## A novel framework inspired by human behavior for peg-in-hole assembly

# A novel framework inspired by human behavior for peg-in-hole assembly

*Peng Guo*
School of Automation Science and Engineering, South China University of Technology, Guangzhou, China
*Weiyong Si*
School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK, and
*Chenguang Yang*
Department of Computer Science, University of Liverpool, Liverpool, UK

**Abstract**

**Purpose** – The purpose of this paper is to enhance the performance of robots in peg-in-hole assembly tasks, enabling them to swiftly and robustly accomplish the task. It also focuses on the robot's ability to generalize across assemblies with different hole sizes.

**Design/methodology/approach** – Human behavior in peg-in-hole assembly serves as inspiration, where individuals visually locate the hole firstly and then continuously adjust the peg pose based on force/torque feedback during the insertion process. This paper proposes a novel framework that integrate visual servo and adjustment based on force/torque feedback, the authors use deep neural network (DNN) and image processing techniques to determine the pose of hole, then an incremental learning approach based on a broad learning system (BLS) is used to simulate human learning ability, the number of adjustments required for insertion process is continuously reduced.

**Findings** – The author conducted experiments on visual servo, adjustment based on force/torque feedback, and the proposed framework. Visual servo inferred the pixel position and orientation of the target hole in only about 0.12 s, and the robot achieved peg insertion with 1–3 adjustments based on force/torque feedback. The success rate for peg-in-hole assembly using the proposed framework was 100%. These results proved the effectiveness of the proposed framework.

**Originality/value** – This paper proposes a framework for peg-in-hole assembly that combines visual servo and adjustment based on force/torque feedback. The assembly tasks are accomplished using DNN, image processing and BLS. To the best of the authors' knowledge, no similar methods were found in other people's work. Therefore, the authors believe that this work is original.

**Keywords** Peg-in-hole assembly, Deep neural network, Broad learning system, Robot manipulation

**Paper type** Research paper

## 1. Introduction

Peg-in-hole assembly is a challenging task in the dexterous manipulation of robots, primarily applied in industrial assembly(Huang *et al.*, 2013; Fang *et al.*, 2016; Song *et al.*, 2021). The main challenge lies in its extremely low error tolerance, particularly in transitional or interference fits, where minor differences in position or orientation can lead to assembly failure. Traditional visual servos used in robot manipulations are often inadequate for the requirements of peg-in-hole assembly tasks, as they are susceptible to errors influenced by device performance and noise. Numerous researchers have focused on this area, and typically conducted research from two aspects: visual servo localization (Chang *et al.*, 2011; Liu *et al.*, 2015; Chang and Wu, 2017) and adjustment based on force/torque feedback (Chen and Liu, 2013; Song *et al.*, 2016). However, their research on visual servo often requires high-precision devices or camera equipment, and on adjustment based on force/torque feedback is less flexible, adjusting the orientation only in the hole and taking longer to complete the task.

Visual servo is the most commonly used method in peg-in-hole assembly, and researchers use various approaches to enhance the accuracy(Fei *et al.*, 2023). Traditional image processing methods typically includes edge detection and template matching to detect target positions (Zhang *et al.*, 2017; Gu *et al.*, 2018). Chang *et al.* (2011) designed a visual servo assembly system for microsystems using traditional image processing methods and position-based control, requiring precise visual guidance that is complex and expensive. The time to complete an assembly was approximately 4 min. Liu *et al.* (2015) proposed an efficient relative pose estimation method based on multi-microvision for long cylindrical components assembly, using an estimation approach from coarse to fine, more than ten adjustments in position and orientation respectively are required to complete the task. Chang and Wu (2017) designed a closed-loop visual servo controller for USB insertion based on visual error, but the drawback is the lengthy time required, with each assembly taking 74 s. Ma *et al.* (2020)

introduced an image-based pose alignment system that simultaneously adjusts position and orientation, applied in the assembly of small parts, but here the assembly task is carried out in a single plane, and 12 adjustments are needed to complete a task in about 56 s. Two-dimensional images can only provide position information and adjust position or orientation within a single plane, while the use of 3D point cloud data enhances pose estimation efficiency. Li *et al.* (2021) proposed a coarse-to-fine hole pose estimation method based on 3D point clouds, capable of estimating poses for arbitrary orientations. In recent years, learning-based visual servo have rapidly advanced. The structure of DNN, characterized by multiple layers of interconnected neurons, allows them to learn hierarchical representations of data (Xie *et al.*, 2019; Zhao *et al.*, 2019; Xie *et al.*, 2022). The key advantage of learning-based visual servo is its ability to automatically learn complex visual representations and control policies from large amounts of training data. Haugaard *et al.* (2021) used DNN based on visual sensors to simultaneously estimate the positions of the peg and hole for guiding assembly. Puang *et al.* (2020) introduced a novel learning-based visual servo method called keypoint-based visual servoing framework (KOVIS), where one network learns keypoint representations from images, and another network learns robot motions based on these key points, but there are only positional differences between the peg holes.

In part of small-part precision assembly tasks, researchers could use high-precision devices or camera equipment to complete the task, but this would undoubtedly be costly (Fan *et al.*, 2023). For some large-size assembly tasks, adjustment based on force/torque feedback is essential. Chen and Liu (2013) proposed a robust impedance control algorithm for intelligent force control during PCB insertion. However, this method requires knowledge of the robot's dynamic model. Song *et al.* (2016) investigated the insertion assembly of irregular geometric shapes, focusing on learning operational forces during human assembly using a Gaussian mixture model, and the time to complete an assembly was about 20 s. The success rate of this method is relatively low, and stability is poor. Park *et al.* (2013) mimicked human actions to locate holes along a specific trajectory, using a combination of force/position control and passive compliant control during insertion, it takes about 20 s to complete an assembly, not taking into account orientation differences either. Jasim *et al.* (2014) matched peg-in-hole contact states with force information to guide the search for hole poses. Park *et al.* (2017) established a geometric model of contact states to guide pose adjustment alignment, using a force/position hybrid control during the insertion process. The experiments do not account for orientation differences, and it takes approximately 6.3 s to complete an assembly. Song *et al.* (2021) proposed an assembly strategy based on the passive alignment principle, simultaneously using Gaussian mixture models and regression learning of human assembly's compliant control skills. However, this method has a time-consuming teaching and learning process and limited generalization capability. Zou *et al.* (2020) designed a multilayer perceptron (MLP) network to search for peg positions and proposed a variable impedance controller based on fuzzy Q-learning, achieving smooth insertion through force feedback control. In our previous work (Dong *et al.*, 2023), we used BLS to transfer human assembly skills to robot, but the peg-in-hole assembly was only performed in the vertical direction. Spector and Di Castro (2021) used deep reinforcement learning to learn a residual impedance strategy, enabling trajectory correction for assembly completion based on force/torque feedback.

