Review of Scientific Instruments ARTICLE scitation.org/journal/rsi

Convolutional neural network-based pose mapping estimation as an alternative to traditional hand–eye calibration

Cite as: Rev. Sci. Instrum. 94, 065002 (2023); doi: 10.1063/5.0147783
Submitted: 26 February 2023 • Accepted: 13 May 2023 •
Published Online: 1 June 2023

Kuai Zhou,[1] Xiang Huang,[1,a)] Shuanggao Li,[1] and Gen Li[2]

AFFILIATIONS

[1] College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[2] Suzhou Research Institute, Nanjing University of Aeronautics and Astronautics, Suzhou, China

[a)] **Author to whom correspondence should be addressed:** xhuang@nuaa.edu.com

ABSTRACT

The vision system is a crucial technology for realizing the automation and intelligence of industrial robots, and the accuracy of hand–eye calibration is crucial in determining the relationship between the camera and robot end. Parallel robots are widely used in automated assembly due to their high positioning accuracy and large carrying capacity, but traditional hand–eye calibration methods may not be applicable due to their limited motion range and resulting accuracy problems. To address this issue, we propose using a pose, nonlinear mapping estimation method to solve the hand–eye calibration problem and have constructed a 1-D pose estimation convolutional neural network (PECNN) with excellent performance, through experiments and discussions. The PECNN achieves an end-to-end mapping of the variation of the target object pose to the variation of the robot end pose. Our experiments have shown that the proposed hand–eye calibration method has high accuracy and can be applied to the automated assembly tasks of vision-guided parallel robots. Moreover, the method is also applicable to most parallel robots and tandem robots.

I. INTRODUCTION

Industrial robotics has been in development for several decades, and it initially gained popularity in the automotive industry.[1] The automotive industry extensively uses industrial robots to handle work with high-precision requirements and high repeatability, including welding, painting, dispensing, and assembly. Apart from the automotive industry, industrial robots are used in aircraft manufacturing,[2,3] pharmaceutical manufacturing,[4,5] metal processing,[6,7] and other fields as well. In the past, industrial robots mainly relied on programming to automate tasks, but this method was rigid and required a lot of time and effort for programming and debugging. To address this issue, vision-guided robots were created, which can achieve guidance of robot motion through vision systems. These robots can accurately identify and locate objects and, thus, perform tasks more flexibly.[8,9] The importance of the vision system for industrial robots is self-evident and is one of the key technologies used to realize robot automation and intelligence.[10] Figure 1 depicts a typical vision-guided robot with a camera near the end that senses the working environment.

The robot can have a camera mounted in a fixed position with a fixed field of view (eye-to-hand), or the camera can be mounted at the end of the robot, and the camera can acquire a new image with the robot's motion (eye-in-hand). However, the robot can only perceive the external world through its coordinate system. Therefore, to obtain the poses of the target object in its own coordinate system in the workspace, it is necessary to determine the pose relationships between the object and the camera, between the camera and the end of the robot, and between the end of the robot and its base coordinate system. These three tasks can be achieved by visual measurement, hand–eye calibration, and robot calibration, respectively. The focus of this paper is on hand–eye calibration. In addition to its application to vision-guided robots, the hand–eye calibration technique can solve multi-sensor calibration problems, such as integrated 3D scanning systems,[11] human eye navigation vision inertial integration,[12] and camera and inertial/magnetic sensor calibration.[13]

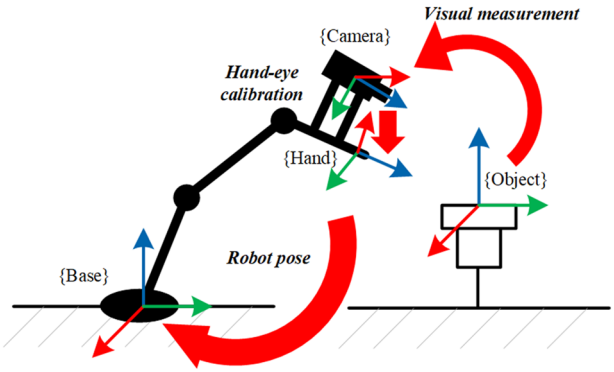


FIG. 1. Vision-guided robot.

Hand–eye calibration, as its name implies, is the process of determining the transformation relationship between the robot end coordinate system and the camera coordinate system. Depending on the camera mounting method, it can be divided into two categories: eye-in-hand and eye-to-hand, and our research focuses on eye-in-hand calibration. In an eye-in-hand setting, as shown in Fig. 2, the transformation relationship between the robot base coordinate system and the target object coordinate system is determined, along with the transformation relationship between the robot end coordinate system and the camera coordinate system. With two motion poses at the end of the robot, an equation can be established:

$$H_{HB1} \times H_{CH} \times H_{OC1} = H_{HB2} \times H_{CH} \times H_{OC2}, \quad (1)$$

where H_{HB} denotes the pose of the robot end in the robot base coordinate system, H_{CH} denotes the pose of the camera in the robot end coordinate system, and H_{OC} denotes the pose of the target object in the camera coordinate system. Further transforming this equation, we can get

$$\left(H_{HB2}^{-1} \times H_{HB1}\right) \times H_{CH} = H_{CH} \times \left(H_{OC2} \times H_{OC1}^{-1}\right), \quad (2)$$

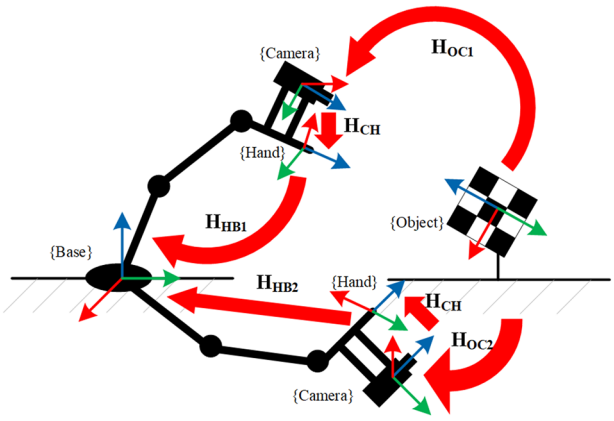


FIG. 2. Eye-in-hand calibration principle.

which can be further abstracted as

$$AX = XB. \quad (3)$$

X is the hand–eye relationship matrix that needs to be requested. The various methods that have emerged in recent years are not so new methods of hand–eye calibration as they are new solutions to this equation $AX = XB$.

Since the introduction of the concept of hand–eye calibration by Shiu and Ahmad[14] and Tsai and Lenz[15] in the 1980s, various hand–eye calibration algorithms have emerged, broadly classified into two major categories: The first type calculates the rotation matrix in the hand–eye relationship matrix first and then the translation vector, which is referred to as the separation method. The second type calculates the rotation matrix and translation vector simultaneously, which is known as the synchronous method. Several hand–eye calibration methods belonging to the separation method include the representation based on rotation matrix by Shiu and Ahmad,[14] Tsai and Lenz,[15] and Horaud and Dornaika,[16] the representation based on Lie group and Lie algebra by Park and Martin,[17] and the representation based on quaternion by Chou and Kamel.[18] The synchronous method can be subdivided into analytical and numerical methods. Representative analytical methods are the methods based on quaternions by Lu and Chou[19] and Daniilidis,[20] the methods based on Sylvester's equation by Andreff *et al.*,[21] and the methods based on double tensor by Condurache and Burlacu.[22] Numerical methods differ according to the optimization algorithm, including Newton's gradient method by Gwak *et al.*,[23] the linear matrix inequality method by Heller *et al.*,[24] an alternative linear programming method by Zhao,[25] and the pseudo-inverse matrix method by Zhang[26] and Zhang *et al.*[27] In the separation method, the rotation matrix needs to be solved first and then the translation vector, and the errors generated when solving the rotation matrix will be substituted into the translation vector solution process, resulting in inaccurate results. Therefore, the synchronous method tends to have higher accuracy due to solving of the translation vector and rotation matrix simultaneously.

Compared to tandem robots, parallel robots have several advantages such as a stable structure, high positioning accuracy, space-saving design, high rigidity, load capacity, and speed.[28] They

FIG. 3. Parallel robot for aircraft assembly.

are particularly suitable for applications that require high assembly accuracy, such as aircraft assembly, as shown in Fig. 3. However, parallel robots have limited movement range, and the current method is no longer suitable for hand–eye calibration of parallel robots due to accuracy issues. The data collected for hand–eye calibration under restricted movement range has low robustness to noise.[14,15] In recent years, some scholars have attempted to apply deep learning technology to robot hand–eye calibration. Hua solved the hand–eye calibration of eye-to-hand using an optimized neural network, to map image coordinates to position coordinates.[29] However, compared to hand–eye calibration, the application of deep learning technology to industrial robot error compensation is more prevalent,[2,30] since the input and the output of error compensation are also pose data, which can provide some reference for hand–eye calibration.

To address the limitations of traditional hand–eye calibration algorithms in achieving high accuracy for parallel robot hand–eye calibration in manufacturing settings, our research proposes a novel approach based on a convolutional neural network (CNN) for pose mapping estimation. The hand–eye relationship matrix is implicitly encoded in the weight parameters of the trained neural network, which can seamlessly integrate with existing vision-guided robot architectures. When used, the CNN-based method can directly obtain variations in the robot's poses by inputting variations in target poses in the camera coordinate system. Our contributions are summarized as follows:

(1) We propose a novel pose mapping estimation method to replace traditional hand–eye calibration.
(2) We develop a new 1-D CNN model and a data acquisition method for training, after experimenting with various classical network architectures.

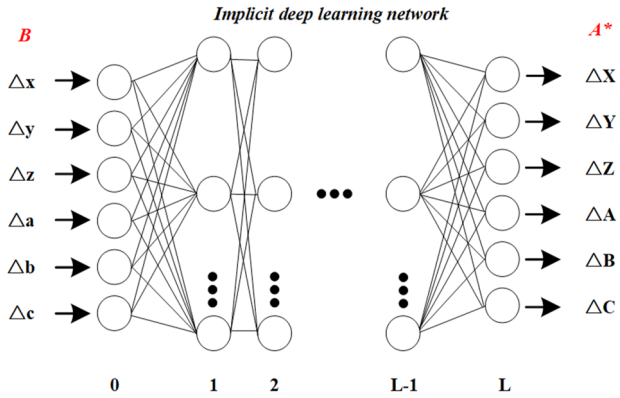


FIG. 5. Pose mapping estimation with the implicit network.

(3) Through a series of experiments, we demonstrate the accuracy and applicability of the proposed method.

