



Article

# Absolute Positioning Accuracy Improvement in an Industrial Robot

Yizhou Jiang , Liandong Yu \*, Huakun Jia , Huining Zhao  and Haojie Xia

School of Instrument Science and Opto-Electronics Engineering, Hefei University of Technology, Hefei 230009, China; jiangyizhou1@mail.hfut.edu.cn (Y.J.); huakun\_jia@mail.hfut.edu.cn (H.J.); hnzhaohao@mail.hfut.edu.cn (H.Z.); hxia@hfut.edu.cn (H.X.)

\* Correspondence: liandongyu@hfut.edu.cn; Tel.: +86-138-5606-1480

Received: 6 July 2020; Accepted: 31 July 2020; Published: 5 August 2020



**Abstract:** The absolute positioning accuracy of a robot is an important specification that determines its performance, but it is affected by several error sources. Typical calibration methods only consider kinematic errors and neglect complex non-kinematic errors, thus limiting the absolute positioning accuracy. To further improve the absolute positioning accuracy, we propose an artificial neural network optimized by the differential evolution algorithm. Specifically, the structure and parameters of the network are iteratively updated by differential evolution to improve both accuracy and efficiency. Then, the absolute positioning deviation caused by kinematic and non-kinematic errors is compensated using the trained network. To verify the performance of the proposed network, the simulations and experiments are conducted using a six-degree-of-freedom robot and a laser tracker. The robot average positioning accuracy improved from 0.8497 mm before calibration to 0.0490 mm. The results demonstrate the substantial improvement in the absolute positioning accuracy achieved by the proposed network on an industrial robot.

**Keywords:** absolute positioning accuracy; industrial robot; neural network; differential evolution algorithm

## 1. Introduction

Although industrial robots are flexible platforms and provide high repeatability for the automation of a variety of manufacturing tasks, a low absolute positioning accuracy may limit their applicability [1]. Error sources in robots can be either kinematic or non-kinematic [2–4]. Manufacturing and assembly tolerances cause deviation of the actual kinematic parameters from their nominal values, causing kinematic errors [5]. On the other hand, non-kinematic errors are usually neglected by their difficult modeling. Nevertheless, non-kinematic errors caused by factors such as temperature variations, joint and link compliance, and gear backlash have a considerable effect on the absolute positioning accuracy. Thus, an efficient method to effectively compensate positioning deviations caused by both kinematic and non-kinematic errors should be devised.

Extensive reports have shown that kinematic errors account for about 90% of the total positioning error [6]. Therefore, kinematic calibration effectively improves the absolute positioning accuracy of robots. This type of calibration comprises four steps: modeling, measurement, identification, and compensation or correction [7]. Thus, the kinematic model plays a critical role during robot calibration and should meet the following requirements for parameter identification: continuity, completeness, and minimality [8]. The Denavit–Hartenberg convention is a widely used modeling method to describe the robot kinematics [9]. However, this method is not continuous when two adjacent joint axes are parallel or nearly parallel, thus presenting singularities. To overcome the singularity problem, many studies have suggested alternative models. For instance, Hayati added a revolute

parameter to the Denavit–Hartenberg convention, establishing the modified Denavit–Hartenberg model [10]. The complete and parametrically continuous model proposed by Meng and Zhuang provides completeness and continuity, but some of its parameters are redundant [11]. The product of exponentials (POE) model provides a complete geometric and parameterized representation of the robot motion, greatly simplifying the kinematic analysis of robotic mechanisms. The POE formula satisfies continuity, completeness, and minimality, being suitable and widely used for robot kinematic modeling [12].

Robot calibration aims to optimize an objective function and obtain accurate kinematic parameters [13]. The most common calibration algorithms are those based on the least squares method, such as the Gauss–Newton and Levenberg–Marquardt (LM) algorithms, with the latter being robust against interference and enabling global search [14,15]. However, neglecting error sources such as temperature variations and gear backlash limits the effectiveness and accuracy of robot calibration. Various correction methods have been developed to mitigate the influence of non-kinematic errors. Ma et al. demonstrated the importance of non-kinematic errors and generalized them by a representation with error matrices containing high-order Chebyshev polynomials that reflect individual error terms [16]. Whitney et al. developed a forward calibration method using joint encoder offset, link length, and consecutive-axis relative orientations as parameters and experimentally evaluated the effects of joint backlash, gear transmission, and compliance errors [17]. Chen et al. parameterized the joint flexibility error for estimation to improve the absolute positioning accuracy. However, the mathematical modeling of these methods is complex and various non-kinematic errors are ignored, limiting their applicability [18].

Given the difficulty to model non-kinematic errors, artificial neural networks have been used as an alternative to compensate the absolute positioning error [19]. Neural networks allow one to compensate positioning errors, avoid complex modeling, and comprehensively consider the influence of every error source. Ding et al. proposed a neural network combining with Faugeras vision system calibration technology for accurate calibration of a delta-robot vision system [20]. Likewise, Jang et al. used a radial basis function network to approximate the relationship between the robot joint readings and corresponding position errors [21]. Wang et al. developed a neural network to extract local features of error surface and estimated the positioning errors during calibration of a robot manipulator [22].

The performance of neural networks is influenced by their structure and parameters. To optimize neural networks, Rouhani et al. introduced one-pass heuristic rules to determine the center, number, and spread of hidden neurons in the network [23]. Feng et al. used evolutionary particle swarm optimization and developed a neural network with a self-generating radial basis function [24]. González et al. developed a multiobjective evolutionary algorithm to optimize neural networks. Differential evolution algorithms enable global optimization with high identification accuracy and optimal rate [25]. Accordingly, we adopt differential evolution for pre-training of a neural network to optimize the thresholds, initial weights, and the number of hidden neurons for maximizing accuracy and efficiency. We conducted simulations and experiments to verify the performance of the proposed network by employing a six-degree-of-freedom (DOF) industrial robot and a laser tracker as reference.

The main contributions of this work can be summarized as follows. (1) The influence of kinematic and non-kinematic errors on the absolute positioning accuracy is analyzed, and the limitations of kinematic calibration are addressed. (2) A neural network optimized using differential evolution is proposed to enhance the absolute positioning accuracy. Thresholds, initial weights, and the number of hidden neurons in the neural network are optimized using differential evolution to improve the performance of the neural network. The proposed network mitigates the effects of kinematic and non-kinematic errors and thus improves the absolute positioning accuracy of a robot. (3) The experiment and simulation are completed with the six-DOF robot as the research object and the laser tracker as the measuring instrument. The theoretical correctness and effectiveness of the proposed neural network are verified by simulations and experiments.

