Article:
Li, H., Ali, S.G., Zhang, J. et al. (8 more authors) (2022) Video-based table tennis tracking and trajectory prediction using convolutional neural networks. Fractals, 30 (05). 2240156. ISSN 0218-348X

https://doi.org/10.1142/S0218348X22401569

OPEN ACCESS
Fractals, Vol. 30, No. 5 (2022) 2240156 (23 pages)

DOI: 10.1142/S0218348X22401569

World Scientific
www.worldscientific.com

VIDEO-BASED TABLE TENNIS TRACKING AND TRAJECTORY PREDICTION USING CONVOLUTIONAL NEURAL NETWORKS

HAOXUAN LI,[*] SABA GHAZANFAR ALI,[*] JUNHAO ZHANG,[*]
BIN SHENG,[*,††,¶¶] PING LI,[†] YOUNHYUN JUNG,[‡] JIHONG WANG,[§,¶¶]
PO YANG,[¶] PING LU,[‖,¶¶] KHAN MUHAMMAD[**,‡‡,¶¶]
and LIJIUAN MAO[§,§§,¶¶]

[*]*Department of Computer Science and Engineering*
Shanghai Jiao Tong University, Shanghai, China

[†]*The Hong Kong Polytechnic University*
Hong Kong, China

[‡]*Gachon University, Gyeonggi-do, Korea*

[§]*Shanghai University of Sport, Shanghai, China*

[¶]*The University of Sheffield, Sheffield, UK*

[‖]*State Key Laboratory of Mobile Network*
and Mobile Multimedia Technology
ZTE Corporation, Shenzhen, China

[**]*Visual Analytics for Knowledge Laboratory (VIS2KNOW Lab)*
Department of Applied Artificial Intelligence, School of Convergence
College of Computing and Informatics, Sungkyunkwan University
Seoul 03063, Republic of Korea

[††]*shengbin@sjtu.edu.cn*
[‡‡]*khan.muhammad@ieee.org*
[§§]*maolijuan@sus.edu.cn*

[¶¶]Corresponding authors.

Received August 6, 2021
Accepted November 10, 2021
Published July 4, 2022

Abstract

One of the fascinating aspects of sports rivalry is that anything can happen. The significant difficulty is that computer-aided systems must address how to record and analyze many game events, and fractal AI plays an essential role in dealing with complex structures, allowing effective solutions. In table tennis, we primarily concentrate on two issues: ball tracking and trajectory prediction. Based on these two components, we can get ball parameters such as velocity and spin, perform data analysis, and even create a ping-pong robot application based on fractals. However, most existing systems rely on a traditional method based on physical analysis and a non-machine learning tracking algorithm, which can be complex and inflexible. As mentioned earlier, to overcome the problem, we proposed an automatic table tennis-aided system based on fractal AI that allows solving complex issues and high structural complexity of object tracking and trajectory prediction. For object tracking, our proposed algorithm is based on structured output Convolutional Neural Network (CNN) based on deep learning approaches and a trajectory prediction model based on Long Short-Term Memory (LSTM) and Mixture Density Networks (MDN). These models are intuitive and straightforward and can be optimized by training iteratively on a large amount of data. Moreover, we construct a table tennis auxiliary system based on these models currently in practice.

Keywords: Table Tennis; Deep Learning; Object Tracking; Trajectory; Fractal AI Prediction.

1. INTRODUCTION

The fractal AI method plays a significant role in learning complex, and patterns and features that have no characteristics length but are of self-similarity.[1–3] Therefore, in recent years, with the development and reliability of computer vision technology, specific applications of computers in the field of sports have also continued to appear. The fractal method aims to estimate the fractal dimension and analyze the relationship between fractal dimension and other parameters.[4] Some are used as auxiliary systems in sports games, such as eagle eyes in tennis, badminton, and other games; goal determination in football matches; and as data visualization and analysis tools, such as player movement distances in football games, heatmaps, and technical statistics. Several models have been developed to calculate fractal dimension according to different subjects of interest, e.g. tracking algorithms in computer vision and machine learning related theories.[5]

In table tennis, another application that requires the use of computer vision technology is table tennis robots. In this field, Zhang and Xiong,[6] Mülling *et al.*[7] and Japan's OMRON company developed robot games that can fight human table tennis players. Although these applications involve computer vision, sensors, robot control, and other theories, computer vision plays a significant role. For example, the accurate and fast-tracking of the ping-pong ball and the estimation of information such as trajectory, speed, and rotation speed are all crucial to the subsequent strokes and the usability of the entire system.

Generally, designing a table tennis robot system must solve two fundamental problems. First, the ping-pong ball is tracked in each frame of the camera to record the position information of the ball. Second, based on the sequence of position information, it must predict the ball's trajectory and landing point on the table. The significance of prediction is to obtain the ball's movement

information in advance to allow time for the robot arm to move to a suitable position and determine the way of hitting the ball. Because table tennis has the characteristics of small size, few features, and fast movement, a tracking algorithm needs to be specially designed to meet these requirements. In addition, hitting a ping-pong ball will produce various rotations, bringing significant challenges to trajectory prediction.

Though traditional methods have established complete frameworks and can achieve good practical results, they still have some shortcomings. First, the characteristics based on manually selective color or shape indicate that the robustness is insufficient. The performance will significantly fluctuate in video shooting conditions, the present occasion, or a complex background. Second, the features are fixed to form when the algorithm is designed and cannot be flexible, and it is impossible to obtain better characteristics by more sample training. In addition, traditional table tennis trajectory predictions include three stages, including rebound and rebound before a rebound. For the problems mentioned earlier, traditional machine learning methods show limitations.

However, deep learning-based approaches have overcome these problems by improving the performance of table tennis robots. With the improvement of computer computing power and the emergence of many labeled datasets for model training, the potential of deep neural networks has been continuously explored, and deep learning has become an advanced research direction in recent years. Fractal Net,[8] first proposed by Gustav, was widely revived due to its powerful feature learning capability. There are two main contributions. On the one hand, they introduce Fractal Net as the first simple alternative to ResNet. At the same time, Fractal Net certificates that explicit residual learning is not a requirement for building ultra-deep neural networks.[2,8] In Ref. 9, winning the ImageNet image classification competition, deep learning techniques based on models such as CNN can extract better features and are widely used in computer vision, natural language processing, and other fields. The problem even exceeds the traditional manual feature selection method in some respects. For example, R-CNN [10] is used for object detection in images, LSTM [11] is used for various sequence prediction problems, and Q-Learning[12] is used for strategy learning problems such as game AI.

In recent years, the real-time performance of the depth tracking framework has attracted widespread attention. Researchers have proposed a lightweight deep learning-based framework that can be tracked in real-time. For example, in Ref. 13, a five-layer convolutional neural network is used to return to the location of the target in the image, and remove the online learning step, allowing the algorithm to process more than 100 frames per second. On the other hand, object detection is also a widely used area. Even if the target detection task is primarily classified in the target object compared to the target tracking, the object's position is in the image. But some of the critical techniques in target object detection have a great inspiration to our tracking model.

In addition to convolutional neural networks, deep learning, a Recursive Neural Network (RNN) is specifically used to resolve sequence prediction, such as natural language processing. For example, in Ref. 14, we use RNN to train on man handwriting sequences, allowing the writing to predict and even mimic different styles to automatically generate. Likewise, in Ref. 15, training LSTM on basketball game data predicts the ball trajectory and judges whether it goes in the hoop. Therefore, in this work, we solve the challenges of trajectory prediction of table tennis, which can be visually converted to the prediction problem of the coordinate sequence designed from deep learning, and propose a novel table tennis tracking system. In sum, the following are the contributions of our paper:

- CNN-based Tracking Algorithm: Based on a structural output convolutional neural network, our network outputs a probability map that represents the location of the tracking target in the image. We also refer to the target assessment, designed an enclosure back to the back lapse, and enhanced tracking result accuracy.
- Trajectory Prediction Model Based on LSTM: We employ a three-dimensional coordinate sequence of table tennis to train the model, predict the entire table tennis trajectory, and automatically process the ball rebound on the play. Thus, it significantly simplifies the problem of table tennis trajectory and uses it as a tracking framework.
- Construction of Tracking Framework: We have designed a complete tracking framework. At the same time, we use this framework to collect a host of table tennis trajectory data for analysis.

2. METHODOLOGY

The framework proposed in this paper can be used in ship trajectory classification. Taking advantage of the high timeliness of CNN's model construction, the modular reading and identification input of data information are carried out. At the same time, in order to prevent the loss of local information in the modeling process, the data is serialized into multiple sub-sequences and transmitted to the CNN network to improve the integrity of prediction data. Due to the spatial and temporal correlation of ships in navigation, the LSTM model with memory characteristics is used to predict the ordered data of ships. Therefore, this combined model can not only support more data input, but also ensure that the predicted time state is more stable.

2.1. Tracking Algorithm Based on Structured CNN Output

Since the table tennis tracking framework needs to meet real-time tracking requirements, the candidate image of the neural network needs to be as small as possible. Due to this, the CNN model can directly output a surrounding box indicating the target position without a feature vector. Thus, simplifying the network structure, avoiding more complex calculations. SO-DLT[16] provides an idea to solve this problem.

The CNN model of this paper is shown in Fig. 1. Similarly, in SO-DLT, the network is input to an image of 100×100 and outputs a probability map of 50×50. The convolution layer uses a pre-trained CaffeNet and fine-tuning on a table tennis training set. Above the convolution layer, there is a space pyramid pooling layer, which is used to retain more location information. Finally, two full-connection layers output 2500-dimensional vector transform in 50×50 matrices. Since the CNN model outputs more than just the target's position, it can also distinguish between table tennis, other objects, and non-objects. Therefore, after the convolution layer, a classification head is added, and in the training phase, two tasks are jointly trained. The loss function is defined as

$$L(p, s) = L_{\text{cls}}(s, s^*) + \lambda \sum_{i}^{50} \sum_{j}^{50} -(1 - t_{ij}) \times \log(1 - p_{ij}) - t_{ij} \log(p_{ij}), \quad (1)$$

where p and s are the probability maps and category scores of the output, respectively. s^* is the target output category, t_{ij} is an element of the target output probability map, with a value of 0 or 1. L_{cls} is a Softmax loss function, usually used for classification tasks. λ adjusts the weight of the loss function of the two tasks.

We employed a table tennis image from a table tennis competition shooting from the high-speed camera to train the network. These pictures contain positive and antique examples. The correct example contains images of table tennis. For the typical example, the displacement is randomly added, making the target (table tennis) uniformly distributed in the picture, a scaled zoom within a certain range, and finally the unified cut into 100× sizes. The antique includes a background image and an interference image. Target output refers to the description in SO-DLT; for the typical example, the value of the probability map is set to 1; for the reverse example, the probability map, all values are 0. Figure 2 shows some training data. The two columns of the left are numbered input images and target outputs, and the right side is two listed as an example.