The structure of this paper is outlined as follows: In Sec. II, the focus is on the neural network-based hand–eye calibration method, highlighting its advantages and the need for its use, as well as the principles of the pose mapping estimation. Section III provides details on the implementation of the proposed convolutional neural network, including the acquisition and pre-processing of training data, the determination of the basic network architecture, and the discussion of the optimal network architecture and hyperparameters. In Sec. IV, a series of experiments are conducted to compare the proposed deep learning-based hand–eye calibration method

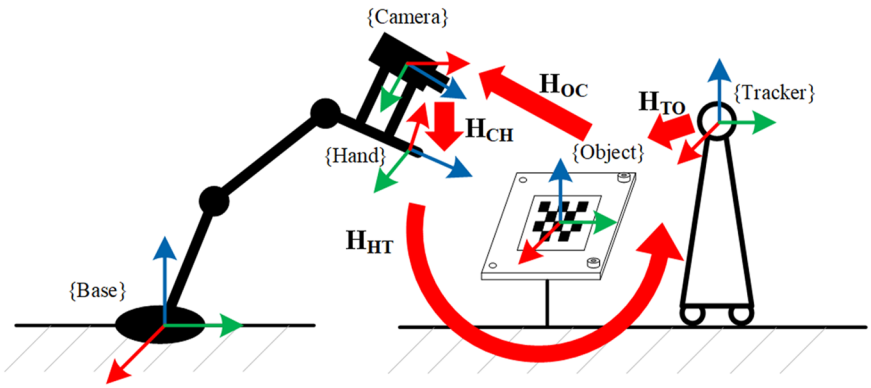
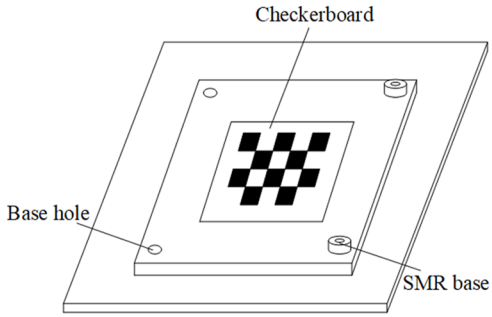


FIG. 4. Principle of direct measurement method.



with various hand–eye calibration methods, to verify its accuracy, performance in practical applications, and applicability.

II. NEURAL NETWORK-BASED HAND–EYE CALIBRATION

A. Advantages and necessity of using neural networks for hand–eye calibration

In a complete hand–eye calibration process, several factors can contribute to errors in the final calibration results due to noise interference with the calibration data. These factors include measurement errors caused by the visual measurement process and positioning errors caused by the robot motion process. The variation of these errors is often non-linear and does not vary with the amount of input. Traditional hand–eye calibration methods rely on an exact linear model and do not account for these errors. Therefore, applying traditional hand–eye calibration methods to data containing non-linear errors will inevitably lead to inaccurate hand–eye calibration results. In contrast, artificial neural networks use non-linear activation functions such as Sigmoid and ReLU (Rectified Linear Unit), which can efficiently map non-linearities and cover the above-mentioned non-linear errors during the learning phase of the network. This results in higher accuracy in the final inference phase, making neural networks a more suitable method for handling non-linear errors in hand–eye calibration, as demonstrated by Hua and Zeng's work.[29]

TABLE I. Mechanical parameters of the parallel robot.

Parameter	Value
Load capacity	0.5t
Moving speed	0–0.5 m/min
Maximum acceleration	0.1 m/s^2
Displacement resolution	0.001 mm
X stroke	±50 mm
Y stroke	±50 mm
Z stroke	150 mm
$A(RX)$ stroke	±14°
$B(RY)$ stroke	±7°
$C(RZ)$ stroke	±5°
Positioning accuracy	≤0.01 mm
Repeated positioning accuracy	≤0.005 mm

The traditional hand–eye calibration method is extremely inaccurate for parallel robots with restricted range of motion. In preliminary practical industrial applications, we temporarily use the direct measurement method based on a laser tracker for hand–eye calibration of parallel robots,[31] as shown in Fig. 4, because the neural network-based hand–eye calibration method has not been developed yet.

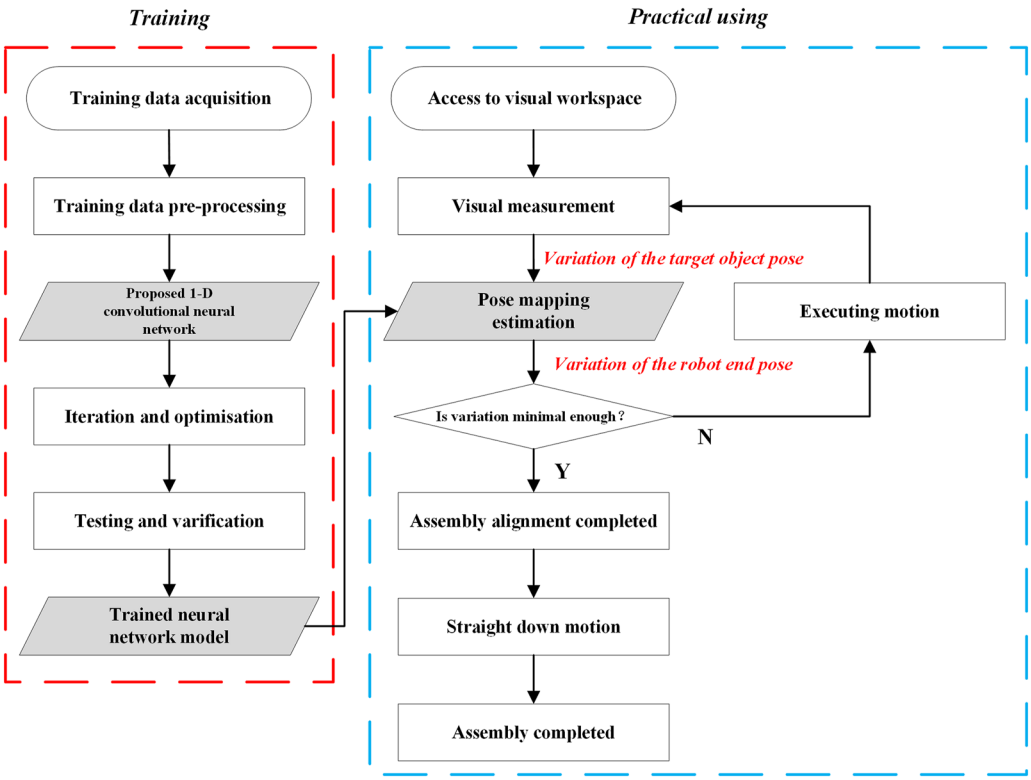


FIG. 6. Flow chart of pose mapping estimation.

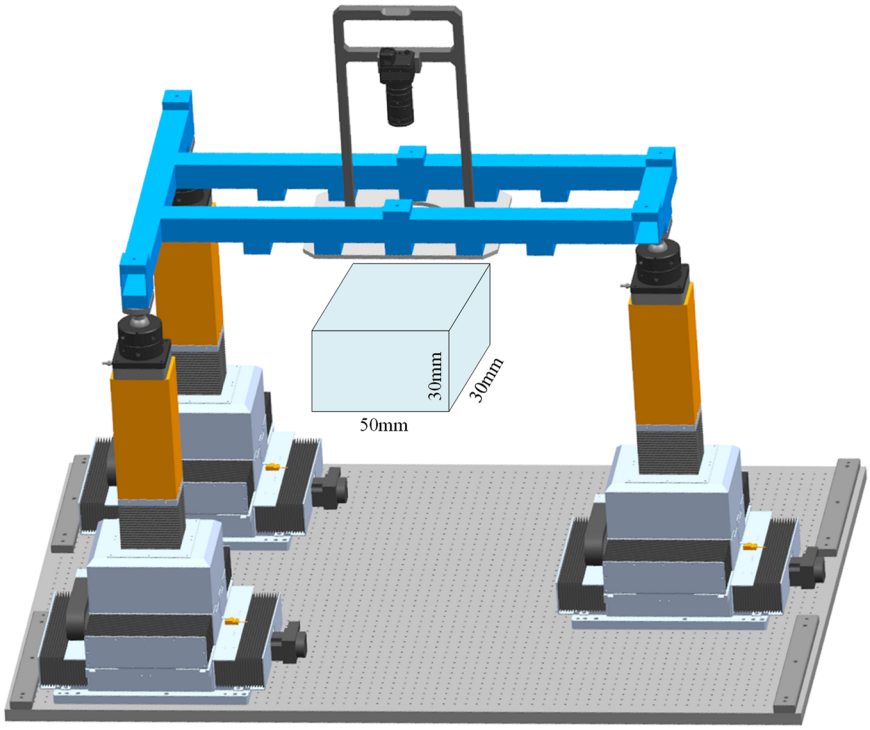


FIG. 7. Motion space of the parallel robot.

The key to the direct measurement method is a home-made calibration object consisting of a vision-oriented checkerboard calibration plate and four surrounding SMR (Spherically Mounted Retroreflector) bases to which laser tracker-oriented target balls can be mounted. The relation between the check board and the base holes is obtained by advance calibration with a hexagon image measuring instrument. The calibration process of the direct measurement method does not require robot motion and we can obtain

$$H_{HT} = H_{OT} \times H_{CO} \times H_{HC}, \tag{4}$$

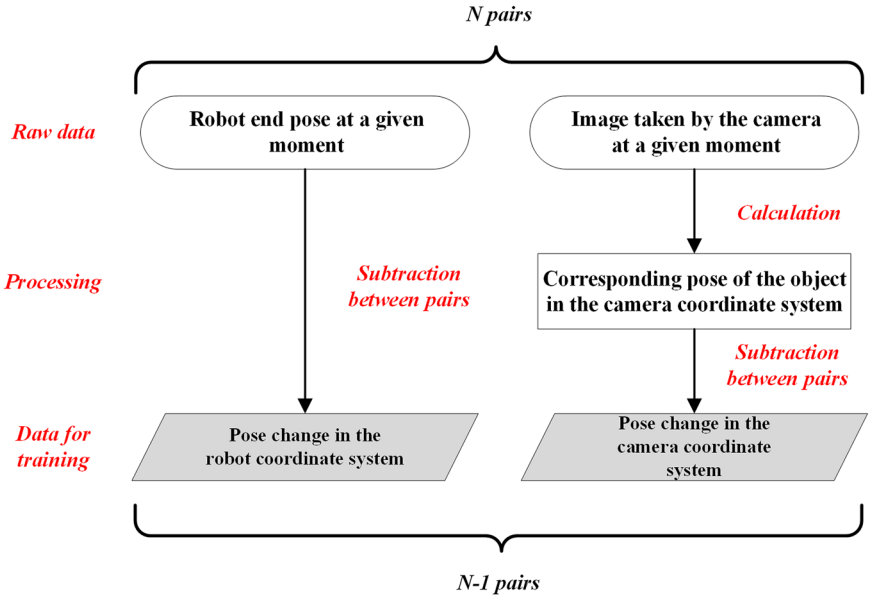


FIG. 8. Training data pre-processing process.

$$H_{HC} = {H_{OT}}^{-1} \times {H_{CO}}^{-1} \times H_{HT}, \tag{5}$$

$$H_{HC} = H_{TO} \times H_{OC} \times H_{HT}. \tag{6}$$

Specify $H_{AB} = {H_{BA}}^{-1}$; H_{HT} denotes the pose of the robot end in the coordinate system of the laser tracker, H_{OT} denotes the pose of the calibration object in the coordinate system of the laser tracker, H_{CO} denotes the pose of the camera in the coordinate system of the calibration object, H_{HC} denotes the pose of the robot end in the coordinate system of the camera, while the required H_{OC} can be calculated by using the industrial camera to photograph the checkerboard calibration plate, H_{TO} can be obtained by measuring the target ball position of the calibration object with a laser tracker, and H_{HT} can be obtained by measuring the target ball position at the end of the robot with a laser tracker.