Current adjustment based on force/torque feedback studies are performed when the position of the hole is known. That is, this method is only used to softly insert the hole at the final stage, not to find the exact position of the hole. To establish a comprehensive assembly system, it is imperative to integrate both these techniques. Zhao *et al.* (2020) integrated visual servo and force/torque feedback to achieve peg-in-hole alignment. They used admittance control during the insertion process. Kang *et al.* (2022) used visual estimation to determine peg orientation and then located the peg position through trajectory search. Liu *et al.* (2014) designed an automated precision assembly system, using traditional image processing methods for error measurement to guide robot arm and platform alignment, ensuring assembly safety through force control, one assembly task can be completed in less than 2 min. Learning-based methods are equally applicable when combining these two techniques. Triyonoputro *et al.* (2019) trained a DNN based on images to predict hole positions, using iterative visual servo to move the robot arm to the target position. Impedance control was used during the insertion process. It takes 70 s to complete an assembly. Spector and Zacksenhouse (2021) and Spector *et al.* (2022) framed peg-in-hole assembly as a regression problem and proposed the InsertionNet, a fusion of visual and force information, to accomplish routine insertion tasks. Shen *et al.* (2023) introduced a DNN to estimate workpiece feature points for inferring peg poses, combining visual servo and force control for assembly realization.

Through the above literature research, we found the following shortcomings in the current research on peg-in-hole assembly:

- The use of visual servo alone is only feasible in the small-part precision assembly, but this requires multiple high-precision devices and camera equipment, which are high cost; low-cost visual servo is difficult to satisfy the large-size assembly tasks and must be combined with other technologies;
- The research related to adjustment based on force/torque feedback mainly focuses on controlling the insertion process smoothly, the research on how to find the hole position is relatively small and most of them are only regulated in a single plane, which is inconsistent with the actual assembly scenarios;
- Learning-based approaches are limited by factors such as data sets, training equipment, etc., and it is difficult to meet the need for rapid redeployment and lack of versatility (Wu *et al.*, 2023).

Humans' behavior in peg-in-hole assembly has inspired our approach, where individuals typically use visual cues to locate the hole and then adjust the peg pose based on force/torque feedback during the insertion process, achieving successful assembly after multiple adjustments. Also due to the learning ability of humans, when individuals assemble the same target

hole multiple times, the number of adjustments will continue to decrease.

Taking this inspiration, we propose an assembly framework, as illustrated in Figure 1. Initially, we use DNN and image processing to preliminarily determine the pose of the target hole. The DNN used is an object detection network based on YOLOv7, while image processing focuses on target hole fitting and normal vector extraction. During the peg insertion process, we use an incremental learning approach to train a BLS, inputting force/torque data at the point of contact between the peg and the hole and outputting the next action for the peg movement. The purpose of introducing incremental learning is to simulate this human learning ability. Compared with previous studies, our proposed framework and experimental scenario setup have the following features:

- We use monocular vision for initial pose estimation, and the vision devices used are all consumer grade and low cost.
- The visual servo workflow we designed is fast and efficient, requiring only 0.12 s to infer the pixel position and orientation of the target holes, and the training time of the object detection network is 0.95 h.
- We perform the peg pose adjustment based on force/torque feedback and introduce an incremental learning method, which ultimately requires only 1–3 adjustments to complete the assembly. This is the first time that a learning approach is introduced into the peg-in-hole insertion process.
- The pose adjustment based on force/torque feedback is with four degrees of freedom to adjust the position and orientation at the same time, this is different from the previous studies where only two degrees of freedom adjustment is usually performed.

The main contributions of this paper are:

- We proposed a novel peg-in-hole assembly framework inspired by human behavior, which combines the visual servo and adjustment based on force/torque feedback. Experimental results show that the framework is effective and stable.
- We introduced DNN-based object detection into assembly tasks, where the chosen DNN is able to strike a balance between GPU selection and training duration. Our proposed visual servo workflow is able to quickly obtain the target hole pixel position and orientation.
- We used the BLS for peg-in-hole insertion, starting the task from any pose and adjusting the position and orientation simultaneously. An incremental learning method is introduced for the training of BLS, and the trained BLS only needs 1–3 adjustments to complete the assembly and has the ability of generalization.

The organization of this paper is as follows: Section 2 introduces the visual servo based on DNN and image processing, Section 3 presents the adjustment based on force/torque feedback, Section 4 presents experiments to evaluate the effectiveness of the proposed framework, and Section 5 concludes the paper.

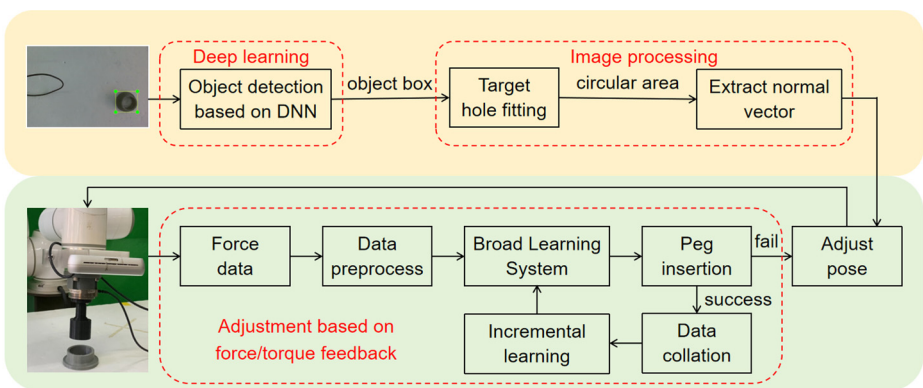
## 2. Visual servo based on deep neural network and image processing

Visual servo serves as the foundation for achieving peg-in-hole assembly. In recent years, DNN-based object detection has rapidly advanced, capable of achieving real-time object detection across different GPU types, making it highly suitable for the speed requirements of this task. The workflow of visual servo is shown in Figure 2. Following object detection, we further process the obtained objects by target hole fitting to acquire more accurate coordinates. Simultaneously, we extract the point cloud normal vectors of the target region to achieve precise retrieval of position and pose.

### 2.1 Deep neural network structure for object detection

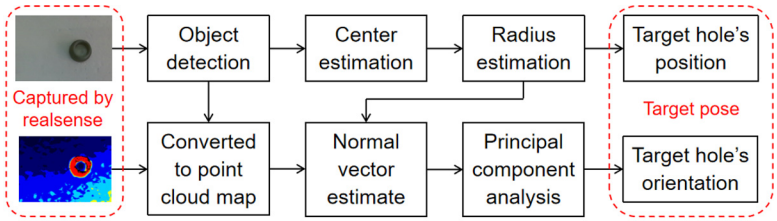
We use the YOLOv7-tiny network for object detection, known for its outstanding detection speed and accuracy in the range of 5FPS to 160FPS. YOLOv7 is a one-stage object detection network (Wang *et al.*, 2023) composed of three parts: input, backbone and head. The input module preprocesses input images to align them to a size of $640 \times 640$, the backbone module is responsible for feature extraction and the head module predicts and outputs results. YOLOv7-tiny, designed for edge GPUs, shows significant improvement in detection performance compared to YOLOv4-tiny.