The remainder of this paper is organized as follows: The influences of kinematic and non-kinematic errors on robot positioning accuracy are analyzed in Section 2. The proposed neural network optimized by differential evolution is detailed in Section 3. In Section 4, simulations and experiments are performed for compensating the absolute positioning error. The discussion and conclusions are presented in Section 5.

## 2. Kinematic and Non-Kinematic Error Analysis

The accuracy of a robot kinematic model depends on robot calibration. We use the POE model to describe the robot kinematics [26], considering the coordinate systems of the robot shown in Figure 1.

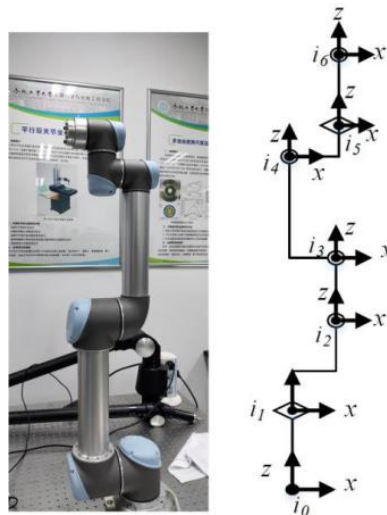


Figure 1. Coordinate systems of the robot manipulator.

The pose of end-effector frame  $\{t\}$  with respect to base frame  $\{s\}$  is given by

$$g_{st}(\boldsymbol{\theta}) = e^{\hat{\xi}_1 \theta_1} e^{\hat{\xi}_2 \theta_2} e^{\hat{\xi}_3 \theta_3} e^{\hat{\xi}_4 \theta_4} e^{\hat{\xi}_5 \theta_5} e^{\hat{\xi}_6 \theta_6} e^{\hat{\xi}_{st}} = \begin{bmatrix} \mathbf{p}(\boldsymbol{\theta}) & \mathbf{r}(\boldsymbol{\theta}) \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & r_x \\ n_y & o_y & a_y & r_y \\ n_z & o_z & a_z & r_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where  $\theta_i$  indicates the robot joint variable and  $\hat{\xi}_{st}$  is the twist of the initial transformation. In Equation (1),  $\hat{\xi}_i$  can be expressed as

$$\hat{\xi}_i = \begin{bmatrix} \hat{\mathbf{w}}_i & \mathbf{v}_i \\ 0 & 0 \end{bmatrix} \quad (2)$$

where  $\hat{\mathbf{w}}_i$  and  $\mathbf{v}_i$  are given by

$$\hat{\mathbf{w}}_i = \begin{bmatrix} 0 & -w_{zi} & w_{yi} \\ w_{zi} & 0 & -w_{xi} \\ -w_{yi} & w_{xi} & 0 \end{bmatrix} \quad (3)$$

$$\mathbf{v}_i = -\mathbf{w}_i \times \mathbf{q}_i \quad (4)$$

where  $\mathbf{q}_i = [q_{xi}, q_{yi}, q_{zi}]$  represents the coordinate of the origin of each axis in the base coordinate system  $\{s\}$ ;  $\mathbf{w}_i = [w_{xi}, w_{yi}, w_{zi}]$  is the direction vector in system  $\{s\}$  of each rotation axis.

For a rotational joint, the exponential matrix of the motion screw can be represented as

$$e^{\hat{\xi}_i \theta_i} = \begin{bmatrix} e^{\hat{\mathbf{w}}_i \theta_i} & (I - e^{\hat{\mathbf{w}}_i \theta_i})(\mathbf{w}_i \times (-\mathbf{w}_i \times \mathbf{q}_i)) \\ 0 & 1 \end{bmatrix} \quad (5)$$

The forward kinematics of a six-DOF serial robot is given by

$$f_c = g(w_1, \dots, w_6, q_1, \dots, q_6, \theta) = e^{\hat{\xi}_1 \theta_1} e^{\hat{\xi}_2 \theta_2} \dots e^{\hat{\xi}_6 \theta_6} e^{\hat{\xi}_{st}} \quad (6)$$

The kinematic parameters of the robot generally deviate from their designed values due to different kinematic errors such as mechanical deformation, assembly and machining errors etc., as shown in Figure 2.

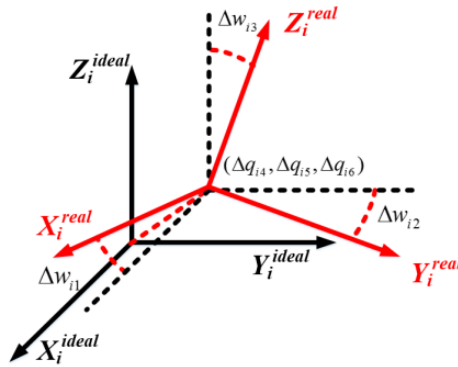


Figure 2. Definition of kinematic errors for link  $i$ .

When the kinematic errors are only considered, the actual robot position  $f_{real}$  can be expressed as

$$f_{real} = g(w_1 + \Delta w_1, \dots, w_6 + \Delta w_6, q_1 + \Delta q_1, \dots, q_6 + \Delta q_6, \theta) \quad (7)$$

where  $\Delta q_i$  and  $\Delta w_i$  represent the kinematic errors to be corrected. The mathematical expression of kinematic errors can be obtained by kinematic modeling. These errors exist whether the robot is moving or not. Therefore, kinematic errors can be identified by the calibration method so as to effectively improve the positioning accuracy of the robot.

However, non-kinematic errors undermine accuracy, but they are generally neglected due to their modeling difficulty and complexity. All error sources whose contributions to positioning errors cannot be characterized by kinematic parameters are herein described as non-kinematic errors [18]. The non-kinematic errors relate to pose and dynamical behavior of the robot, so it is difficult to obtain the exactly mathematical expression. Therefore, calibration cannot mitigate the effects of non-kinematic errors. Non-kinematic errors with considerable effects, such as those related to strain wave gearing and joint flexibility, occur between the angular encoder and output shaft at the joint, and the nominal joint variable should be compensated for computing the actual joint rotation.

