



# A Tandem Robotic Arm Inverse Kinematic Solution Based on an Improved Particle Swarm Algorithm

Guojun Zhao<sup>1,2</sup>, Du Jiang<sup>1,3\*</sup>, Xin Liu<sup>1,2\*</sup>, Xiliang Tong<sup>1\*</sup>, Ying Sun<sup>1,2\*</sup>, Bo Tao<sup>1,4</sup>, Jianyi Kong<sup>1,2,3</sup>, Juntong Yun<sup>2,4</sup>, Ying Liu<sup>2,4</sup> and Zifan Fang<sup>5\*</sup>

<sup>1</sup>Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, Wuhan, China, <sup>2</sup>Research Center for Biomimetic Robot and Intelligent Measurement and Control, Wuhan University of Science and Technology, Wuhan, China, <sup>3</sup>Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan, China, <sup>4</sup>Precision Manufacturing Research Institute, Wuhan University of Science and Technology, Wuhan, China, <sup>5</sup>Hubei Key Laboratory of Hydroelectric Machinery Design & Maintenance, China Three Gorges University, Yichang, China

## OPEN ACCESS

### Edited by:

Tinggui Chen,  
Zhejiang Gongshang University, China

### Reviewed by:

Jianhua Zhang,  
Tianjin University of Technology, China  
Ting Wang,  
Shandong University of Science and  
Technology, China  
Mihai Crenganis,  
Lucian Blaga University of Sibiu,  
Romania

### \*Correspondence:

Du Jiang  
jiangdu@wust.edu.cn  
Xin Liu  
liuxin3058@wust.edu.cn  
Ying Sun  
sunying65@wust.edu.cn  
Xiliang Tong  
tongxiliang@wust.edu.cn  
Zifan Fang  
zxf@ctgu.edu.cn

### Specialty section:

This article was submitted to  
Bionics and Biomimetics,  
a section of the journal  
Frontiers in Bioengineering and  
Biotechnology

**Received:** 10 December 2021

**Accepted:** 24 February 2022

**Published:** 19 May 2022

### Citation:

Zhao G, Jiang D, Liu X, Tong X, Sun Y,  
Tao B, Kong J, Yun J, Liu Y and Fang Z  
(2022) A Tandem Robotic Arm Inverse  
Kinematic Solution Based on an  
Improved Particle Swarm Algorithm.  
Front. Bioeng. Biotechnol. 10:832829.  
doi: 10.3389/fbioe.2022.832829

The analysis of robot inverse kinematic solutions is the basis of robot control and path planning, and is of great importance for research. Due to the limitations of the analytical and geometric methods, intelligent algorithms are more advantageous because they can obtain approximate solutions directly from the robot's positive kinematic equations, saving a large number of computational steps. Particle Swarm Algorithm (PSO), as one of the intelligent algorithms, is widely used due to its simple principle and excellent performance. In this paper, we propose an improved particle swarm algorithm for robot inverse kinematics solving. Since the setting of weights affects the global and local search ability of the algorithm, this paper proposes an adaptive weight adjustment strategy for improving the search ability. Considering the running time of the algorithm, this paper proposes a condition setting based on the limit joints, and introduces the position coefficient  $k$  in the velocity factor. Meanwhile, an exponential product form modeling method (POE) based on spinor theory is chosen. Compared with the traditional DH modeling method, the spinor approach describes the motion of a rigid body as a whole and avoids the singularities that arise when described by a local coordinate system. In order to illustrate the advantages of the algorithm in terms of accuracy, time, convergence and adaptability, three experiments were conducted with a general six-degree-of-freedom industrial robotic arm, a PUMA560 robotic arm and a seven-degree-of-freedom robotic arm as the research objects. In all three experiments, the parameters of the robot arm, the range of joint angles, and the initial attitude and position of the end-effector of the robot arm are given, and the attitude and position of the impact point of the end-effector are set to verify whether the joint angles found by the algorithm can reach the specified positions. In Experiments 2 and 3, the algorithm proposed in this paper is compared with the traditional particle swarm algorithm (PSO) and quantum particle swarm algorithm (QPSO) in terms of position and direction solving accuracy, operation time, and algorithm convergence. The results show that compared with the other two algorithms, the algorithm proposed in this paper can ensure higher position accuracy and orientation accuracy of the robotic arm end-effector. The position error of the algorithm proposed in this paper is 0 and the

maximum orientation error is  $1.29 \times 10^{-8}$ , while the minimum position error of the other two algorithms is  $-1.64 \times 10^{-5}$  and the minimum orientation error is  $-4.03 \times 10^{-6}$ . In terms of operation time, the proposed algorithm in this paper has shorter operation time compared with the other two algorithms. In the last two experiments, the computing time of the proposed algorithm is 0.31851 and 0.30004s respectively, while the shortest computing time of the other two algorithms is 0.33359 and 0.30521s respectively. In terms of algorithm convergence, the proposed algorithm can achieve faster and more stable convergence than the other two algorithms. After changing the experimental subjects, the proposed algorithm still maintains its advantages in terms of accuracy, time and convergence, which indicates that the proposed algorithm is more applicable and has certain potential in solving the multi-arm inverse kinematics solution. This paper provides a new way of thinking for solving the multi-arm inverse kinematics solution problem.

**Keywords:** particle swarm algorithm, joint limiting, adaptive strategy, spinor theory, robot inverse kinematics solution

## 1 INTRODUCTION

For the trajectory planning as well as control of the robotic arm, its inverse kinematic solution is the key. The inverse kinematic solution can directly affect the control accuracy and the success of trajectory planning of the robot arm. However, the process of solving the inverse kinematic solution for the robot arm is not only tedious, but also impossible to solve. The emergence of bionic intelligent algorithms provides new ideas for solving inverse kinematics solutions. By transforming the tedious inverse kinematics solution process into an optimization problem with minimum value, it not only simplifies the solution process, but also improves the solution efficiency. The particle swarm algorithm is more advantageous in terms of accuracy, speed and applicability at the level of robot inverse kinematics solution due to its simple programming and easy implementation. Therefore, this paper selects the particle swarm algorithm and further improves it for solving the robot inverse kinematics solution.

In this paper, an inverse kinematic solution method based on an improved particle swarm algorithm is proposed for the inverse kinematic solution of an arbitrary tandem robotic arm, with the following innovative points.

- 1) A particle swarm algorithm is introduced to solve the inverse kinematic solution of the tandem multi-degree-of-freedom robotic arm, which transforms the inverse kinematic solution process of the robotic arm into a multi-objective optimization problem and gives a suitable fitness function based on the inverse kinematic problem.
- 2) In this paper, an exponential product form modeling method (POE) based on spinor theory is chosen. Compared with the traditional DH modeling method, the spinor approach describes the motion of a rigid body as a whole and avoids the singularities that arise when described by a local coordinate system.
- 3) This paper proposes a condition setting based on limit joints and introduces a position factor  $k$  in the velocity factor. The

reasonable condition setting provides a reference standard for the initialization of position and velocity, and reduces the running time of the algorithm at the same time. Among them, the operation time of the algorithm proposed in this paper is 0.31851 and 0.30004s in the second as well as the third experiment, while the shortest operation time of the other two algorithms is 0.33359 and 0.30521s, respectively.

- 4) An adaptive weight adjustment strategy is proposed to improve the stable search capability of the algorithm.
- 5) Three experiments are designed to illustrate the solution accuracy, operation time, and convergence of the algorithm. The experimental objects include: general six-degree-of-freedom robotic arm, PUMA560 robotic arm, and 7-degree-of-freedom robotic arm. The experimental method is: setting the position of the impact point and the attitude of the end-effector of the robotic arm, by bringing the relevant parameters of the robotic arm, the value range of the joint angle, the initial attitude and position of the end-effector of the robotic arm into the algorithm, finding the joint angle that meets the conditions, and finally getting the actual position and attitude of the end-effector of the robotic arm. The comparison algorithms include: the algorithm proposed in this paper, the traditional particle swarm algorithm (PSO), and the quantum particle swarm algorithm (QPSO). The comparison is done by comparing the error of the actual position and attitude of the robotic arm end-effector with the position and attitude of the given impact point, comparing the computing time of a set of data, and comparing the convergence of different algorithms according to the change of the fitness function with the number of iterations in a set of data. The results show that compared with the other two algorithms, the algorithm proposed in this paper can ensure higher position accuracy and orientation accuracy of the robotic arm end-effector. The position error of the algorithm proposed in this paper is 0 and the maximum orientation error is  $1.29 \times 10^{-8}$ , while the minimum position error of the other two algorithms is  $-1.64 \times 10^{-5}$  and the minimum orientation error is  $-4.03 \times 10^{-6}$ . In terms of

operation time, the proposed algorithm in this paper has shorter operation time compared with the other two algorithms. In the last two experiments, the computing time of the proposed algorithm is 0.31851 and 0.30004s respectively, while the shortest computing time of the other two algorithms is 0.33359 and 0.30521s respectively. In terms of algorithm convergence, the proposed algorithm can achieve faster and more stable convergence than the other two algorithms.

The rest of this paper is described as follows. **Section 2** introduces the current domestic and foreign methods for the inverse kinematics solution of robotic arms, the improvement methods of particle swarm algorithms, and the advantages and improvements of particle swarm algorithms for the inverse kinematics solution of robotic arms. **Section 3** takes a general six-degree-of-freedom industrial robotic arm as the research object and analyzes its positive kinematics based on the spin volume theory, and also gives the specific calculation steps. **Section 4** introduces two particle swarm optimization algorithms, explains the implementation steps of the general particle swarm algorithm, and illustrates the improvements of the algorithm compared with other improved particle swarm algorithms. **Section 5** sets the experimental conditions and gives the specific form of the fitness function, the flowchart of the algorithm, and the pseudo-code. **Section 6** conducts three experiments for different research objects, shows the simulation results under different conditions, and compares this algorithm with other algorithms in terms of solution accuracy, operation time, and convergence. Finally, the discussion and conclusion sections of this paper are presented.