Figure 3 shows the performance of the model on the validation set. It can be seen that for the input

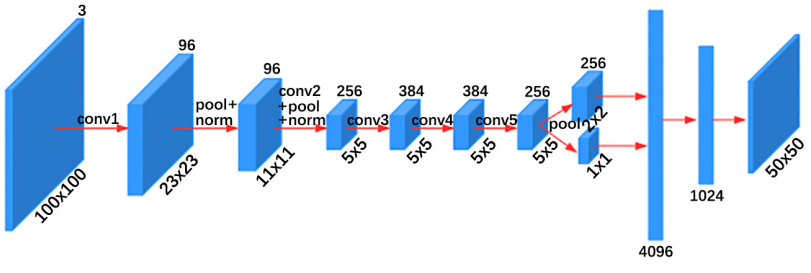


Fig. 1 CNN model network structure.

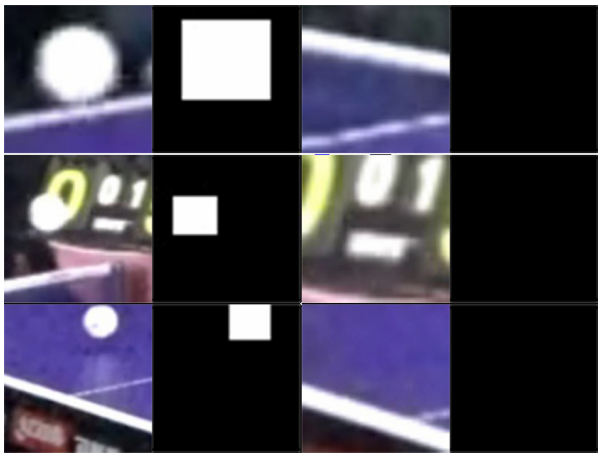


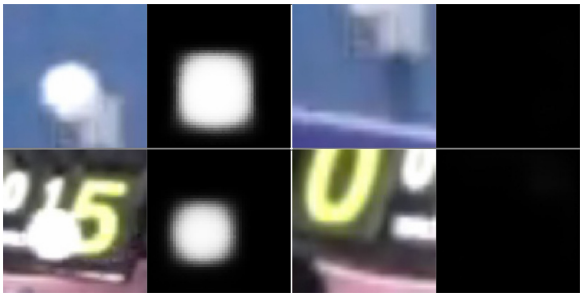
Fig. 2 Results of the CNN model on the validation set.

image containing the target, the output is displayed as a mass consisting of a high probability value, and the group corresponds to the position of the table tennis in the input image. This can be more clearly seen on the three-dimensional probability map shown in Fig. 3b. The farther from the center, the lower the probability value. Therefore, a simple threshold method can be used to obtain a surrounding box, and this enclosure is used as the model's output.

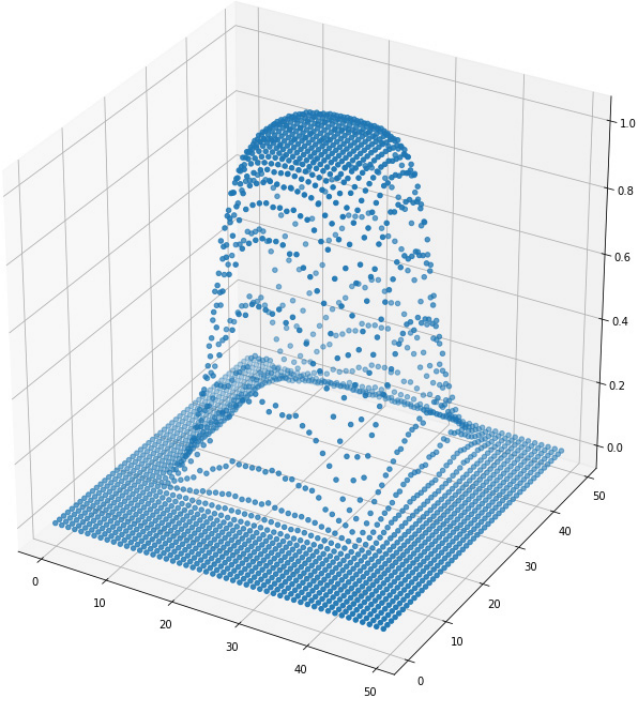
2.1.1. *Regression Layer*

However, from Fig. 3, the envelope from the probability map is not ideal. An incorrect surrounding box means the error of location information, but it will also cause drift or even loss of the entire tracking framework. Although the probability map can give a more accurate estimate of the object's position, it is poor in the object's border. The shape of the probability map changes and a simple threshold method may cause more errors. Infested envelopes not only mean the error of location information, but they will also cause drift or even loss of the entire tracking framework.

We focus on the object detection field. The enveloped box obtained after the transformation is the output of the entire network. The high-level semantic information (candidate area layer) and low layer location information can correctly identify the object in the input image and give the envelope box. For example, the probability map's envelope case can be considered a candidate area in the above-structured output CNN. Unlike the production of this envelope, we proposed to realize this enclosure to the low-level convolution layer. The feature is input to the returning layer to obtain a more precise chamber.



(a)



(b)

Fig. 3 Sample figure caption. **(a)** For inputs containing the target, the network generates a white communication shape; for the reverse example, the output is close to all 0. **(b)** 3D representation of the output probability map.

This paper refers to the Fast R-CNN[17] design, adding a regional pooling layer (ROI pooling layer) to the network. This pilot layer uses the envelope case obtained from the above probability map, which is cropped on a characteristic of a low-level convolution layer and is scaled to a new feature map of a seven × seven size. A returning layer above this feature is added. Considering that the positional accuracy of the convolution layer cannot be too low, this paper selects a circuit in Conv 1. The returning layer is essentially a thorough attachment layer, but the output is fixed to 4 real numbers, representing the displacement and long width of the XY direction. The newly added neural network layer is directly attached to the pre-training structured output CNN. During training, the returning layer will be trained separately while the other layers are frozen. For the loss function, we refer to the method in the R-CNN, which uses a Smooth L1 Loss layer.

First, the four simulations predicted by the returning layer are not an absolute value of the displacement and size of the target enclosure but are a transformation of the candidate zone to the target enclosure. These indicate that the transformed value can be normalized, and the training difficulty can be reduced. At the same time, we refer to the design of the R-CNN loss function, when comparing network output and target output, the SmoothL1 process is used instead of simple L1 or L2

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if} |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \quad (2)$$

L2 is used in the vicinity of the origin, and the remaining portion is the same as L1. This can effectively avoid gradient explosion. The newly added network layer is shown in Fig. 4. Like the training data used and the previous structured output CNN, only the target output is changed to 4 real numbers, indicating the target enclosure's XY coordinates and long width. The effect of the returning layer can be seen in Fig. 5. The returning layer can be effectively fine-tuned to the original envelope box, closer to the detection target.

2.2. Trajectory Prediction Model Based on LSTM

Recurrent Neural Network (RNN) and its variant Long Short-Term Memory (LSTM) are widely used to solve sequence prediction. In table tennis robot system design, two problems need to be solved by using the new model. One is the motion model in the tracking framework. Given the position of the last frame of the tracking target, it predicts the place that may appear in the next frame. The other

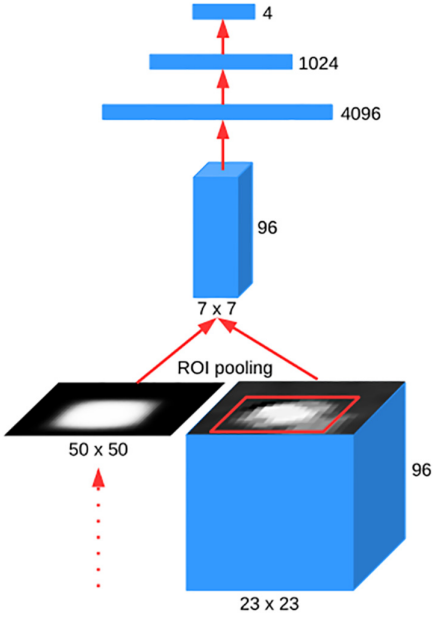


Fig. 4 Network structure of the returned layer. The network structure of the returned layer. Use the candidate area in the low-level convolution layer feature map cut, and the pool is used to become a feature map of the fixed size. Subsequently, four real numbers are output, indicating the inclusions of the return.

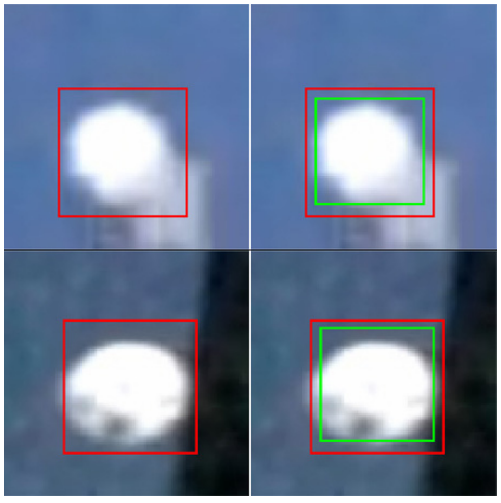


Fig. 5 Results of the returning layer on the verification set. The result of the returning layer on the verification set. The red rectangle comes from the structured output CNN; the green rectangle is the output of the returning layer. At this point, this paper has implemented an end-to-end CNN model that can directly obtain the precise position of the target object from the input image, and only the forward operation needs to be performed.

is trajectory prediction. Given a small section of table tennis coordinate sequence, the subsequent coordinate sequence is automatically generated. As we can see, these two problems are suitable for solving a recurrent neural network model. For the motion model, the known table tennis coordinate sequence can be input into the neural network, and the final output can be used to predict the coordinates in the next frame. The known coordinate series is also input into the neural network to update the internal state vector for trajectory prediction. Then, the new coordinates are recursively generated and connected to the input to create a sequence. From this sequence, the landing point information of table tennis and some attributes in the flight process can be calculated.

The recurrent neural network gets its name because of its internal recursive structure. In the most straightforward cyclic neural network, the network parameters only include the state transformation matrix W_hh, the input transformation matrix W_xh, and the output transformation matrix W_hy. There are only two kinds of operations: receiving input and updating status and computational output.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t), \tag{3}$$

$$y_t = W_{hy}h_t. \tag{4}$$

By simply overlaying the hidden layer, a more complex network structure can be achieved. Usually, we can also connect the output of other networks to the hidden layer of RNN to provide initial information. For example, in the image capturing task, we connect the output of CNN to RNN as the initial state, and then make the recursive structure of RNN to generate a text sequence to describe the input image.

Due to structural defects, the state of RNN at one recursion may affect the state at a future recursion.

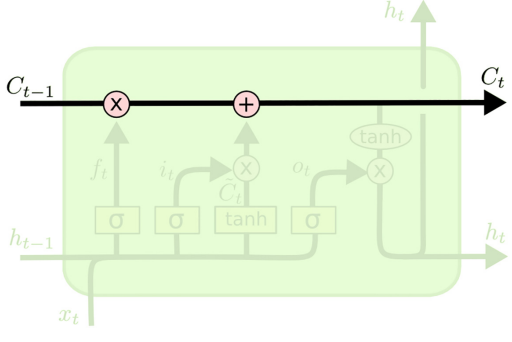


Fig. 6 Structure of LSTM.