Considering non-kinematic errors, the actual robot joint variable can be expressed as

$$\theta_{i\_real} = \hat{\theta}_i + C_i(\hat{\theta}_i) \quad (8)$$

where  $\hat{\theta}_i$  is the nominal robot joint variable and  $C_i(\hat{\theta}_i)$  represents the compensation factor for non-kinematic errors.

We can use Chebyshev polynomials to calculate  $C_i(\hat{\theta}_i)$  [16]:

$$C_i(\hat{\theta}_i) = a_0 + a_1 c_1(\hat{\theta}_i) + \dots + a_m c_m(\hat{\theta}_i) \quad (9)$$

with

$$\begin{aligned} c_0(\hat{\theta}_i) &= 1, c_1(\hat{\theta}_i) = \lambda, c_2(\hat{\theta}_i) = 2\hat{\theta}_i^2 - 1, c_3(\hat{\theta}_i) = 4\hat{\theta}_i^3 - 3\hat{\theta}_i, \\ c_4(\hat{\theta}_i) &= 8\hat{\theta}_i^4 - 8\hat{\theta}_i^2 + 1, \dots, c_{m+1}(\hat{\theta}_i) = 2\hat{\theta}_i c_m(\hat{\theta}_i) - c_{m-1}(\hat{\theta}_i) \end{aligned} \quad (10)$$

where  $m$  denotes the order of the Chebyshev polynomial and  $a_0, a_1, \dots, a_m$  are the polynomial coefficients to be calculated.

Thus, when nominal joint variables and theoretical kinematic parameters of a robot are known, its actual position is given by

$$f_{real} = g(\Delta w_1, \dots, \Delta w_6, \Delta q_1, \dots, \Delta q_6, C_1(\hat{\theta}_1), \dots, C_6(\hat{\theta}_6)) \quad (11)$$

The complexity of non-kinematic errors and the coupling between the compensation of nominal joint variables and robot pose hinder the determination of the actual mathematical representation of the robot position. Therefore, we use a neural network to directly estimate the robot position based on the joint variables. This method avoids complex calibration and modeling, effectively improves the absolute positioning accuracy of the robot and mitigates the influence of kinematic and non-kinematic errors.

### 3. The Proposed Neural Network

Different neural networks such as radial basis function and back-propagation networks have been widely used to compensate the absolute positioning error [19–22]. The compensation results depend on the network structure and parameters including thresholds, initial weights, and number of hidden neurons. Thus, we adopt differential evolution to optimize the structure and parameters of the proposed neural network for maximizing the compensation effect.

#### 3.1. The Back-Propagation Neural Network

The back-propagation neural network is a kind of artificial neural network trained by the error back-propagation method. Its outstanding advantages are its strong linear mapping ability and flexible network structures. They provide strong linear mapping and a flexible structure.

Figure 3 shows the proposed neural network that consists of input, hidden, and output layers. In the proposed network, the input layer has six nodes representing the robot joint variables, and the output layer has three nodes representing the coordinates of the robot position obtained from the laser tracker.

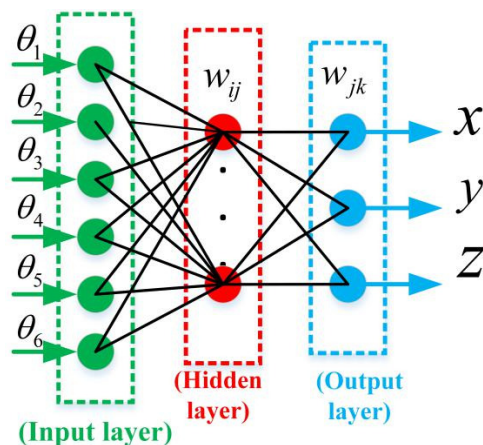


Figure 3. Structure of the back-propagation neural network.

In Figure 3,  $w_{ij}$  is the connection weight between neuron  $i$  in the input layer and neuron  $j$  in the hidden layer, and  $w_{jk}$  is that from neuron  $j$  in the hidden layer to neuron  $k$  in the output layer. The output from the hidden layer is expressed as

$$H_j = G\left(\sum_{i=1}^6 w_{ij}\theta_i\right) \quad j = 1, 2, \dots, l \quad (12)$$

where  $l$  is the number of nodes in the hidden layer and  $G$  is an activation function, which we set as follows:

$$G(x) = \frac{1}{1 + e^x} \quad (13)$$

For  $x$ , the output layer provides

$$x = \sum_{j=1}^l H_j w_{jx} \quad (14)$$

The residual errors are obtained by subtracting the predicted values from the expected ones, which can be considered as position error values. The residual mean squared error is given by

$$F = \frac{1}{m} \sum_{v=1}^m (T_v - O_v)^2 \quad v = 1, 2, \dots, m \quad (15)$$

where  $m$  is the number of training samples,  $O_v = [x_v, y_v, z_v]$  is the neural network output, and  $T_v$  is the expected output.

As long as the residual mean squared error is equal to or higher than the target training error or the number of iterations does not reach its limit, the biases and weights are updated to reduce the deviation between the predicted and expected values. Using gradient descent, the weights are updated as follows:

$$\Delta w_{ij} = -\eta \frac{\partial F}{\partial w_{ij}} \quad (16)$$

$$w_{ij} = \Delta w_{ij} + w_{ij} \quad (17)$$

Figure 4 shows the training flowchart of the neural network.

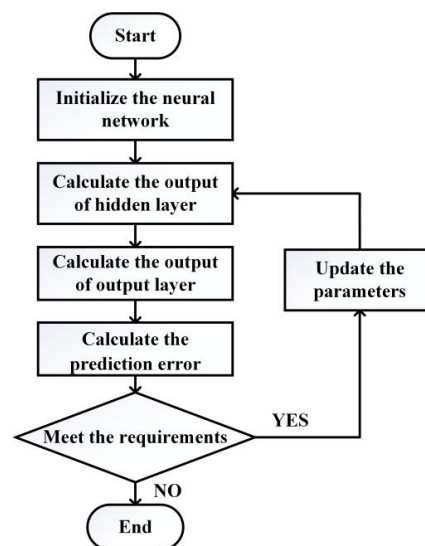


Figure 4. Training flowchart of the neural network.

### 3.2. Differential Evolution Optimization

Differential evolution allows one to perform global optimization with high identification accuracy and optimal rate. Differential evolution is a population-based stochastic optimizer that explores the search space by sampling from multiple random initial points [27]. We use differential evolution to pre-train the proposed neural network for optimizing its thresholds, initial weights, and number of hidden neurons.