## 2 RELATED WORK

The main types of robot inverse kinematics solutions are analytical, numerical, geometric, and intelligent algorithms. The analytical method is mainly used to solve the robotic arm with a definite configuration, i.e., a robotic arm that satisfies the “Pieper” criterion - three joint axes intersecting at one point (Tong et al., 2021; Liu F. et al., 2021; Xiao et al., 2021; Chen et al., 2021a). When the criterion is satisfied, the joint angles of the robotic arm have a definite analytical solution form. The advantage of the analytical method is that it is fast to solve, and the disadvantage is that it has a more limited use. The numerical method has a wider application compared to the analytical method, but the solution speed is slow and there are numerical stability problems. The numerical method is based on the Jacobi matrix and approximates the optimal solution by numerical iteration. The geometric method has a narrower application than the analytical method. This method solves the inverse kinematic solution of the robotic arm mainly by the geometric configuration of the robotic arm. The robotic arm that does not satisfy the “Pieper” criterion can often be solved by the geometric method, and specific applications include the inverse kinematic solution of the three subproblems of “Paden-Kahan” based on the rotation theory (Wang et al., 2021; Weng et al., 2021;

Duan et al., 2021; Cheng et al., 2021). The advantage of the geometric method is that it has a clear geometric meaning and can solve some problems that cannot be solved by the analytical method. Compared with the first two methods, the intelligent algorithm does not involve the inverse kinematic solution part, and solves the inverse kinematic solution mainly by deriving the end position change matrix based on the positive kinematics of the robot, and finally approximating the correct joint angle gradually by randomly generating the joint angle values and error functions to achieve the solution of the inverse kinematic solution. Intelligent algorithms tend to avoid some of the problems present in the process of solving conventional inverse kinematics solutions, such as the existence of singularities when the Jacobi determinant is zero, which cannot be solved (Kucuk and Bingul, 2014; Liao et al., 2021; Jiang et al., 2021a; Liu X. et al., 2021). Therefore it is of great research significance for the study of intelligent algorithms. The existing intelligent algorithms are artificial neural networks, adaptive neuro-fuzzy inference systems, and genetic algorithms, particle swarm search algorithms, etc. (EI-Sherbiny et al., 2018; Huang et al., 2021; Tao et al., 2021; Hao et al., 2021a).

Particle swarm algorithm is widely used as an intelligent algorithm compared to other algorithms because of its simple programming and easy implementation, as well as its better final solution. Many researchers have improved the particle swarm algorithm. Netjinda et al. optimized the search diversity of PSO algorithm by re-updating the position and velocity based on the principle of bird flock frightening (Netjinda et al., 2015; Jiang et al., 2021b; Ma et al., 2020; Sun et al., 2020a). Yang et al. improve the convergence of the algorithm generate the initial population by Halton sequence and adjust the inertia weights based on the variation property of the nonlinear function (Yang et al., 2015; Sun et al., 2020b; Luo et al., 2020; Sun et al., 2020c). Harrison et al. further study on the inertia weight adjustment strategy of particle swarm algorithm (Harrison et al., 2016; Li et al., 2020; Tan et al., 2020; Huang et al., 2020). Chen et al. proposed a double cluster and double layer structure, with the best particles as the top layer and all particles as the bottom layer to improve the search ability and efficiency of the algorithm (Chen et al., 2017; He et al., 2019; Jiang et al., 2019a; Chen et al., 2021b). Tanweer et al. divided the particle swarm into three groups, each with a different speed configuration. The speed of each group is configured with a different update strategy so that the algorithm achieves faster convergence and higher accuracy (Tanweer et al., 2016; Huang et al., 2019; Jiang et al., 2019b; Yu et al., 2019). Li et al. combine the particle swarm optimization algorithm with the artificial bee colony algorithm to improve the search capability and convergence speed of the algorithm (Li et al., 2015a; Sun et al., 2021; Tao et al., 2022a; Li et al., 2019b; Jiang et al., 2019c). Aydilek combined particle swarm optimization algorithm with firefly algorithm to improve the running time and convergence accuracy of the algorithm (Aydilek, 2018; Liu et al., 2021c; Liu et al., 2021d; Bai et al., 2022). Ngo et al. proposed a particle movement strategy for overcoming the situation that traditional particle swarm algorithms converge too early and fall into local optimum (Ngo et al., 2016; Liu et al., 2022d; Yang et al., 2021; Mao et al., 2017). Thangaraj et al. summarized the fusion

algorithm of particle swarm algorithm with various other intelligent algorithms and conducted an experimental comparison (Thangaraj et al., 2011; Yang et al., 2021; Hao et al., 2021b; Tao et al., 2022b). Wei et al. proposed an adaptive two-layer particle swarm algorithm based on learning strategy by dividing the population into two parts (Lim and Mat Isa, 2014; Liu et al., 2022a; Yun et al., 2022; Liu et al., 2022b). Taherkhani et al. determines the inertia weight for each position based on the distance of each particle's performance from the optimal position and ultimately improves the solution quality as well as the convergence speed (Taherkhani and Safabakhsh, 2016; Cheng et al., 2020; Wu et al., 2022; Yu et al., 2020).

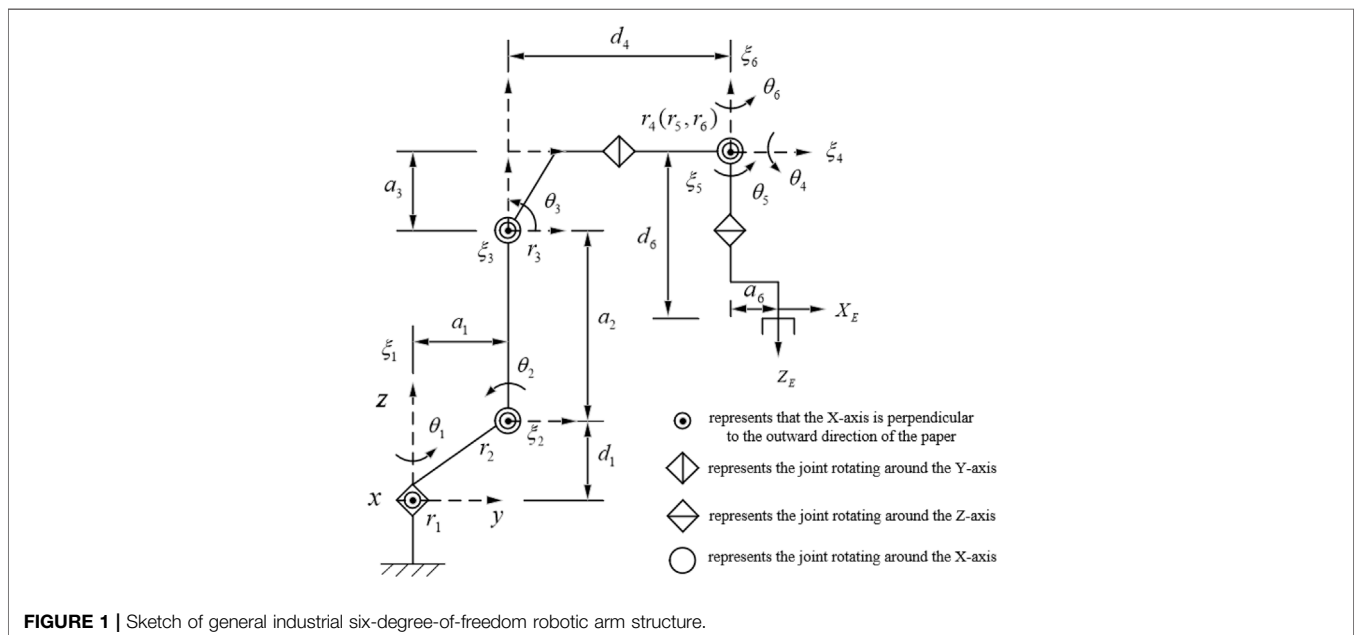
For robot inverse kinematics solution solving, particle swarm algorithm is more advantageous in terms of accuracy, speed and applicability. Ayyildiz et al. compared genetic algorithm, particle swarm algorithm, quantum particle swarm algorithm and gravitational search algorithm for solving robot inverse kinematics solution and finally found that particle swarm algorithm has higher accuracy compared to other algorithms (Ayyildiz and Centinkaya, 2016). Dereli et al. proposed an improved PSO algorithm which discarded the traditional position and velocity update approach and chose to use a quantum mechanics based position update approach for solving the seven degree of freedom robotic arm inverse kinematics solution (Dereli and Koker, 2020). Liu et al. proposed an improved PSO algorithm for simultaneous optimization of multiple populations to enhance the search capability during population iteration (Liu F. et al., 2021; Chen et al., 2022; Sun et al., 2020d). Deng et al. proposed an adaptive particle swarm algorithm by improving the learning factor, adopting an adaptive weighting strategy, and proposing a special boundary handling method thereby optimizing the case where the particles fall into local optima (Deng and Xie, 2021; Xu et al., 2022; Sun et al., 2020a). Dereli et al. changed the fixed

weights to variable random weights, while applying the improved PSO algorithm to the estimation of the end position of a seven-degree-of-freedom redundant robotic arm, ultimately improving its solution accuracy (Dereli and Koker, 2018). Pathak et al. proposed a bi-directional particle swarm optimization algorithm for solving the optimization problem of the inverse kinematic solution of a robotic arm (Pathak et al., 2019; Li et al., 2015b). Liu et al. proposed a new parallel learning particle swarm optimization algorithm (PLPSO) that divides the original population into two independently evolving subpopulations. The algorithm was compared with other algorithms and tested for the UR5 robotic arm, which finally showed the good performance of the algorithm (Liu et al., 2022c; Yiyang et al., 2021; Sun et al., 2018). Shastri et al., 2019 combined neural network with particle swarm algorithm to solve the robotic arm inverse kinematics solution from the operation time and complexity level (Shastri et al., 2019; Li et al., 2019c; Liu et al., 2022d; Li et al., 2019a).