The direct measurement method obtains the hand–eye transformation matrix by multiplying the transformation matrices obtained from three single measurements. However, the accuracy of this method is limited by the measurement uncertainty of the laser tracker and the processing progress of the calibrated object. Both are essentially one order of magnitude lower than the visual measurement accuracy, making it challenging to enhance the accuracy of the direct measurement method based on the laser tracker. To achieve higher accuracy in digital assembly, it is still essential to study the hand–eye calibration method based on deep learning.

B. Pose mapping estimation

Observe $AX = XB$; X is deterministic due to the fixed positions of the camera and the robot end. Furthermore, it can be considered that a deterministic A must obtain a deterministic B through the operation, i.e., A and B are considered to be in one-to-one correspondence; specifically,

$$A = {H_{HB2}}^{-1} \times H_{HB1} \overset{Correspondence}{\rightarrow} A^* = H_{HB2} \times {H_{HB1}}^{-1}, \tag{7}$$

$$B = H_{OC2} \times {H_{OC1}}^{-1}. \tag{8}$$

A corresponds to A^*, and A^* and B denote the variation of the robot end pose and the variation of the target object pose in the camera coordinate system in the two movements of the robot, respectively. A functional relation can be constructed:

$$A^* = f(B). \tag{9}$$

The above functional relation is trained and learned by the proposed convolutional neural network to achieve a pose nonlinear mapping of the variation B to the variation A^*. In the actual assembly process, B can be obtained by visual measurement, and the mapped A^* can be directly used for assembly alignment by adding it to the robot's current pose.

It is worth mentioning that throughout the hand–eye calibration process, we do not obtain the explicit hand–eye relationship matrix X, but implicitly include it in the deep neural network model, as shown in Fig. 5.

Unlike hand–eye calibration, in eye-to-hand, which also uses artificial neural networks, the eye-to-hand fits the poses rather than the pose variation.[29] Therefore, the raw data needed are in the same

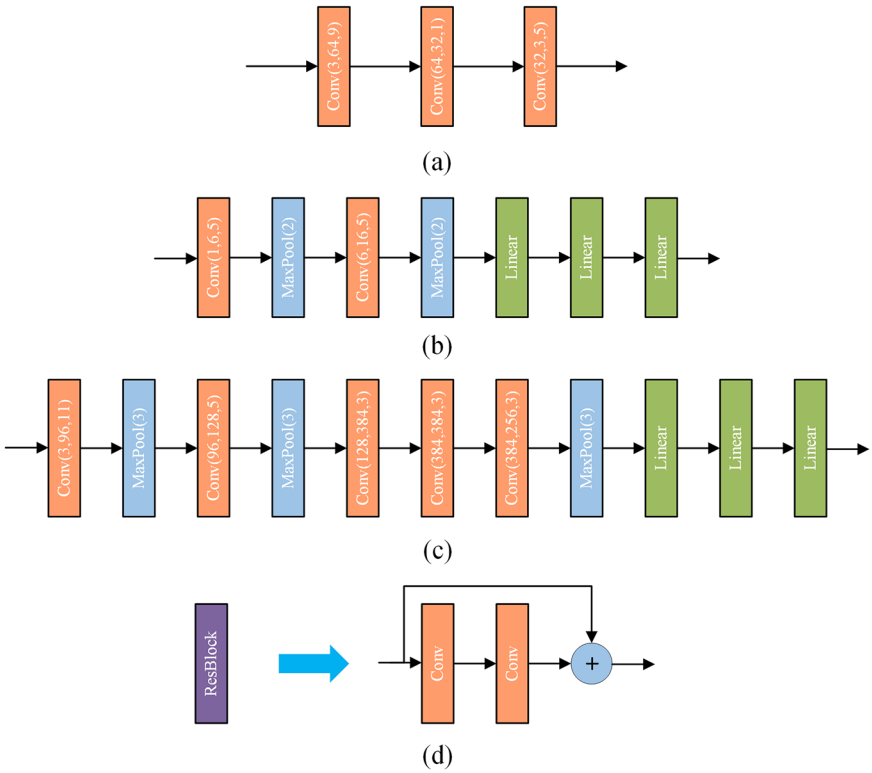


FIG. 9. Four classical network architectures: (a) SRCNN; (b) LeNet; (c) AlexNet; (d) ResBlock.

form as those required by traditional hand–eye calibration methods. In order to improve the accuracy of hand–eye calibration for eye-in-hand scenarios, we propose a deep learning-based method, and a complete flow chart is shown in Fig. 6.

Specifically, the entire process is divided into a training phase and a practical using phase:

Training phase:

(1) Acquire and pre-process data.
(2) Design a suitable convolutional neural network (often requires a lot of experimentation).
(3) Iterative optimization to complete the training.
(4) Test and validate to obtain the trained network model.

Practical using phase:

(1) Access to visual work range.
(2) Visual measurement to obtain the variation of the target object pose in the camera coordinate system.
(3) Based on the trained network model, input the variation and output the variation of the robot end pose.
(4) Determine whether the variation is small enough; if yes, then finish adjusting the pose.
(5) If not, adjust the pose and repeat step 2.

III. PROPOSED CONVOLUTIONAL NEURAL NETWORK

A. Training data acquisition and processing

The mechanical parameters of the parallel robot used in this study are presented in Table I. To take into account the range of motion of the parallel robot and the depth of field of the industrial camera, the motion space of the parallel robot during the training data acquisition is illustrated in Fig. 7.

We use the Latin Hypercube Sampling (LHS) method[32] for sampling planning to ensure that the sampling points can reflect the state of the entire workspace of the robot as much as possible.[2] As shown in Fig. 7, a rectangular, parallel, hexahedral region with dimensions of 50 × 30 × 30 mm^3 and attitude angles of 10°, 4°, and 3° are selected as the adopted space. According to the LHS method, 25 position coordinates and 40 attitude angles are randomly generated in the sampling space, and then they are arranged and combined, to obtain the poses of 1000 sampling points. We send the sampling point poses obtained by the above sampling method to the control system of the parallel robot, control the robot to move to

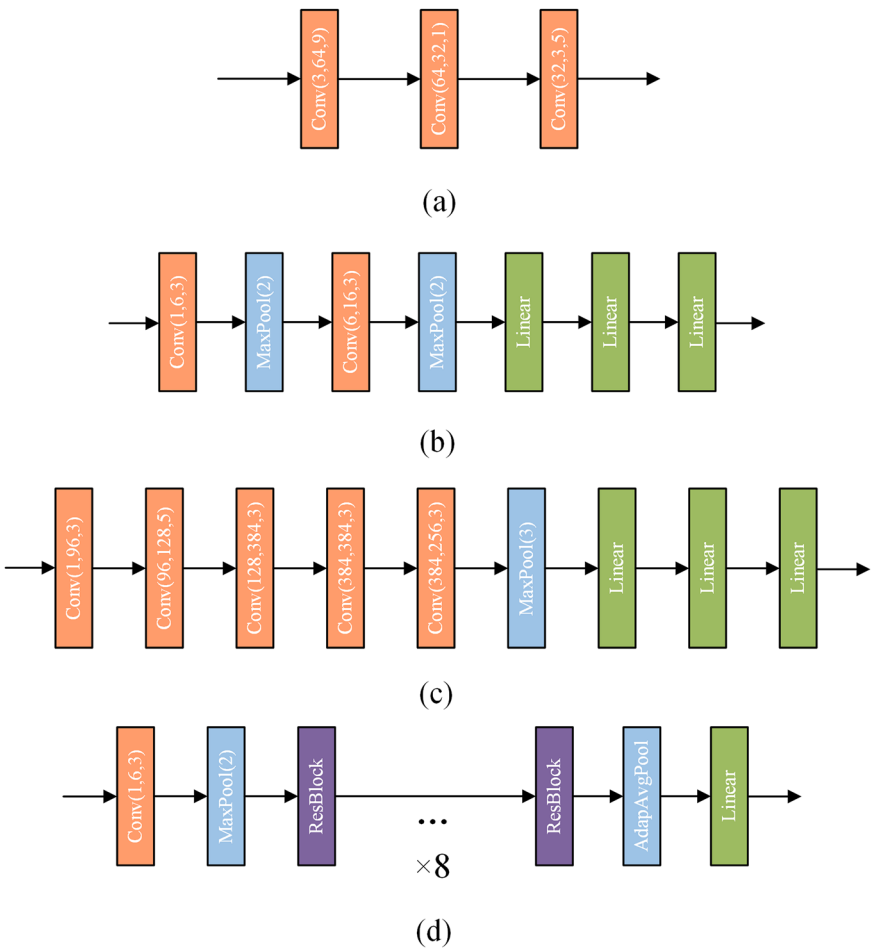


FIG. 10. Four new network architectures for pose estimation: (a) SRCNN*; (b) LeNet*; (c) AlexNet*; (d) ResBlock*.

the commanded pose sequentially, and then collect the corresponding 1000 photos of the calibration plate with an industrial camera. The training data pre-processing flow is shown in Fig. 8.

The specific process of training data pre-processing is as follows:

(1) Pre-calibrate the camera by Zhang's method[33] to get the internal parameters of the camera.
(2) Combine the camera's internal parameters with 1000 calibration plate photos to calculate the pose of 1000 corresponding calibration plates in the industrial camera coordinate system.
(3) Pair the above 1000 poses with the 1000 corresponding robot end poses to get 1000 pairs of pose data.
(4) Divide the training dataset and the test dataset by a certain ratio. In this study, a ratio of 9:1 was chosen, i.e., 900 pairs of pose data were used for training and 100 pairs of pose data were used for testing.
(5) The order of the 900 pairs of training data is disordered, and the difference between the front and back is made to form the pose variation data for deep learning network training.

TABLE II. Experimental parameters set for the network layer control experiments.