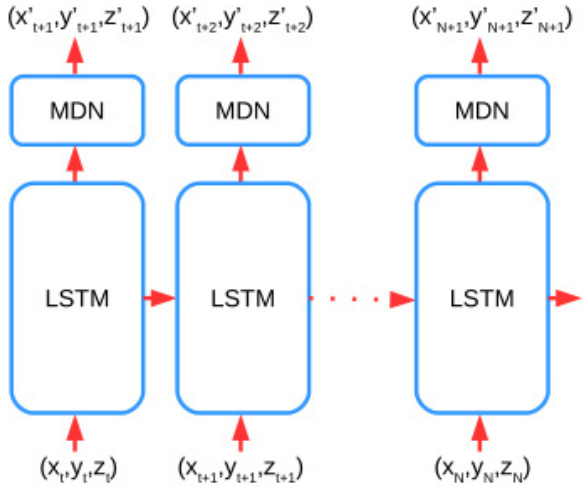


Fig. 7 Internal structure of LSTM.

LSTM has made a unique design for this problem. As shown in Fig. 6, the state transition is modified by adding a new internal state vector Ct and a path to connect the two states so that the old information can be passed to multiple recursions. The figure can also show the structure of the multiplication operation of σ functions followed by an element. This structure allows the neural network to learn what information is retained in the previous state and added data.

2.2.1. *Trajectory Prediction Model Design*

Figure 7 shows the structure of the trajectory prediction model used in this paper. The LSTM model accepts a three-dimensional vector: the xyz coordinate of the ping-pong ball at t time as the input. For the output, we follow the practice in Ref. 15, which is not only to output a three-dimensional coordinate of the moment $t+1$ but also to add a Mixture of Density Networks (MDN) to LSTM to output multiple parameters of multi-dimensional Gaussian distribution, which represents the probability distribution of the next coordinate predicted by the model on xyz. In this work, the Gaussian distribution represents the displacement from the current coordinate to the next coordinate, the probability distribution of velocity.

First, the model, which only outputs three-dimensional coordinates and the loss function during training, generally uses the Euclidean distance between the target coordinates and the actual coordinates. For the model that outputs the parameters of Gaussian distribution, the probability value

of the target results in this Gaussian distribution can be used as the loss function to increase the probability value as much as possible. Because there may be some errors in the training data itself, the Gaussian distribution can tolerate such errors more than the absolute distance, which is helpful in the convergence of the model. Second, hybrid density networks are often used to model objects with multiple states. For table tennis, because the ball has flight, collision, and other conditions, a single Gaussian distribution may not be able to describe all possible state changes. When multiple Gaussian distributions are used, the neural network can determine which state to predict by adjusting the weight of each distribution. For example, when the ball is about to collide with the table, the importance of the Gaussian distribution that predicts the state of "bounce" should be more significant, while the weight of the distribution that indicates "keep flying" should be smaller.

MDN works like a fully connected network. It accepts the intermediate state of LSTM as input and outputs multiple parameters of multi-dimensional Gaussian distribution. Seven parameters represent each Gaussian distribution: the mean and standard deviation on the xyz axis $\mu_x\sigma_x\mu_y\sigma_y\mu_z\sigma_z$, and the correlation ρ_{xy} of the xy axis. Because in the trajectory of table tennis, the xy-axis is parallel to the table, while the z-axis is perpendicular to the table, it is only assumed that xy correlates. For a single Gaussian distribution, the probability of input state c and output velocity v is

$$p(v\,|\,c) = \frac{1}{2}\exp\left(-\frac{1}{2}\begin{pmatrix} v_x-\mu_x \\ v_y-\mu_y \\ v_z-\mu_z \end{pmatrix}^T \times \begin{pmatrix} \sigma_x^2 & \sigma_x\sigma_y\sigma_{xy} & 0 \\ \sigma_x\sigma_y\sigma_{xy} & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{pmatrix}^{-1} \times \begin{pmatrix} v_x-\mu_x \\ v_y-\mu_y \\ v_z-\mu_z \end{pmatrix}\right). \quad (5)$$

For multiple Gaussian distributions, the model sets a weight for each Gaussian distribution, and the sum of the weights is 1 so that the sum of the whole probability distribution is 1. For example, for MDN, input state c with K Gaussian distributions, the probability of output velocity v is

$$p(v\,|\,c) = \sum_k^K \theta_k(c)p(v|c), \quad \text{s.t.} \sum_k^K \theta_k(c) = 1, \quad (6)$$

where θ_k is the weight of each Gaussian distribution, and p_k is the probability distribution function of each Gaussian distribution.

To train the network with MDN, because the network's output represents a mixed Gaussian distribution, the Euclidean distance from the target coordinates cannot be directly used as the loss function. Instead, the network can be trained by maximizing the probability value of the target coordinates in the Gaussian distribution. The loss function is

$$L(v^*, x) = -\log\left(\sum_k^K \theta_k(c(x))p_k(v^*|c(x))\right), \quad (7)$$

where v^* is the target output, x is the input three-dimensional coordinate, and $c(x)$ represents the intermediate state of the output after LSTM accepts x.

In trajectory prediction, the model takes a sample from the output mixed Gaussian distribution, and the calculated new coordinates are re-input to the network. A predicted trajectory is obtained after several cycles. When LSTM is used as the motion model of the tracking framework, it is like a quantum filter to sample multiple candidate regions from the mixed Gaussian distribution.

3. INTEGRATED TRACKING FRAMEWORK

The system proposed in this paper needs to be used in a specific actual environment configuration. Specifically, two high-speed cameras are placed on one side of the table, and the area of the table is photographed synchronously through hardware triggers in parallel and does not move in the whole process. To ensure that the three-dimensional coordinates can be calculated accurately later, the models and specifications of the camera need to be consistent. The world coordinate system is established with a corner of the ball table as the origin, as shown in Fig. 8. The x-axis is along the bottom line of the table, the y-axis is along the edge of the table, and the z-axis is perpendicular to the table.

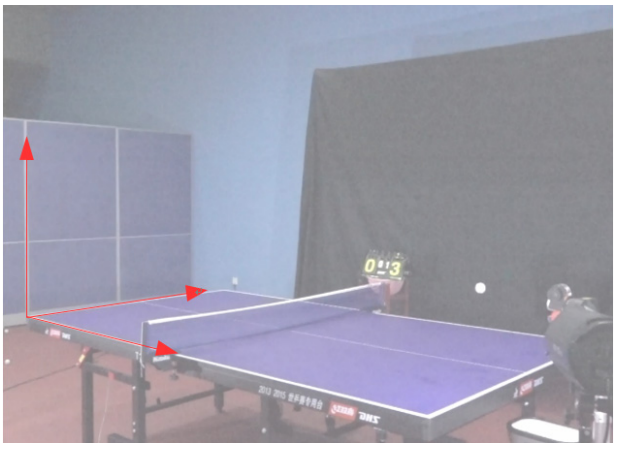


Fig. 8 World coordinate settings.

The projection matrix of the two cameras to the ball table needs to be calculated in advance to calculate the three-dimensional coordinates from the two-dimensional coordinates obtained by the two cameras or to project the three-dimensional coordinates to the camera plane.

3.1. Framework Design

The next step is to build the entire tracking framework. At each moment, it obtains an image from two synchronous cameras, tracks and receives the three-dimensional coordinates of the table tennis ball after a series of calculations and predicts the trajectory simultaneously. The whole process can be divided into the three steps described below and shown in Fig. 9.

3.1.1. *Extract region of tracking targets*

The first thing to solve is how to get the initial bounding box, that is, how to detect the tracking target in the image. This framework first looks for possible areas of table tennis. Because the camera is permanently fixed and there are not too many moving objects in the whole scene, the foreground region can be extracted by background subtraction to narrow the search range. These regions are used as target candidate regions, and then input into the CNN model for calculation. Finally, the candidate image with the highest probability value is output as the tracking target. Note that this system does not rely heavily on background subtraction. It only needs to find the target's initial position in the first few frames, and then it can continue to track using the other steps of the algorithm. This means that you do not need to keep the entire scene still all the time. Of course, in the central part of the algorithm, we can still use background subtraction as an auxiliary method to provide candidate regions for tracking or skip this step directly.

The test method is similar to the landing point prediction, and the results are shown in Table 7.

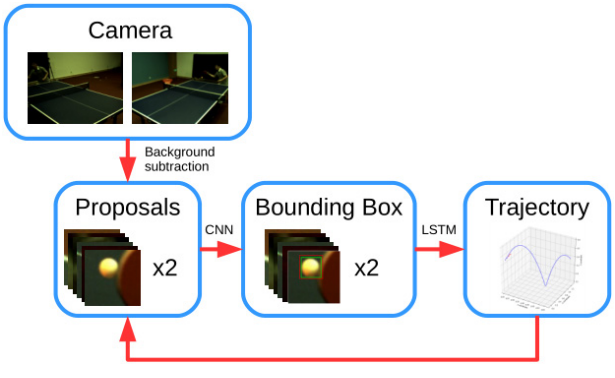


Fig. 9 Flowchart of tracking framework.

3.1.2. *Model tracking using CNN*

With the candidate region obtained in the previous step, the bounding box of the target can be obtained using the CNN model. For performance reasons, the CNN model works in two modes. If no target is found in the current tracking framework, the CNN model acts as the target detector and takes all candidate regions as input. If the target has been found in the previous frame, as a tracker, only the candidate area near the current target position is taken as input. Before entering the network, the candidate area is enlarged by several pixels to ensure that the complete target can be included. Then, depending on the number of candidate areas we input into the network uniformly as one or more batches. The bounding box with the highest probability value is taken as the final output in the output result. After this step, the two cameras receive a two-dimensional coordinate, respectively. Through the camera projection matrix obtained in advance, the three-dimensional coordinates of table tennis can be obtained.

3.1.3. *Predict trajectory using LSTM*

After several cycles, the frame can get a continuous sequence of coordinates. The trajectory and landing point predictions can be carried out by inputting this sequence into the LSTM model and automatically generating the subsequent coordinate

sequence. In addition, we can also use the LSTM model as the motion model in the tracking framework, input the coordinates obtained in each cycle, and make it output by only one parameter of the mixed Gaussian model, like the Kalman filter, to predict the possible position of the tracking target in the next frame and reduce the search range of the tracking. In addition, because table tennis movement is regular, the LSTM model can directly predict the position of the ball after several frames through the trajectory prediction method. When an occlusion occurs, or the speed of table tennis is too fast, the CNN tracking model may lose the target. At this point, the prediction of LSTM is used as the tracking result, and when the CNN model searches for the target again, the whole tracking framework can continue to work.

In addition to the above main modules, the tracking algorithm includes some steps to deal with exceptional cases. These steps come from many best practices in target tracking and are critical to the robustness of the entire tracking framework. Algorithm 1 shows the specific steps of single view tracking.

In the above algorithm, first, the candidate regions of the input are screened. Since the motion of the target object is usually continuous, the candidate regions far from the previous frame can be excluded. The context of the last frame position is also taken as a candidate region. The candidate region is then entered into the CNN model, which returns the probability value and bounding box for each region. When all probability values are less than a set threshold, the algorithm determines that the tracking has failed. Otherwise, the algorithm selects the highest probability value and the corresponding bounding box. Finally, according to the size of the probability value, the bounding box is returned directly or the size of the bounding box is averaged with the previous frames and then returned. This step is due to the slow change of the bounding box size in a short time, while the bounding box size of the output of the CNN model fluctuates to a certain extent. When the probability value is greater than a higher threshold (such as 0.95), the framework considers the output of CNN to be more reliable. Otherwise, based on the above assumptions, it will be averaged with the previous frames. In the actual measurement, when the averaging method is not used, the tracker can easily fail because the predicted bounding box is larger than the actual one such that the search box is too large to detect the targets inside the search box.

