}]$, named the



(a) (b)

Fig. 4. (Color online) (a) Definition of coordinate systems and (b) geometry between the world and left-camera coordinate systems.



(a) (b)

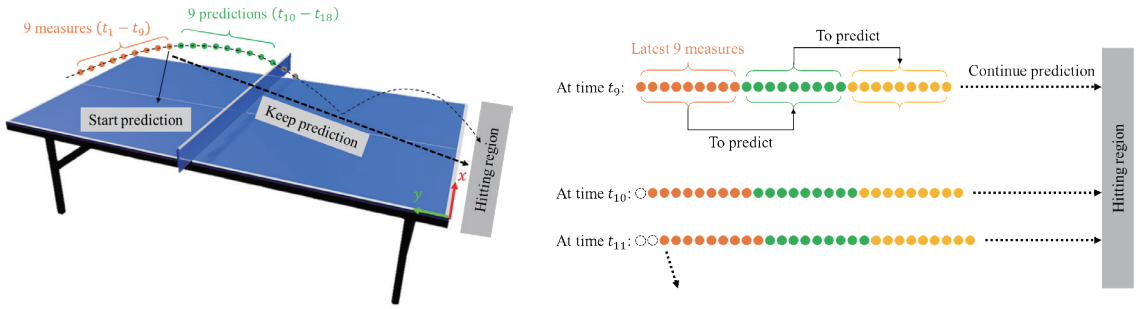
Fig. 5. (Color online) (a) Performing period of ball spin classification and (b) input sequence at different times during the performing period.

performing period. To fix the length of the input to the model, we define the input as a 30 × 1 vector, as presented in Fig. 5(b). The input sequence is filled with the measured coordinates $\boldsymbol{h}(t)$s for $t = t_{30}$ and the empty element is zero-padded for $t < t_{30}$. Now, we describe the output dimension of the ball spin classification. Generally, most commercially available ping-pong ball machines offer four spin modes: topspin, backspin, left-side spin, and right-side spin. Accordingly, we designed an LSTM-based classification model that can identify the four spin types at S different flying speeds, i.e., there are $4S$ spin types that our proposed model can classify. Hence, the output is set as a $4S \times 1$ vector, indicating the classified categories. The implementation details will be discussed in the experimental section.

After obtaining the classification results, the next step is to predict the trajectory in accordance with the spin classes. We trained $4S$ LSTM models corresponding to the $4S$ spin categories for prediction accuracy. Figure 6(a) shows our concept of how the prediction model works. More specifically, upon receiving nine measures from consecutive image frames, one of the trained prediction models begins to predict the future trajectory. This continues until the predicted trajectory reaches the hitting region where the robot is expected to strike the ball. Here, the trajectory prediction process adopts an iterative scheme that uses the predicted output to be fed back to the input, as plotted in Fig. 6(b). More precisely, the prediction model collected nine measures (orange dots), used them to predict the next nine predicted points (green dots), and then used these nine green dots to predict the next nine yellow points. Therefore, the entire trajectory from the latest measures to the hitting plane can be predicted by repeating the prediction model several times.

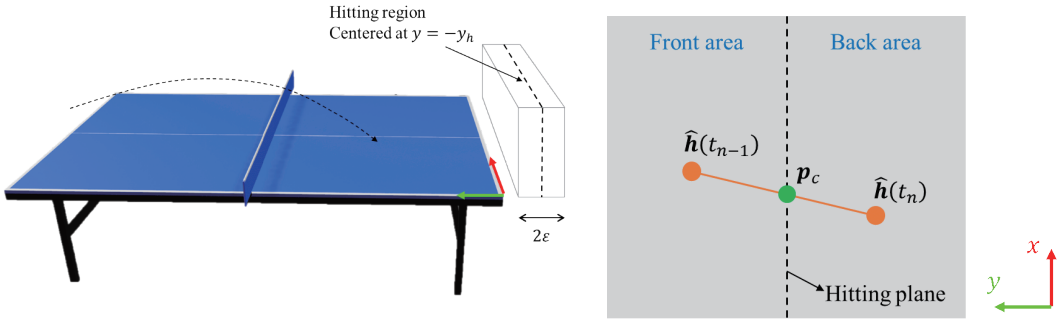
3.2 Contact point position and hitting time estimation