The differential evolution algorithm typically consists of three elements:

- (1) Encoding solution population;

- (2) Fitness function to evaluate solution optimality;
- (3) Evolutionary operation comprising selection, crossover and mutation.

Individual  $\epsilon_i = \{w_i, B_i\}$  is initialized to represent different numbers of hidden layers, weight values  $w_i$ , and threshold values  $B_i$ . The length of  $\epsilon_i$  is obtained from the number of hidden neurons and calculated as follows:

$$\text{length}(\epsilon_i) = n_{in}n_{hid} + n_{hid} + n_{hid}n_{out} + n_{out} \quad (18)$$

where  $n_{in}$ ,  $n_{hid}$  and  $n_{out}$  are the number of neurons in the hidden, input, and output layers, respectively. Different values of  $\epsilon_i$  determine the performance of the neural network, as few neurons undermine accuracy, whereas many neurons increase the training time and lead to overfitting. In addition, the initial weights strongly affect the performance and convergence rate of the neural network. Hence, differential evolution allows one to enhance the learning rate and prediction accuracy.

For differential evolution, mutation is expressed as

$$\epsilon_{mi}^g = \epsilon_{gbest}^g + 0.5(\epsilon_a^g - \epsilon_b^g). \quad (19)$$

where  $g$  is the number of iterations,  $\epsilon_{gbest}^g$  is the neural network parameter providing the highest performance, and  $\epsilon_a^g$  and  $\epsilon_b^g$  are two particles randomly selected from the population for differential evolution. As the lengths of  $\epsilon_{gbest}^g$ ,  $\epsilon_a^g$ , and  $\epsilon_b^g$  are different due to the varying number of hidden neurons, mutation cannot be directly performed. Instead, we apply the best selection method to guarantee the same vector lengths during mutation. Specifically, the length of the best individual,  $\epsilon_{gbest}^g$ , is employed as the basis to unify the lengths of the other individuals and perform mutation. When  $\text{length}(\epsilon_a^g) < \text{length}(\epsilon_{gbest}^g)$ , the missing parameters are randomly added to  $\epsilon_a^g$  to obtain  $\text{length}(\epsilon_a^g) = \text{length}(\epsilon_{gbest}^g)$ . When  $\text{length}(\epsilon_a^g) > \text{length}(\epsilon_{gbest}^g)$ , the excess parameters of  $\epsilon_a^g$  are considered as interferent and disregarded during mutation.

Test individuals  $\epsilon_T$  are then generated through crossover using Equation (20), CR is a real-valued crossover probability factor in range [0,1] that controls the probability that a trial vector parameter will be randomly chosen. Generally, CR affects the convergence velocity and robustness of the search process. In this paper, CR = 0.9 to ensure a fast convergence rate.

Thus, for each parameter of the particle, crossover can be expressed as

$$\epsilon_T(j) = \begin{cases} \epsilon_{mi}^g(j), \text{rand}(0,1) \leq CR \\ \epsilon_i^g(j), \text{rand}(0,1) > CR \end{cases} \quad (j = 1, 2, \dots, n_{hid}) \quad (20)$$

Selection is based on a greedy search strategy expressed as

$$\epsilon_i^{g+1} = \begin{cases} \epsilon_T, F(\epsilon_T) < F(\epsilon_i^g) \\ \epsilon_i^g, F(\epsilon_T) > F(\epsilon_i^g) \end{cases} \quad (21)$$

The optimized parameters during pre-training are applied to the proposed neural network. If the predicted values after pre-training do not reach the expected deviation, the number of hidden layers of each particle is updated using Equation (22). This procedure is repeated until the desired values are obtained.

$$K_i = \begin{cases} K_i - 1 & \text{if}(K_{best} < K_i) \\ K_i + 1 & \text{if}(K_{best} \geq K_i) \end{cases} \quad (22)$$

where  $K_i$  is the number of hidden layers for individual  $i$  and  $K_{best}$  is the best solution. The differential evolution algorithm is described in Figure 5 and proceeds until the desired prediction error is reached.

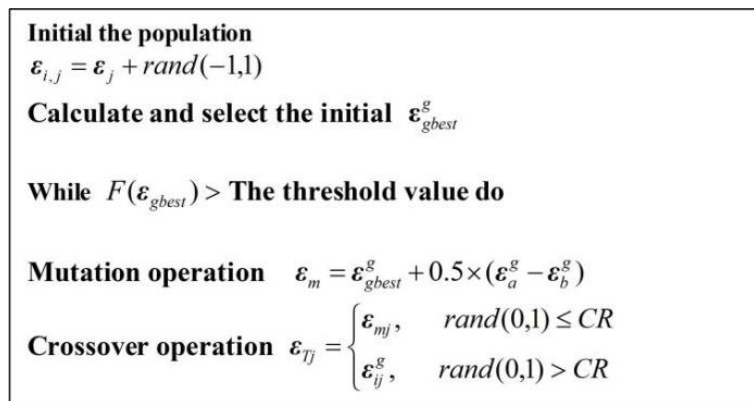


Figure 5. Differential evolution for pre-training the proposed neural network.

#### 4. Simulations and Experiments

We validated the performance of the proposed neural network through simulations and experiments of error compensation on a six-DOF robot manipulator. For comparison, the robot kinematic parameters were calibrated using the POE model [12] and LM algorithm [14,15].

As described in standard ISO 9283 [28], the positioning accuracy is the difference between the position of a command pose and the barycenter of the attained positions, as illustrated in Figure 6.

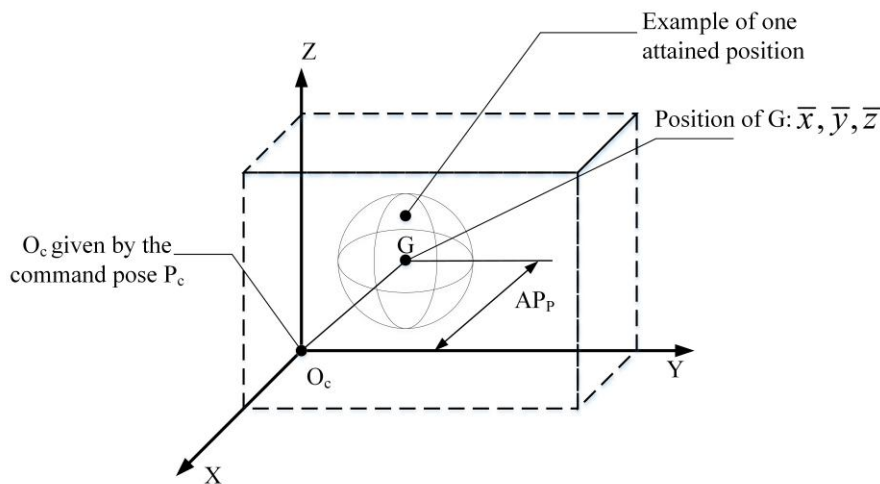


Figure 6. Positioning accuracy.