### 3 MODEL AND KINEMATIC ANALYSIS

#### 3.1 Robotic Arm Model

In this paper, a common six-degree-of-freedom robotic arm in industry is selected as the research object, and its structural sketch is shown in **Figure 1**. The robotic arm shown in **Figure 1** is a six-degree-of-freedom tandem robotic arm, and all six joints are rotating joints. Its structure is characterized by the first three joints not intersecting, the second and third joint axes are parallel and anisotropic to the first joint axis, and the fourth, fifth and sixth joints intersect at a point, satisfying Pieper's criterion, so there exists an inverse kinematic closure solution. Considering that the process of establishing the coordinate system by DH kinematic modeling method is too complicated, not only the



world coordinate system needs to be established, but also the relative coordinate system between joints and joints, and the form of the final solution obtained is often inconsistent for different DH modeling methods. While based on the spinor theory only needs to establish a world coordinate system, which not only optimizes the modeling process, but also has good geometric meaning. Therefore, in this paper, we choose to establish the joint coordinate system based on the Lie group and spinor theory (Wang et al., 2018; Liao et al., 2020; Sun and Zhao, 2022; Zhang et al., 2022).

### 3.2 Kinematic Analysis

Compared with the traditional DH modeling method, the spinor approach describes the motion of a rigid body as a whole, avoiding the singularities that arise when described by a local coordinate system. One of the positive kinematic modeling processes based on the spinor theory is shown below.

The coordinates of a rigid body in space can be expressed as the transformation of the pose of the rigid body with respect to the base coordinate system as well as the transformation of the position. The specific representation is shown in Eq. 1.

$$SE(3) = \left\{ \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}; R \in R^{3 \times 3}, t \in R^3, R^T R = I, \det(R) = 1, \right\} \quad (1)$$

where R is a 3-by-3 matrix representing the pose transformation of the rigid body with respect to the base coordinate system. t is a 3-by-1 column vector representing the position transformation of the rigid body with respect to the base coordinate system. Chasles' theorem proves that the rigid body motion of any object from one position pattern to another can be realized by a compound rotation around a certain line and a movement along that line, and that the compound motion is called spiral motion, whose infinitesimal quantity is the element of the Lie algebra, i.e., the kinematic spin. the Lie algebra of SE(3) is denoted as se(3), where the elements are defined as follows:

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} = [\omega^T; v^T]^T \in R^6 \quad (2)$$

where  $\omega$  denotes the angular velocity of rotation of the rigid body around the rotation axis.  $v$  denotes the corresponding translational velocity of the rigid body. According to the rotation theory, if the spiral motion of a rigid body is known, the linkage position and attitude transformation matrix can be expressed in the form of an exponential product (POE). The specific form is shown in Eq. 3.

$$e^{\xi\theta} = \begin{cases} \begin{bmatrix} e^{\hat{\omega}\theta} & (I - e^{\hat{\omega}\theta})(\omega \times v) + \theta\omega\omega^T v \\ 0 & 1 \end{bmatrix}, \omega \neq 0 \\ \begin{bmatrix} I & \theta v \\ 0 & 1 \end{bmatrix}, \omega = 0 \end{cases} \quad (3)$$

The above equation represents the posture transformation matrix corresponding to the joint angle when the rigid body is in spiral motion. where  $e^{\hat{\omega}\theta}$ ,  $I$  denotes attitude change and

$(I - e^{\hat{\omega}\theta})(\omega \times v) + \theta\omega\omega^T v$ ,  $\theta v$  denotes position change.  $\omega$  denotes the angular velocity of rotation of the rigid body around the rotation axis.  $v$  denotes the corresponding translational velocity of the rigid body.  $\theta$  indicates the joint angle. The positive kinematic expression of the robot is obtained by multiplying the initial positional matrix with the transformation matrix corresponding to each joint angle, provided that the initial position, pose and each joint angle of the end-effector of the robot arm are known. Based on the spinor theory, the transformation matrix corresponding to each joint angle can be replaced by the form of exponential product (POE). The specific positive kinematic expression is shown in Eq. 4.

$$e = e^{\xi_1\theta_1} e^{\xi_2\theta_2} e^{\xi_3\theta_3} e^{\xi_4\theta_4} e^{\xi_5\theta_5} e^{\xi_6\theta_6} e_0 \quad (4)$$

Where  $e_0$  represents the initial position, attitude matrix of the end-effector of the robot arm.  $e^{\xi_1\theta_1}$  represents the transformation matrix corresponding to joint angle 1,  $e^{\xi_2\theta_2}$  represents the transformation matrix corresponding to joint angle 2,  $e^{\xi_3\theta_3}$  represents the transformation matrix corresponding to joint angle 3,  $e^{\xi_4\theta_4}$  represents the transformation matrix corresponding to joint angle 4,  $e^{\xi_5\theta_5}$  represents the transformation matrix corresponding to joint angle 5, and  $e^{\xi_6\theta_6}$  represents the transformation matrix corresponding to joint angle 6.  $e$  represents the final position, attitude matrix of the end-effector of the robot arm. The specific calculation process of the positive kinematics of the six-degree-of-freedom industrial robotic arm (Figure 1) based on the rotational volume theory is as follows.

Step 1: Determine the angular speed of rotation  $\omega_i$  of each linkage ( $i = 1, 2, 3, 4, 5, 6$ ).

$$\omega_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \omega_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \omega_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \omega_4 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \omega_5 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \omega_6 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Step 2: Determine the position  $r_i$  of each linkage ( $i = 1, 2, 3, 4, 5, 6$ ).

$$r_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, r_2 = \begin{bmatrix} 0 \\ a_1 \\ d_1 \end{bmatrix}, r_3 = \begin{bmatrix} 0 \\ a_1 \\ d_1 + a_2 \end{bmatrix}, r_4 = r_5 = r_6 = \begin{bmatrix} 0 \\ a_1 + d_4 \\ d_1 + a_2 + a_3 \end{bmatrix}$$

Where  $a_1, a_2, a_3, d_1, d_4$  are the relevant parameters for a general six-degree-of-freedom industrial robotic arm.

Step 3: Determine the intermediate parameters  $\hat{\omega}_i$  of each linkage ( $i = 1, 2, 3, 4, 5, 6$ ).

$$\hat{\omega}_1 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \hat{\omega}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \hat{\omega}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \hat{\omega}_4 = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \hat{\omega}_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \hat{\omega}_6 = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Step 4: Determine the translational speed  $v_i$  of each linkage ( $i = 1, 2, 3, 4, 5, 6$ ).

$$v_i = -\omega_i \times r_i \quad (i = 1, 2, 3, 4, 5, 6) \quad (5)$$

Step 4: Determine the attitude change matrix  $R_i$  for each linkage ( $i = 1, 2, 3, 4, 5, 6$ ).

$$R_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \hat{\omega}_i * \sin(\theta_i) + \hat{\omega}_i^2 * (1 - \cos(\theta_i)) \quad (i = 1, 2, 3, 4, 5, 6) \quad (6)$$

where  $\theta_i$  represents the joint angle of the corresponding linkage  $i$ .

Step 6: Determine the position change matrix  $t_i$  for each linkage ( $i = 1, 2, 3, 4, 5, 6$ ).

$$t_i = \left( \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - R_i \right) * (\omega_i \times v_i) + \omega_i * \omega_i^T * v_i * \theta_i \quad (i = 1, 2, 3, 4, 5, 6) \quad (7)$$

Step 7: Determine the position, attitude change matrix  $e^{\xi_i \theta_i}$  of each linkage ( $i = 1, 2, 3, 4, 5, 6$ ).

$$e^{\xi_i \theta_i} = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix} \quad (i = 1, 2, 3, 4, 5, 6) \quad (8)$$

Step 8: Find the robotic arm end-effector end position, attitude matrix  $e_0$  with the known robotic arm end-effector initial end position and attitude matrix  $e$ .

$$e = e^{\xi_1 \theta_1} e^{\xi_2 \theta_2} e^{\xi_3 \theta_3} e^{\xi_4 \theta_4} e^{\xi_5 \theta_5} e^{\xi_6 \theta_6} e_0 \quad (9)$$

Among them, the positive kinematic solution idea for the position coordinates of the robotic arm end-effector and Euler angles of rotation along the  $x, y, z$  axes is as follows: first set the initial position coordinates of the robotic arm end-effector and Euler angles of rotation along the  $x, y, z$  axes to get the initial end position pose matrix of the robotic arm end-effector. Then the kinematic analysis of the robot arm is carried out to obtain the position pose change matrix of the end-effector of the robot arm, and finally the actual position pose matrix of the end-effector of the robot arm is obtained by multiplying the position pose change matrix with the initial position pose matrix. According to the actual position pose matrix, the actual position coordinates of the end-effector of the robot arm and Euler angles of rotation along the  $x, y, z$  axes can be obtained. The whole process realizes the transformation of the robotic arm from one position pose to another position pose. Among them, the initial position pose matrix of the robotic arm end-effector in experiment 1, the associated change matrix and Euler angles of rotation along the  $x, y, z$  axes of the end-effector, and the position expressions are shown below.