The hitting region in Fig. 7(a) indicates the area within which the table tennis robot would hit the ball. It is a 3D space with a height of about 60 cm above the table, the same width as the table, and is distributed in the interval $[-y_h - \varepsilon, -y_h + \varepsilon]$. It depends on the working area of the employed table tennis robot. In our experiments, $y_h = 45$ cm and $\varepsilon = 5$ cm. The central line located at $y = y_h$ is regarded as the hitting plane. With the central line of the region as the boundary, the area near the table is called the front area, and the opposite side is called the back



(a) (b)

Fig. 6. (Color online) (a) Ball trajectory prediction and (b) iterative scheme for predicting the trajectory.



(a) (b)

Fig. 7. (Color online) (a) Hitting region presented as a 3D space, and (b) interpolation of contact point.

area, as shown in Fig. 7(b). Since the predicted trajectory point does not exactly fall on the hitting plane every time, we use an interpolation method to find the contact point that exactly falls on the hitting plane. Figure 7(b) shows a zoom-in part around the hitting plane, in which the orange dots are the two points closest to the hitting plane and on different sides. Let $\hat{\boldsymbol{h}}(t_{n-1})$ and $\hat{\boldsymbol{h}}(t_n)$ be two predicted points before and after the hitting plane, respectively.

Assuming that the contact point is denoted by $\boldsymbol{p}_c = [x_c, y_c, z_c]^T$, the distance ratio is

$$\delta = \frac{y_c - \hat{y}(t_{n-1})}{\hat{y}(t_n) - y(t_{n-1})}. \tag{9}$$

Therefore, the contact point $\boldsymbol{p}_c$ and the hitting time t_c can be interpolated as follows.

$$\begin{cases} \boldsymbol{p}_c = (1-\delta)\hat{\boldsymbol{h}}(t_{n-1}) + \delta\hat{\boldsymbol{h}}(t_n) \\ t_c = (1-\delta)t_{n-1} + \delta t_n \end{cases} \tag{10}$$

Notably, the above derivations are based on the assumption that the ball is moving in a straight line with constant speed in a very short time around the hitting plane.

4. Implementation and Experimental Results

We focused on a binocular vision system and ball trajectory prediction that can be used in a robotic system for playing table tennis, particularly to strike the spinning ball. Hence, implementation details and some experimental results are discussed in this section.

4.1 Binocular vision establishment

As mentioned at the beginning of Sect. 2.1, we first used the toolkit provided by Matlab software to obtain the camera parameters. Table 1 lists the intrinsic parameters of two cameras in which the lens distortion coefficients contain the radial distortion factors (k_1, k_2, k_3) and the tangential factors (p_1, p_2). Precisely, the image resolution in our system is 2048 × 1536 pixels.

The extrinsic parameters describing the relative geometry between the two cameras are as follows:

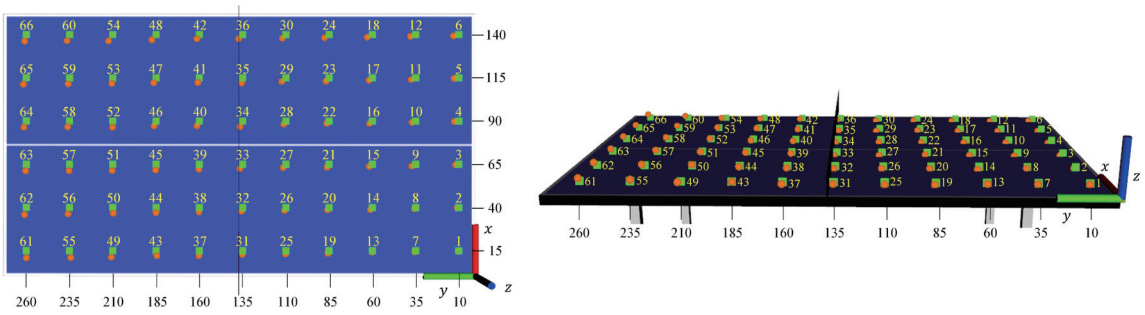
$$\begin{cases} \boldsymbol{R}_r^l = \begin{bmatrix} 0.9633 & -0.1044 & 0.2474 \\ 0.1166 & 0.9925 & -0.0353 \\ -0.2419 & 0.0628 & 0.9683 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -840.3844 \\ -115.5291 \\ 232.1445 \end{bmatrix} \\ \boldsymbol{R}_w^l = \begin{bmatrix} 0.9994 & -0.0019 & -0.0106 \\ -0.0091 & -0.3722 & -0.9281 \\ -0.0057 & 0.9281 & -0.3722 \end{bmatrix}, \quad \boldsymbol{b}_0^w = \begin{bmatrix} -699.6207 \\ 450.7032 \\ 2042.7389 \end{bmatrix} \end{cases}.$$

To evaluate the positioning accuracy of our vision system, we manually placed balls uniformly distributed on the table and then calculated the positioning errors. Figure 8 shows the positioning results, where the green squares represent the actual ball position we placed in the experiment, and the orange dots are the vision-based positioning results. The average positioning errors are 2.28 cm on the x axis, 1.08 cm on the y axis, and 1.10 cm on the z axis. The maximum error at the farthest point is 4.2 cm. More specifically, the closer to the camera, the more accurate the positioning.

Table 1
Intrinsic parameters of our cameras.

Camera	Parameter			
	Focal length	Optical center	Radial distortion factors	Tangential distortion factors
Left camera	$f_u^l = 1766.56$	$u_o^l = 1038.99$	$k_1^l = 0.00826$	$p_1^l = -0.00005$
	$f_v^l = 1766.80$	$v_o^l = 783.18$	$k_2^l = -0.01795$, $k_3^l = 0$	$p_2^l = -0.00192$
Right camera	$f_u^r = 1764.20$	$u_o^r = 1056.78$	$k_1^r = 0.00772$	$p_1^r = -0.00074$
	$f_v^r = 1765.55$	$v_o^r = 773.98$	$k_2^r = -0.01115$, $k_3^r = 0$	$p_2^r = -0.00084$

Note: Each camera (GS3-U3-32S4C-C) is produced by FLIR Systems, Inc. and is equipped with a SONY CMOS sensor. This system provides an image resolution of 2048 × 1536 pixels and operates at a frame rate of 60 FPS.



(a) (b)

Fig. 8. (Color online) Positioning results: (a) top view and (b) side view.

4.2 Experiments of ball spin classification

The proposed ball spin classification model can identify four spin classes with S different velocities, i.e., the model's output is a $4S \times 1$ vector that presents the probabilities of all classes. Specifically, our classification model comprised two parts, namely, a feature extractor and a classifier, in which six cascaded LSTM-based layers formed the extractor, and the classifier comprised dense and softmax layers, as illustrated in Fig. 9. In the experiments, we set $S = 2$, denoting two different velocities, "LOW" velocity (level-5) and "HIGH" velocity (level-6), of the ping-pong ball thrown by the ball launcher (CRACK V-981). Hence, the total number of classes is 8, and the detailed descriptions for all considered classes are summarized in Table 2. The notations TS, LS, RS, and BS represent topspin, left-side spin, right-side spin, and backspin, respectively.