Thus, the positioning accuracy can be calculated as

$$AP_P = \sqrt{(\bar{x} - x_c)^2 + (\bar{y} - y_c)^2 + (\bar{z} - z_c)^2} \tag{23}$$

with

$$\bar{x} = \frac{1}{n} \sum_{j=1}^n x_j, \bar{y} = \frac{1}{n} \sum_{j=1}^n y_j, \bar{z} = \frac{1}{n} \sum_{j=1}^n z_j \tag{24}$$

where  $\bar{x}, \bar{y}, \bar{z}$  are the coordinates of the barycenter of the cluster of points obtained after executing the same pose  $n$  times,  $x_c, y_c, z_c$  are the coordinates of the command pose, and  $x_j, y_j, z_j$  are the coordinates of the  $j$ -th attained pose along the respective axes.

Analogously, the distance accuracy expresses the deviation in positioning and orientation between the command distance and mean of the attained distances, as illustrated in Figure 7.



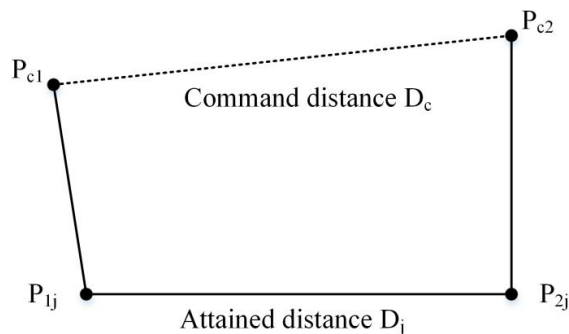


Figure 7. Distance accuracy.

The distance accuracy can be calculated as

$$AP_d = \bar{D} - D_c \tag{25}$$

with

$$\bar{D} = \frac{1}{n} \sum_{j=1}^n D_j \tag{26}$$

$$D_j = |P_{1j} - P_{2j}| = \sqrt{(x_{1j} - x_{2j})^2 + (y_{1j} - y_{2j})^2 + (z_{1j} - z_{2j})^2} \tag{27}$$

$$D_c = |P_{c1} - P_{c2}| = \sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2 + (z_{c1} - z_{c2})^2} \tag{28}$$

We use both the distance accuracy and positioning accuracy to quantify error compensation.

#### 4.1. Simulations

To perform simulations, we added random parameter errors to the theoretical robot kinematic parameters as actual parameters and non-kinematic errors as the compensation of the nominal joint variables of the robot. The actual and theoretical end-effector position coordinates of the robot were calculated using the POE model.

We applied 1000 pairs of joint variables and position coordinates to optimize the thresholds, initial weights, and number of hidden neurons in the proposed network using differential evolution. The same 1000 samples were applied to train the optimized neural network, and 100 command points were employed for verification of the prediction accuracy of the trained neural network. In addition, the robot kinematic parameters were calibrated using the POE model and LM algorithm for comparison by applying the same samples. The compensation results are shown in Figures 8 and 9.

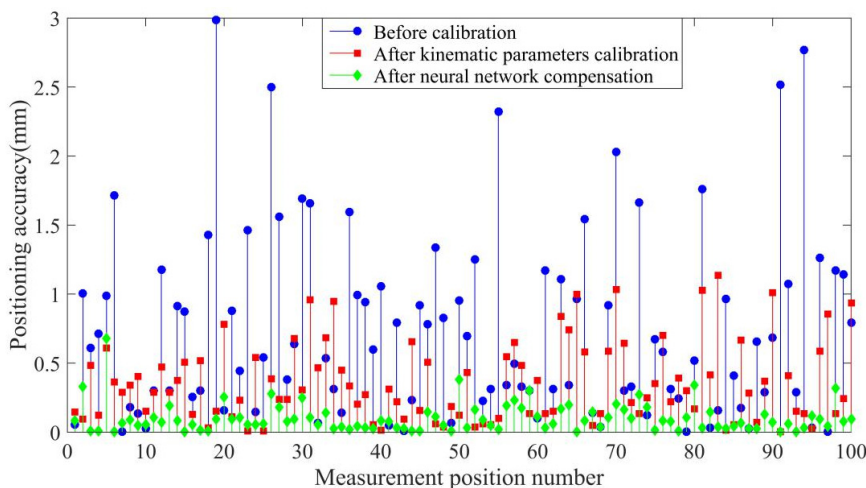
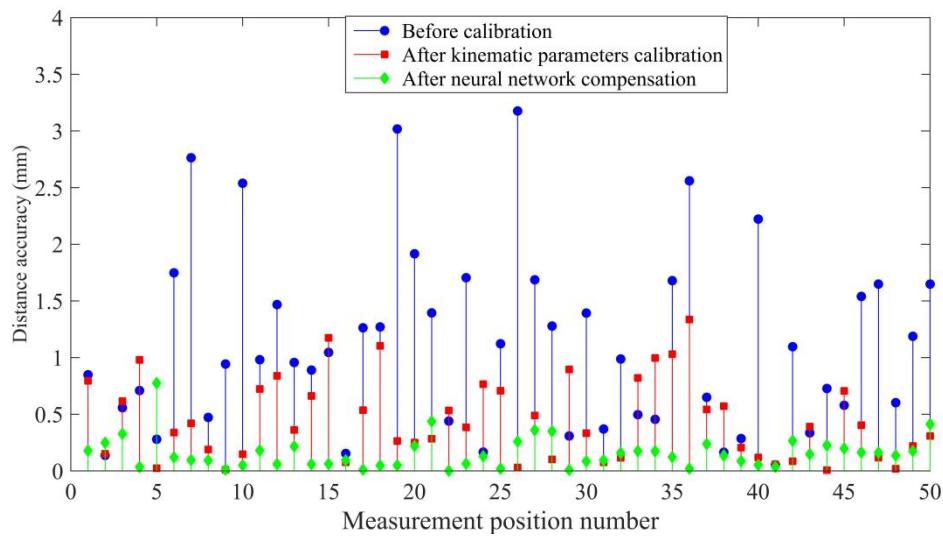


Figure 8. Positioning accuracy of the simulations.