$$g = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & a_1 + d_4 + a_6 \\ 0 & 0 & 1 & d_1 + a_2 + a_3 - d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

$$e^{\xi_1 \theta_1} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} e^{\xi_2 \theta_2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_2 & -s_2 & d_1 s_2 - a_1 (c_2 - 1) \\ 0 & s_2 & c_2 & -a_1 s_2 - d_1 (c_2 - 1) \\ 0 & 0 & 0 & 1 \end{bmatrix} e^{\xi_3 \theta_3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_3 & -s_3 & s_3 (a_2 + d_1) - a_1 (c_3 - 1) \\ 0 & s_3 & c_3 & -a_1 s_3 - (a_2 + d_1) (c_3 - 1) \\ 0 & 0 & 0 & 1 \end{bmatrix} e^{\xi_4 \theta_4} = \begin{bmatrix} c_4 & 0 & -s_4 & s_4 (a_2 + a_3 + d_1) \\ 0 & 1 & 0 & 0 \\ s_4 & 0 & c_4 & -(c_4 - 1) (a_2 + a_3 + d_1) \\ 0 & 0 & 0 & 1 \end{bmatrix} e^{\xi_5 \theta_5} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_5 & -s_5 & s_5 (a_2 + a_3 + d_1) - (a_1 + d_4) (c_5 - 1) \\ 0 & s_5 & c_5 & -(c_5 - 1) (a_2 + a_3 + d_1) - s_5 (a_1 + d_4) \\ 0 & 0 & 0 & 1 \end{bmatrix} e^{\xi_6 \theta_6} = \begin{bmatrix} c_6 & -s_6 & 0 & s_6 (a_1 + d_4) \\ s_6 & c_6 & 0 & -(a_1 + d_4) (c_6 - 1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

$$\alpha = atan2 \left( c_6 (c_5 (c_2 s_3 + c_3 s_2) + c_4 s_5 (c_2 c_3 - s_2 s_3)) \right.$$

$$\left. - s_4 s_6 (c_2 c_3 - s_2 s_3), c_4 c_5 (c_2 c_3 - s_2 s_3) - s_5 (c_2 s_3 + c_3 s_2) \right) \quad (12)$$

$$\beta = atan2 \left( \begin{matrix} -[s_6 (c_5 (c_2 s_3 + c_3 s_2) + c_4 s_5 (c_2 c_3 - s_2 s_3)) + c_6 s_4 (c_2 c_3 - s_2 s_3)] \\ sqrt \left( [c_6 (c_5 (c_2 s_3 + c_3 s_2) + c_4 s_5 (c_2 c_3 - s_2 s_3)) - s_4 s_6 (c_2 c_3 - s_2 s_3)]^2 \right. \right. \\ \left. \left. + [c_4 c_5 (c_2 c_3 - s_2 s_3) - s_5 (c_2 s_3 + c_3 s_2)]^2 \right) \right) \quad (13)$$

$$\eta = atan2 \left( \begin{matrix} s_6 (c_5 (c_1 c_2 c_3 - c_1 s_2 s_3) - s_5 (s_1 s_4 + c_4 (c_1 c_2 s_3 + c_1 c_3 s_2))) \\ + c_6 (c_4 s_1 - s_4 (c_1 c_2 s_3 + c_1 c_3 s_2)), c_6 (c_1 c_4 + s_4 (c_2 s_1 s_3 + c_3 s_1 s_2)) \\ - s_6 (c_5 (c_2 c_3 s_1 - s_1 s_2 s_3) + s_5 (c_1 s_4 - c_4 (c_2 s_1 s_3 + c_3 s_1 s_2))) \end{matrix} \right) \quad (14)$$

$$P_x = a_2 s_1 s_2 - a_1 s_1 + a_3 c_2 s_1 s_3 + a_3 c_3 s_1 s_2 + d_4 s_1 s_2 s_3 - a_6 c_1 c_4 s_6 - c_2 c_3 d_4 s_1 + c_1 c_3 d_6 s_4 - a_6 c_1 c_6 s_4 s_5 - c_2 c_3 d_6 s_1 s_5 + d_6 s_1 s_2 s_3 s_5 - a_6 c_2 c_3 c_5 c_6 s_1 - c_2 c_4 c_5 d_6 s_1 s_3 - c_3 c_4 c_5 d_6 s_1 s_2 + a_6 c_5 c_6 s_1 s_2 s_3 - a_6 c_2 s_1 s_3 s_4 s_6 - a_6 c_3 s_1 s_2 s_4 s_6 + a_6 c_2 c_4 c_6 s_1 s_3 s_5 + a_6 c_3 c_4 c_6 s_1 s_2 s_5 \quad (15)$$

$$P_y = a_1 c_1 - a_2 c_1 s_2 - a_6 c_4 s_1 s_6 - c_1 d_4 s_2 s_3 + c_2 d_6 s_3 s_4 + c_1 c_2 c_3 d_4 - a_3 c_1 c_2 s_3 - a_3 c_1 c_3 s_2 + c_1 c_2 c_3 d_6 s_5 - a_6 c_6 s_1 s_4 s_5 - c_1 d_6 s_2 s_3 s_5 + a_6 c_1 c_2 c_3 c_5 c_6 + c_1 c_2 c_4 c_5 d_6 s_3 + c_1 c_3 c_4 c_5 d_6 s_2 - a_6 c_1 c_5 c_6 s_2 s_3 + a_6 c_1 c_2 s_3 s_4 s_6 + a_6 c_1 c_3 s_2 s_4 s_6 - a_6 c_1 c_2 c_4 c_6 s_3 s_5 - a_6 c_1 c_3 c_4 c_6 s_2 s_5 \quad (16)$$

$$P_z = d_1 + a_2 c_2 + a_3 c_2 c_3 + c_2 d_4 s_3 + c_3 d_4 s_2 - a_3 s_2 s_3 + c_2 d_6 s_3 s_5 + c_3 d_6 s_2 s_5 - c_2 c_3 c_4 c_5 d_6 + a_6 c_2 c_5 c_6 s_3 + a_6 c_3 c_5 c_6 s_2 - a_6 c_2 c_3 s_4 s_6 + c_4 c_5 d_6 s_2 s_3 + a_6 s_2 s_3 s_4 s_6 + a_6 c_2 c_3 c_4 c_6 s_5 - a_6 c_4 c_6 s_2 s_3 s_5 \quad (17)$$

where  $s_i = \sin \theta_i (i = 1 \sim 6)$ ,  $c_i = \cos \theta_i (i = 1 \sim 6)$ .  $a_1, a_2, a_3, a_6, d_1, d_4, d_6$  denotes the parameters of a general six-degree-of-freedom industrial robot arm.  $g$  represents the initial position attitude matrix of the end-effector,  $e^{\xi_1 \theta_1} \sim e^{\xi_6 \theta_6}$  represents the variation matrix of the joint angles from 1 to 6,  $\alpha, \beta, \eta$  represents Euler angles of rotation along the  $x, y, z$  axes after the end-effector moves from the initial position, and  $P_x, P_y, P_z$  represents the new position of the end-effector after the end-effector moves from the initial position. The initial position pose matrix of the robotic arm end-effector in experiment 2, the associated change matrix and the three Euler angles of the end-effector, and the position expressions are shown below.

$$g = \begin{bmatrix} 0.9998 & 0.0175 & 0 & a_2 + a_3 \\ -0.0175 & 0.9998 & 0 & d_2 \\ 0 & 0 & 1 & -d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (18)$$

$$T_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_3 = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_4 = \begin{bmatrix} c_4 & -s_4 & 0 & a_3 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (19)$$

$$\alpha = \text{atan2} \left( \frac{(4999s_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2)))/5000 - (7c_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2)))/400 + (7s_4s_6(c_2s_3 + c_3s_2))/400}{c_4s_5(c_2s_3 + c_3s_2) - c_5(c_2c_3 - s_2s_3)} \right) \quad (20)$$

$$\beta = \text{atan2} \left( \frac{-[(4999s_4s_6(c_2s_3 + c_3s_2))/5000 - (7s_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2)))/400] - (4999c_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2)))/5000}{-[(4999s_4s_6(c_2s_3 + c_3s_2))/5000 - (7s_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2)))/400] - (4999c_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2)))/5000}}{\text{sqrt} \left( \frac{-(7c_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2)))/400 + (7s_4s_6(c_2s_3 + c_3s_2))/400 + (4999c_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2)))/5000}{+ [c_4s_5(c_2s_3 + c_3s_2) - c_5(c_2c_3 - s_2s_3)]^2} \right)} \right) \quad (21)$$