First, we used the ball launcher to launch eight categories of ball trajectories, and each trajectory type had a total of 750 shots from a fixed position and 750 shots from random positions. Hence, data of a total of 12000 trajectories were collected to form training datasets. In addition, we collected 2000 trajectories shot from a fixed position and 2000 trajectories from random shooting positions for the test. Tables 3 and 4 list the confusion matrix for testing the data shot from fixed and random positions. The overall accuracy of the test data was 99.475%. The experiments with results presented in Table 3 were carried out with the ball launcher fixed at a specific position to serve the ball, and the ball was consistently shot from position $\boldsymbol{h}^w = [76.5, 309, 85]^T$. In this case, classification errors occurred only with velocity, while the classification of spin types was accurate. In contrast, the experiments with results in Table 4 involved random shots from various positions, where $-5 \le h_x^w \le 170$ and $275 \le h_y^w \le 335$. In this table, misclassifications occurred for both velocity and spin types.

4.3 Experiments of trajectory prediction

We conducted a thorough analysis of trajectory prediction accuracy by evaluating the position of the contact point and the duration of the hitting time. In this section, we introduce the proposed trajectory prediction model, which is structured as an encoder–decoder architecture,

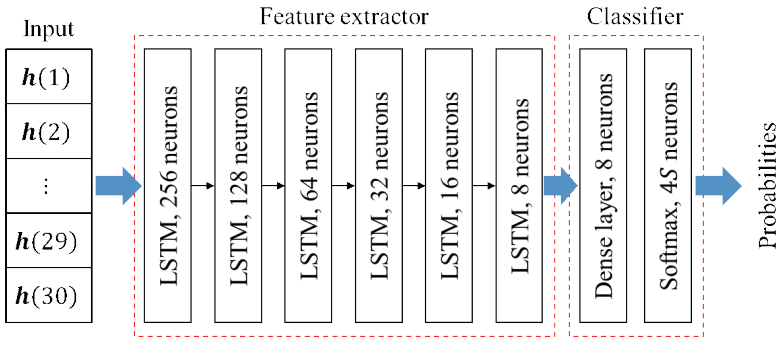


Fig. 9. (Color online) Architecture of our proposed classification model.

Table 2
Spin classes in our experiments.

Category number	1	2	3	4	5	6	7	8
Spin type	TS	TS	LS	LS	RS	RS	BS	BS
Velocity	LOW	HIGH	LOW	HIGH	LOW	HIGH	LOW	HIGH

Table 3
Confusion matrix of the ball spin classification on test data (from a fixed position).

		Predicted class							
		1	2	3	4	5	6	7	8
True class	1	0.98	0.02	0	0	0	0	0	0
	2	0.008	0.992	0	0	0	0	0	0
	3	0	0	0.996	0.004	0	0	0	0
	4	0	0	0.008	0.992	0	0	0	0
	5	0	0	0	0	1	0	0	0
	6	0	0	0	0	0.008	0.992	0	0
	7	0	0	0	0	0	0	1	0
	8	0	0	0	0	0	0	0	1

Table 4
Confusion matrix of the ball spin classification on test data (from random positions).

		Predicted class							
		1	2	3	4	5	6	7	8
True class	1	1	0	0	0	0	0	0	0
	2	0	0.992	0	0.004	0.004	0	0	0
	3	0	0	0.996	0.004	0	0	0	0
	4	0	0	0	0.996	0	0.004	0	0
	5	0	0	0	0	0.996	0.004	0	0
	6	0	0.004	0	0	0.004	0.992	0	0
	7	0	0	0	0.004	0	0	0.992	0.004
	8	0	0	0	0	0	0	0	1

as depicted in Fig. 10. The model comprises an encoder and a decoder, with each containing a single 256-neuron LSTM layer. A more accurate estimation of the contact point and hitting time implies a better trajectory prediction; thus, the table tennis robot is likelier to hit the ball.