**Figure 9.** Distance accuracy of the simulations.

Compared to the calibration method, the proposed neural network optimized using differential evolution provides better compensation. The maximum distance accuracy is improved from 3.1760 mm to 0.7743 mm, and the average distance accuracy is improved from 1.1118 mm to 0.1564 mm by using the proposed network. The maximum positioning accuracy is improved from 3.1760 mm to 0.7743 mm and the average positioning accuracy is improved from 0.7411 mm to 0.1007 mm by using the proposed algorithm.

As summarized in Tables 1 and 2, the proposed method has the best absolute positioning accuracy and distance accuracy.

**Table 1.** Positioning accuracy of the simulations.

Positioning Accuracy (mm)	Mean Error	Max Error
Before calibration	0.7411	2.9500
After kinematic parameter calibration	0.3611	1.1379
After neural network compensation	0.1007	0.6803

**Table 2.** Distance accuracy of the simulations.

Distance Accuracy (mm)	Mean Error	Max Error
Before calibration	1.1180	3.1760
After kinematic parameter calibration	0.4461	1.3360
After neural network compensation	0.1564	0.7743

#### 4.2. Experiment

The calibration system, as shown in Figure 10, consists of a six-DOF serial robot (Universal Robot 5, Universal Robots), a Laser Tracker (API T3, Automated Precision Inc., Maryland, USA) with an accuracy of 0.005 mm/m, and an accompanying laser reflector. The reflector was fixed at an assigned location on the robot end-effector. The robot moved within the workspace and spatial position coordinates of the end-effector data were collected using a laser tracker as the output of the neural network. Additionally, the corresponding joint angles of the sampling points were recorded as the input of the neural network.

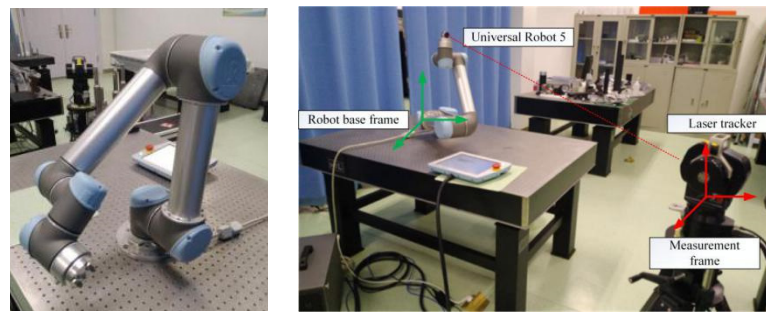


Figure 10. The experimental setup.

We measured 600 samples in the workspace of the robot for optimization and training of the proposed network. The same samples were used for calibration of kinematic parameters using the POE model and LM algorithm. The samples were selected such that the corresponding joint variables covered the robot joint working ranges, as shown in Figure 11. In addition, 100 command points in the robot workspace were randomly selected to verify the accuracy of the calibration methods and proposed network.

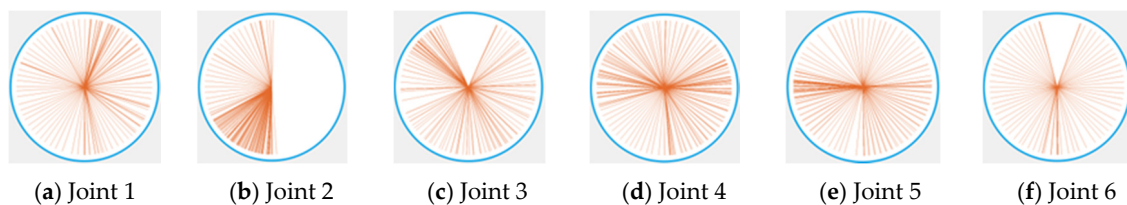


Figure 11. Angular distribution covering the joint space.

Figure 12 shows the robot base and measurement coordinate systems considered in the experiment. Gan et al. presented a simple but effective calibration of the relative rotation matrix and translation vector for converting the base frames of two coordinate systems using quaternions [29]. We used this method to determine the transformation between the measurement and robot base frames. The standard position coordinates measured using the laser tracker can be expressed into the robot base coordinate system. Therefore, we conducted the prediction and training of the proposed network by integrating robot joint variables with exact positioning with respect to the robot base.

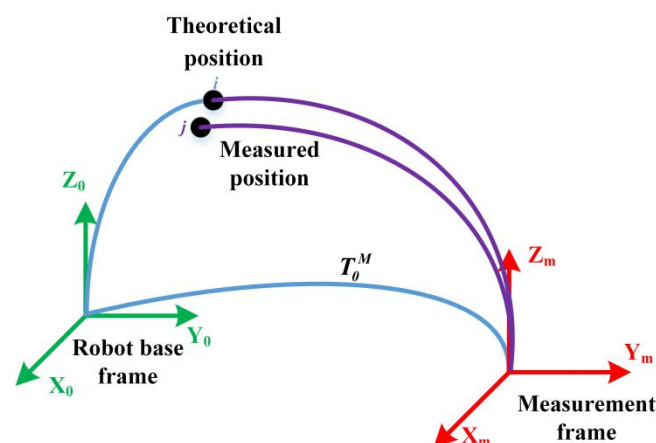


Figure 12. Experiment coordinate systems considering the measurement and robot base.

Figures 13 and 14 and Tables 3 and 4 show the experimental results, with the optimized network accurately predicting the position coordinates. Before calibration, the maximum positioning accuracy is 3.1760 mm and the maximum distance accuracy 1.1118 mm, which is affected by kinematic and

non-kinematic errors. After calibration, using the POE model and the LM algorithm, the positioning accuracy and distance accuracy of the robot were significantly improved, which is due to the reduction in kinematic errors through calibration. However, neglecting error sources such as temperature variations and gear backlash limits the effectiveness and accuracy of robot calibration. Therefore the proposed network provides the best absolute positioning accuracy and distance accuracy because the proposed network mitigates the effects of kinematic and non-kinematic errors. After error compensation, the maximum positioning accuracy is improved from 3.1760 mm to 0.7743 mm and the average positioning accuracy is improved from 1.1118 mm to 0.1564 mm by using the proposed algorithm. The maximum distance accuracy is improved from 2.0140 mm to 0.2392 mm and the average distance accuracy is improved from 0.8497 mm to 0.0493 mm.