Network	Number of convolutional layers
Original network	2
Control group 1	4
Control group 2	6
Control group 3	10

B. Basic architecture of the convolutional neural network

The pose estimation convolutional neural network (PECNN) proposed in this study is inspired by four classical network architectures, SRCNN (Super Resolution Convolutional Neural Network),[34] LeNet,[35] AlexNet,[36] and ResNet (Residual Network),[37] which are shown in Fig. 9, where, due to the ResNet having too many

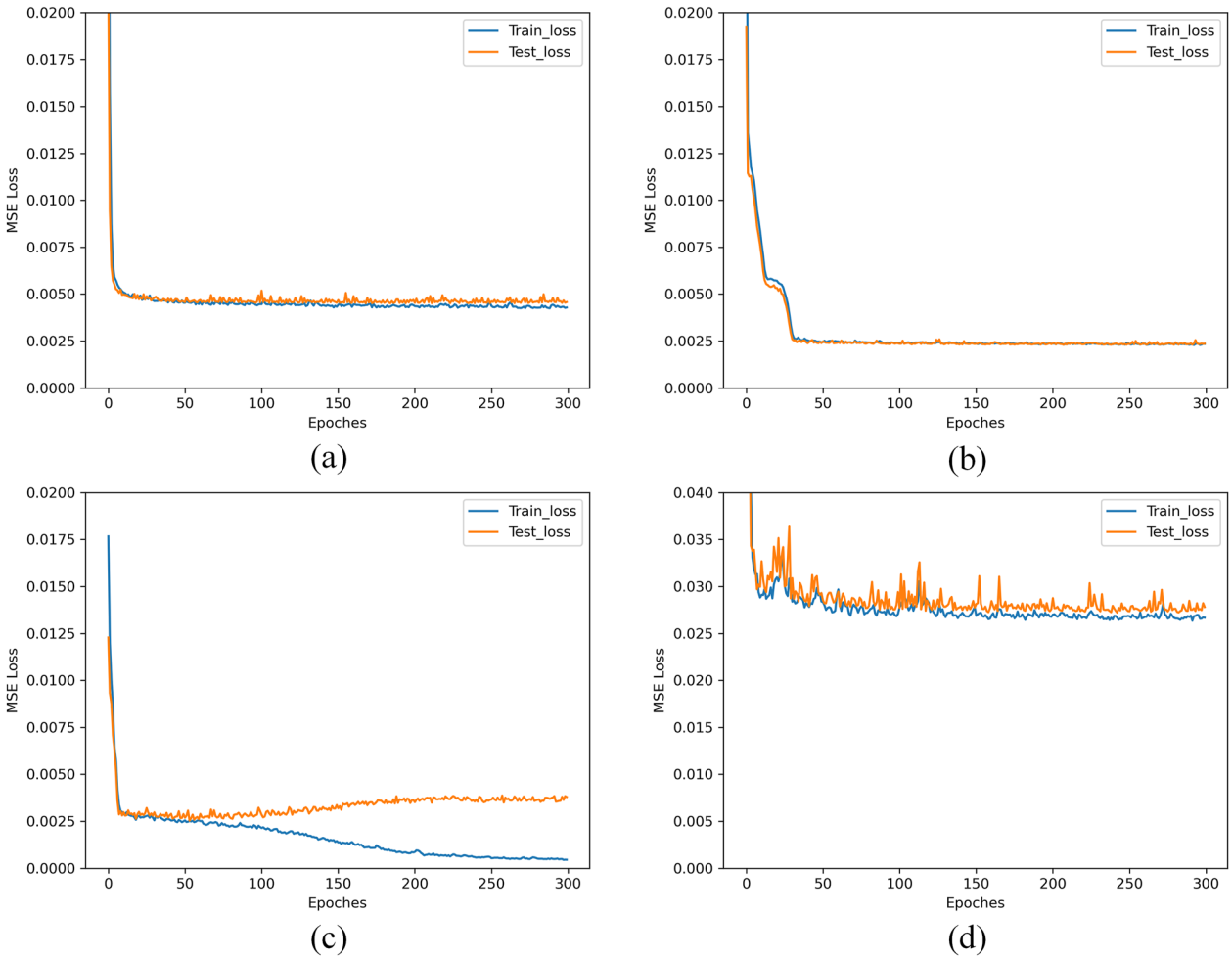


FIG. 11. Loss descent curves of four new network architectures: (a) SRCNN*; (b) LeNet*; (c) AlexNet*; (d) ResBlock*.

layers, the core component of ResNet, ResBlock (Residual Block), is substituted in the figure.

In Fig. 9, $Conv(i,o,k)$ denotes the convolutional layer, i denotes the number of channels in the input of the convolutional layer, o denotes the number of channels in the output of the convolutional layer, and k denotes the kernel size. *Linear* denotes the linear layer. $MaxPool(k)$ denotes the maximum pooling layer. Among these four network architectures, SRCNN is the pioneer of deep learning super-resolution, while LeNet, AlexNet, and ResNet are all used for the image classification recognition tasks (these three network architectures recognize slightly different image objects). In order to apply these four classical network architectures to the pose mapping estimation task in this paper, these network structures need to be adapted to the inputs and outputs of the task, to obtain four new network architectures, as shown in Fig. 10.

In Fig. 10, *AdapAvgPool* denotes the adaptive averaging pooling layer. The four networks are trained on the acquired pose data using an improved MSE loss as the loss function, the epoch was set to 300, and the loss descent curves during training are shown in Fig. 11.

TABLE III. Experiment parameters set for the pooling layer control experiments.

Network	Number of pooling layers
Original network	2, preserve the complete pooling layers
Control group 1	1, preserve the pooling layer before the linear layer
Control group 2	0

Analyzing the results in Fig. 11, we can observe that the LeNet* architecture has the minimum training loss, while ResNet* has the maximum training loss. Although AlexNet* has results that are second only to LeNet*, it suffers from a more severe overfitting phenomenon during the training process. Additionally, due to the deeper network layers of AlexNet*, the time required for training is

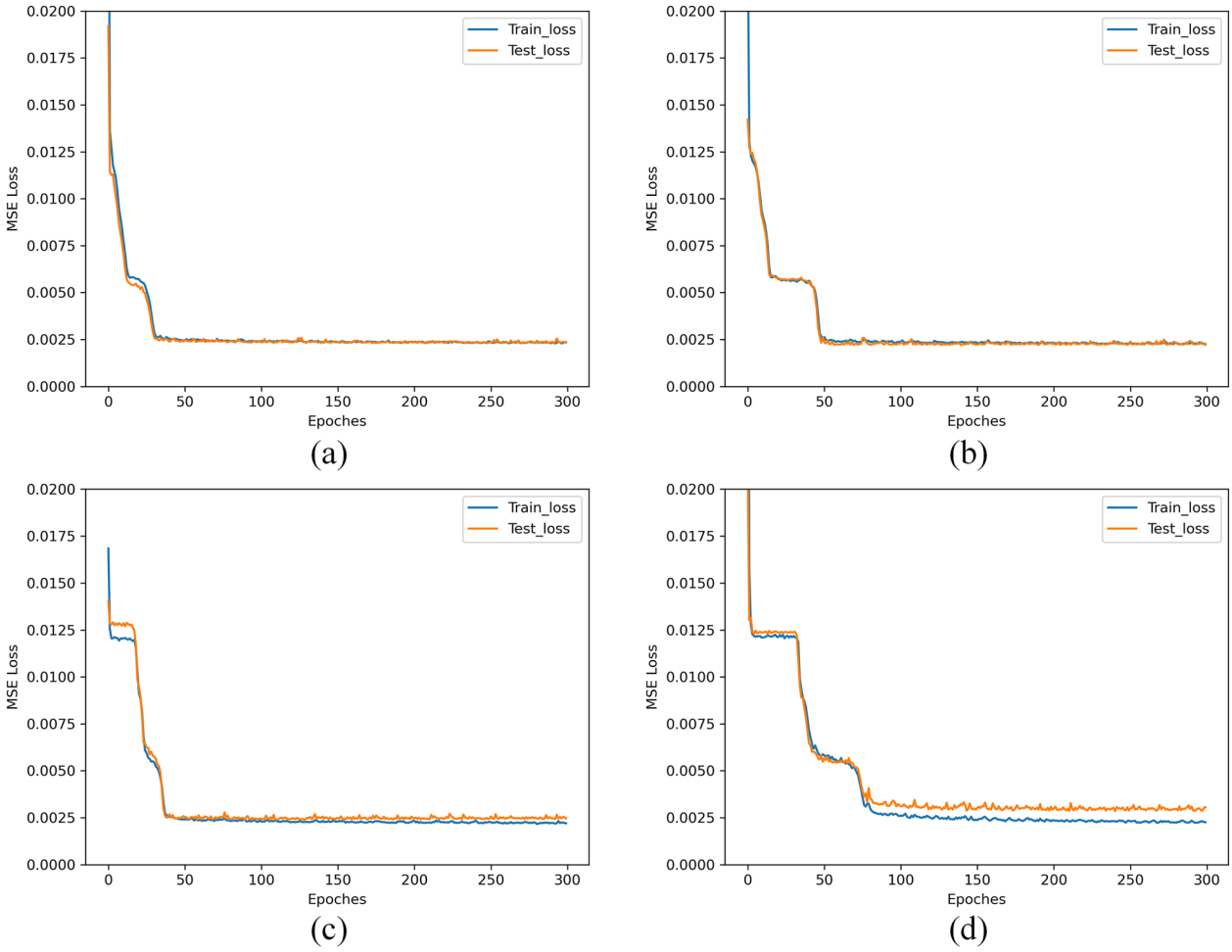


FIG. 12. Loss descent curves in network layer control experiments: (a) Original network; (b) control group 1; (c) control group 2; (d) control group 3.

tens of times longer than the remaining three architectures. Based on these results, the LeNet* architecture is chosen as the basis for the proposed PECNN architecture in this paper.

C. Network optimal architecture discussion

The following training experiments are used to analyze and discuss the optimal architecture of the PECNN network. First, the number of convolutional layers of the network is increased, and the experimental parameters set for the network layer control experiments are shown in Table II.

The loss descent curves of these four architectures with different numbers of network layers during the training process are shown in Fig. 12.

By analyzing the results in Fig. 12, it can be observed that there is no significant improvement in the performance of the network when the number of convolutional layers in the network is increased from 2 to 4. However, when the number of convolutional layers in the network is further increased to 6, a slight overfitting phenomenon can be observed. Finally, when the number of convolutional layers is increased directly to 10, a significant overfitting phenomenon occurs. Based on these results, the number of convolutional layers in the network is determined to be 2 for the proposed PECNN architecture.

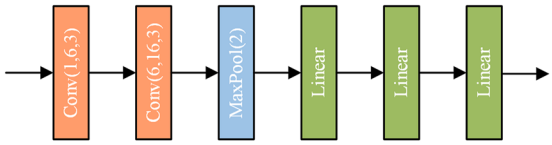


FIG. 14. Optimal architecture of PECNN.