**Figure 1** The proposed peg-in-hole assembly framework




**Source:** Authors' own work

**Figure 2** The workflow of visual servo




**Source:** Authors' own work

### 2.2 Image processing

The target holes detected by DNN is labeled by square bounding box, which is different from the actual circular assembly holes. Extracting normal vectors using the square bounding box leads to excessive offsets and assembly failures. We use the Hough gradient method for circular fitting (Illingworth and Kittler, 1987), consisting of two steps: center estimation and radius estimation. During center estimation, we perform Canny edge detection, use the Sobel operator to calculate pixel neighborhood gradient values and identify the circle center at the intersection point along the gradient direction of edge pixels. For radius estimation, we calculate distances from edge pixels to the circle center and select the most frequent distance value as the radius.

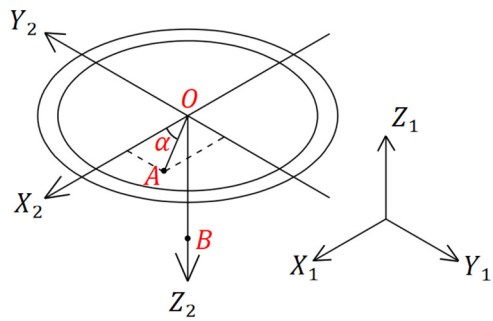
Based on the aforementioned methods, we obtain the position of target holes. However, obtaining orientation information is also crucial for guiding assembly. Using the circular bounding box fitted by image processing, we solve for the normal vector of the plane within the circular bounding box, which means we obtain the normal vector of the target hole. During the data set collection process, we simultaneously obtain the depth map. First, the depth map of the target area is converted into a point cloud by using the camera intrinsic parameters; second, the plane within the neighborhood of each point cloud is estimated using the least squares method and the normal vector perpendicular to this plane is calculated to obtain the normal vector at each point cloud. Finally, all the obtained normal vectors are subjected to principal component analysis to obtain the normal vector of the target hole.

## 3. Adjustment based on force/torque feedback

### 3.1 Data description and preprocessing

In the context of using the BLS for adjustment, it is essential to first define the data description used. This paper is based on the contact force/torque during peg-in-hole assembly to predict actions that can correctly complete the assembly. The model's input comprises six-dimensional force/torque data denoted as $F = [f_x \quad f_y \quad f_z \quad r_x \quad r_y \quad r_z]$, obtained through force/torque sensor. The model's output is the difference between the current pose and the successfully assembled pose, denoted as $\delta = [\delta_T \quad \delta_R]$. Here, $\delta_T = [\delta_x \quad \delta_y]$ represents the translational difference, indicated by a vector pointing from the current position to the target position, and $\delta_R = [\delta_{R_x} \quad \delta_{R_y}]$ signifies the rotational difference, represented by the rotation angles from the current orientation to the target orientation. Because the cross-sections of the peg and hole are both circular, we currently do not consider rotation around the $z$-axis. The calculation of $\delta$ is illustrated in Figure 3 and is obtained through the following equation:

**Figure 3** The coordinate relationship for computing $\delta$




**Source:** Authors' own work

$$\begin{aligned} \delta_x &= -OA \cdot \cos\alpha \\ \delta_y &= -OA \cdot \sin\alpha \\ \delta_{R_x} &= -\arctan\frac{\delta_y}{OB} \\ \delta_{R_y} &= -\arctan\frac{-\delta_x}{OB} \end{aligned} \tag{1}$$

where $OA$ represents the distance from the center point of the peg's end plane to the target position, and $OB$ represents the depth of the hole. The calculation of $\delta$ is conducted in the $O - X_1Y_1Z_1$ coordinate frame, aiming to facilitate robot control.

Due to the influence of various factors on the contact force/torque during assembly, there is a certain degree of measurement error. Additionally, the torque values in different directions are relatively small compared to the contact force data. Therefore, it is necessary to preprocess the force/torque to enhance the model's fitting capability. We use a normalization approach for preprocessing force information. For the six-dimensional force/torque $F = [f_x \quad f_y \quad f_z \quad r_x \quad r_y \quad r_z]$, with mean and standard deviation denoted as $F_{mean}$ and $F_{std}$, the normalized values can be expressed as follows:

$$F_{norm} = \frac{F - F_{mean}}{F_{std}} \tag{2}$$

### 3.2 Broad learning system

In this paper, we use BLS to model the insertion process, input the contact force/torque data, and output the peg pose adjustment. BLS (Chen and Liu, 2017) is based on the random vector functional link network. The network structure is relatively concise, lacking multiple layers of connections,

without coupling between layers and it does not require gradient descent to update weights. The computation speed is significantly superior to deep learning. Precision in prediction can be enhanced by increasing the "broad" of the network. The additional computational load is also relatively lower compared to deep learning, making it highly suitable for systems with high demands on real-time prediction. Comparative experiments on modelling using BLS and deep learning will be shown in Section 4 B.

The input of the BLS consists of mapped features and enhancement nodes. The mapped feature is obtained through linear transformation and activation function mapping of the original input, while the enhancement node is obtained through linear transformation and activation function mapping of the mapped feature. Both the mapped features and enhancement nodes consist of several nodes, each node encapsulating different sets of nodes within the layer. This facilitates the assembly of features extracted by other machine learning models into the broad learning system.

As shown in Figure 4, the network structure diagram of the broad learning system is illustrated, where $X$ represents the original input data, $Y$ represents the output, $W$ is the weights and $Z^n = [Z_1 \quad Z_2 \quad \dots \quad Z_n]$ and $H^m = [H_1 \quad H_2 \quad \dots \quad H_m]$ are the mapped feature and enhancement node, calculated through the following equations:

$$\begin{aligned} Z_i &= \phi\left(XW_{e_i} + \beta_{e_i}\right), \quad i = 1, \dots, n \\ H_j &= \xi\left(Z^n W_{h_j} + \beta_{h_j}\right), \quad j = 1, \dots, m \end{aligned} \tag{3}$$

where $\phi$ and $\xi$ are activation functions, $W_{ei}$ and $W_{h_j}$ are randomly generated weight matrices with appropriate dimensions. Typically, to prevent correlation in the input of enhancement nodes, $W_{h_j}$ is often orthogonal. $\beta_{ei}$ and $\beta_{h_j}$ are randomly generated biases, and $n$ and $m$ represent the number of mapped features and enhancement nodes, respectively. The network's input is a combination of the mapped features and the enhancement nodes:

$$\begin{aligned} Y &= \left[Z_1, \dots, Z_n | \xi\left(Z^n W_{h_1} + \beta_{h_1}\right), \dots, \xi\left(Z^n W_{h_m} + \beta_{h_m}\right)\right] W^m \\ &= [Z_1, \dots, Z_n | H_1, \dots, H_m] W^m \\ &= [Z^n | H^m] W^m \end{aligned} \tag{4}$$