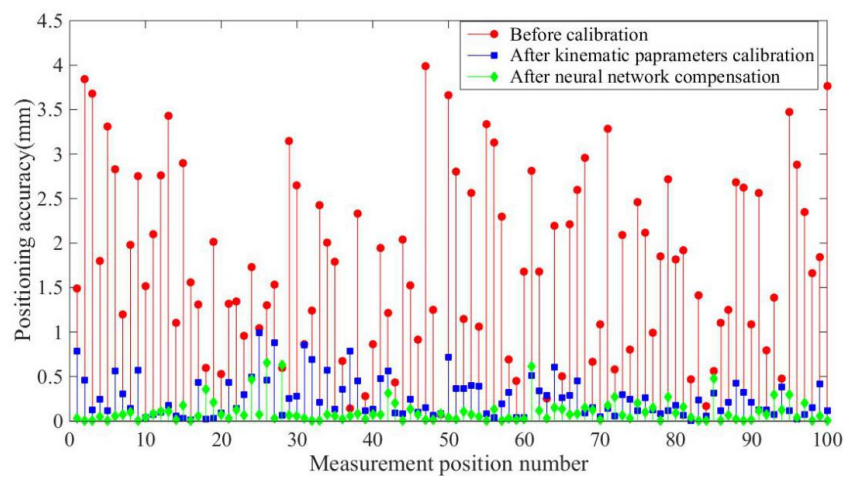


Figure 13. Positioning accuracy of the experiments.

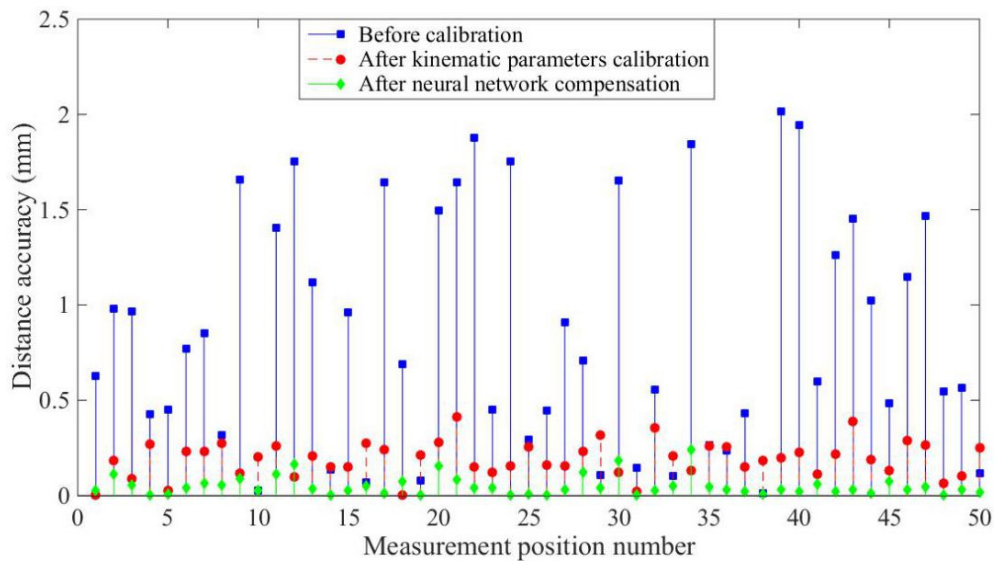


Figure 14. Distance accuracy of the experiments.

Table 3. Positioning accuracy of the experiments.

Positioning Accuracy (mm)	Mean Error	Max Error
Before calibration	1.7730	3.9930
After kinematic parameter calibration	0.2595	0.9877
After neural network compensation	0.1041	0.6559

**Table 4.** Distance accuracy of the experiments.

Distance Accuracy (mm)	Mean Error	Max Error
Before calibration	0.8497	2.0140
After kinematic parameter calibration	0.1915	0.4113
After neural network compensation	0.0493	0.2392

## 5. Discussion and Conclusions

The absolute positioning accuracy of robots has become increasingly important to support their widespread applications, particularly when offline programming is required. Error sources in robots can be classified into kinematic and non-kinematic errors. Conventional calibration can reduce the influence of kinematic errors on positioning accuracy, but the complexity of non-kinematic error sources and modeling hinders the compensation of such errors.

We applied a neural network to compensate both kinematic and non-kinematic errors with the aim of improving the absolute positioning accuracy of robot manipulators. The proposed network can avoid complex modeling while comprehensively considering the influence of all error sources. As the performance of a neural network is influenced by its structure and parameters, we optimized the proposed network using differential evolution. The thresholds, initial weights, and number of hidden neurons in the network are optimized using differential evolution to improve efficiency and performance. Using the proposed network, the effects of kinematic, and non-kinematic errors decreased, thus improving the absolute positioning accuracy of an industrial robot.

The theoretical correctness and effectiveness of the proposed method are verified by simulations and experiments on a six-DOF serial robot and compared with calibration using the POE model and LM algorithm. The absolute positioning accuracy improves using the proposed network, and the experimental results verify the correctness and effectiveness of the network. After calibration, the robot average distance accuracy improved from 0.8497 mm before calibration to 0.04933 mm. Likewise, the robot average positioning accuracy improved from 3.176 mm before calibration to 0.7743 mm.