$$\eta = \text{atan2} \left( \frac{(7c_6(c_1c_4 + s_4(c_2c_3s_1 - s_1s_2s_3)))/400 - (7s_6(s_5(c_2s_1s_3 + c_3s_1s_2) + c_5(c_1s_4 - c_4(c_2c_3s_1 - s_1s_2s_3))))/400 - (4999c_6(s_5(c_2s_1s_3 + c_3s_1s_2) + c_5(c_1s_4 - c_4(c_2c_3s_1 - s_1s_2s_3))))/5000 - (4999s_6(c_1c_4 + s_4(c_2c_3s_1 - s_1s_2s_3)))/5000}{(4999s_6(c_4s_1 - s_4(c_1c_2c_3 - c_1s_2s_3)))/5000 - (7s_6(s_5(c_1c_2s_3 + c_1c_3s_2) - c_5(s_1s_4 + c_4(c_1c_2c_3 - c_1s_2s_3))))/400 - (7c_6(c_4s_1 - s_4(c_1c_2c_3 - c_1s_2s_3)))/400} \right) \quad (22)$$

$$P_x = d_2(s_6(s_5(c_1c_2s_3 + c_1c_3s_2) - c_5(s_1s_4 + c_4(c_1c_2c_3 - c_1s_2s_3))) + c_6(c_4s_1 - s_4(c_1c_2c_3 - c_1s_2s_3))) - d_2s_1 + d_4(c_5(c_1c_2s_3 + c_1c_3s_2) + s_5(s_1s_4 + c_4(c_1c_2c_3 - c_1s_2s_3))) + a_3(c_1c_2c_3 - c_1s_2s_3) - d_4(c_1c_2s_3 + c_1c_3s_2) - (a_2 + a_3)(-s_6(c_4s_1 - s_4(c_1c_2c_3 - c_1s_2s_3))) \Big) + a_2c_1c_2 \quad (23)$$

$$P_y = c_1d_2 - (a_2 + a_3)(c_6(s_5(c_2s_1s_3 + c_3s_1s_2) + c_5(c_1s_4 - c_4(c_2c_3s_1 - s_1s_2s_3))) + s_6(c_1c_4 + s_4(c_2c_3s_1 - s_1s_2s_3))) + d_2(s_6(s_5(c_2s_1s_3 + c_3s_1s_2) + c_5(c_1s_4 - c_4(c_2c_3s_1 - s_1s_2s_3))) - c_6(c_1c_4 + s_4(c_2c_3s_1 - s_1s_2s_3))) + d_4(c_5(c_2s_1s_3 + c_3s_1s_2) + s_5(s_1s_4 - c_4(c_2c_3s_1 - s_1s_2s_3))) + a_3(c_1c_2s_3 - c_1c_3s_2) - d_4(c_2s_1s_3 + c_3s_1s_2) - a_2c_1s_1 \quad (24)$$

$$P_z = d_1(c_5(c_2c_3 - s_2s_3) - c_4s_5(c_2s_3 + c_3s_2)) - a_3(c_2s_3 + c_3s_2) - d_4(c_2c_3 - s_2s_3) - (c_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2))) - s_4s_6(c_2s_3 + c_3s_2)(a_2 + a_3) - a_2s_2 + d_2(s_6(s_5(c_2c_3 - s_2s_3) + c_4c_5(c_2s_3 + c_3s_2))) + c_6s_4(c_2s_3 + c_3s_2) \quad (25)$$

where  $s_i = \sin \theta_i$  ( $i = 1 \sim 6$ ),  $c_i = \cos \theta_i$  ( $i = 1 \sim 6$ ).  $a_2, a_3, d_2, d_4$  denotes the parameters of PUMA560 robot arm.  $g$  represents the initial position attitude matrix of the end-effector,  $T_1 \sim T_6$  represents the variation matrix of the joint angles from 1 to 6,  $\alpha, \beta, \eta$  represents Euler angles of rotation along the  $x, y, z$  axes after the end-effector moves from the initial position, and  $P_x, P_y, P_z$  represents the new position of the end-effector after the end-effector moves from the initial position. The initial position pose matrix of the robotic arm end-effector in experiment 3, the associated change matrix and the three Euler angles of the end-effector, and the position expressions are shown below.

$$g = \begin{bmatrix} 0.9998 & 0 & 0.0175 & a_2 + a_3 + a_4 + a_5 + a_6 + a_7 \\ 0 & 1 & 0 & d_7 \\ -0.0175 & 0 & 0.9998 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (26)$$

$$T_1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_2 = \begin{bmatrix} c_2 & 0 & s_2 & a_2c_2 \\ s_2 & 0 & -c_2 & a_2s_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_3 = \begin{bmatrix} c_3 & 0 & -s_3 & a_3c_3 \\ s_3 & 0 & c_3 & a_3s_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_4 = \begin{bmatrix} c_4 & 0 & s_4 & a_4c_4 \\ s_4 & 0 & -c_4 & a_4s_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_5 = \begin{bmatrix} c_5 & 0 & -s_5 & a_5c_5 \\ s_5 & 0 & c_5 & a_5s_5 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_6 = \begin{bmatrix} c_6 & -s_6 & 0 & a_6c_6 \\ s_6 & c_6 & 0 & a_6s_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (27)$$

$$\alpha = \text{atan2} \left( \frac{c_7(s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 - c_3s_2s_4)) + s_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4))}{(7s_7(s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 - c_3s_2s_4)))/400 - (7c_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4)))/400 + (4999s_5(c_2s_4 + c_3c_4s_2))/5000}{(4999s_5(c_2s_4 + c_3c_4s_2))/5000} \right) \quad (28)$$

$$\beta = \text{atan2} \left( \frac{-[(4999s_7(s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 - c_3s_2s_4)))/5000 - (4999c_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4)))/400 - (7c_5s_2s_3)/400], \text{sqrt} \left( [c_7(s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 - c_3s_2s_4)) + s_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4))]^2 + [(7s_7(s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 - c_3s_2s_4)))/400 - (7c_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4)))/400 + (4999s_5(c_2s_4 + c_3c_4s_2))/5000]^2 \right)} \right) \quad (29)$$

$$\eta = \text{atan2} \left( \frac{(7s_5(c_1(c_4s_1 + c_2c_3s_1) - s_1s_2s_3))/400 - (4999c_7(s_6(s_4(c_1s_3 + c_2c_3s_1) + c_4s_1s_2) - c_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3))))/5000 - (7c_5(c_1c_3 - c_2s_1s_3))/400}{-[(4999s_7(c_6(s_4(c_1s_3 + c_2c_3s_1) + c_4s_1s_2) - c_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3))))/5000 - (4999c_7(s_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) - c_6(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(s_1s_3 - c_1c_2c_3))))/5000 - (7s_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4))/400} \right) \quad (30)$$

$$P_x = \left( c_7(s_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) - c_6(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3))) + s_7(c_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) + s_6(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3))) \right) + d_2(a_2 + a_3 + a_4 + a_5 + a_6 + a_7) + d_4 \left( c_7(c_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) + s_6(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3))) + a_7c_7(s_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) - c_6(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3)))) \right) + d_1(s_5(s_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) - c_5(c_3s_1 + c_1c_2s_3)) + d_2(s_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) - c_5(c_3s_1 + c_1c_2s_3)) - a_5c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + a_7c_7(s_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) - c_6(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(s_1s_3 - c_1c_2c_3)))) + a_6s_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) + a_7s_7(c_6(s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2) + s_6(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3))) - a_6c_6(c_5(c_4(s_1s_3 - c_1c_2c_3) + c_1s_2s_4) + s_5(c_3s_1 + c_1c_2s_3)) - a_4c_4(s_1s_3 - c_1c_2c_3) - a_5s_5(c_3s_1 + c_1c_2s_3) + a_2c_1c_2 - a_3s_1s_3 - a_4c_1s_2s_4 + a_3c_1c_2c_3 \quad (31)$$

$$P_y = a_5c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) - (c_7(s_6(s_4(s_1s_3 + c_2c_3s_1) + c_4s_1s_2) - c_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3))) + s_7(c_6(s_4(s_1s_3 + c_2c_3s_1) + c_4s_1s_2) + s_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3)))) \right) + a_2(a_2 + a_3 + a_4 + a_5 + a_6 + a_7) - d_1(s_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) - c_5(c_1c_3 - c_2s_1s_3)) - d_2(s_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) - c_5(c_1c_3 - c_2s_1s_3)) - d_3(s_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) - c_5(c_1c_3 - c_2s_1s_3)) - d_4 \left( c_7(c_6(s_4(s_1s_3 + c_2c_3s_1) + c_4s_1s_2) + s_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3))) \right) - a_7s_7(s_6(s_4(s_1s_3 + c_2c_3s_1) + c_4s_1s_2) - c_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3))) - a_6s_6(s_4(s_1s_3 + c_2c_3s_1) + c_4s_1s_2) - a_7c_7(s_6(s_4(s_1s_3 + c_2c_3s_1) + c_4s_1s_2) - c_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3))) + a_6c_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3)) + a_4c_4(c_1s_3 + c_2c_3s_1) - a_7s_7(c_6(s_4(s_1s_3 + c_2c_3s_1) + c_4s_1s_2) + s_6(c_5(c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4) + s_5(c_1c_3 - c_2s_1s_3))) + a_3c_1c_3 - a_4s_1s_2s_4 + a_3c_1c_2c_3 \quad (32)$$

$$P_z = d_1 + d_7 \left( c_7(s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 - c_3s_2s_4)) + s_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4)) \right) - a_2s_2 - (c_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4))) - s_7(s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 - c_3s_2s_4))) + d_1(s_5(c_2s_4 + c_3c_4s_2) + c_5s_2s_3) + d_7(s_5(c_2s_4 + c_3c_4s_2) + c_5s_2s_3) - a_6c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) - a_7c_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4))) - a_5c_5(c_2s_4 + c_3c_4s_2) + a_7s_7(s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 - c_3s_2s_4)) - a_6s_6(c_2c_4 - c_3s_2s_4) - a_3c_3s_2 - a_4c_4s_4 + a_5s_2s_3s_5 - a_4c_3c_4s_2 \quad (33)$$

where  $s_i = \sin \theta_i$  ( $i = 1 \sim 6$ ),  $c_i = \cos \theta_i$  ( $i = 1 \sim 6$ ).  $a_1 \sim a_7, d_1 \sim d_7$  denotes the parameters of Seven-degree-of-freedom robot arm.  $g$  represents the initial position attitude matrix of the end-effector,  $T_1 \sim T_7$  represents the variation matrix of the joint angles from 1 to 6,  $\alpha, \beta, \eta$  represents Euler angles of rotation along the  $x, y, z$  axes after the end-effector moves from the initial position, and  $P_x, P_y, P_z$  represents the new position of the end-effector after the end-effector moves from the initial position.