Given $\mathcal{J} = [Z^n | H^m]$, then $W^m = \mathcal{J}^+ Y$. As $\mathcal{J}$ may be a singular matrix, an optimization approach is employed to find the pseudo-inverse:

$$\underset{W}{\operatorname{argmin}} \; : \; ||\mathcal{J}W - Y||_v^{\sigma_1} + \lambda ||W||_u^{\sigma_2} \tag{5}$$

where $\theta_1 > 0$, $\theta_2 > 0$, $u$ and $v$ are typical norm regularization terms and $\lambda$ represents constraints on the sum of weight squares and $W$. When $\lambda = 0$, the above equation transforms into a least squares problem, and the inverse solution can be directly obtained. As $\lambda$ approaches infinity, the solution is constrained and tends to 0. When $\theta_1 = \theta_2 = u = v = 2$, the above equation becomes a ridge regression model, given by:

$$W = \left(\lambda I + \mathcal{J}\mathcal{J}^T\right)^{-1} \mathcal{J}^T Y \tag{6}$$

And:

$$\mathcal{J}^+ = \lim_{\lambda \to 0} \left(\lambda I + \mathcal{J}\mathcal{J}^T\right)^{-1} \mathcal{J}^T \tag{7}$$

This yields the pseudo-inverse $\mathcal{J}^+$ of $\mathcal{J}$, from which the weights $W$ of the network can be determined.

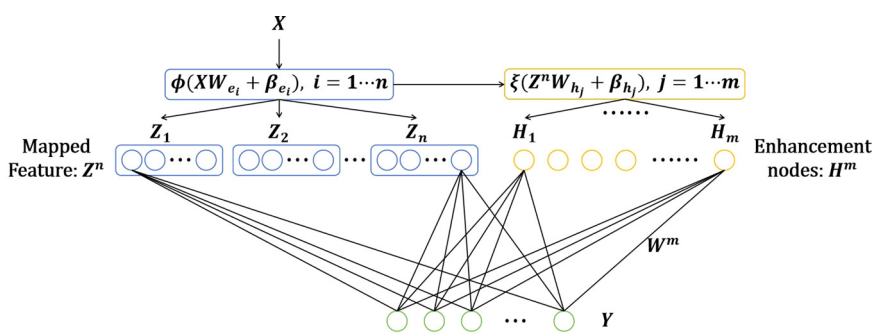
### 3.3 Incremental learning based on successful assembly data

The BLS is highly suitable for incremental learning, and it can be enhanced through methods such as enhancement-node-based, mapped-feature-based and data-based approaches. In the peg-in-hole assembly experiments, collecting a large data set is time-consuming. Therefore, we adopt a data-based incremental learning approach, supplementing new data after each successful assembly to continuously improve the network's predictive accuracy.

When the $k$th adjustment becomes successful assembly, the modified data set from the previous $k-1$ adjustments can be input into the network for training. The modified data set $D_a$ is as follows:

$$D_a = \begin{bmatrix} F_{norm_1} & \sum_{i=1}^{k-1} a_{t_i} & \sum_{i=1}^{k-1} a_{r_i} \\ F_{norm_2} & \sum_{i=2}^{k-1} a_{t_i} & \sum_{i=2}^{k-1} a_{r_i} \\ F_{norm_3} & \sum_{i=3}^{k-1} a_{t_i} & \sum_{i=3}^{k-1} a_{r_i} \\ \vdots & \vdots & \vdots \\ F_{norm_{k-1}} & a_{t_{k-1}} & a_{r_{k-1}} \end{bmatrix} = [X_a \mid Y_a] \tag{8}$$

**Figure 4** The structure of broad learning system




**Source:** Authors' own work

where $F_{norm_i}$ is the recorded force/torque data at the time of contact, $a_{t_i}$ and $a_{r_i}$ are the BLS-predicted next translational and rotational data, respectively. $i \in [1, k-1]$.

At this point, the incremental representations of the mapped features and enhancement nodes are as follows:

$$A_x = \left[\phi(X_a W_{e_1} + \beta_{e_1}), \ldots, \phi(X_a W_{e_n} + \beta_{e_n}) \right. \\ \left. |\xi(Z_x^n W_{h_1} + \beta_{h_1}), \ldots, \xi(Z_x^n W_{h_m} + \beta_{h_m})\right] \quad (9)$$

where $Z_x^n = \left[\phi(X_a W_{e_1} + \beta_{e_1}), \ldots, \phi(X_a W_{e_n} + \beta_{e_n})\right]$ is the group of the incremental features updated by $X_a$. The $W_{e_i}$, $W_{h_j}$ and $\beta_{e_i}$, $\beta_{h_j}$ are randomly generated during the initial of the network.

According to the derivations in references (Chen and Liu, 2017), the weight update for the network is given by:

$$^{x}W_n^m = W_n^m + (Y_a^T + A_x^T W_n^m) B \quad (10)$$

where $B^T = \begin{cases} (C)^+ & if\ C \neq 0 \\ (1 + D^T D)^{-1} (A_n^m)^+ D & if\ C = 0 \end{cases}$, $D^T = A_x^T A_n^{m+}$ and $C = A_x^T - D^T A_n^m$.

## 4. Experiment

### 4.1 Experiments for visual servo

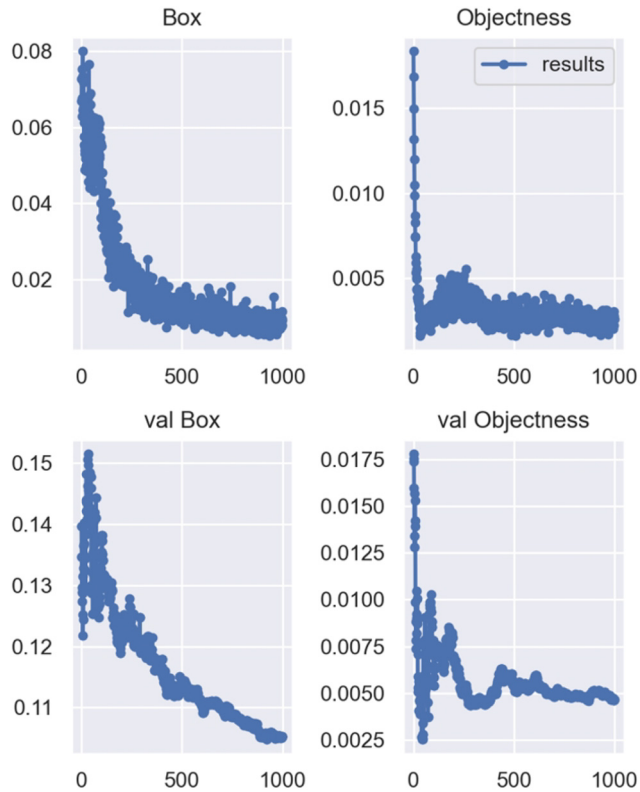
Visual servo consists of both DNN-based object detection and image processing. For the peg-in-hole assembly task, our first goal is to use DNN to get the location of holes. We collected a data set of 50 images with a training, testing and validation set ratio of 7:2:1. Figure 5 illustrates the change in the loss function during the training process, which shows the localization and classification losses in the training and validation sets, respectively, both of which indicating the effectiveness of the training. We used an NVIDIA GeForce GTX 750 Ti edge GPU, and the training took 0.95 h, demonstrating the balance achieved by YOLOv7-tiny in terms of efficiency on edge GPUs and training time, allowing for quick deployment and retraining in new scenarios.