Algorithm 1. Single View Tracking Algorithm

```
Input: Proposals candidate area array
Output: the bounding box of the tracking target
Ensure: Augmented image I_aug
if last_frame_tracked then
    for all p in Proposals and TOO_FAR_AWAY(last_box, p) do
        remove p from Proposals
    end for
    Proposals Proposals+ GET_CONTEXT(last_box)
end if
Probabilities, Boxes CNN_PREDICT( Proposals)
Max_id ARGMAX(Probabilities)
Probability Probabilities[Max_id]
Box Boxes[Max_id]
if Box Boxes[Max_id] then
    execute last_frame_tracked false;
    return None;
end if
if last_frame_tracked or Probability > threshold_high then
    returnBox
else
    return RUNNING_AVERAGE(Box, last_box)
end if
```

When a single tracking fails, the algorithm will also select the next frame of coordinates predicted by the trajectory prediction model as the tracking result according to the number of consecutive failures or determine the loss of the whole tracking process and re-carry out the first step of the frame and afterward search the tracking target in the entire image.

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1. Structured CNN Output

For the implementation of the CNN model, this paper uses an open-source depth learning framework Caffe,[18] which is primarily used in deep learning research. Many of the famous depth learning models are implemented on Caffe, such as AlexNet and VGGNet[19] in image classification, FCNT[20]

in target tracking, and R-CNN in target detection. The network structure is modified directly by Caffe to alter the network structure and make fine adjustments of some network layers. Compared with retraining the entire network, the fine-tuning task is quite complicated, and the number of training data required is more minor, reducing the time and effort needed to collect and label data. We used the prototype declaration file to define the network of the structured output CNN and the regression layer of the box and use the Caffe command-line interface to complete training and testing. In addition, this paper uses Caffe's C ++ interface to encapsulate functions using the model for use in the production environment. First, five convolutional layers are obtained. Their weights from the pre-trained CaffeNet are fine-tuned, the original fully connected layer is removed, the new network layer is connected to, and training is from scratch. The whole training is divided into two steps. First, joint training is performed on classified tasks and structured output CNN so that the network can distinguish between table tennis objects and output probability. The second step removes the classification layer, adding a bounding box regression layer, freeing all the remaining network layers, and training the regression layer.

We use learning rate 10^{-6} for the first layer, and 5×10^{-6} for other layers. The weight decay value is 10 to prevent training. This paper collects table tennis pictures from 30 table tennis videos, randomly adds displacement and zoom, and uniformly cuts to 100×100 sizes. Approximately 40,000 images were gathered, of which about 20,000 were included in table tennis, and the rest as a background or contained examples of other interferers (such as racket, skin). For the first stage, a total of 5 cycles of training is 20,000 cycles, and the learning rate of each cycle is multiplied by the attenuation coefficient of 0.1. In the second stage, only table tennis pictures are used to train the regression layer, with four cycles and 12,000 cycles. Figure 10 shows the primary interface of the tool. The upper left corner is the input image.

The central part of the interface is the intermediate result of a layer in the network, which is the characteristic diagram of conv5 in this figure. A significant map of the neuron generated using the deconvolution layer is displayed in the lower-left corner when a neuron is selected. The upper right corner shows the image that maximizes the response of the neuron. The middle and lower right are the pictures in the dataset that make the neuron produce the maximum response and their saliency maps.

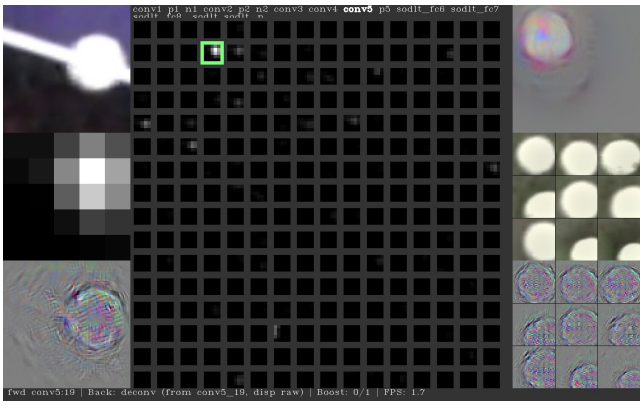


Fig. 10 Deep network visualization tool interface. The interface of the deep network visualization tool and the running status of the CNN network.

By changing the input image, it can be found that the No. 19 neuron in the conv5 layer responds to the corresponding position of the target object, and it is more evident than other neurons. The specific observation of this neuron shows that the response of this neuron mainly comes from the ping-pong part of the input graph from the prominent map in the lower-left corner of Fig. 11. The generated image in the upper right corner also shows that the neuron mainly learns the characteristics of "white" and "round."

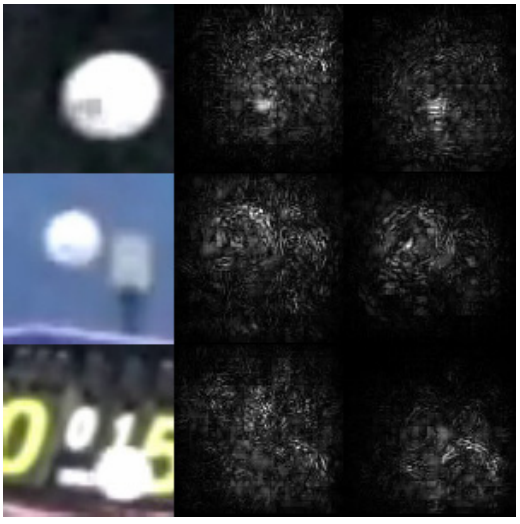


Fig. 11 Comparison of saliency maps of different convolutional layers. The three columns from left to right are the input image, the saliency map of $conv3_5 9$ and the saliency map of $conv5_1 9$.

Compared with the lower layer neurons, it can be found that the features learned by the conv5 layer are more robust. In contrast, the No. 59 neuron, which can also produce an obvious response in the target object's corresponding position, are selected from the conv3 layer. This time we use a saliency map calculated backward, that is, neurons, to compare the gradients of the input images. As shown in Fig. 11, for an input image (the first line) that does not contain an occlusion, there is little difference in the saliency map of the two neurons. However, for the input containing interference, the confidence map range of the conv3 neuron is more extensive; that is, the interference will also cause the neuron's response, while the reaction of the conv5 neuron is limited to the range of the target object. Thus, it can be seen that the higher convolution layer can learn semantic features and can distinguish between target objects and distractors.

In addition to the internal situation of the neural network, we can also test whether the whole neural network has learned the ability to predict the target object's position in the input image. For the last layer of the structured output CNN, that is, the probability graph of 50×50 sizes, using the above toolbox to do the same operation, you can get the result of Fig. 12. It can be seen from the graph that the significant image obtained by deconvolution operation on the structured output layer also corresponds to the position of the target object in the input image. The complete response image generated is also like a white circular object. The most responsive picture in the dataset also happens to contain the entire ping-pong ball. From this, it can be inferred that the neural network has been studied effectively.

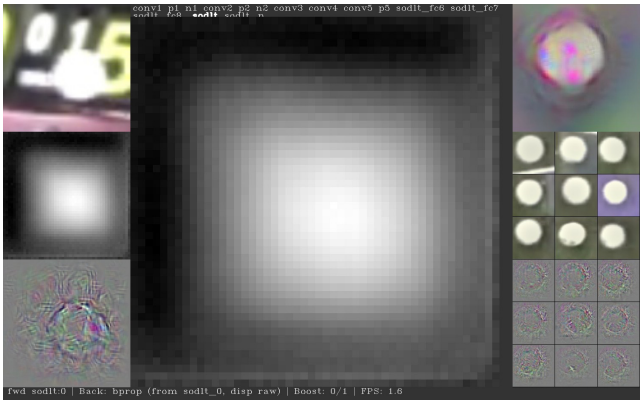


Fig. 12 Visualization results of structured output layer.

4.1.1. *Performance analysis*

For comparison, this paper trains a new model based on VGGNet. VGGNet has a deeper network level than CaffeNet. Therefore, more layers of the network can learn more semantic features. The achievements in image classification can also show that improving the depth of the network can effectively enhance accuracy. The VGGNet-16 selected in this paper contains 13 convolution layers, which are divided into five groups from conv1 to conv5. Each group of convolution layers relates to a two pooling layer. The size of the conv1-2-layer feature map used as the input of the ROI pooling layer is 50×50. Compared with the 23×23 of the CaffeNet, the position accuracy of the regression layer can be improved to a certain extent. But also, because there are more layers of VGGNet, the amount of calculation is more enormous, and the running time will be significantly higher than that of CaffeNet. The training of the model is also more complex, and the super-parameters need to be fine-tuned carefully.

First, the classification performance is tested on the test set of the dataset mentioned above. As shown in Table 1, because the two-classification task itself is relatively simple, the accuracy of different models has reached a higher value. Thus, the classification accuracy to a certain extent ensures that the tracking algorithm will not misclassify interfering objects as tracking targets. Second, Table 2 shows the IoU scores of prediction bounding boxes and target bounding boxes of different models on the test set of the above regression tasks. The model's score with the regression layer is higher than that without the regression layer, while VGGNet gets a higher score than CaffeNet with or without the regression layer.

Table 1 Classification Performance on Test Dataset.

CaffeNet	VGGNet
0.985	0.997

Table 2 Comparison of IoU Scores of CNN Models on the Regression Task Test Set.

	CaffeNet	VGGNet
Structured output CNN	0.51	0.61
Regression layer	0.61	0.75

Table 3 Comparison of CNN Model Running Speed.

	CaffeNet	VGGNet
Convolution layer (ms)	1	9.87
Structured output (ms)	0.47	0.61
Regression layer (ms)	1.26	1.65
Overall run time (ms)	2.93	12.63

In this paper, the time performance of the model is tested on a host configured with GTX 1060 3G. As can be seen from Table 3, the running speed of CaffeNet is faster than that of VGGNet, and the difference is mainly reflected in the convolution layer. There is little difference in time between the structured output and regression layers because they use the same structure.

During tracking, the CNN framework often needs to process multiple input images in one frame. Using the parallel computing ability of GPU, putting multiple input images into a batch, and inputting them into the network simultaneously, instead of processing only one idea at a time, can improve the processing speed. This section tests the uptime of the network under different batch size settings, as shown in Fig. 13. It can be seen that the intercept of the two curves on the y-axis is greater than 0, which means that it is more efficient to process a batch at the same time than to process a single image multiple times.

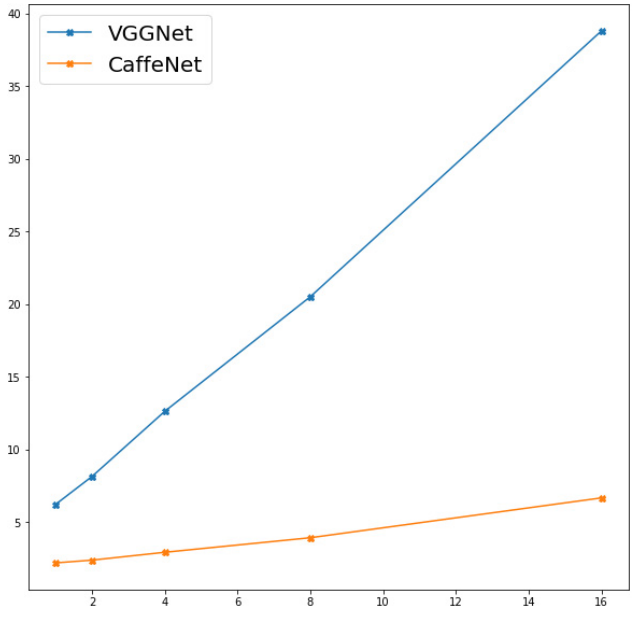


Fig. 13 Comparison of running speeds of different batch sizes.