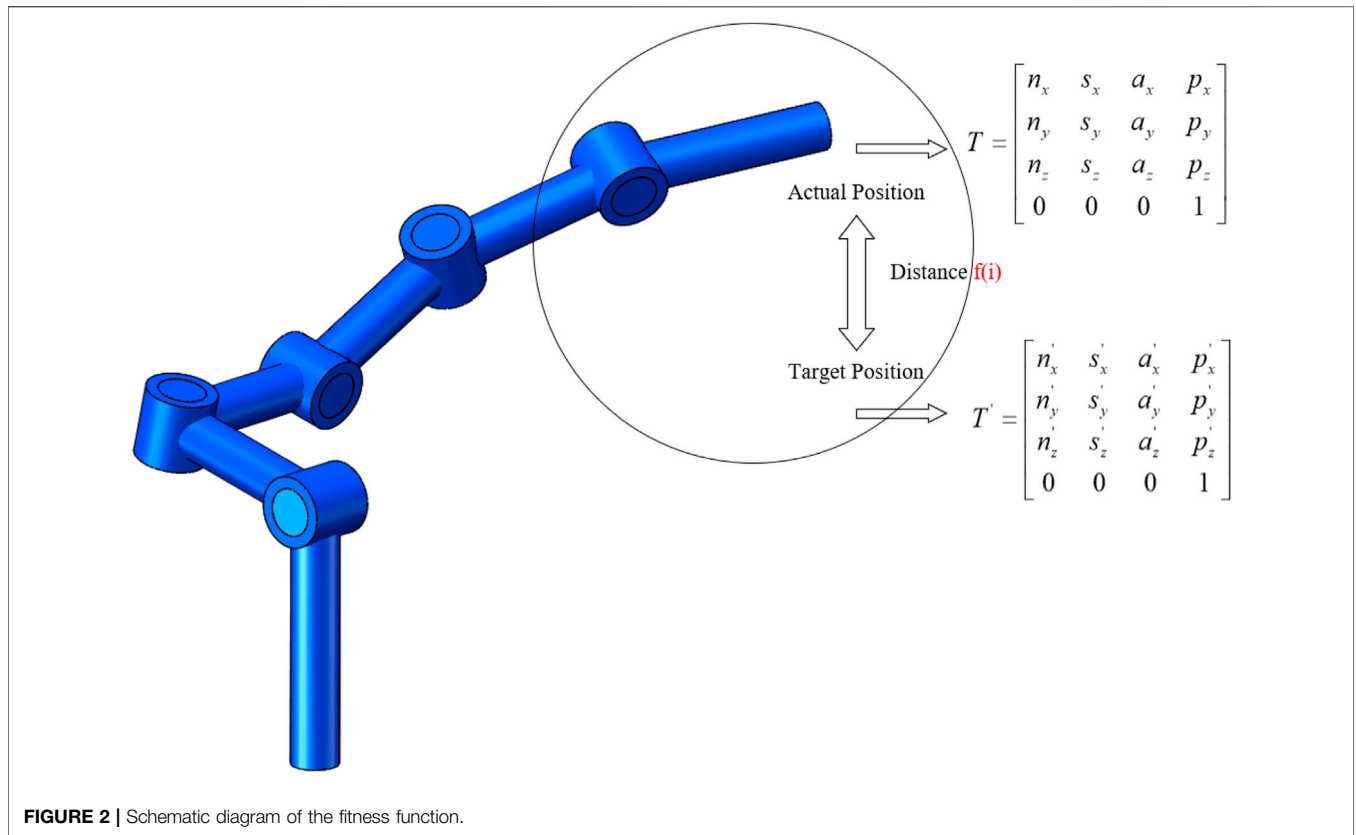


FIGURE 2 | Schematic diagram of the fitness function.

## 4 PARTICLE SWARM ALGORITHM AND ENHANCEMENT

### 4.1 Two Particle Swarm Algorithms

To solve practical engineering applications, researchers have invented metaheuristic algorithms based on the laws of nature. Particle swarm optimization algorithm, as a kind of metaheuristic algorithm, is an algorithm invented to simulate bird flock predation. Based on the feature that the flock of birds close to the predation target will drive the flock of birds at a distance to the predation target and eventually drive the flock as a whole to the predation target, the particles of the population in the particle swarm algorithm will carry two variables,  $x$  and  $v$ . Meanwhile, the solution will be approximated to the optimal solution by updating  $x$  and  $v$  in real time. Where  $x$  denotes the position of the particle in the search space and  $v$  denotes the step length and direction of movement. For the classical particle swarm algorithm, the update formula for  $v$  and  $x$  is shown below.

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 (P_{best} - x_i^t) + c_2 r_2 (G_{best} - x_i^t) \quad (34)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (35)$$

Where  $i$  represents the position of the particle in the population,  $t$  represents the number of iterations, and  $\omega$  represents the inertia weight.  $c_1$  represents the cognitive learning factor, and  $c_2$  represents the social learning factor, both of which are generally taken between 1 and 2.  $r_1$  and  $r_2$  are random numbers between 0 and 1.  $P_{best}$  represents the individual optimal position, and  $G_{best}$  represents the global optimal

position. The above is the update formula of position as well as velocity of the classical particle swarm algorithm. In addition to the above update formula, there is a quantum mechanics-based position update method in the QPSO algorithm, which discards the velocity update and chooses the position update method, which is a novel attempt, and its update formula is shown below.

$$g_i = \varphi pbest_i + (1 - \varphi)gbest_d \quad (36)$$

$$x_i(t + 1) = g_i + \beta |mbest_d - x_i(t)| \log\left(\frac{1}{u}\right) \quad u > 0.5 \quad (37)$$

$$x_i(t + 1) = g_i - \beta |mbest_d - x_i(t)| \log\left(\frac{1}{u}\right) \quad u > 0.5 \quad (38)$$

$$mbest_d = \frac{1}{M} \sum_{i=1}^M pbest_i \quad (39)$$

where  $i$  represents the position of the particle in the population,  $t$  represents the number of iterations, and  $M$  represents the number of populations.  $\varphi$ ,  $u$  is a random number between 0 and 1, and  $\beta$  is a constant number between 0 and 1. The  $pbest$  represents the best position of the individual particle,  $gbest$  represents the optimal position of the population, and  $mbest$  represents the average of the best position of the individual particle. The basic steps of solving the particle swarm algorithm are shown below.

Step 1: Initialization of the population particles.

Step 2: Calculate the fitness function.

Step 3: Update particle position and velocity.