Next, the analysis of the need for pooling layers in the original network architecture is continued with control experiments. The experiment parameters for the pooling layer control experiments are set, as shown in Table III.

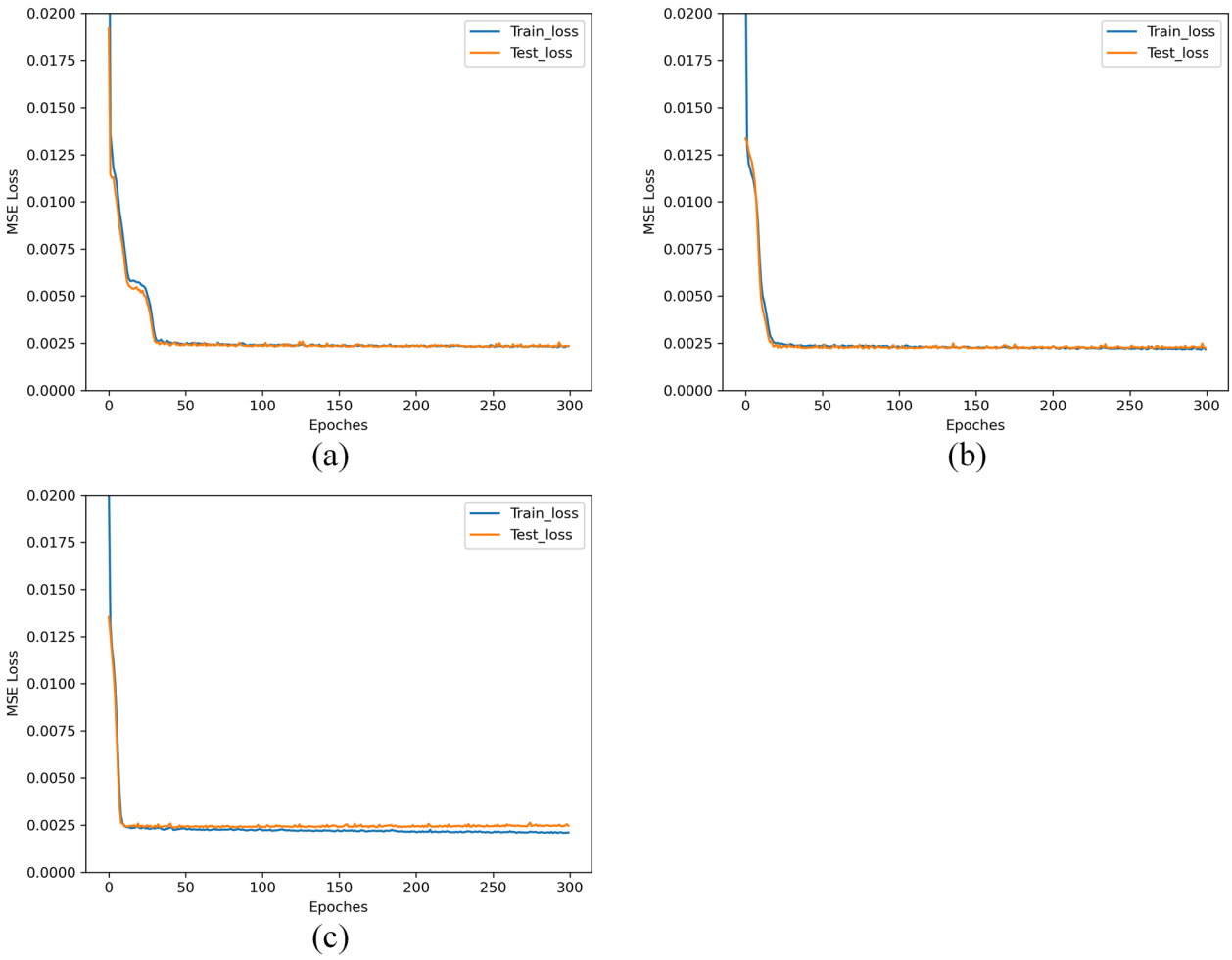


FIG. 13. Loss descent curves in pooling layer control experiments: (a) Original network; (b) control group 1; (c) control group 2.

TABLE IV. Experimental parameters set for the convolutional layer parameter control experiments.

Network	Layer 1 parameters	Layer 2 parameters
Original network	Number of input channels: 1 Number of output channels: 6	Number of input channels: 6 Number of output channels: 16
Control group 1	Number of input channels: 1 Number of output channels: 16	Number of input channels: 16 Number of output channels: 32
Control group 2	Number of input channels: 1 Number of output channels: 64	Number of input channels: 64 Number of output channels: 128
Control group 3	Number of input channels: 1 Number of output channels: 128	Number of input channels: 128 Number of output channels: 512

The loss descent curves of these three architectures with different numbers of pooling layers during the training process are shown in Fig. 13.

By analyzing the results in Fig. 13, it can be observed that removing the pooling layer at the front end of the network structure and retaining the pooling layer in front of the linear layer does not cause a significant change in the loss of final convergence of the network, but it converges better and earlier. In contrast, when all pooling layers in the network structure are removed, the network converges faster overall, but overfitting occurs. Based on these

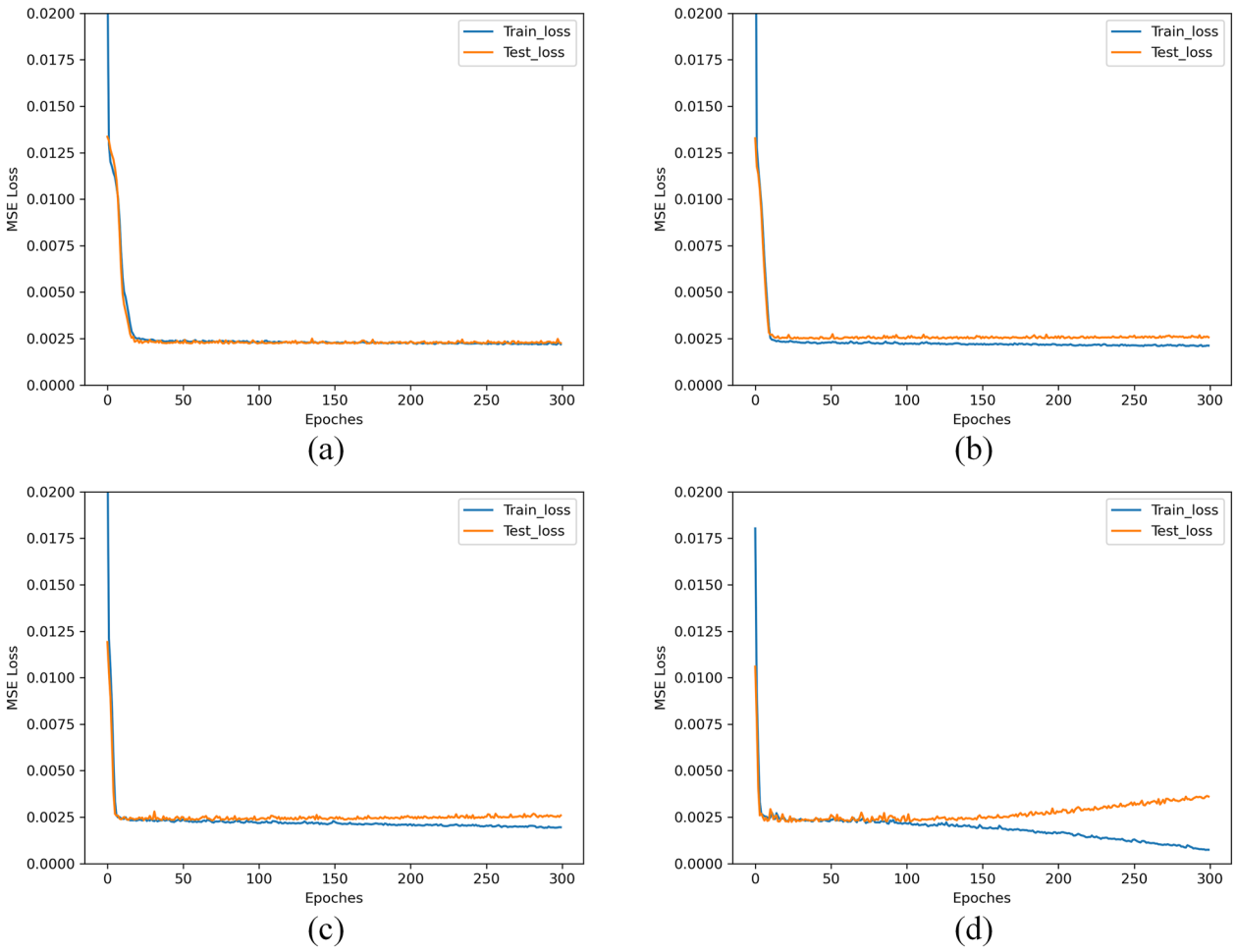


FIG. 15. Loss descent curves in convolutional layer parameter control experiments: (a) Original network; (b) control group 1; (c) control group 2; (d) control group 3.

TABLE V. Experimental parameters set for the learning rate control experiments.

Network	Learning rate
Original network	0.001
Control group 1	0.005
Control group 2	0.0005
Control group 3	0.0001

TABLE VI. Experimental parameters set for the epoch control experiments.

Network	Epoch
Original network	300
Control group 1	600
Control group 2	100

results, the final architecture of PECNN is determined to be the one with the pooling layer at the end of the network and without the pooling layer at the front end, as shown in Fig. 14.

D. Network optimal hyperparameters discussion

Next, we conduct further experiments to analyze and discuss the optimal hyperparameters of PECNN. First, we conduct control experiments of the convolutional layer parameters. Since the input data are in the form of 1×6, we did not modify the convolutional kernel size of the convolutional layers, and the experimental parameters set are shown in Table IV.

The loss descent curves of these four architectures with different convolutional layer parameters during the training process are shown in Fig. 15.