4.1.2. *Tracking framework*

In this simple framework, only a small amount of samples are taken at each frame near the target position of the previous frame. The candidate region is input into the CNN model to obtain the object's bounding box in this frame. Although in the following test, the framework even carries out only one sampling, only intercepting the context of the target location of the previous structure as a candidate region, it can still achieve good tracking results.

The simple framework is tested on the table tennis video test set, and according to the performance index mentioned in Ref. 21, the success curve and the precision curve are drawn, as shown in Fig. 14a. The success curve indicates that in the tracking process, the coincidence rate of the predicted bounding box and the target bounding box is more significant than a certain threshold, that is, the ratio of the number of frames tracked successfully to the total number of frames. The abscissa is the threshold, and the ordinate is the ratio of the number of successful frames. Usually, the curve's Area Under Curve (AUC) is calculated to get a number between 0 and 1. The higher the value, the better the performance of the tracker. The accuracy curve represents the ratio of the number of frames in which the distance between the predicted target position and the actual place is less than a certain threshold.

This index is also significant for table tennis tracking because of the need to obtain accurate coordinates for trajectory prediction. Usually, the value of the curve is selected when the threshold is 20 pixels (precision@20) to measure the performance of the tracker. The value range is also in the field of 0–1. The tracking framework is tested using the above different CNN models, and the comparison results are shown in Fig. 14b.

A comparison of AUC and precision 20 scores is shown in Table 4. As can be seen from the chart and table, the performance of the VGGNet-based model is better than that of the CaffeNet model in all indicators. From the tracking test of a single video, it can be observed that due to the lower

Table 4 Target Tracking Indicators Tested on the Table Tennis Video Dataset.

	CaffeNet	VGGNet
AUC	0.51	0.61
PRECISION@20	0.61	0.75

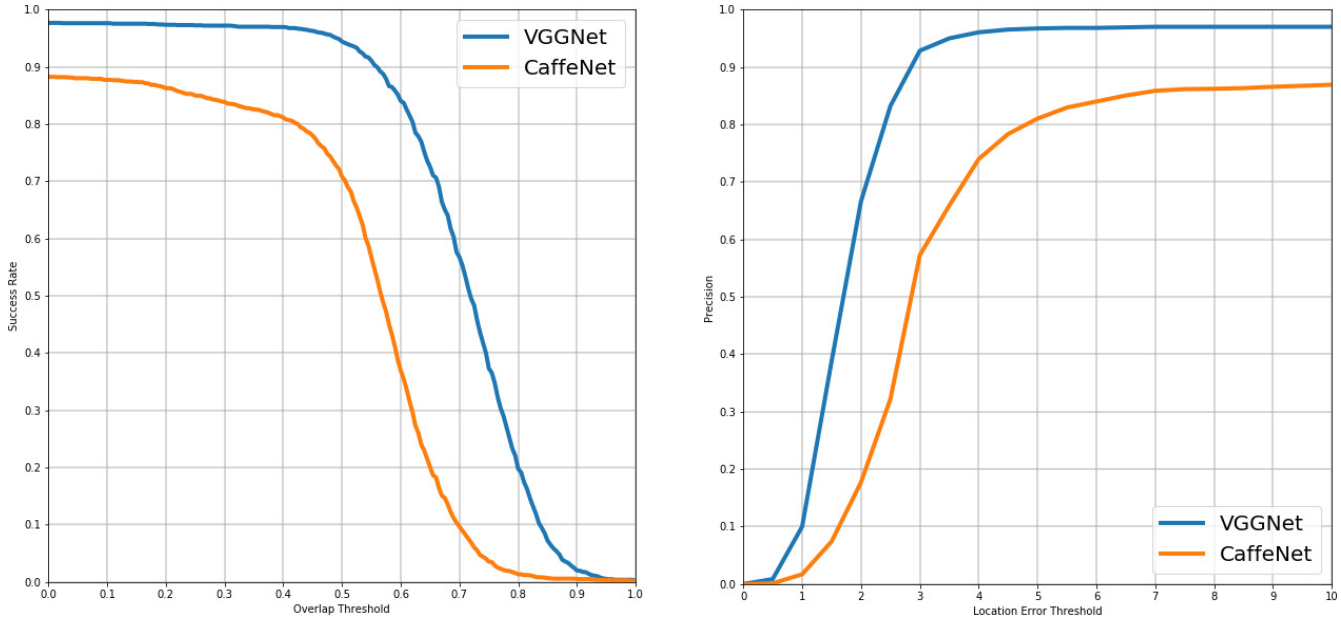


Fig. 14 Target tracking indicators tested on the table tennis video dataset. The commonly used target tracking indicators, as well as the comparison of different models on these indicators. **(a)** success curve. **(b)** precision curve.

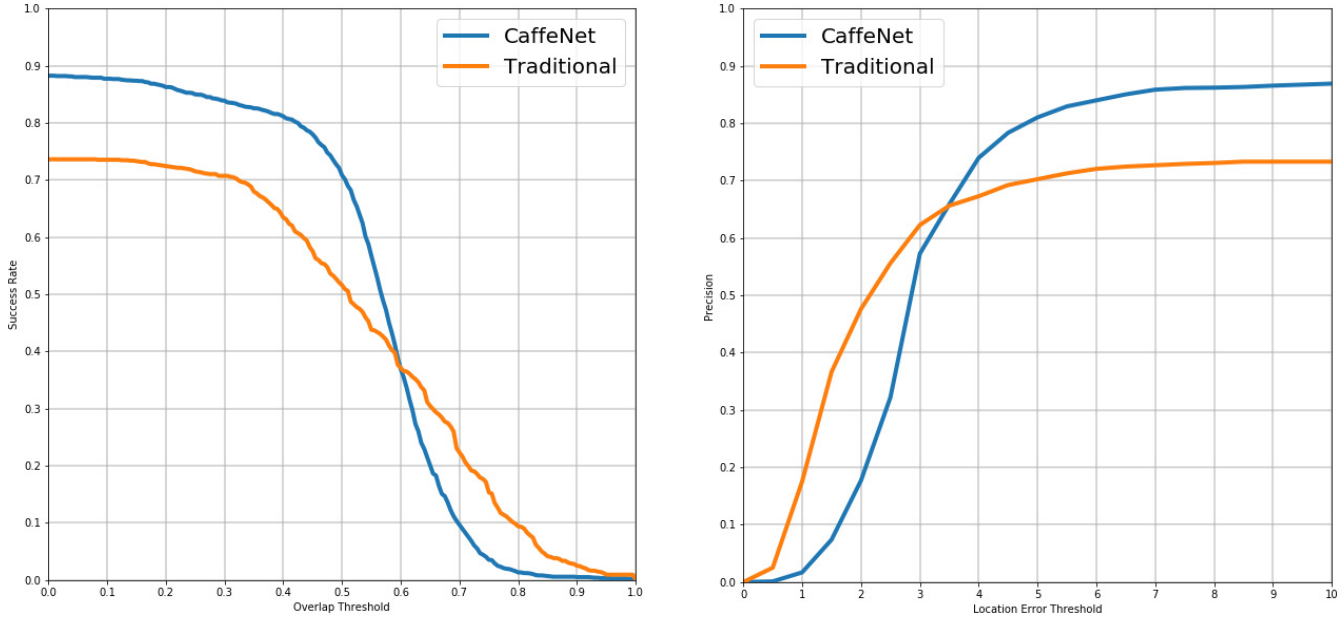


Fig. 15 Comparison of traditional tracking algorithms and deep learning tracking algorithms. **(a)** success curve. **(b)** precision curve.

accuracy of the CaffeNet model, it is easy to see that the target cannot be fully included in the search box, resulting in the loss of the target. In addition, because the search box of the next frame is determined by the target bounding box of the previous frame, when the predicted bounding box is larger than the actual bounding box, it will cause the search box to become larger and larger such that the model cannot recognize the target objects that are too small in the search box.

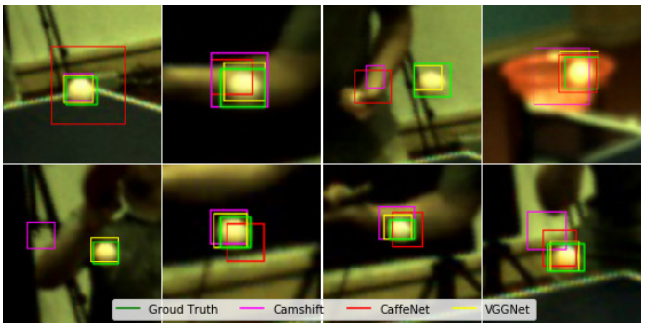


Fig. 16 Visualization of the tracking performance of each model on the table tennis video dataset.

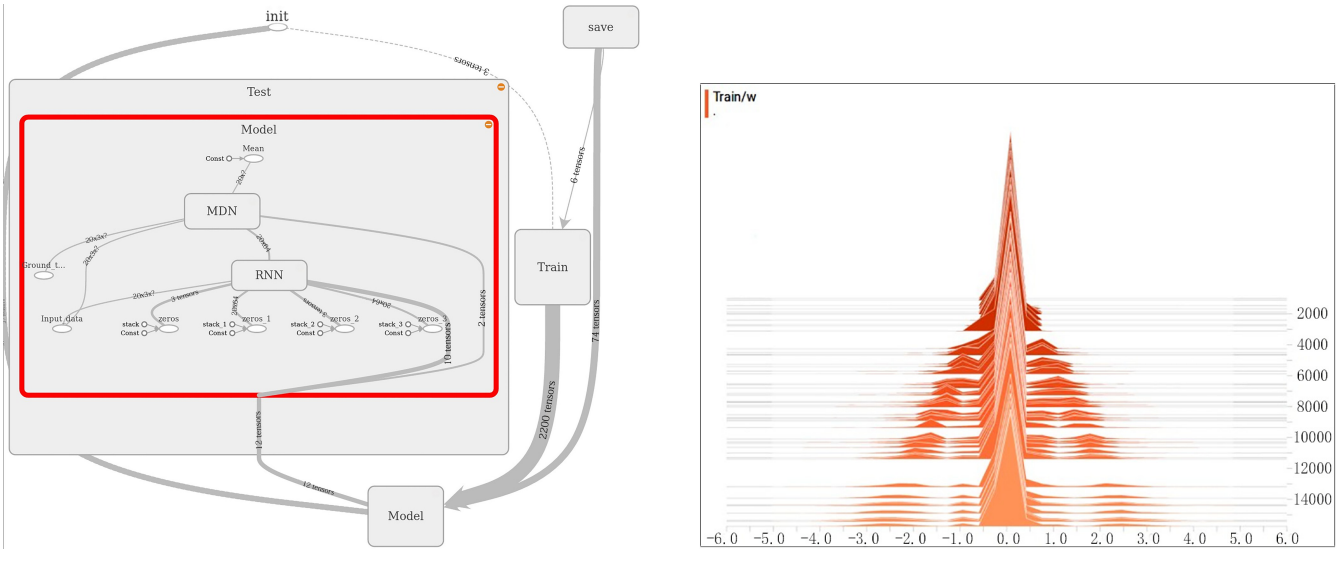
This paper also implements a simple tracking framework based on color features and the CamShift algorithm to compare with the traditional methods. The tracking framework is compared with the deep learning framework on the same video data set, and the results are shown in Fig. 15b. In the actual test, it is found that the traditional method is easy to drift when there are background interferers (arm, table sideline). Even if the tracking is successful, the target cannot be surrounded by a small response area. In fact, in the use of traditional methods, the environment will be constrained.[6] For example, only shoot the still table area, only use orange table tennis. In addition, more auxiliary methods will be used, such as background subtraction and motion model based on Kalman filter.