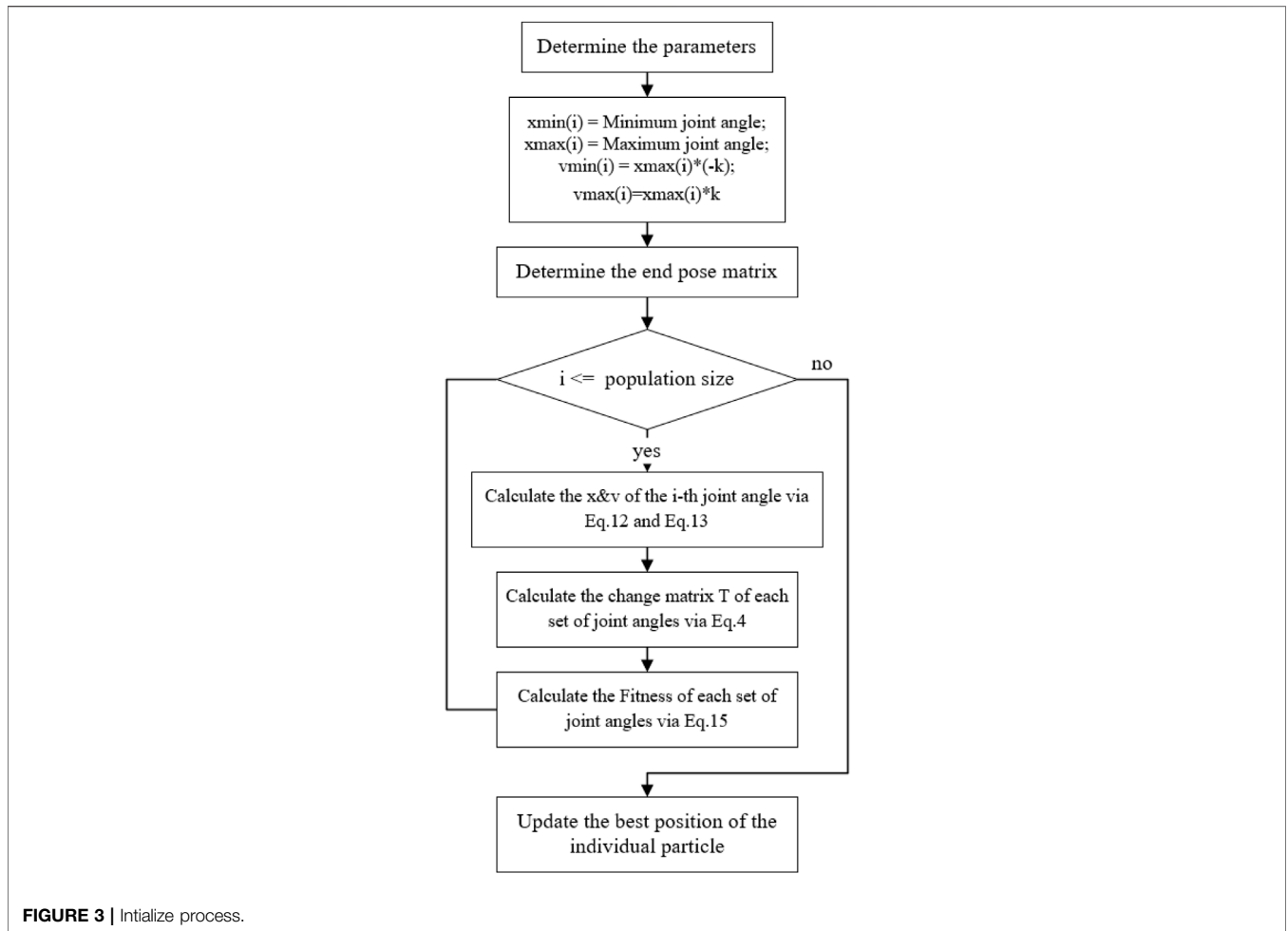


FIGURE 3 | Initialize process.

Step 4: Update individual best position and group best position.

### 4.2 Improvement

For setting the weights when updating the velocity in the particle swarm algorithm, this paper adopts the adaptive weights. The most important feature of the adaptive weights is that the weights change with the change of the fitness function value of the particles. For different weights, the local search as well as the global search capability of the algorithm will be very different. Generally large values of inertia weights are more favorable for global search, and small values of inertia weights are favorable for local search. When the algorithm falls into the local optimum, it tends to miss the optimal solution and then converge too early, while when the global search ability of the algorithm is too strong, the final accuracy of the algorithm is often not too high. In order to balance the local search as well as the global search ability, adaptive weights are the best choice. The basic idea is as follows:

Step 1: Calculate value of fitness function  $fitness$ , minimum fitness function value  $fitness_{min}$  and average fitness function value  $fitness_{avg}$ .

Step 2: If  $fitness$  is less than or equal to  $fitness_{avg}$ , the weight become:

$$\omega = \omega_{min} + \frac{(\omega_{max} - \omega_{min})(fitness - fitness_{min})}{(fitness_{avg} - fitness_{min})} \quad (40)$$

If the  $fitness$  is larger than  $fitness_{avg}$ , the weight become:  $\omega = \omega_{max}$ .

Step 3: Update  $fitness$ .

Particle swarm algorithms require random initialization of positions and velocities, and there is no fixed standard for initialization, which makes it difficult to guarantee the accuracy and precision of the final solution, and the search time is often too long, resulting in low efficiency of the algorithm. Most researchers use particle swarm algorithms to study the robot inverse kinematics solution without further description of the initialization process of the population, but only add boundary conditions to the algorithm to ensure the executability of the algorithm. Based on the consideration of the shortest algorithm operation time, this paper adopts the conditional restriction based on the limit joints, and the maximum as well as the minimum values of the position are determined by the range of values of each joint angle. For the maximum and minimum values of velocity, this paper introduces the position coefficient  $k$  and adopts the form of multiplying

position and coefficient to determine the velocity factor, and the specific conditions are set as follows.

Step 1: Determine maximum and minimum position  $x_{max}$ ,  $x_{min}$  according to the range of values of the joint angle.

Step 2: Determine maximum and minimum speed  $v_{max}$ ,  $v_{min}$  according to  $x_{max}$ ,  $x_{min}$ , where:

$$v_{max} = x_{max} \cdot k; v_{min} = x_{max} \cdot (-k)$$

(Based on actual experience  $k$  takes the value of 0.5)

Step 3: Determine position  $x$  and speed  $v$ .

$$x = \text{rand}(0, 1)(x_{max} - x_{min}) + x_{min} \quad (41)$$

$$v = \text{rand}(0, 1)(v_{max} - v_{min}) + v_{min} \quad (42)$$

## 5 INVERSE KINEMATIC SOLUTION USING IMPROVED PARTICLE SWARM ALGORITHM

### 5.1 Fitness Function

The selection of the fitness function greatly affects the efficiency of the particle swarm algorithm. For the robot arm inverse kinematics solution problem, in order to better ensure the accuracy of position and direction solution, the form of the fitness function in this paper is the error value of the robot arm end-effector position and direction angle. The error is selected as the Euclidean distance between the target position, direction angle and the actual position and direction angle. The specific form of the fitness function is shown in Eq. 43. A geometric illustration of the fitness function is shown in Figure 2.

$$f(i) = \sqrt{\sum_{i=1}^3 (T'_{i1} - T_{i1})^2 + \sum_{i=1}^3 (T'_{i2} - T_{i2})^2 + \sum_{i=1}^3 (T'_{i3} - T_{i3})^2 + \sum_{i=1}^3 (T'_{i4} - T_{i4})^2} \quad (43)$$

In Eq. 43, the first to third terms under the root sign represent the directional angle error of the deflection of the robotic arm end-effector along the positive direction of the  $x$ ,  $y$ , and  $z$  axes, respectively, and the fourth term under the root sign represents the position error of the robotic arm end-effector. In Figure 2, the geometric meaning of the adaptation function is further illustrated. When the robotic arm end-effector cannot reach the specified target position, the Euclidean distance between the actual position and attitude of the robotic arm end-effector and the ideal position and attitude is represented as the error, and this error is reflected in the form of the fitness function. Eq. 43 and the matrix  $T$  in Figure 2 represent the actual end position pose matrix of the robotic arm end-effector.  $t'$  represents the ideal end position pose matrix of the robotic arm end-effector. Where the matrix forms of  $T$  and  $T'$  are shown in Eqs. 44 and 45, respectively.

$$T = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (44)$$

$$T' = \begin{bmatrix} n'_x & s'_x & a'_x & p'_x \\ n'_y & s'_y & a'_y & p'_y \\ n'_z & s'_z & a'_z & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (45)$$

In Eq. 44, the first three rows and the first three columns of the matrix  $T$  form the rotation transformation matrix, which represents the actual attitude transformation of the end-effector of the robot arm, and the first three rows of the last column of the matrix  $T$  form the column vector, which represents the actual position of the end-effector of the robot arm with respect to the base coordinate system. In Eq. 45, the first three rows and the first three columns of the matrix  $T'$  form the rotation transformation matrix, which represents the target attitude transformation of the end-effector of the robot arm, and the first three rows of the last column of the matrix  $T'$  form the column vector, which represents the target position to be reached by the end-effector of the robot arm.

### 5.2 Flowchart and Pseudocode

The flowchart and the pseudo-code of the improved PSO algorithm proposed in this paper are shown in Figures 3, 4, and Table 1, respectively. Where Figure 3 represents the parameter initialization process based on limit joints. Figure 4 shows the main iterative loop process for updating the position as well as the velocity.

```

Algorithm: Improved Particle Swarm Optimization
1  ##### Initialize parameters #####
2  Determine the parameters
3  ##### Initialize positions and velocities #####
4  xmin(i) = Minimum joint angle; xmax(i) = Maximum joint angle
5  vmin(i) = xmin(i)*(-k); vmax(i) = xmax(i)*k
6  for i = 1 to population size
7      The position of the i-th joint angle(i)=rand(0,1)*(xmax(i)-xmin(i))+xmin(i)
8  end for
9  Determine the end pose matrix T_e
10 ##### Initialize the fitness of each particle #####
11 for i = 1 to population size
12     Calculate the change matrix T of each set of joint angles
13     Calculate the value of the fitness function p(i) according to Equation (19)
14 end for
15 particle.best = particle[i].position
16 for i = 1 to population size
17     if particle[i].position < particle.best
18         particle.best = particle[i].position
19         particle.bestfitness = particle[i].fitness
20     end if
21 end for
22 ##### Enter the main loop #####
23 for t = 1 to max_iteration
24     for i = 1 to population size
25         Determine the end pose matrix T_e
26         Calculate the value of the fitness function f(i) according to Equation (19)
27         ##### Calculate the average_fitness #####
28         favg_fitness = sum(fit)/ population size
29         ##### Update the best position of the individual particle #####
30         if particle[i].position < particle.best
31             particle.best = particle[i].position
32             particle.bestfitness = particle[i].fitness
33         end if
34         ##### Update the best position of the group #####
35         if particle[i].position < particle.groupbest
36             particle.groupbest = particle[i].position
37             particle.groupbestfitness = particle[i].fitness
38         end if
39         ##### Calculate the minimum_fitness #####
40         min_fitness = min(fit)
41         if particle[i].fitness <= average_fitness
42             Assign a value to ω according to Equation (16)
43         else
44             ω = ω_max
45         end if
46         Updates to velocity and position of individual particles
47     end for
48     return particle.groupbestfitness
    
```