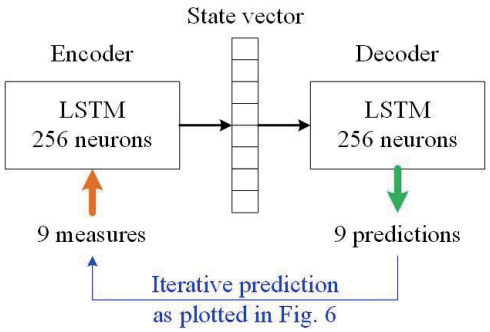


Fig. 10. (Color online) Architecture of our proposed trajectory prediction model.

Additionally, we compared two conditions in the experiments: one was that nine measures were captured for trajectory prediction, and the other was that five measures were used for prediction. Figure 11 shows the error bars of hitting time prediction (when the spin category was #5 in Table 2), where the horizontal axis indicates the time stamps since the forecast started. Precisely, the average prediction error is near zero, and its standard deviation is reduced to less than 10 ms. Moreover, we completed the experiments for all spin classes, and their experimental results were consistent with a small error. Similarly, we analyzed the prediction error of the contact point's position in Fig. 12, in which the prediction errors on the x and z axes were drawn when the hitting plane was set at $y = -45$ cm. Specifically, the prediction errors decreased as time elapsed, and finally, the errors became less than 2–3 cm. Because the length and width of a ping-pong racket is 15–17 cm, we believe that such an error range is sufficient to hit the ball. In our experiments, the total computation time for ball spin classification and trajectory prediction is always less than 16.67 ms. In this work, we compared two different conditions: one with 5 time steps and the other with 9 time steps, as indicated in the legends of Figs. 11 and 12. After receiving nine measurements from consecutive image frames, one of the trained prediction models began to forecast the future trajectory, referred to as "9 time steps." Similarly, the term case of "5 time steps" describes the process of initiating trajectory prediction after obtaining five measurements from consecutive image frames. In the sequence-to-sequence prediction mechanism, longer input sequences typically contain more information, leading to more accurate predictions. Conversely, models with shorter input sequences generate results more quickly. Our experiments indicate that the model based on "9 time steps" produces slower, yet more precise predictions compared with the case of "5 time steps". The prediction results from the "9 time steps" model are slightly better than those from the "5 time steps" model. Since the prediction time for the entire trajectory using the "9 time steps" model is faster than the intervals at which the camera captures images, we have chosen to adopt the "9 time steps" model owing to its greater accuracy.

4.4 Estimation error of contact point and hitting time

The objective of this study is to introduce a visual system with the capability to guide a table tennis robot accurately in hitting a ping-pong ball. As such, the precision of determining

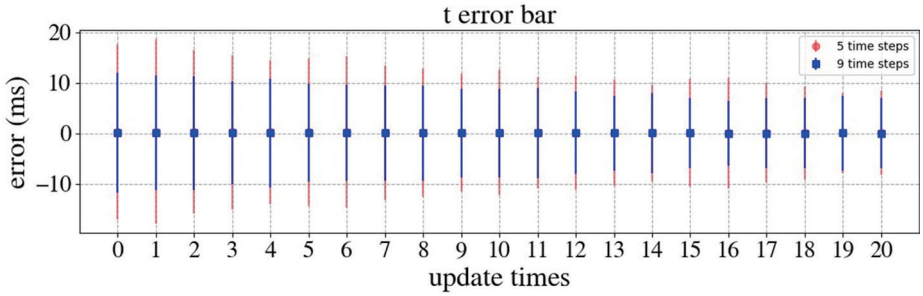


Fig. 11. (Color online) Prediction error of hitting time.