Finally, Fig. 16 also shows the actual tracking effect of each of the above models, as well as the comparison with the actual value. It can be seen that the traditional method easily loses the target in the case of complex background, and in the case of successful tracking, the output of the model based on deep learning has a higher coincidence rate with the real value.

4.2. LSTM: Trajectory Prediction Model

For the implementation of the LSTM model, we use the open-source Tensorflow1 deep learning framework. Due to MDN and other operations such as Gaussian distribution sampling, the derivation process is more complex, so the Tensorflow framework is chosen to implement. For example, Fig. 17a generates a network structure based on the calculation diagram, which can help to check whether the operation is correct. With the statistics of the network weight and the intermediate results in the training process in Fig. 17b, we can have a more specific understanding of the quality of the training results and improve the efficiency of parameter adjustment.

This paper uses the Python interface of Tensorflow to write the structure, training, and testing code of the LSTM model. To use it in the real environment, we used the more efficient C++ interface to write the code that uses the LSTM model to predict.



(a) **(b)**

Fig. 17 Network structure and statistics generated by Tensorflow. **(a)** Network structure of the LSTM model generated by Tensorflow. **(b)** Distribution of a certain weight in the network statistics.

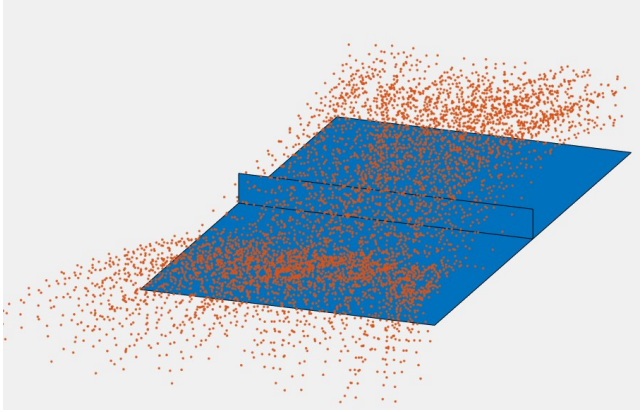


Fig. 18 Training data of LSTM model.

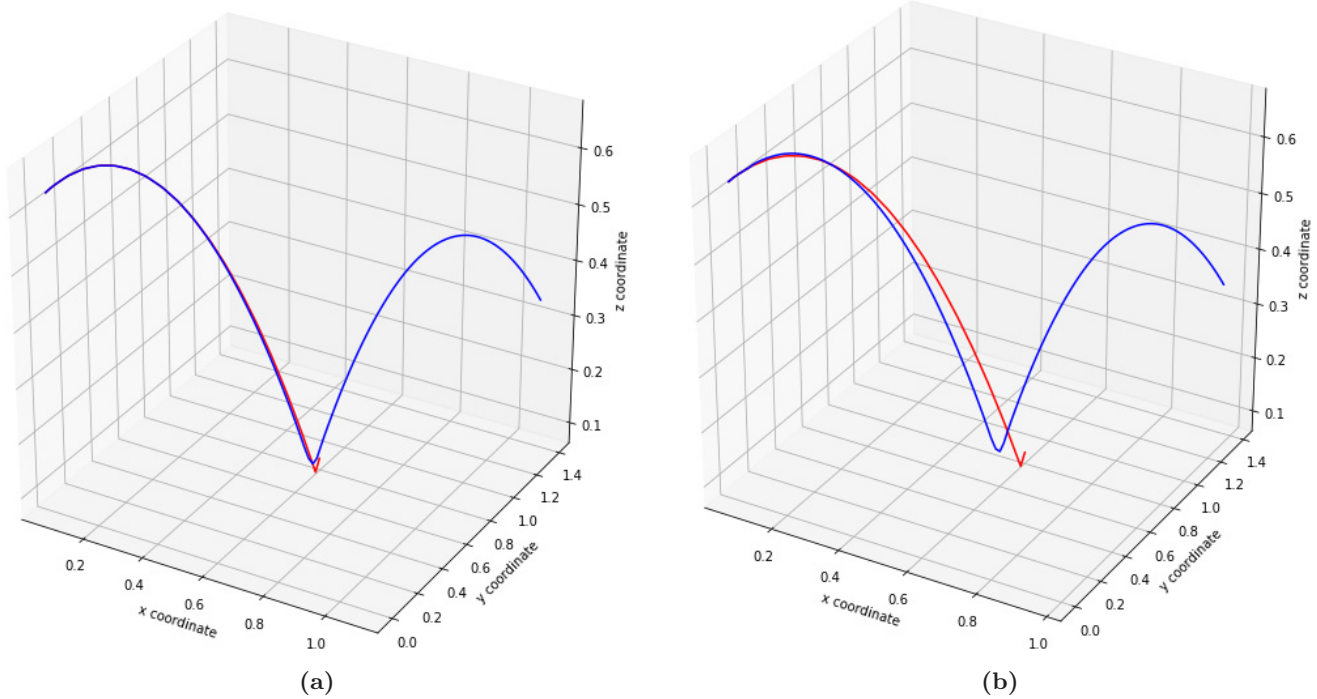
4.2.1. *Model training*

This paper trains the LSTM model on the 3D coordinate dataset of table tennis trajectories. First of all, 20,000 ping-pong trajectories are generated using the open-source physics engine Bullet3. Then, the initial speed, position, and direction are randomly selected to diversify the data as much as possible. Part of the trajectory is shown in Fig. 18. In addition, a small number of ping-pong tracks are collected by high-speed cameras. In the training process, we intercept a complete track of table tennis as input and translate the same track backward one unit in time as the target output. Thus, for three-dimensional coordinate information at time t, the target output is the coordinate of the trajectory at time t. To restore the error of the real measurement trajectory, noise is added to the input data, but the target output is not processed.

The LSTM model used in this paper contains two hidden layers, and the length of the state vector of each layer is 64. The mixed Gaussian network predicts two multidimensional Gaussian distributions. The number of Gaussian distributions depends on the number of states. Because table tennis mainly has two conditions of flight and rebound, two Gaussian distributions are enough to describe these states, but too many distributions may lead to overfitting. The model trains 30 cycles on the training set, traverses the whole training set every process. The initial learning rate is $5 \times 10^{-}3$, and each cycle is multiplied by the attenuation coefficient of 0.95.

4.2.2. *Performance analysis*

Figure 19 shows the results of the trained model on the validation set. It can be seen that a predictable trajectory that is very close to the target trajectory can be generated by inputting the initial coordinates into the neural network. The



(a) **(b)**

Fig. 19 Trajectory prediction results. **(a)** The prediction result when 30 coordinates are input. **(b)** The result of inputting 4 coordinates.

fewer coordinates entered means that the model can make predictions earlier. With the increase of the number of input coordinates, the predicted trajectory is closer to the target. It can also be seen that the model can accurately predict the collision event and change the trajectory direction when the coordinates are relative to the ball table.

As mentioned earlier, an essential task of trajectory prediction of a table tennis robot is to predict the landing point accurately. The next part is to test the performance of the LSTM model landing point prediction. For a point in the sequence, if its z coordinate is less than the z coordinate of the two adjacent points, it will be regarded as the landing point. The quality of the model is measured by calculating the distance between the predicted landing point and the target landing point. A test is carried out on a test set containing 4000 tracks, and a comparison is made by changing the initial sequence length of the input. Because there is noise in the input track, which may impact the prediction results, we have carried out the same test on the input without noise, and the results are shown in Table 5. It can be seen that with the increase in the number of input coordinates, the prediction is more accurate. When the input is in an ideal state, the prediction accuracy of different input lengths is improved. For example, when the input sequence size is 30, the error is less than 40 mm. The actual diameter of the ping-pong ball is 40 mm, so this error is acceptable.

To compare with the traditional methods, this paper also implements a trajectory prediction model based on the Extended Kalman Filter and tests the landing point prediction according to Ref. 22. The results are shown in Table 6. Compared with the LSTM model, the traditional model has higher accuracy when there are more input coordinates. However, as the input decreases, the accuracy of the conventional model falls faster. This paper found that this is due to the lack of input information, and the model cannot obtain additional information from pre-training.

Table 5 Performance of Landing Point Prediction of LSTM Model.

Input Sequence Length	4	10	30
Average error (mm)	181.93	79.73	36.48
Ideal average error (mm)	162.6	73.26	33.65

Table 6 Performance of Landing Point Prediction of LSTM Model with Extended Kalman Filter.[22]

Input Sequence Length	4	10	30
Average error (mm)	621.16	90.16	17.93
Ideal average error (mm)	599.28	79.38	14.05

Table 7 Performance of Landing Point Prediction of LSTM Model with Extended Kalman Filter.[22]

Input Sequence Length	4	10	30
Average error (mm)	6.39	6.38	6.38
Ideal average error (mm)	0.72	0.64	0.62

For the LSTM model, we also made a network analysis to know its internal operation to observe whether the MDN network has learned to predict the rebound of ping-pong balls on the table. As mentioned above, the MDN model outputs two parameters of the multi-dimensional Gaussian distribution. When a bounce occurs, the Gaussian weight of the predicted "bounce" action should be higher. Because it is mainly the z coordinate that changes during the bounce, the multi-dimensional Gaussian distribution is projected onto the z-axis. The result is shown in Fig. 20. Because the Gaussian distribution represents the displacement from the current coordinate to the next frame, the instantaneous velocity on the z-axis should be less than 0 before the bounce and greater than 0 after the bounce. The figure shows this situation correctly: before the rebound, the mean of the two Gaussian distributions is less than 0, but after the rebound, the standard deviation of the Gaussian distribution with a mean greater than 0 is more minor and has a higher weight.

Finally, we also test the time performance of the LSTM model. This time the model is run in CPU mode. On the CPU of Intel i7-6700 3.40 GHz, the average time for making a prediction is 0.35 ms.

4.3. Integrated Tracking Structure

This paper uses the Codebook [23] algorithm to subtract the background on the YCrCb channel of the image. Because the video is filmed with a high-speed camera under indoor lighting conditions, there will be frequent brightness changes and noise, as shown in Fig. 21. However, by operating on the YCrCb channel, the Codebook method can reasonably deal with regular brightness changes and ignore the noise generated by the light.