After acquiring the target, we use the image processing methods from Section 2 B to fit a circle and calculate the normal vector of the target hole, Figure 6 shows the fitted circle, demonstrating a close match with the actual assembly holes, Figure 6 shows the normal vector, which aligns well with the actual assembly hole. Thanks to the real-time detection capabilities of the DNN and the efficient image processing, the visual servo process is very quick. In our 30 tests, it takes an average of only 0.12 s to infer the pixel position and orientation of the target hole in a photograph.

### 4.2 Experiments for adjustment based on force/torque feedback

In this paper, we will use the number of peg pose adjustments to judge the validity of the model. Although this is different from some experiments that take the time spent as a judgment criterion, we think it is scientific in the experimental scenario of this paper. The reason is that the time spent for successful assembly depends not only on the number of adjustments but also to a large extent on the speed of the robot arm. If it is necessary to reduce the assembly success time, it is only necessary to increase the speed of the robot arm. To ensure the safety of the experiment, we set a smaller speed of the robot arm, so the use of the number of peg pose adjustment can better reflect the performance of the model.

**Figure 5** Localization (box) and classification (objectness) loss function decreases during training




**Source:** Authors' own work

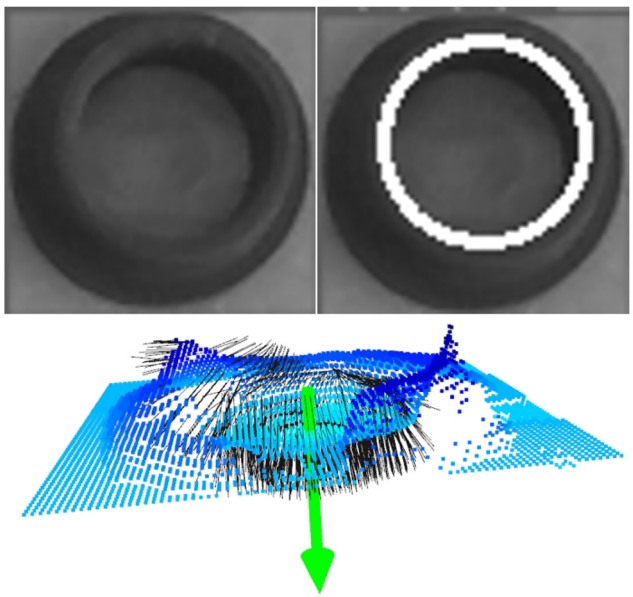
#### 4.2.1 *Simulation experiments*

In the CoppeliaSim simulator, we constructed a robot platform, as shown in Figure 7. The robot arm model used was UR5, with a circular peg of diameter 36 *mm* mounted at the end, and a circular hole of diameter 38 *mm* placed on the ground.

Generally, when the peg cannot be correctly assembled with the hole, three typical situations, as illustrated in Figure 8, usually occur. In robot assembly systems with visual servo, the second and third situations are more likely to happen. Meanwhile, the force/torque data generated in the first situation is similar to that in the second situation. Therefore, we primarily selected data collection points based on the second and third situations. Based on several tests, the selection of data collection points was determined. Taking OB = 20*mm*, on two circular paths with radius of 4.5 *mm* and 1.5 *mm*, respectively, we evenly distributed 16 data collection points, as shown in Figure 9, where the red points represent the data collection points.

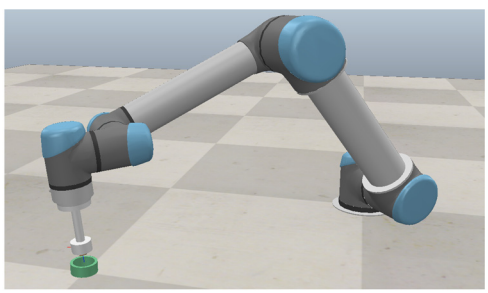
Due to the small tolerance required for peg-in-hole assembly, CoppeliaSim's collision model struggles to accurately simulate such fine contacts, resulting in inaccurate readings from the force/torque measurement module. This discrepancy leads to

**Figure 6** Circular holes obtained by image processing, labeled by white lines (up); normal vector of the target assembly holes, marked by green arrows (down)



**Source:** Authors' own work

**Figure 7** The robot platform in CoppeliaSim simulator

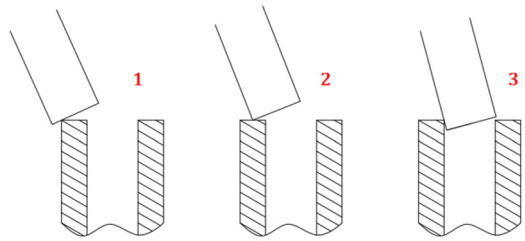


**Source:** Authors' own work

erroneous data, rendering it unsupportive for data-based incremental learning. Therefore, in the simulator, we primarily aimed to verify the effectiveness of the proposed data preprocessing and BLS. According to the data definition in Section 3 A, we randomly generated data combinations within the ranges $OA \in [3, 5mm]$, $OB \in [17, 23mm]$ and $\alpha \in [0, 360^o]$, defining them as the initial assembly poses for the peg. We used collision force/torque data in the trained BLS to predict the next action of the peg and recorded the number of experimental trials until successful assembly. The distribution map of randomly chosen initial positions is shown in Figure 9, where the blue region represents the distribution of randomly chosen initial positions, and the green region represents the area where successful assembly is achieved.

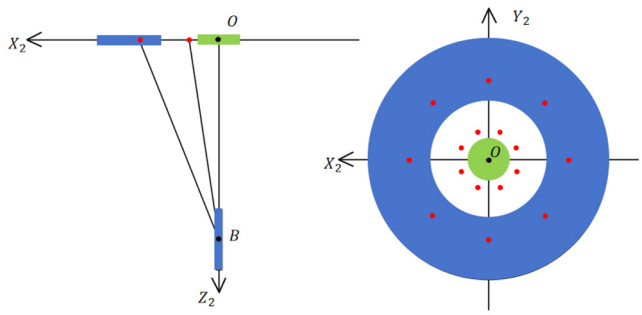
We conducted three sets of experiments, each consisting of assembly starting from 50 randomly generated positions. An experiment was considered a failure if adjustments exceeded 30 times. The success rates for the three sets of experiments were 92%, 98% and 98%, respectively, indicating a certain feasibility of our

**Figure 8** Three typical assembly failure scenarios




**Source:** Authors' own work

**Figure 9** The schematic of data collection and random initial points




**Source:** Authors' own work

assembly method. The average number of adjustments required for successful assembly was calculated every five positions, as shown in Table 1. Due to the randomness in the selection of initial positions, the results also indicate a significant variation in the average number of adjustments, mostly ranging between four and ten adjustments in successful assembly scenarios.