By analyzing the results presented in Fig. 15, it can be observed that increasing the number of input channels and output channels in

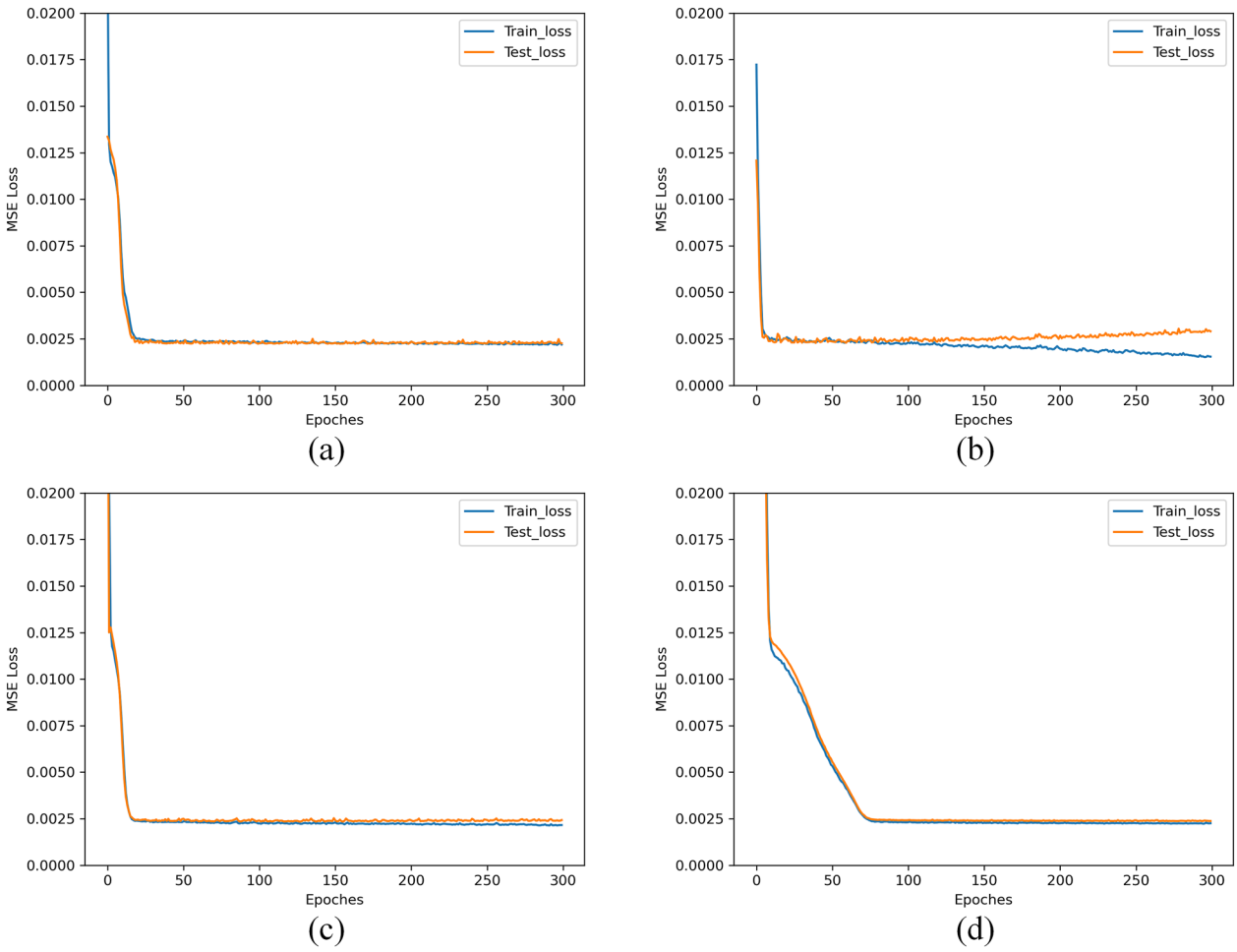


FIG. 16. Loss descent curves in learning rate control experiments: (a) Original network; (b) control group 1; (c) control group 2; (d) control group 3.

the convolutional layers results in overfitting, which becomes more pronounced as the number of channels increases. Therefore, based on these results, the decision is made not to increase the number of channels in the convolutional layers.

The following are learning rate control experiments with the experiment parameters set, as shown in Table V.

The loss descent curves of these four architectures with different learning rates during the training process are shown in Fig. 16.

After analyzing the results in Fig. 16, it can be observed that when the learning rate is increased to 0.005, the network fails to converge, while reducing the learning rate to 0.0005 shows a slight overfitting phenomenon. Based on these results, a learning rate of 0.001 is chosen, as it shows the best performance, without overfitting or underfitting.

Finally, epoch control experiments are performed, and the experimental parameters set are shown in Table VI.

The loss descent curves of these three architectures with different epochs during the training process are shown in Fig. 16.

TABLE VII. Optimal hyperparameters of PECNN.

Hyperparameter	Set
Convolutional layer 1	Number of input channels: 1 Number of output channels: 6
Convolutional layer 2	Number of input channels: 6 Number of output channels: 16
Learning rate	0.001
Epoch	100

After analyzing the results shown in Fig. 17, it can be observed that increasing the number of epochs to 600 leads to an obvious overfitting phenomenon, while reducing the number of epochs to 100 results in a better fitted loss. Therefore, in this paper, the final epoch is set to 100.

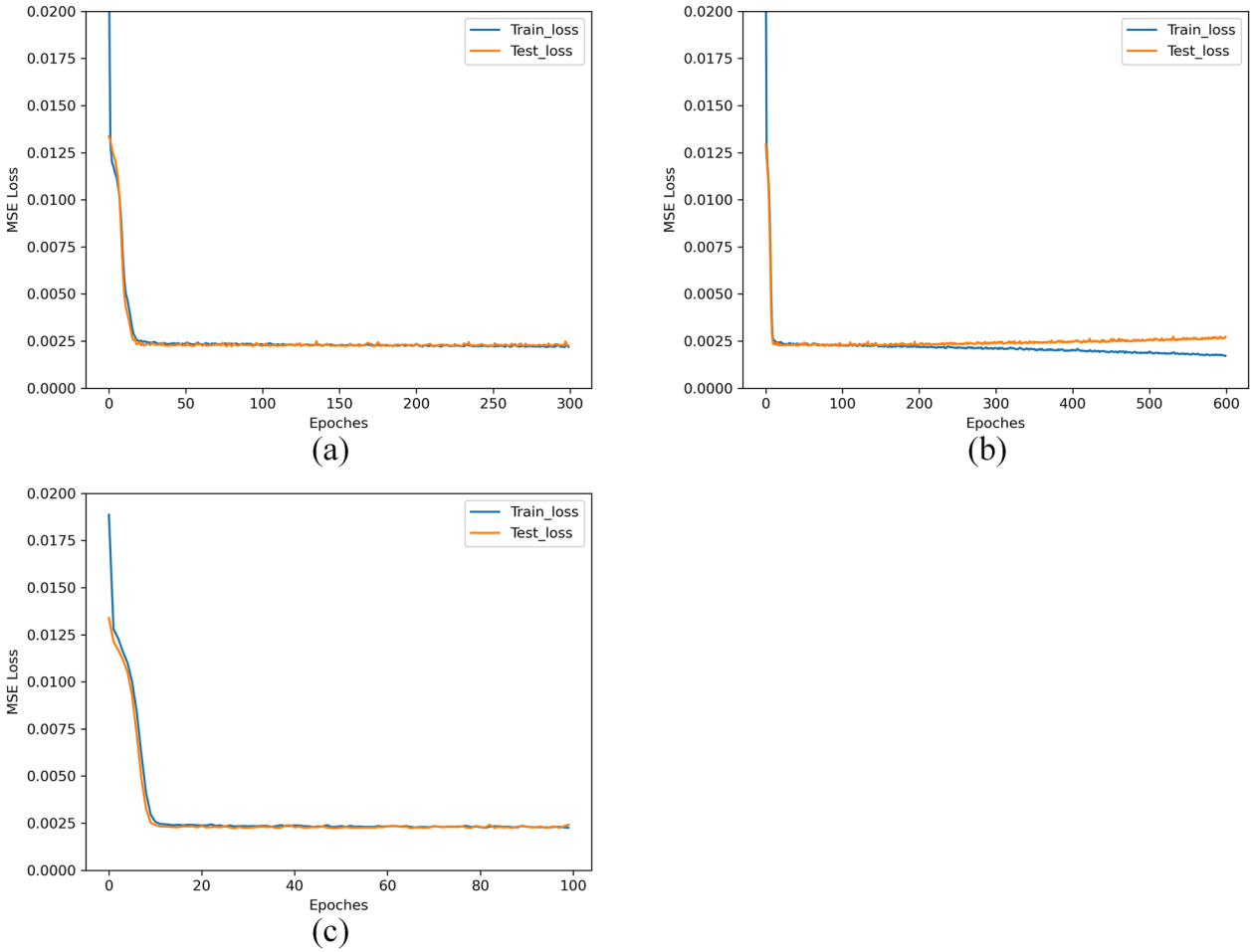


FIG. 17. Loss descent curves in epoch control experiments: (a) Original network; (b) control group 1; (c) control group 2.

After conducting the experiments discussed earlier, the optimal hyperparameters for the PECNN network have been determined and are presented in Table VII. To train the PECNN network, the ADAM optimizer is used, and the network weight parameters are initialized using the Kaiming method. Finally, with the hyperparameters set as in Table VII, the PECNN network is trained, to obtain the final trained model.

IV. EXPERIMENT

A. Hand-eye calibration accuracy experiment

To investigate the generalization capability of PECNN, we obtain 100 new sets of data for hand–eye calibration accuracy experiments using the method described in Sec. III A. In these experiments, we compare the performance of the trained PECNN model with Tsai's hand–eye calibration method,[15] a laser tracker-based direct measurement method, Hua's network,[29] and Wang's network.[30]

The hand–eye relationship matrix is obtained directly by the direct measurement method and is calculated by Tsai's method using data from the training dataset. Hua's network and Wang's network use the same principle as the PECNN proposed in this paper. Wang's network has the same input and output as PECNN, both being 6-D pose data, while Hua's network has 3-D input and output data. We fine-tune its network structure so that its input and output data structures are the same as those of PECNN, and, finally, obtain the implicit hand–eye relationship matrix through training.

All 100 new sets of data are used to verify the accuracy of the above hand–eye calibration methods, and Eq. (10) is used as the error expression to judge the accuracy.

TABLE VIII. Average and standard deviation of hand–eye calibration errors.

No.	Method	Average (mm)	Standard deviation (mm)
1	Tsai	0.6296	0.1623
2	Direct	0.0682	0.0098
3	Hua's network	0.1146	0.0203
4	Wang's network	0.0327	0.0064
5	Proposed PECNN	0.0083	0.0009

$$Error = \sqrt{(X-x)^2 + (Y-y)^2 + (Z-z)^2}. \tag{10}$$

X, Y, and Z represent the true values, i.e., the labels in the test dataset, and x, y, and z represent the calculated values, which are obtained by the hand–eye relationship that was obtained. The error results are shown in Fig. 18, Table VIII, and Fig. 19.

After analyzing the results, it can be concluded that the overall hand–eye calibration accuracy achieved by PECNN is better than 0.01 mm, which is significantly better than the other hand–eye calibration methods. The depth confidence network proposed by Wang achieves an accuracy close to 0.03 mm, which is the second-best after PECNN. The network proposed by Hua has a slightly lower accuracy than the direct measurement method, and the traditional hand–eye calibration method has the lowest accuracy.