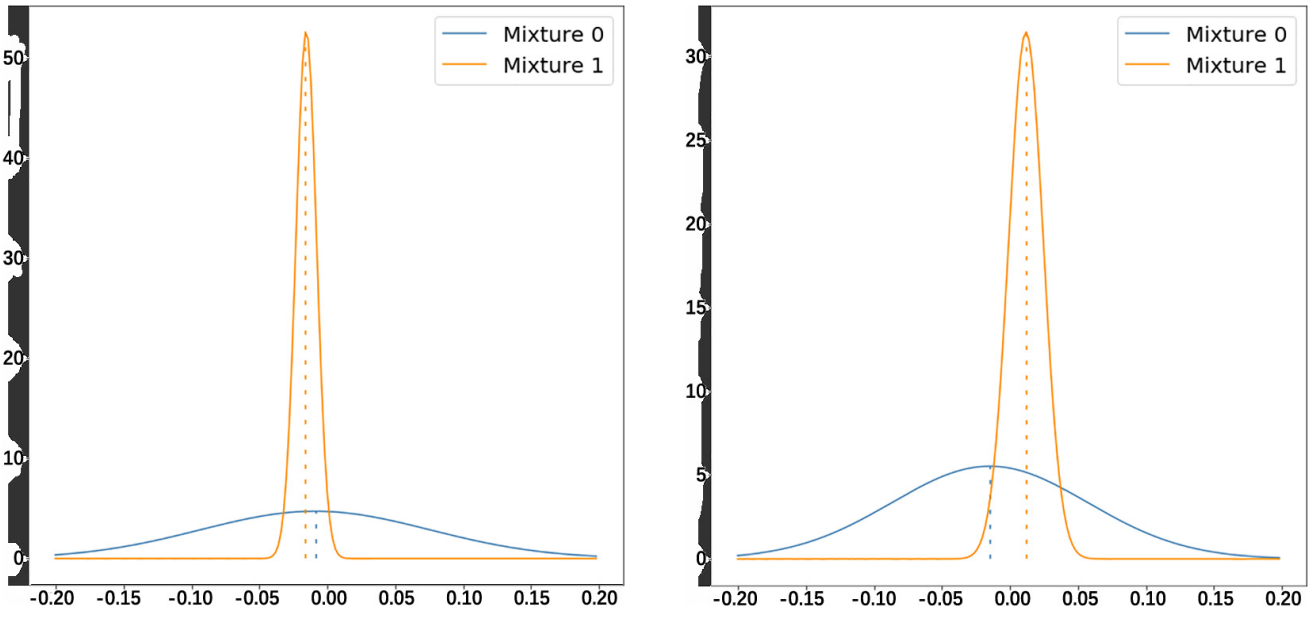


Fig. 20 TMDN predicts table tennis rebound action. The internal state of MDN during the rebound of the ping-pong ball. **(a)** The output of MDN before rebound. **(b)** MDN after rebound Output.

Fig. 21 Noise of input video.

A separate candidate region is obtained from the foreground image by morphological operation, and the candidate region is filtered according to the shape, size, and other features. Then the candidate region is input into the CNN model. The result is shown in Fig. 22.

In the stage of collecting training data, we also use background subtraction to automatically cut table tennis pictures in videos with less motion interference to improve tagging efficiency.

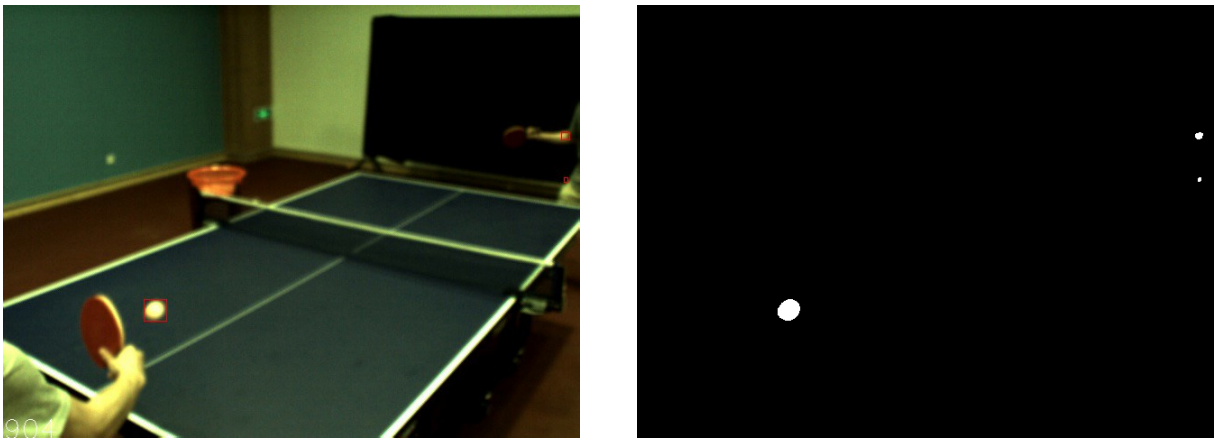
4.3.1. 3*D coordinate reconstruction*

First of all, using the chessboard calibration method, the chessboard pictures are taken from multiple angles, and the internal parameter matrix $M_{3\times3}$ and the distortion coefficient of the camera are obtained by using the built-in calibration function of OpenCV. Then, the internal parameter matrix is used to convert the 3D coordinates of the camera coordinate system into the 2D coordinates of the camera plane

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M_{3\times3} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}. \tag{8}$$

Then, the ping-pong table region in the image is identified by the color feature, and the boundary line is obtained by Hough transform, as shown in Fig. 23. The coordinates of the four corners of the ball table are obtained through the intersection of the boundary line. Then the external parameter matrix from the camera to the ball table is calculated, including the rotation matrix $R_{3\times3}$ and the displacement matrix $T_{3\times1}$. Finally, the external parameter matrix is used for the conversion between the camera coordinate system and the world coordinate system

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \tag{9}$$



(a) (b)

Fig. 22 Background subtraction result. (a) Input image. (b) The extracted foreground area.

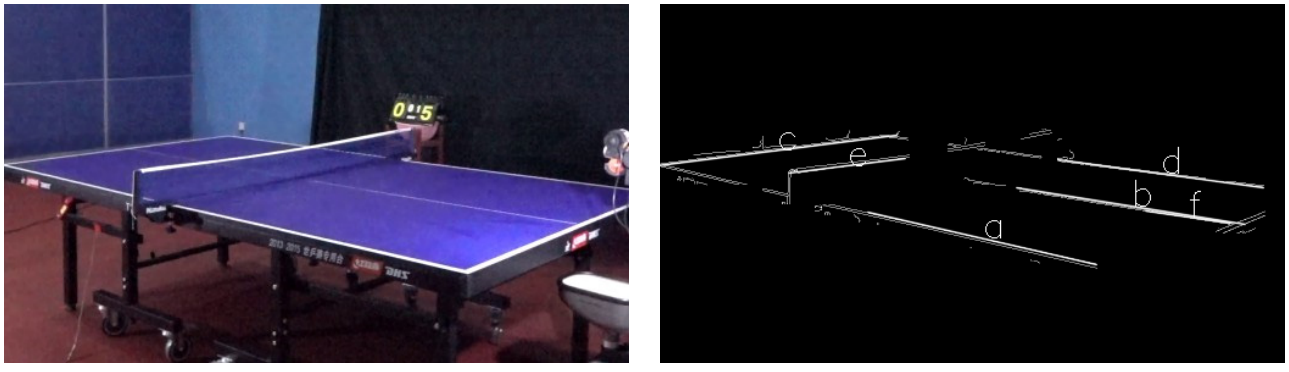


Fig. 23 Extracting the boundary line of the table.

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M_{3\times3} \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \quad (10)$$

Finally, combining the above two formulas, we know the two-dimensional coordinates of a certain point in the two camera planes and use the following formula to calculate the three-dimensional coordinates.

Z_c is the Z coordinate of the point in a camera coordinate system, which is an unknown number, and u, v are the coordinates of the point in the plane of the camera. R is the rotation matrix, and T is the displacement matrix. Finally, X_w, Y_w, Z_w are the three-dimensional coordinate in the world coordinate system, which requires the solution. There are four unknowns, and each of the two cameras provides one of the above equations to solve directly by linear algebra.

The drop-off statistics shown in Fig. 28 can better characterize the player's offensive options and help players better train.

4.4. Tracking Framework Performance

To meet the real-time requirements, this paper uses C++ to implement the tracking framework and uses Qt to write a graphical interface. The operation of the program is shown in Fig. 24. The top of the interface offers tracking from two perspectives: table tennis's instantaneous speed at the bottom left. The top view of the table at the lower right shows the position of table tennis (green) and the actual landing point (red) in real-time, as well as the predicted location of the impact point (yellow). We encapsulate the CNN model based on Caffe, the LSTM model based on Tensorflow, and the computer vision method used in the framework into dynamic libraries, which are linked in the main

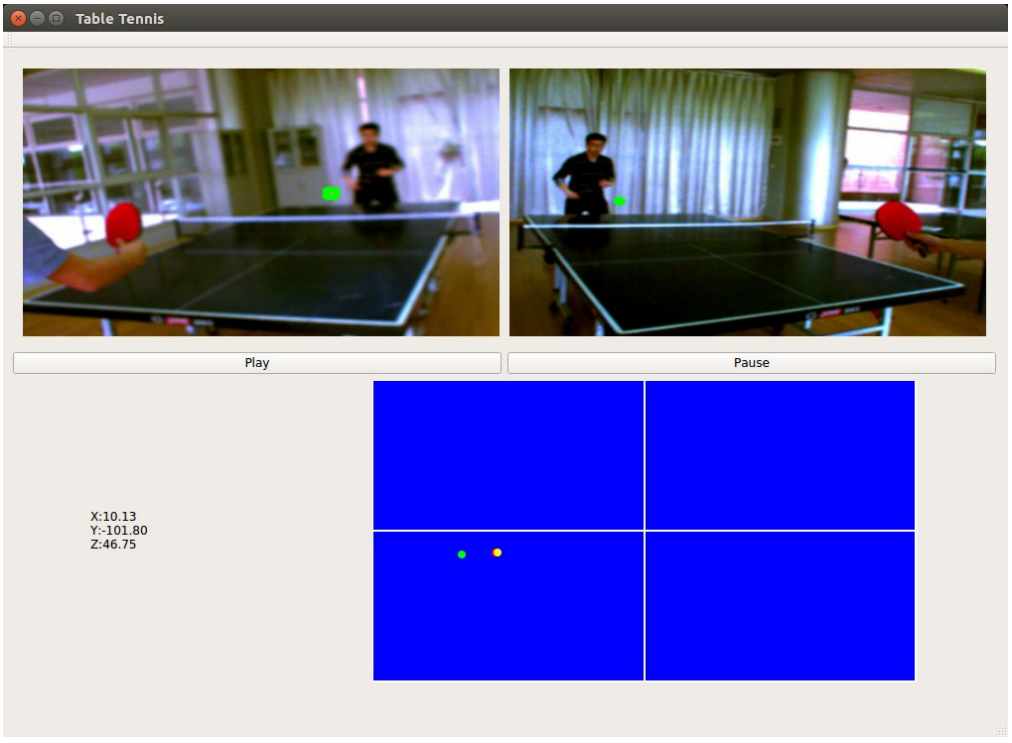


Fig. 24 Graphical interface of ping-pong tracking system.