We also use deep learning to build the model. Generally speaking, a DNN consists of an input layer, a hidden layer and an output layer, and each layer contains multiple neurons. In this paper, the input layer contains six nodes representing the input contact force/torque data, and the output layer contains four nodes corresponding to the peg pose adjustment $\delta$ as described above. According to the number of hidden layers, they are named NN_3, DNN_10, DNN_20 and DNN_50, corresponding to the number of layers 3, 10, 20 and 50, respectively. Each hidden layer contains 64 nodes, the activation function used is ReLU and the residual connection and dropout are also used in DNN_50.

Figure10 show the loss function changes during the training process of each network. Due to the random nature of the initialization of the neural network, although the initial loss values of each network differ greatly, they all eventually reach convergence, indicating that the network training is effective. However, when the above trained neural networks were used for peg-in-hole insertion, all of them showed a large error in the prediction value leading to assembly failure. After adjusting the peg's pose, the peg has left the target hole, and it is impossible to use the model for the next prediction.

Due to less research on the neural network model for peg-in-hole insertion, there is no better design paradigm, and the

**Table 1** Results in simulator: the average number of adjustments for successful assembly in every five positions

| Exp. No. | 1–5 | 6–10 | 11–15 | 16–20 | 21–25 | 26–30 | 31–35 | 36–40 | 41–45 | 46–50 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3.4 | 5.8 | 3.5 | 8.4 | 4.3 | 4.4 | 6.2 | 5.4 | 3.3 | 9.8 |
| 2 | 5 | 6.8 | 6 | 4.4 | 5.4 | 6.6 | 5.2 | 6.4 | 3.8 | 4.6 |
| 3 | 12.4 | 5.2 | 7.4 | 14.6 | 6.2 | 3.2 | 9.2 | 3.8 | 6.6 | 8.8 |

**Source:** Authors' own work

design of this model needs to be tried extensively. After these comparisons, we believe that the model built using the BLS has good performance, we will continue to use the model for subsequent experiments.

*4.2.2 Robot platform experiments*

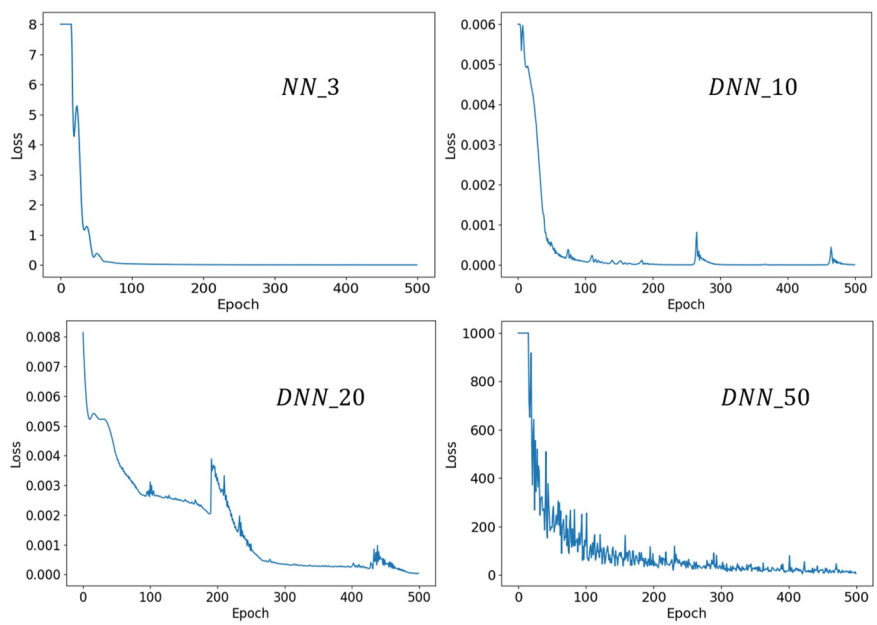
The robot platform we used consists of an Elite EC66 robot arm with an ATI mini-45 force sensor mounted at the end and a shaft of diameter 36 *mm*. On the desk, there is a hole with a diameter of 38 *mm*.

Consistent with the experiments in the simulator, the approach of selecting data collection points and randomly generating initial positions are the same as described previously. In the experiments, we initially conducted BLS without any data preprocessing. The peg gradually moves away from the hole, leading to experiment termination and failure. This indicates that data preprocessing is essential for peg-in-hole assembly experiments.

After introducing data preprocessing, we conducted three sets of experiments, each starting assembly from 50 randomly generated positions. An experiment was considered a failure if adjustments exceeded 30 times, and the success rates for the three sets of experiments were 100%. Simultaneously, we performed data-based incremental learning, continually adding new data to enhance the network's fitting capability, as per the data correction method in Section 3 C. The average number of adjustments required for successful assembly was calculated every five positions, as shown in Table 2. The results demonstrate a gradual reduction in the average number of adjustments with the progression of incremental learning. Initially, an average of more than six adjustments was needed for assembly, ultimately reducing to 1–3 adjustments, proving the effectiveness of data preprocessing and data-based incremental learning. As we set a smaller speed of robot arm movement, the time to complete a set of experiments is about 2 h. Figure 11 illustrates an experimental scenario where successful assembly is achieved after only two adjustments.

Figure 12 shows the force/torque data and the robot arm's $z$-axis data corresponding to this successful assembly. Our criteria for successful assembly are as follows: when the contact force/torque reaches a threshold, we check whether the $z$-axis has reached the assembly success limit. If both conditions are met, it is considered a successful assembly. If either condition is not met, the assembly is deemed unsuccessful, requiring further pose adjustments. From Figure 12, it can be observed that the peg hole did not meet the assembly success criteria in the first two contact but achieved successful assembly after two pose adjustments. It should be noted that during successful assembly, significant forces in the z-direction are recorded, indicating the peg has fully passed through the hole and was in contact with the desk, as corroborated by the $z$-axis data.