B. Practical application

An automatic assembly system based on monocular visual measurement and a parallel robot is successfully implemented and

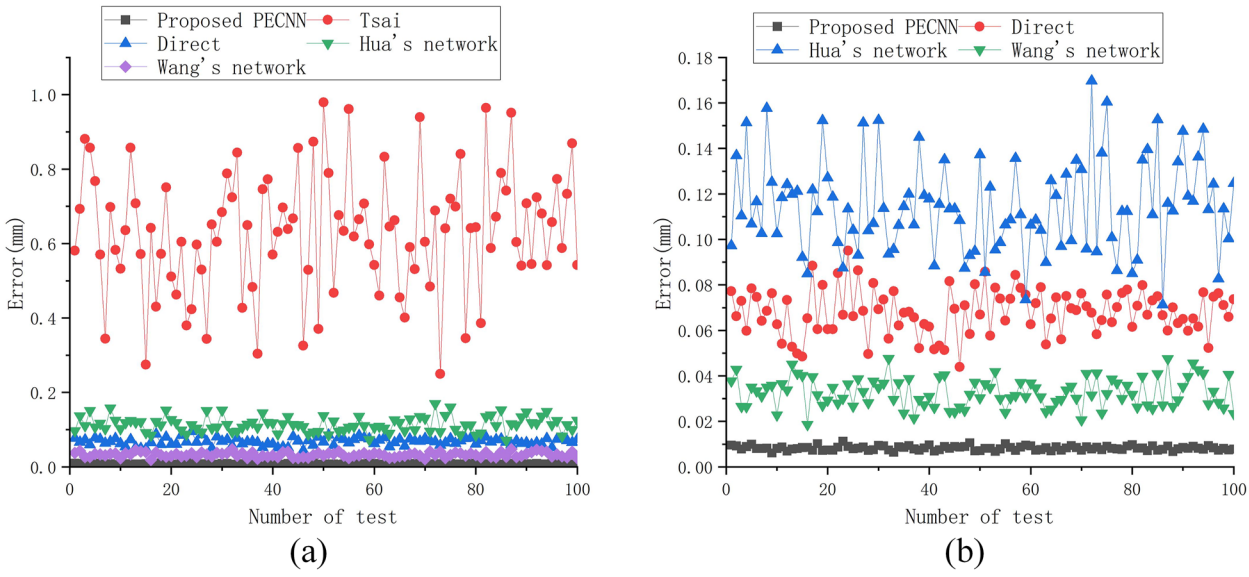


FIG. 18. Results of the hand–eye calibration accuracy experiment: (a) Results of five methods; (b) results of four methods without Tsai.

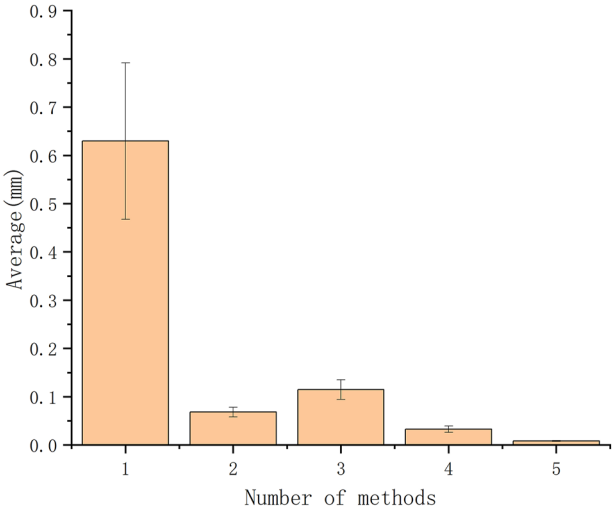


FIG. 19. Histograms of hand–eye calibration errors.

applied to the assembly of a large helicopter lift system, as shown in Fig. 20. The hand–eye calibration method proposed in this study is successfully applied to this system, and the hole–axis assembly is achieved with the inter-axis clearance less than or equal to 0.05 mm.

A highly accurate hand–eye calibration method is not only essential for reducing the number of visual measurement iterations during assembly but also a key factor for successful assembly. If the hand–eye calibration method is not accurate enough, it can cause the end of the parallel robot to hover around the assembly pose, but never truly converge, resulting in unsuccessful assembly.

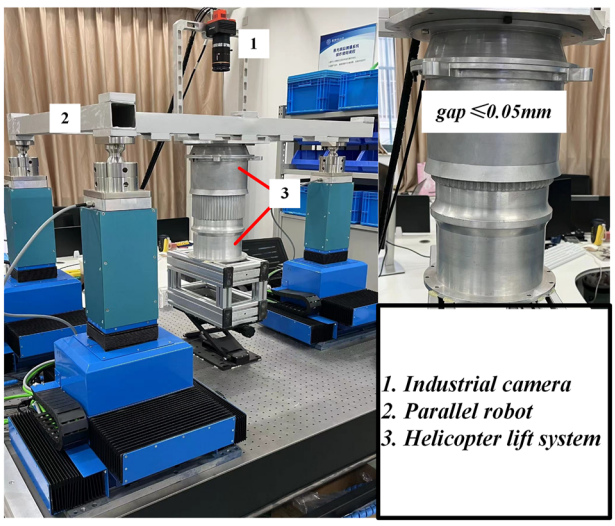


FIG. 20. Automatic assembly system.

C. Method applicability experiment

In order to evaluate the applicability of the proposed hand–eye calibration method on tandem robots or parallel robots of different configurations, the complete method proposed in this paper, which includes not only the PECNN network structure, but also the data acquisition and preprocessing, network training, and practical use processes, is applied to a KUKA tandem robot, as shown in Fig. 21. The data acquisition and preprocessing, network training to obtain the model, and accuracy verification processes are repeated for this specific robot configuration.

As shown in Fig. 21, a rectangular, parallel, hexahedral region with dimensions of 100 × 100 × 30 mm^3 and attitude angles of

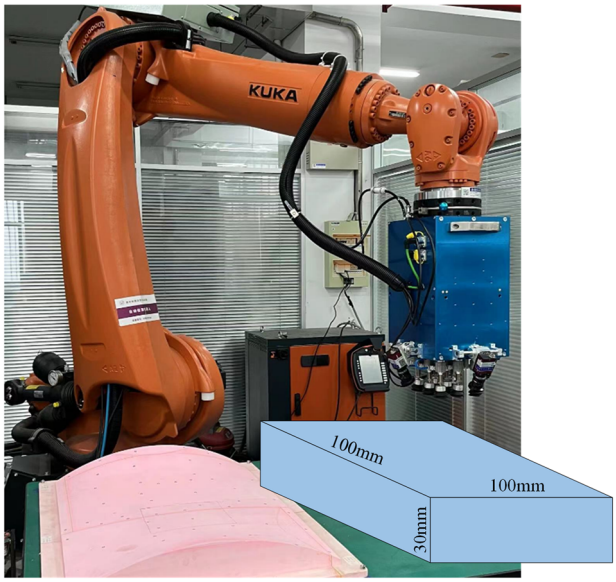


FIG. 21. Motion space of the KUKA tandem robot.

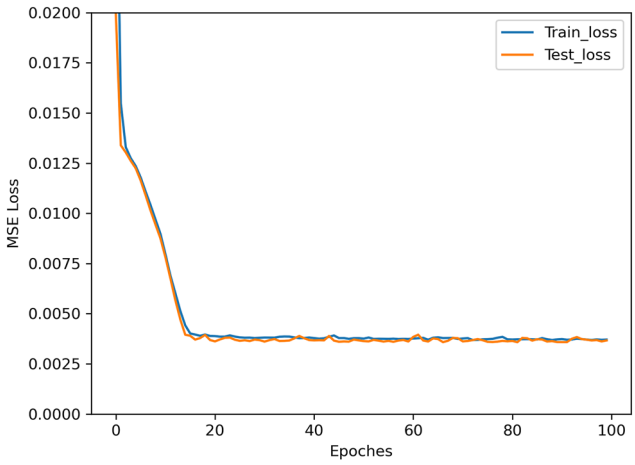


FIG. 22. Loss descent curves in applicability experiment.

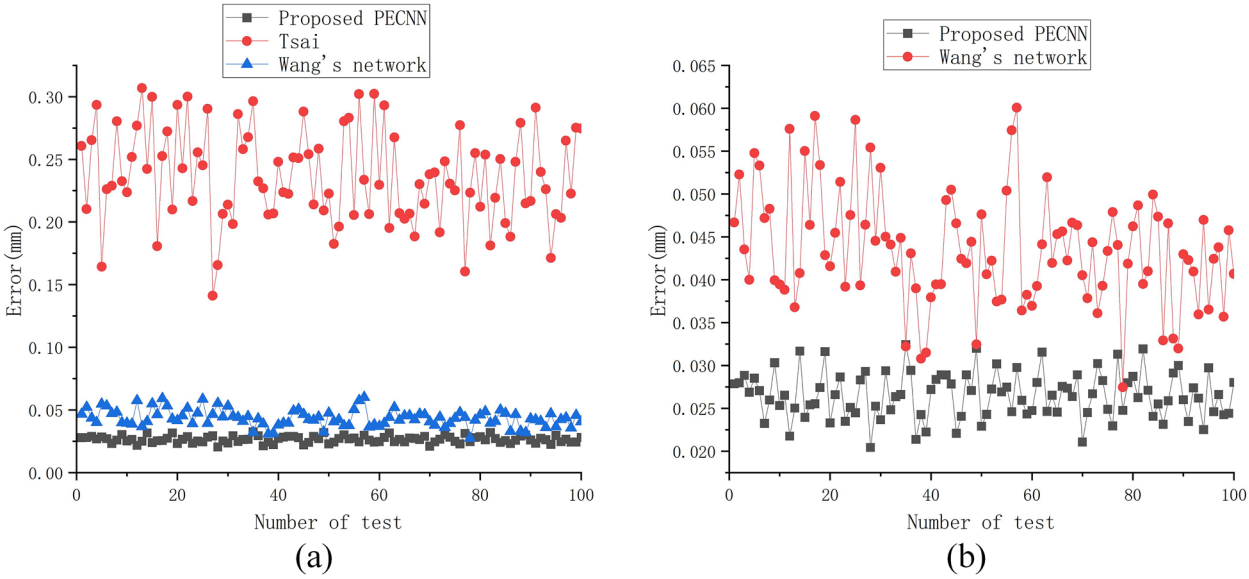


FIG. 23. Results of hand–eye calibration accuracy in applicability experiment: (a) Results of three methods; (b) results of two methods without Tsai.

10°, 10°, and 10° are selected as the adopted space. First, 1000 sets of data are acquired for training and testing of PECNN, and then 100 sets of data are acquired for hand–eye calibration accuracy verification.

The loss descent curves of PECNN in applicability experiment during the training process are shown in Fig. 22.