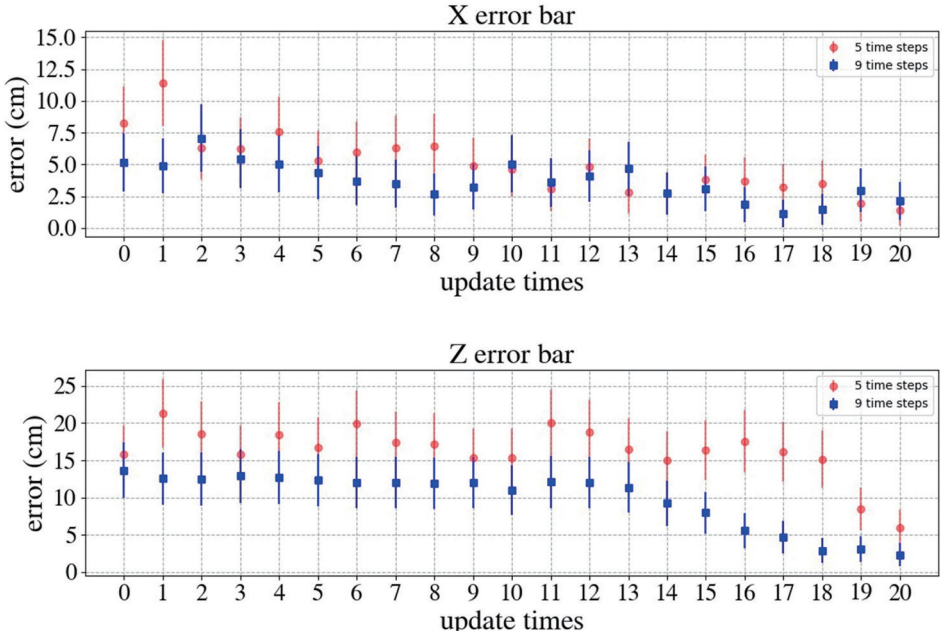


Fig. 12. (Color online) Prediction error of contact point position.

the exact point of contact and time of the strike is of utmost importance and will be elaborated upon in detail within this subsection. On the basis of observations from Fig. 13, where the horizontal axis shows the elapsed time after receiving 9 measures (time t_9), it can be concluded that the proposed method accurately estimated the contact point and hitting time over time. The first bin represents the estimation error at t_9, whereas the second bin represents the error at t_{10}, and so on. It was observed that the estimation error decreased significantly over time, as observed from the decrease in the length of the bins in Fig. 13. Notably, the number of errors decreased significantly after bin #5 (at time t_{13}), as observed in our experiments.

4.5 Short discussion of our experiments

In this study, we successfully integrated a ball spin classification model and a trajectory prediction model to estimate the position of the contact point and the hitting time. Initially, the ball's flight speed (low or high) and spinning direction were classified, and then the entire

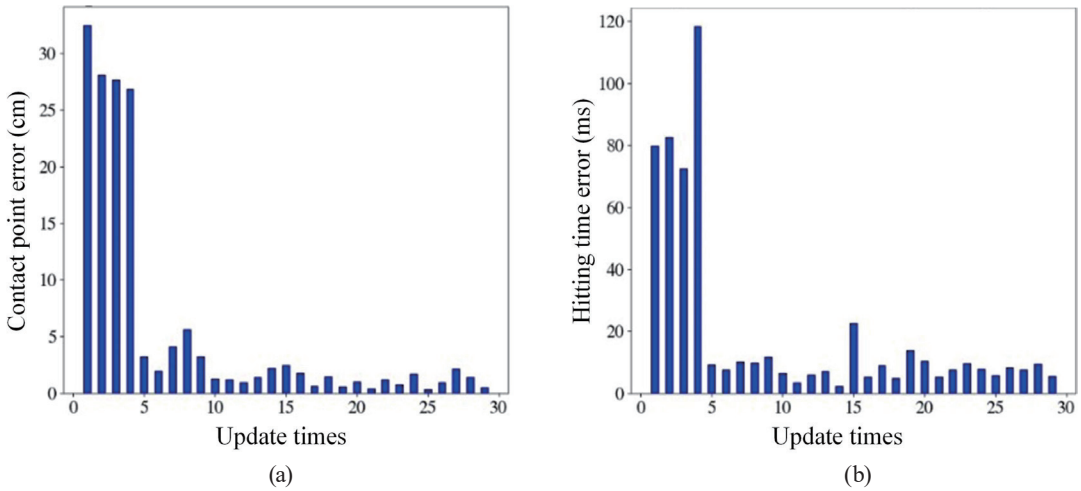


Fig. 13. (Color online) Prediction errors of (a) contact position and (b) hitting time over ball-flying duration.