**Figure 10** Plot of loss function changes during model training using deep learning building




**Source:** Authors' own work

**Table 2** Results in real: the average number of adjustments for successful assembly in every five positions

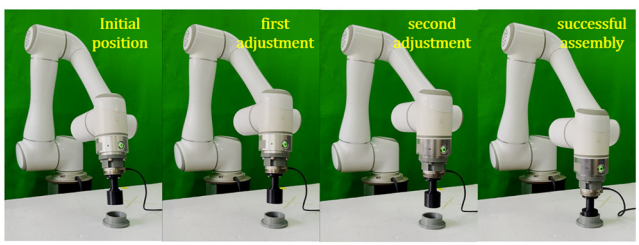
| Exp. No. | 1–5 | 6–10 | 11–15 | 16–20 | 21–25 | 26–30 | 31–35 | 36–40 | 41–45 | 46–50 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 6 | 2.6 | 2.4 | 2.4 | 2 | 2.4 | 2.4 | 2.8 | 2 | 1.6 |
| 2 | 6.2 | 2.4 | 2.4 | 1.6 | 2 | 2.6 | 1.8 | 1.6 | 2.2 | 2.4 |
| 3 | 6.6 | 3.8 | 2 | 2.4 | 1.8 | 2.4 | 1.6 | 1.6 | 1.6 | 1.8 |

**Source:** Author's own work

*4.2.3 Generalization experiment*

In the experiments, we also validated the generalization capability of the proposed method. We transferred the trained BLS through incremental learning to assemblies with a peg diameter of 18 *mm* and a hole diameter of 20 *mm*, maintaining the same random initial position selection as described earlier. We conducted two sets of experiments, each starting assembly from 50 random positions, and the experimental results are presented in Table 3. Figure 13 illustrates a schematic representation of successful assembly after only two adjustments. The results indicate that our proposed method exhibits good generalization capability, enabling the trained network to be transferred to assemblies with different sizes of pegs and holes.

**Figure 11** A successful assembly after two adjustments




**Source:** Authors' own work

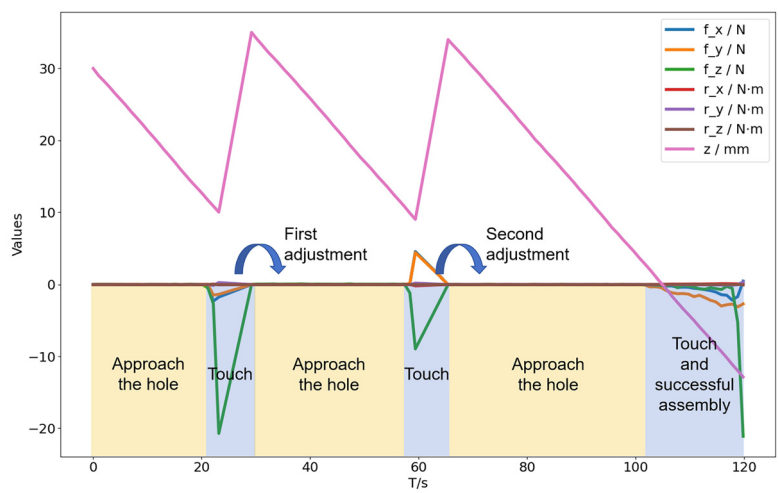
### 4.3 Experiments for the proposed framework

We validated the effectiveness of the entire framework on a robot platform, the camera used was a RealSense D455 and other settings remained consistent with the aforementioned. We conducted a total of 30 experiments, achieving a 100% success rate in assembly. Figure 14 shows a complete assembly at one time. Benefiting from the excellent real-time detection performance of YOLOv7, this framework is capable of swiftly completing initial visual localization. With force/torque feedback, the assembly is accomplished in 1–3 adjustments.

## 5. Conclusion

This paper introduced a framework inspired by human behavior for peg-in-hole assembly. The framework comprises two components: visual servo and adjustment based on force/torque feedback. The visual servo part uses monocular vision, using the YOLOv7-tiny object detection network to determine the position of the target hole. Further precision localization and extraction of the target hole's normal vector are achieved through image processing. The adjustment based on force/torque feedback uses BLS, taking peg and hole contact force/torque data as input and producing the next action for the peg. Incremental learning is used to enhance the network's predictive capability, ultimately achieving assembly completion in 1–3 adjustments. We also validated the generalization capability of the BLS, demonstrating that

**Figure 12** Force/torque and z-axis data for successful assembly after two adjustments



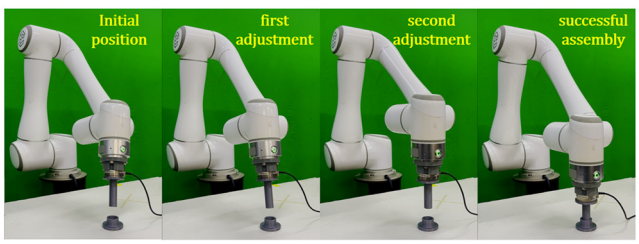

**Source:** Authors' own work

**Table 3** The results of generalization capability experiments

| Exp. No. | Assembly success Rate (%) | 3 or fewer Adjustments (%) | 5 or fewer Adjustments (%) |
|---|---|---|---|
| 1 | 100 | 70 | 82 |
| 2 | 100 | 66 | 88 |

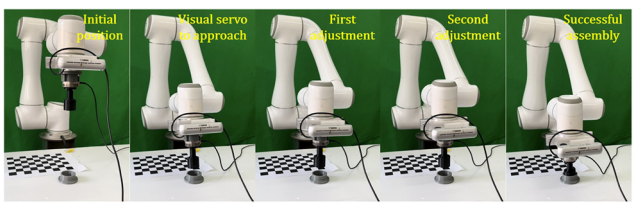
**Source:** Authors' own work

**Figure 13** A successful assembly after two adjustments in generalization experiments




**Source:** Authors' own work

**Figure 14** A complete assembly at one time




**Source:** Authors' own work

the trained network can be applied to peg-in-hole assemblies of different sizes.

The advantages of this framework lie in its simplicity of structure, using only a monocular camera for visual servo. The selected DNN achieves a good balance between equipment selection and training time, facilitating easy redeployment. The BLS, based on incremental learning, exhibits fast training speed, strong fitting capability and data preprocessing that imparts a degree of generalization to the network. Our experimental setup aligned with practical assembly scenarios, starting assembly from any initial position and offering broad application prospects. In the future, we plan to extend this work to support the peg-in-hole assembly with different shapes.

## References


Chang, W.C. and Wu, C.H. (2017), "Automated USB peg-in-hole assembly employing visual servoing", *Proceedings of 2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, IEEE, pp. 352-355.

Chang, R., Lin, C. and Lin, P. (2011), "Visual-based automation of peg-in-hole microassembly process", *Journal of Manufacturing Science and Engineering*, Vol. 133 No. 4, pp. 1087-1357.

Chen, H. and Liu, Y. (2013), "Robotic assembly automation using robust compliant control", *In: Robotics and Computer-Integrated Manufacturing*, Vol. 29 No. 2, pp. 293-300.

Chen, C.P. and Liu, Z. (2017), "Broad learning system: an effective and efficient incremental learning system without the need for deep architecture", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29 No. 1, pp. 10-24.