Fig. 25 Video test taken with actual environment.

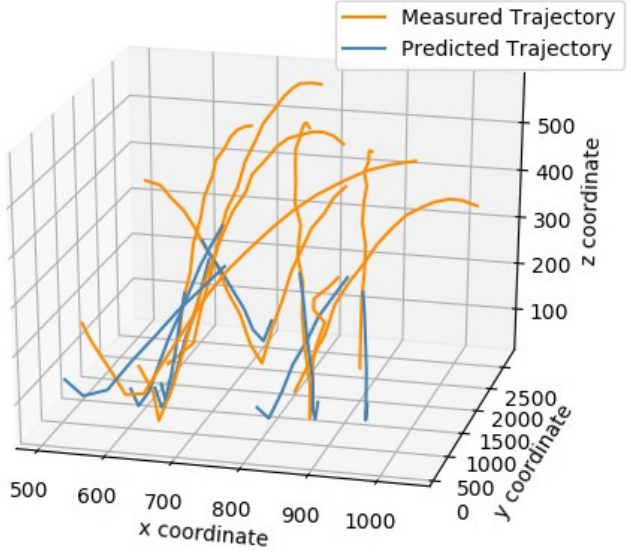


Fig. 26 Trajectory prediction in actual environment.

program of the framework. The program can run successfully on Ubuntu16.04.

This work tests the reliability of the tracking framework in a real environment. As shown in Fig. 25, when a synchronized camera is used to shoot a video of two table tennis players playing against each other, there will be disturbances (players moving in the distance) and occlusion (occlusion of nearby rackets) continuously throughout the process. Figure 26 shows the trajectory prediction in the virtual environment. Prediction starts when the ball flies over the net until it bounces off the table. It can be seen from Fig. 24 that the tracking framework records the trajectory of the table tennis relatively completely and predicts the trajectory more accurately.

We have fully implemented the table tennis tracking framework. However, the framework only

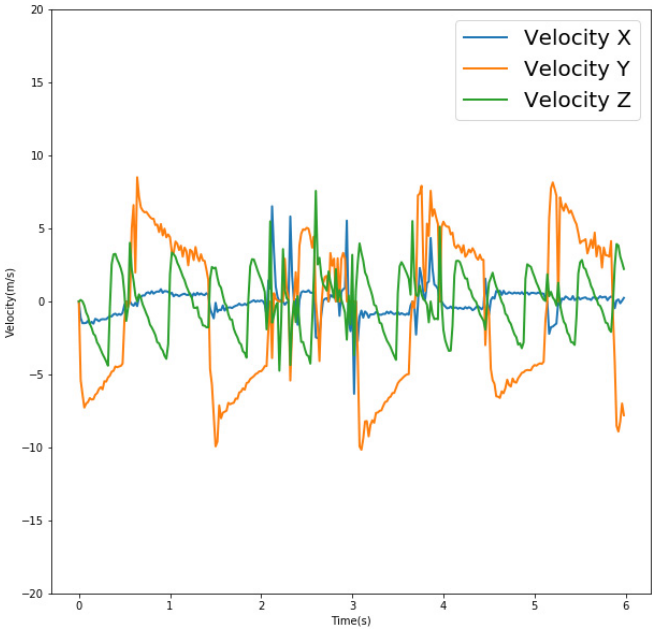


Fig. 27 Instantaneous speed statistics of table tennis.

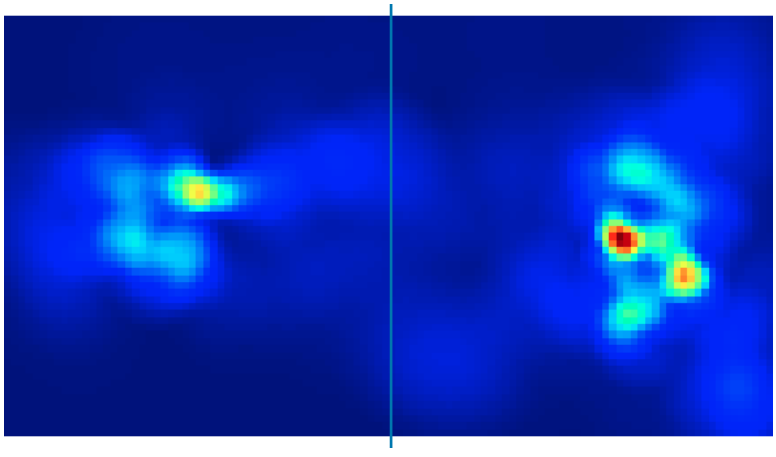


Fig. 28 Statistics of table tennis placement.

accomplishes two fundamental problems: table tennis tracking and trajectory prediction. More work can be done based on the framework. In addition to the landing point prediction and hitting time determination of the table tennis robot mentioned at the beginning of this paper can also be statistically and analyzed based on many trajectory data obtained by tracking. For example, the instantaneous speed curve of a table tennis ball is shown in Fig. 27. Excluding the noise caused by a small amount of tracking failure, some physical laws can be observed, such as the z-axis speed (perpendicular to the table) reverses when bouncing and decreases linearly during flight. For example, the y-axis speed (parallel to the sideline) is reversed when hitting the ball and decreases during flight.

5. CONCLUSION

In this paper, fractal AI provides an effective solution to deal with the complex structural prediction and tracking via designing a framework based on deep learning, the aim of which is to replace the traditional methods with more intuitive ideas and more robust performance, to solve the two fundamental problems of table tennis tracking system, namely, tracking and trajectory prediction. First, we proposed a structured output CNN and

bounding box regression model, successfully applied to the table tennis track. In addition, we convert the LSTM model used for sequence prediction to the problem of ping-pong trajectory prediction and puts forward a more straightforward and intuitive solution than the traditional method. Finally, it integrates each model to achieve a complete tracking framework. We have carried out various experiments on the network model, explored the internal operation mode of the network, and confirmed the availability of the model. We also write a graphical interface program to show the actual effect of the framework.

Compared with the traditional computer vision methods, the proposed framework does not need to manually select features to track table tennis and does not need to predict the trajectory based on many physical formulas. Input and output are clearly defined in all models, and many exceptional cases can be solved automatically within the model. Compared with the general tracking algorithm, the framework is specially designed for the tracking problem of table tennis, which solves the problem of finding the tracking target in the first frame, removes the online learning process through the pre-training of a large number of table tennis pictures, and improves the processing speed. A more reasonable motion model is designed according to the characteristics of table tennis.

These are the several future directions where we can employ tracking and trajectory predictions; (1) The rotation information of table tennis is measured by the method based on computer vision, and the measurement results are integrated into the existing framework to improve the performance of tracking and trajectory prediction. (2) Deep learning can be used to recognize human actions. In table tennis video, athletes' body movements are also an important source of information. By identifying the player's swing, additional information can be provided for ball tracking. It can also analyze the human body movements separately, such as the offensive characteristics and tactical analysis of athletes. Based on the strong ability of deep learning in image recognition and the existing work related to deep learning attitude analysis, it can be easily transferred to table tennis video analysis. (3) The reinforcement learning method is used to realize the design of table tennis robot. In depth learning methods, such as reinforcement learning, can directly take the game picture pixels as input, and ++reward or punish the network through the game operation, so that it can learn to make correct actions at a specific time and obtain higher game scores. More powerful network structure and simple training methods make deep learning become the main direction of table tennis robot design.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grants 62077037 and 61872241, in part by Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102, in part by the Science and Technology Commission of Shanghai Municipality under Grants 18410750700 and 17411952600, in part by Shanghai Lin-Gang Area Smart Manufacturing Special Project under Grant ZN2018020202-3, and in part by Project of Shanghai Municipal Health Commission(2018ZHYL0230). Haoxuan Li and Saba Ghazanfar Ali contributed equally to this work.

REFERENCES

1. L. Wang, Y. Wu, J. Xu, H. Zhang, X. Wang, J. Yu, Q. Sun and Z. Zhao, Status prediction by 3D fractal net CNN based on remote sensing images, *Fractals* **28** (2020) 2040018.
2. B. Liu, L. Jin and C. Hu, Fractal characterization of silty beds/laminae and its implications for the prediction of shale oil reservoirs in Qingshankou formation of Northern Songliao Basin, Northeast China, *Fractals* **27** (2019) 1940009.
3. Y. Karaca, D. Baleanu, M. Moonis, K. Muhammad, Y.-D. Zhang and O. Gervasi, Editorial, *Fractals* **29** (2021) 2102002.
4. Y. Liu, X. Pang, X. Zhao, A. Stein, X. Zhang, Q. Ji, M. Qu and J. Dong, Prediction of the Antarctic marginal ICE zone extent based upon its multifractal property, *Fractals* **29** (2021) 2102002-1.
5. R. Piña-Vega, M. Valtierra-Rodriguez, C. A. Perez-Ramirez and J. P. Amezquita-Sanchez, Early prediction of sudden cardiac death using fractal dimension and ECG signals, *Fractals* **29** (2021) 2150077.
6. Y. Zhang and R. Xiong, Real-time vision system for a ping-pong robot, *Sci. Sin. Inf.* **42** (2012) 1115.
7. K. Mülling, J. Kober, O. Kroemer and J. Peters, Learning to select and generalize striking movements in robot table tennis, *Int. J. Robot. Res.* **32** (2013) 263.
8. L. Xi, B. Sun and J. Yao, Fractality of substitution networks, Imagenet classification with deep convolutional neural networks, *Fractals* **27** (2019) 1950034.

9. A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Adv. Neural Inf. Process. Syst.* **25** (2012) 1097.
10. R. Girshick, J. Donahue, T. Darrell and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2014), pp. 580–587.
11. S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.* **9** (1997) 1735.
12. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, Playing atari with deep reinforcement learning, arXiv:1312.5602.
13. D. Held, S. Thrun and S. Savarese, Learning to track at 100 FPS with deep regression networks, in *European Conference on Computer Vision* (Springer, 2016), pp. 749–765.
14. A. Graves, Generating sequences with recurrent neural networks, arXiv:1308.0850.
15. R. Shah and R. Romijnders, Transferring rich feature hierarchies for robust visual tracking, arXiv:1608.03793.
16. N. Wang, S. Li, A. Gupta and D.-Y. Yeung, arXiv:1501.04587.
17. R. Girshick, Fast R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, 2015), pp. 1440–1448.
18. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in *Proceedings of the 22nd ACM International Conference on Multimedia* (ACM, 2014), pp. 675–678.
19. K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.
20. L. Wang, W. Ouyang, X. Wang and H. Lu, Visual tracking with fully convolutional networks, in *IEEE International Conference on Computer Vision* (IEEE, 2015), pp. 3119–3127.
21. Y. Wu, J. Lim and M.-H. Yang, Online object tracking: A benchmark, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2013), pp. 2411–2418. .
22. Y. Zhang, R. Xiong, Y. Zhao and J. Wang, Real-time spin estimation of ping-pong ball using its natural brand, *IEEE Trans. Instrum. Measure.* **64** (2015) 2280.
23. K. Kim, T. H. Chalidabhongse, D. Harwood and L. Davis, Real-time foreground–background segmentation using codebook mode, *Real-time Imag.* **11** (2005) 172.