The error results in applicability experiment are shown in Fig. 23, Table IX, and Fig. 24.

After analyzing the results, it can be concluded that the KUKA tandem robot has lower positioning accuracy compared to the parallel robot, but a larger range of motion, especially for angles. Since the KUKA tandem robot has a larger range of motion, the traditional hand–eye calibration method represented by Tsai's method achieves better accuracy. This is consistent with the conclusions obtained by Shiu and Ahmad[14] and Tsai and Lenz[15] in the error analysis phase. The hand–eye calibration accuracy of both PECNN and Wang's network has degraded to some extent, which can be attributed to two reasons. First, the positioning accuracy of the tandem robot is low, which leads to the training data being noisier, reducing the training quality of the networks, as can be seen in Fig. 21. Second, the tandem robot has a larger range of motion, which requires more data for training, in order to make the network model fully describe its state in the range of motion and further improve the model accuracy. The 1000 sets of data collected in the applicability experiment are relatively small. Based on these results, it is reasonable to deduce that the proposed method can be applied to most parallel robots and tandem robots.

TABLE IX. Average and standard deviation of errors in applicability experiment.

No.	Method	Average (mm)	Standard deviation (mm)
1	Tsai-parallel robot	0.6296	0.1623
2	Tsai-tandem robot	0.2362	0.0371
3	Wang's network-parallel robot	0.0327	0.0064
4	Wang's network-tandem robot	0.0436	0.0066
5	Proposed PECNN-parallel robot	0.0083	0.0009
6	Proposed PECNN-tandem robot	0.0265	0.0027

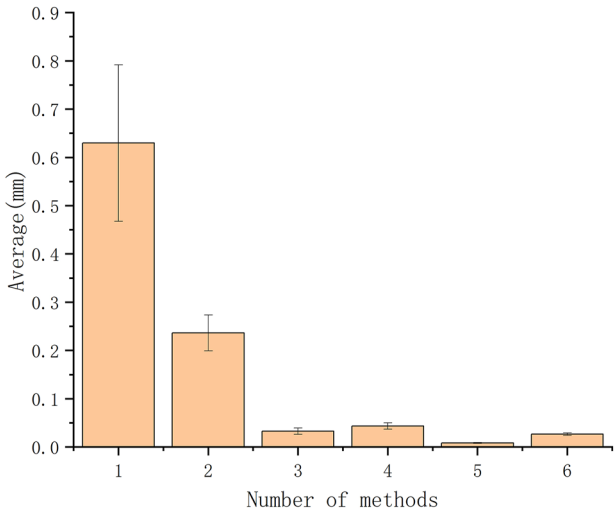


FIG. 24. Histograms of errors in applicability experiment.

V. CONCLUSION

In this study, we propose a deep learning-based method for hand–eye calibration of parallel robots, which solves the problem of the traditional hand–eye calibration methods being inapplicable to parallel robots. We start by collecting and preprocessing the dataset for network training, followed by selecting and experimenting with four classical convolutional neural network architectures, to determine the most suitable basic architecture for the hand–eye calibration task. We then discuss the optimal architecture and hyperparameters of the network, and finally determine the architecture and hyperparameters of PECNN. Through hand–eye calibration accuracy experiments and practical applications, we demonstrate that the proposed method achieves higher accuracy, surpassing the traditional hand–eye calibration method by several orders of magnitude, and can be applied to actual assembly. Additionally, through applicability experiments, we verify that the proposed method also performs well on tandem robots, leading us to conclude that it can be applied to most parallel robots and tandem robots.

Although PECNN has been demonstrated to be effective, there is still potential for further improvement. During our research, we have found that collecting more structured data during the data collection phase may enable semi-supervised learning, which can reduce the amount of data required for training. Additionally, when applying this method to tandem robots, more data may need to be collected to improve the results. These points will be the focus of our next research. Moreover, we hope that our study can shed light on this problem and encourage more researchers to apply deep learning techniques for robot hand–eye calibration.

ACKNOWLEDGMENTS

This work was supported by the Defense Industrial Technology Development Program (Grant No. JCKY2020605C014) and Fundamental Research Funds for the Central Universities (Grant No. NP2022421).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Kuai Zhou: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Methodology (equal); Software (equal); Writing – original draft (equal). **Xiang Huang**: Funding acquisition (equal); Supervision (equal). **Shuanggao Li**: Project administration (equal). **Gen Li**: Project administration (equal).

DATA AVAILABILITY

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due [STATE RESTRICTIONS SUCH AS PRIVACY OR ETHICAL RESTRICTIONS].

REFERENCES

[1]G. Anzolin, A. Zanfei, and A. Andreoni, "Robot adoption and FDI driven transformation in the automotive industry," Int. J. Automot. Technol. Manage. **20**(2), 215–237 (2020).
[2]L. I. Bo, T. Wei, C. Zhang *et al.*, "Positioning error compensation of an industrial robot using neural networks and experimental study," Chin. J. Aeronaut. **35**(2), 346–360 (2022).
[3]K. Zhou, X. Huang, S. Li *et al.*, "6-D pose estimation method for large gear structure assembly using monocular vision," Measurement **183**, 109854 (2021).
[4]J. Wan, S. Tang, D. Li *et al.*, "Reconfigurable smart factory for drug packing in healthcare industry 4.0," IEEE Trans. Ind. Inf. **15**(1), 507–516 (2018).
[5]M. Park, T.-A. Le, A. Eizad, and J. Yoon, "A novel shared guidance scheme for intelligent haptic interaction based swarm control of magnetic nanoparticles in blood vessels," IEEE Access **8**, 106714–106725 (2020).
[6]M. Sehmi, J. Christensen, C. Bastien *et al.*, "Automated post-processing for sheet metal component manufacturing," Adv. Eng. Software **143**, 102794 (2020).
[7]R. Singh and H. K. Verma, "Sintering materials and automation of sintering process for manufacturing bi-metallic engine bearings," Mater. Today: Proc. **12**, 655–664 (2019).
[8]A. O. Fernandes, L. F. E. Moreira, and J. M. Mata, "Machine vision applications and development aspects," in *2011 9th IEEE International Conference on Control and Automation (ICCA)* (IEEE, 2011), pp. 1274–1278.
[9]E. Martinez-Martin and A. P. Del Pobil, "Robot vision for manipulation: A trip to real-world applications," IEEE Access **9**, 3471–3481 (2020).
[10]I. Enebuse, M. Foo, B. S. K. K. Ibrahim *et al.*, "A comparative review of hand-eye calibration techniques for vision guided robots," IEEE Access **9**, 113143–113155 (2021).
[11]S. Yin, Y. Ren, Y. Guo *et al.*, "Development and calibration of an integrated 3D scanning system for high-accuracy large-scale metrology," Measurement **54**, 65–76 (2014).
[12]Y. Tian, W. R. Hamel, and J. Tan, "Accurate human navigation using wearable monocular visual and inertial sensors," IEEE Trans. Instrum. Meas. **63**(1), 203–213 (2013).
[13]H. Liu, Y. Zhou, and Z. Gu, "Inertial measurement unit-camera calibration based on incomplete inertial sensor information," J. Zhejiang University Science C: Computer & Electronics **15**(11), 999–1008 (2014).
[14]Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB," IEEE Trans. Rob. Autom. **5**, 16 (1987).
[15]R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," IEEE Trans. Rob. Autom. **5**(3), 345–358 (1989).
[16]R. Horaud and F. Dornaika, "Hand-eye calibration," Int. J. Rob. Res. **14**(3), 195–210 (1995).
[17]F. C. Park and B. J. Martin, "Robot sensor calibration: Solving AX=XB on the Euclidean group," IEEE Trans. Rob. Autom. **10**(5), 717–721 (1994).
[18]J. C. K. Chou and M. Kamel, "Finding the position and orientation of a sensor on a robot manipulator using quaternions," Int. J. Rob. Res. **10**(3), 240–254 (1991).
[19]Y. C. Lu and J. C. K. Chou, "Eight-space quaternion approach for robotic hand-eye calibration," in *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century* (IEEE, 1995), Vol. 4, pp. 3316–3321.
[20]K. Daniilidis, "Hand-eye calibration using dual quaternions," Int. J. Rob. Res. **18**(3), 286–298 (1999).
[21]N. Andreff, R. Horaud, and B. Espiau, "Robot hand-eye calibration using structure-from-motion," Int. J. Rob. Res. **20**(3), 228–248 (2001).
[22]D. Condurache and A. Burlacu, "Orthogonal dual tensor method for solving the AX = XB sensor calibration problem," Mech. Mach. Theory **104**, 382–404 (2016).
[23]S. Gwak, J. Kim, and F. C. Park, "Numerical optimization on the Euclidean group with applications to camera calibration," IEEE Trans. Rob. Autom. **19**(1), 65–74 (2003).
[24]J. Heller, D. Henrion, and T. Pajdla, "Hand-eye and robot-world calibration by global polynomial optimization," in *2014 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2014), pp. 3157–3164.
[25]Z. Zhao, "Simultaneous robot-world and hand-eye calibration by the alternative linear programming," Pattern Recognit. Lett. **127**, 174–180 (2019).

[26]Z.-Q. Zhang, "Cameras and inertial/magnetic sensor units alignment calibration," IEEE Trans. Instrum. Meas. **65**(6), 1495–1502 (2016).
[27]Z. Zhang, L. Zhang, and G.-Z. Yang, "A computationally efficient method for hand–eye calibration," Int. J. Comput. Assisted Radiol. Surg. **12**, 1775–1787 (2017).
[28]W. Chu, G. Li, S. Li, and X. Huang, "Posture adjustment method of large aircraft components based on multiple numerical control positioners," Int. J. Adv. Manuf. Technol. **126**, 2159 (2023).
[29]J. Hua and L. Zeng, "Hand–eye calibration algorithm based on an optimized neural network," Actuators **10**(4), 85 (2021).
[30]W. Wang, W. Tian, W. Liao, and B. Li, "Pose accuracy compensation of mobile industry robot with binocular vision measurement and deep belief network," Optik **238**, 166716 (2021).
[31]S. Kong, K. Zhou, and X. Huang, "Online measurement method for assembly pose of gear structure based on monocular vision," Meas. Sci. Technol. **34**(6), 065110 (2023).
[32]J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems," Reliab. Eng. Syst. Saf. **81**(1), 23–69 (2003).
[33]Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Proceedings of the 7th IEEE International Conference on Computer Vision* (IEEE, 1999), Vol. 1, pp. 666–673.
[34]C. Dong, C. C. Loy, K. He *et al.*, "Image super-resolution using deep convolutional networks," IEEE Trans. Pattern Anal. Mach. Intell. **38**(2), 295–307 (2015).
[35]Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE **86**(11), 2278–2324 (1998).
[36]A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM **60**(6), 84–90 (2017).
[37]K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 770–778.