Dong, J., Si, W. and Yang, C. (2023), "A novel human-robot skill transfer method for contact-rich manipulation task", *Robotic Intelligence and Automation*, Vol. 43 No. 3.

Fan, B., Dai, Y. and He, M. (2023), "Rolling shutter camera: modeling, optimization and learning", *Machine Intelligence Research*, Vol. 20 No. 6, pp. 783-798.

Fang, S., Huang, X., Chen, H. and Xi, N. (2016), "Dual-arm robot assembly system for 3C product based on vision guidance", *Proceedings of 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, pp. 807-812.

Fei, H., Wang, Z., Tedeschi, S. and Kennedy, A. (2023), "Boosting visual servoing performance through RGB-based methods", *Robotic Intelligence and Automation*, Vol. 43 No. 4, pp. 468-475.

Gu, J., Wang, W., Zhu, M., Lv, Y., Huo, Q. and Xu, Z. (2018), "Research on a technology of automatic assembly based on uncalibrated visual servo system", *Proceedings of 2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, IEEE, pp. 872-877.

Haugaard, R., Langaa, J., Sloth, C. and Buch, A. (2021), "Fast robust peg-in-hole insertion with continuous visual servoing", Proceedings of Conference on Robot Learning, PMLR, pp. 1696-1705.

Huang, S., Murakami, K., Yamakawa, Y., Senoo, T. and Ishikawa, M. (2013), "Fast peg-and-hole alignment using visual compliance", *Proceedings of 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 286-292.

Illingworth, J. and Kittler, J. (1987), "The adaptive Hough transform", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9 No. 5, pp. 690-698.

Jasim, I.F., Plapper, P.W. and Voos, H. (2014), "Position identification in force-guided robotic peg-in-hole assembly tasks", *Procedia Cirp*, Vol. 23, pp. 217-222.

Kang, H., Zang, Y., Wang, X. and Chen, Y. (2022), "Uncertainty-driven spiral trajectory for robotic peg-in-hole assembly", *IEEE Robotics and Automation Letters*, Vol. 7 No. 3, pp. 6661-6668.

Li, C., Chen, P., Xu, X., Wang, X. and Yin, A. (2021), "A coarse-to-fine method for estimating the axis pose based on 3D point clouds in robotic cylindrical shaft-in-hole assembly", *Sensors*, Vol. 21 No. 12, p. 4064.

Liu, S., Xu, D., Liu, F., Zhang, D. and Zhang, Z. (2015), "Relative pose estimation for alignment of long cylindrical components based on microscopic vision", *IEEE/ASME Transactions on Mechatronics*, Vol. 21 No. 3, pp. 1388-1398.

Liu, S., Xu, D., Zhang, D. and Zhang, Z. (2014), "High precision automatic assembly based on microscopic vision and force information", *IEEE Transactions on Automation Science and Engineering*, Vol. 13 No. 1, pp. 382-393.

Ma, Y., Liu, X., Zhang, J., Xu, D., Zhang, D. and Wu, W. (2020), "Robotic grasping and alignment for small size components assembly based on visual servoing", *The*

*International Journal of Advanced Manufacturing Technology*, Vol. 106, pp. 4827-4843.

Park, H., Park, J., Lee, D.H., Park, J.H., Baeg, M.H. and Bae, J.H. (2017), "Compliance-based robotic peg-in-hole assembly strategy without force feedback", *IEEE Transactions on Industrial Electronics*, Vol. 64 No. 8, pp. 6299-6309.

Park, H., Bae, J.H., Park, J.H., Baeg, M.H. and Park, J. (2013), "Intuitive peg-in-hole assembly strategy with a compliant manipulator", Proceedings of IEEE ISR 2013, IEEE, pp. 1-5.

Puang, E.Y., Tee, K.P. and Jing, W. (2020), "Kovis: keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation", *Proceedings of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 7527-7533.

Shen, Y., Jia, Q., Wang, R., Huang, Z. and Chen, G. (2023), "Learning-based visual servoing for high-precision peg-in-hole assembly", *Actuators*, Vol. 12 No. 4, p. 144.

Song, J., Chen, Q. and Li, Z. (2021), "A peg-in-hole robot assembly system based on gauss mixture model", *Robotics and Computer-Integrated Manufacturing*, Vol. 67, p. 101996.

Song, H.C., Kim, Y.L. and Song, J.B. (2016), "Guidance algorithm for complex-shape peg-in-hole strategy based on geometrical information and force control", *Advanced Robotics*, Vol. 30 No. 8, pp. 552-563.

Spector, O. and Di Castro, D. (2021), "Insertionnet-a scalable solution for insertion", *IEEE Robotics Automation Letters*, Vol. 6 No. 3, pp. 5509-5516.

Spector, O. and Zacksenhouse, M. (2021), "Learning contact-rich assembly skills using residual admittance policy", *Proceedings of 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 6023-6030.

Spector, O., Tchuiev, V. and Di Castro, D. (2022), "Insertionnet 2.0: minimal contact multi-step insertion using multimodal multiview sensory input", *Proceedings of 2022 International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 6330-6336.

Triyonoputro, J.C., Wan, W. and Harada, K. (2019), "Quickly inserting pegs into uncertain holes using multi-view images and deep network trained on synthetic data", *Proceedings of 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 5792-5799.

Wang, C.Y., Bochkovskiy, A. and Liao, H.Y.M. (2023), "YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors", *Proceedings of Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7464-7475.

Wu, W., Peng, H. and Yu, S. (2023), "YuNet: a tiny millisecond-level face detector", *Machine Intelligence Research*, Vol. 20 No. 5, pp. 1-10.

Xie, Z., Wang, S., Zhao, W. and Guo, Z. (2019), "Context attention module for human hand detection", *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, IEEE, pp. 555-560.

Xie, Z., Wang, S., Zhao, W. and Guo, Z. (2022), "A robust context attention network for human hand detection", *Expert Systems with Applications*, Vol. 208, p. 118132, doi: 10.1016/j.eswa.2022.118132.

Zhang, J.Y., Liu, Y., Liu, Y. and Hao, Y.P. (2017), "Automatic microassembly method based on teaching playback and visual servo", *Proceedings of 2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, IEEE, pp. 878-882.

Zhao, W., Wang, S., Xie, Z., Shi, J. and Xu, C. (2019), "GAN-EM: GAN based EM learning framework", Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, pp. 4404-4411, doi: 10.24963/ijcai.2019/612.

Zhao, Y., Gao, F., Zhao, Y. and Chen, Z. (2020), "Peg-in-hole assembly based on six-legged robots with visual detecting and force sensing", *Sensors 20*, Vol. 20 No. 10, p. 2861.

Zou, P., Zhu, Q., Wu, J. and Xiong, R. (2020), "Learning-based optimization algorithms combining force control strategies for peg-in-hole assembly", *Proceedings of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 7403-7410.