**Author Contributions:** Y.J. built the error model; Y.J. and L.Y. conceived of and designed the experiment; H.X. designed the computer programs; H.Z. contributed materials and tools; H.J. analyzed the data; Y.J. and L.Y. wrote the paper. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by National Science Foundation of China (Grant No: 51875165 and 51805039) and 111 Project (Grant No: B12019). The authors would like to thank all other members of the research team for their contributions to this research.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Nguyen, H.N.; Zhou, J.; Kang, H.J.; Ro, Y.S. Robot geometric parameter identification with extended Kalman filtering algorithm. In Proceedings of the International Conference on Intelligent Computing 2013, Nanning, China, 28–31 July 2013; pp. 165–170.
2. Veitschegger, W.; Wu, C.H. Robot accuracy analysis based on kinematics. *IEEE J. Robot. Autom.* **1986**, *2*, 171–179. [[CrossRef](#)]
3. Judd, R.; Knasinski, A. A technique to calibrate industrial robots with experimental verification. In Proceedings of the 1987 IEEE International Conference on Robotics and Automation, Raleigh, NC, USA, 31 March–3 April 1987; pp. 351–357.
4. Elatta, A.Y.; Li, P.G.; Fan, L.Z.; Yu, D.Y.; Luo, F. An overview of robot calibration. *Inform. Technol. J.* **2004**, *3*, 74–78.
5. Renders, J.M.; Rossignol, E.; Becquet, M. Kinematic calibration and geometrical parameter identification for robots. *IEEE Trans. Rob. Autom.* **1991**, *7*, 721–732. [[CrossRef](#)]
6. Wu, Y.; Klimchik, A.; Caro, S.; Furet, B.; Pashkevich, A. Geometric calibration of industrial robots using enhanced partial pose measurements and design of experiments. *Robot. Rob. Comput. Integr. Manuf.* **2015**, *35*, 151–168. [[CrossRef](#)]

7. Joubair, A.; Bonev, I.A. Kinematic calibration of a six-axis serial robot using distance and sphere constraints. *Int. J. Adv. Manuf. Technol.* **2015**, *77*, 515–523. [[CrossRef](#)]
8. He, R.; Zhao, Y.; Yang, S.; Yang, S. Kinematic-parameter identification for serial-robot calibration based on POE formula. *IEEE Trans. Robot.* **2010**, *26*, 411–423.
9. Conte, J.; Majarena, A.C.; Aguado, S.; Acero, R.; Santolaria, J. Calibration strategies of laser trackers based on network measurements. *Int. J. Adv. Manuf. Technol.* **2016**, *83*, 1161–1170. [[CrossRef](#)]
10. Long, P.; Khalil, W.; Caro, S. Kinematic and dynamic analysis of lower-mobility cooperative arms. *Robotica* **2015**, *33*, 1813–1834. [[CrossRef](#)]
11. Meng, Y.; Zhuang, H. Autonomous robot calibration using vision technology. *Robot. Rob. Comput. Integr. Manuf.* **2007**, *23*, 436–446. [[CrossRef](#)]
12. Yang, X.; Wu, L.; Li, J.; Chen, K. A minimal kinematic model for serial robot calibration using poe formula. *Robot. Rob. Comput. Integr. Manuf.* **2014**, *30*, 326–334. [[CrossRef](#)]
13. Liu, Y.; Wu, J.; Wang, L. Parameter identification algorithm of kinematic calibration in parallel manipulators. *Adv. Mech. Eng.* **2016**, *8*, 1–16. [[CrossRef](#)]
14. Yang, Z.; Cheon, S.U.; Yang, J. A kinematic calibration method of a 3-dof secondary mirror of the giant magellan telescope based on least square algorithm. *J. Mech. Sci. Technol.* **2013**, *27*, 3779–3786. [[CrossRef](#)]
15. Majarena, A.C.; Santolaria, J.; Samper, D. Analysis and evaluation of objective functions in kinematic calibration of parallel mechanisms. *Int. J. Adv. Manuf. Technol.* **2013**, *66*, 751–761. [[CrossRef](#)]
16. Ma, L.; Bazzoli, P.; Sammons, P.M.; Landers, R.G.; Bristow, D.A. Modeling and calibration of high-order joint-dependent kinematic errors for industrial robots. *Robot. Rob. Comput. Integr. Manuf.* **2018**, *50*, 153–167. [[CrossRef](#)]
17. Whitney, D.E.; Lozinski, C.A.; Rourke, J.M. Industrial robot forward calibration method and results. *J. Dyn. Sys. Meas. Control* **1986**, *108*, 1–8. [[CrossRef](#)]
18. Chen, J.; Chao, L.M. Positioning error analysis for robot manipulators with all rotary joints. *IEEE J. Robot. Autom.* **1987**, *3*, 539–545. [[CrossRef](#)]
19. Nguyen, H.N.; Zhou, J.; Kang, H.J. A calibration method for enhancing robot accuracy through integration of an extended Kalman filter algorithm and an artificial neural network. *Neurocomputing* **2015**, *151*, 996–1005. [[CrossRef](#)]
20. Ding, W.; Gu, J.; Tang, S. Development of a calibrating algorithm for Delta Robot's visual positioning based on artificial neural network. *Optik* **2016**, *127*, 9095–9104. [[CrossRef](#)]
21. Jang, J.H.; Kim, S.H.; Kwak, Y.K. Calibration of geometric and non-geometric errors of an industrial robot. *Robotica* **2001**, *19*, 305–701. [[CrossRef](#)]
22. Wang, D.; Bai, Y. Improving position accuracy of robot manipulators using neural networks. In Proceedings of the 2005 IEEE Instrumentation and Measurement Technology Conference, Ottawa, ON, Canada, 16–19 May 2005; pp. 1524–1526.
23. Rouhani, M.; Javan, D.S. Two fast and accurate heuristic RBF learning rules for data classification. *Neural Netw.* **2016**, *75*, 150–161. [[CrossRef](#)]
24. Feng, H.M. Self-generation RBFNs using evolutionary PSO learning. *Neurocomputing* **2006**, *70*, 241–251. [[CrossRef](#)]
25. González, J.; Rojas, I.; Ortega, J.; Pomares, H.; Fernandez, F.J.; Díaz, A.F. Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. *IEEE Trans. Neural Netw.* **2003**, *14*, 1478–1495. [[CrossRef](#)] [[PubMed](#)]
26. He, R.; Li, X.; Shi, T. A kinematic calibration method based on the product of exponentials formula for serial robot using position measurements. *Robotica* **2014**, *33*, 1–19. [[CrossRef](#)]
27. Noman, N.; Iba, H. Accelerating Differential Evolution Using an Adaptive Local Search. *IEEE Trans. Evol. Comput.* **2008**, *12*, 107–125. [[CrossRef](#)]
28. ISO 9283. *Manipulating Industrial Robots Performance Criteria and Related Test Methods*; International Organization for Standardization: Geneva, Switzerland, 1998.
29. Gan, Y.; Dai, X. Base frame calibration for coordinated industrial robots. *Rob. Autom. Syst.* **2011**, *59*, 563–570. [[CrossRef](#)]