flight trajectory was predicted. The experimental results demonstrated a high level of accuracy with that of the ball spin classification reaching more than 98%. The distance error of the estimated contact point was less than 2–3 cm after time t subscript 13 (bin #5 in Fig. 13), and the predicted hitting time error is often less than 20 ms. Once the visual system received nine measured points, it could begin to predict the ball's flight trajectory. Although there is still some degree of error between the predicted contact point and the actual point, this error falls within the size of the racket, which may allow the robotic arm to strike the ball. Overall, the proposed system is an effective solution that combines ball spin classification and trajectory prediction models to achieve an accurate estimation of the contact point position and hitting time.

In contrast to prediction models that forecast the entire trajectory in one step, the proposed model swiftly generates predictions upon detecting even a small segment of the trajectory. This iterative method facilitates quicker estimation of the ball's hitting position. Each time a new position of the ball is recorded, the predicted trajectory can be promptly adjusted using this updated information. Notably, the closer the ball is to the camera, the smaller the positioning error tends to be. Therefore, continuously refining the predicted trajectory as the ball approaches the camera significantly reduces prediction errors related to both the contact position and the hitting time. Figures 11–13 demonstrate that, with each iterative update, the number of prediction errors progressively decreases.

5. Conclusions

In this study, we introduced a binocular vision system and its algorithm for detecting and positioning a table tennis ball. The system is capable of predicting the future flight trajectory of the ball, estimating the position of the contact point, and determining the hitting time, thereby enabling it to control the striking of a robotic arm, if present. We focused on the algorithm development of the visual system, which comprises several stages, including ball

detection and positioning, ball spin classification, trajectory prediction, and contact point position and hitting time estimation. There are some improvements compared with existing methods. First, we implemented a tracking window that aligns with the trajectory of the table tennis ball, significantly reducing the computation time for ball detection. This allowed us to achieve real-time detection and tracking. Second, while standard binocular vision is commonly used for establishing stereo vision systems, our approach utilized a method called nonSSG for constructing the stereo vision. Third, a primary contribution of our work is the use of the first N consecutive images to initiate ball tracking. This approach enables our method to effectively track the ball until it reaches the hitting region, resulting in a relatively small error in the estimated striking point. Additionally, we combined two models: the ball spin classification model and the trajectory prediction model. The classification of ball spin enhances our ability to predict a more reliable striking point. To verify the performance of our proposed algorithm, we conducted experiments that utilized binocular-vision-based positioning, ball spin classification, and trajectory prediction. The proposed visual system and algorithm have potential applications in robotics, particularly in developing robotic ping-pong players. This study provides valuable insights into designing and implementing an efficient and accurate vision system for ball detection, positioning, and tracking.

Acknowledgments

This research was funded by the National Science and Technology Council, Taiwan, under grant number NSTC 112-2221-E-992-064-MY2.

References

1 L. Acosta, J. J. Rodrigo, J. A. Mendez, G. N. Marichal, and M. Sigut: IEEE Rob. Autom. Mag. **10** (2003) 44. https://doi.org/10.1109/MRA.2003.1256297
2 M. Matsushima, T. Hashimoto, M. Takeuchi, and F. Miyazaki: IEEE Trans. Rob. **21** (2005) 767. https://doi.org/10.1109/TRO.2005.844689
3 Z. Yu, Y. Liu, Q. Huang, X. Chen, W. Zhang, J. Li, G. Ma, L. Meng, T. Li, and W. Zhang: Proc. 2013 IEEE Int. Conf. Robotics and Biomimetics (IEEE, 2013) 911.
4 X. Chen, Y. Tian, Q. Huang, W. Zhang, and Z. Yu: Proc. 2010 IEEE Int. Conf. Robotics and Biomimetics (IEEE, 2010) 603.
5 V. D. Cong, L. D. Hanh, and L. H. Phuong: Proc. 2020 IEEE Int. Conf. Green Technology and Sustainable Development (IEEE, 2020) 9.
6 W. Zhang, D. Xu, and P. Yang: Proc. 2010 IEEE Int. Conf. Robotics and Biomimetics (IEEE, 2010) 376.
7 A. Nakashima, Y. Ogawa, C. F. Liu, and Y. Hayakawa: Proc. 2011 IEEE Int. Conf. Robotics and Biomimetics (IEEE, 2011) 2348.
8 H. Bao, X. Chen, Z. T. Wang, M, Pan, and F. Meng: Proc. 2012 IEEE Int. Conf. Mechatronics and Automation (IEEE, 2012) 2002.
9 J. Achterhold, P. Tobuschat, H. Ma, D. Büchler, M. Muehlebach, and J. Stueckler: Proc. Learning for Dynamics and Control Conf. (PMLR, 2023) 878.
10 P. Payeur, H. Le-Huy, and C. M. Gosselin: IEEE Trans. Ind. Electron. **59** (2010) 3195. https://doi.org/10.1109/41.370380
11 T. Wei: 2020 IEEE Int. Conf. Industrial Application of Artificial Intelligence (IEEE, 2020) 405.
12 J. Tebbe, L. Klamt, Y. Gao, and A. Zell: 2020 IEEE Int. Conf. Robotics and Automation (IEEE, 2020) 9694.
13 Y. Zhang, Y. Zhao, R. Xiong, Y. Wang, J. Wang, and J. Chu: 2014 IEEE Int. Conf. Robotics and Automation (IEEE, 2014) 4108.
14 Y. F. Zhang, R. Xiong, Y. S. Zhao, and J. G. Wang: IEEE Trans. Instrum. Meas. **64** (2015) 2280. https://doi.org/10.1109/TIM.2014.2385173

15 H. Su, Z. Fang, D. Xu, and M. Tan: IEEE Trans. Instrum. Meas. **62** (2013) 2890. https://doi.org/10.1109/TIM.2013.2263672
16 Q. Wang, K. Zhang, and D. Wang: Proc. 2014 IEEE Int. Conf. Cyber Technology in Automation, Control, and Intelligent (IEEE, 2014) 496.
17 Q. Liu and H. Ding: Front. Neurorob. **20** (2022) 820028. https://doi.org/10.3389/fnbot.2022.820028
18 S. Gomez-Gonzalez, S. Prokudin, B. Schölkopf, and J. Peters: IEEE Rob. Autom. Lett. **5** (2020) 970. https://doi.org/10.1109/LRA.2020.2966390
19 W. Li: Soft Comput. **27** (2023) 12769. https://doi.org/10.1007/s00500-023-08962-8
20 Q. Song and R. Lu: Proc. Forth Int. Conf. Signal Processing and Computer Science (SPIE, 2023) 127900. https://doi.org/10.1117/12.3012241
21 H. Li, S. G. Ali, J. Zhang, B. Sheng, P. Li, Y. Jung, J. Wang, P. Yang, P. Lu, K. Muhammad, and L. Mao: Fractals **30** (2022) 2240156. https://doi.org/10.1142/S0218348X22401569
22 Q. Song and R. Lu: Proc. Int. Conf. Image Processing and Artificial Intelligence (SPIE, 2024) 132132D. https://doi.org/10.1117/12.3035167
23 Y. Gao, J. Tebbe, and A. Zell: Proc. Int. Joint Conf. Neural Networks (IEEE, 2022) 1. https://doi.org/10.1109/IJCNN55064.2022.9892776