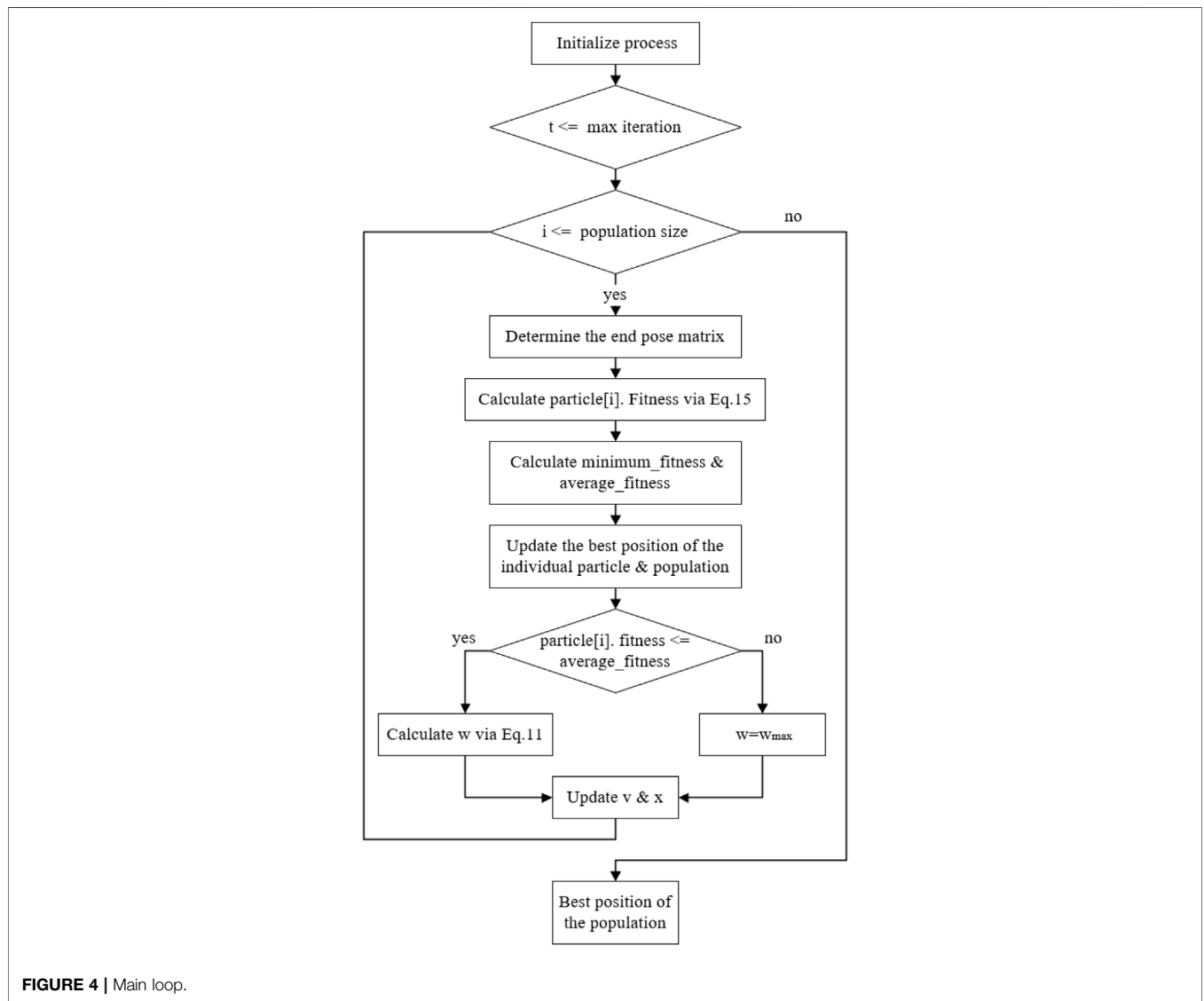


FIGURE 4 | Main loop.

TABLE 1 | Values of relevant parameters of general industrial six-degree-of-freedom robotic arms and the range of values of each joint angle.

$d_1(m)$	$a_1(m)$	$a_2(m)$	$a_3(m)$	$d_4(m)$	$a_6(m)$	$d_6(m)$	Range of $\theta_1 (^\circ)$ ( $i = 1-6$ )
0.25	0.15	0.55	0.16	0.594	0.1	0.1	$(-\pi, \pi]$

TABLE 2 | Initial position of the end-effector of the robot arm and the orientation angle.

Initial Position of the End-Effector			Initial Azimuth of the End-Effector		
$x_0(m)$	$y_0(m)$	$z_0(m)$	$\alpha_0(rad)$	$\beta_0(rad)$	$\eta_0(rad)$
0	$a_1+d_4+a_6$	$d_1+a_2+a_3-d_6$	0	0	0

**TABLE 3** | Position and orientation angle of the end-effector corresponding to the given impact point.

N°	Position Error of the End-Effector			Euler Angles Error of the End-Effector		
	${}^0x_E$ (m)	${}^0y_E$ (m)	${}^0z_E$ (m)	${}^0\alpha_E$ (rad)	${}^0\beta_E$ (rad)	${}^0\eta_E$ (rad)
1	0.1,154,947	0.2,725,154	0.256,376	-2.817,121	0.6,356,652	0.9,973,017
2	-0.0009,583,057	0.2,715,589	0.1,660,651	-2.886,616	0.3,067,386	1.770,458
3	-0.1,268,636	0.3,495,127	0.1,782,351	-2.708,327	0.02,164,443	2.482,109
4	-0.1,676,911	0.4,555,857	0.2,694,087	-2.387,233	-0.07,313,938	3.231,621

**TABLE 4** | Joint angles according to the algorithm.

N°	$\theta_1$ (°)	$\theta_2$ (°)	$\theta_3$ (°)	$\theta_4$ (°)	$\theta_5$ (°)	$\theta_6$ (°)
1	-177.936,340	169.779,771	-58.752,296	91.309,992	125.711,066	32.945,428
2	27.198,690	-43.645,232	-88.014167	35.338,117	-56.159,879	-43.633,188
3	40.364,361	136.560,021	-124.683,465	154.842,073	16.853,445	84.766,356
4	26.260,466	175.113,664	-123.948,475	-95.208,397	107.305,427	-71.495,679

**TABLE 5** | Position and directional angle errors of the end-effector.

N°	Position Error of the End-Effector			Euler Angles Error of the End-Effector		
	${}^0x_E$ (m)	${}^0y_E$ (m)	${}^0z_E$ (m)	${}^0\alpha_E$ (rad)	${}^0\beta_E$ (rad)	${}^0\eta_E$ (rad)
1	0	0	0	$-9.98 \times 10^{-10}$	$4.07 \times 10^{-9}$	$-1.22 \times 10^{-9}$
2	0	0	0	$-1.57 \times 10^{-10}$	$1.28 \times 10^{-9}$	$-1.16 \times 10^{-10}$
3	0	0	0	$6.55 \times 10^{-10}$	$4.10 \times 10^{-10}$	$1.15 \times 10^{-9}$
4	0	0	0	$-2.83 \times 10^{-9}$	$-2.27 \times 10^{-9}$	$6.65 \times 10^{-11}$

## 6 EXPERIMENTS AND RESULTS

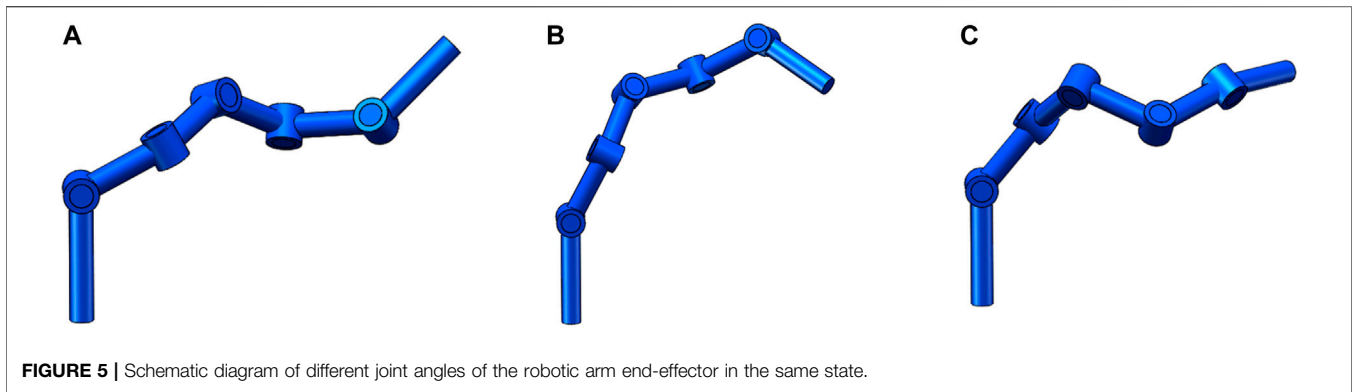
There are three experiments in this paper. The first experiment is an inverse kinematic solution for a general six-degree-of-freedom industrial robotic arm using an improved particle swarm algorithm based on the spinor modeling method. The second experiment compares different improved particle swarm algorithms in terms of algorithm accuracy, convergence and operation time based on existing references for PUMA560 robotic arm. The third experiment replaces the object of study with a seven-degree-of-freedom robotic arm and repeats the steps of experiment 2. The criterion for comparing the algorithms in experiment 2 and experiment 3 was to set the same number of iterations, with the same initial parameters. The criteria for evaluating the algorithm’s capability include the comparison of solution accuracy, solution time, and generalizability. The experiments were all coded in MATLAB R2021b with the processor model: Intel (R) Core (TM) i9-12900KF CPU @ 3.19GHz.

### 6.1 Results Obtained for General Industrial Six-degree-of-freedom Robotic Arm

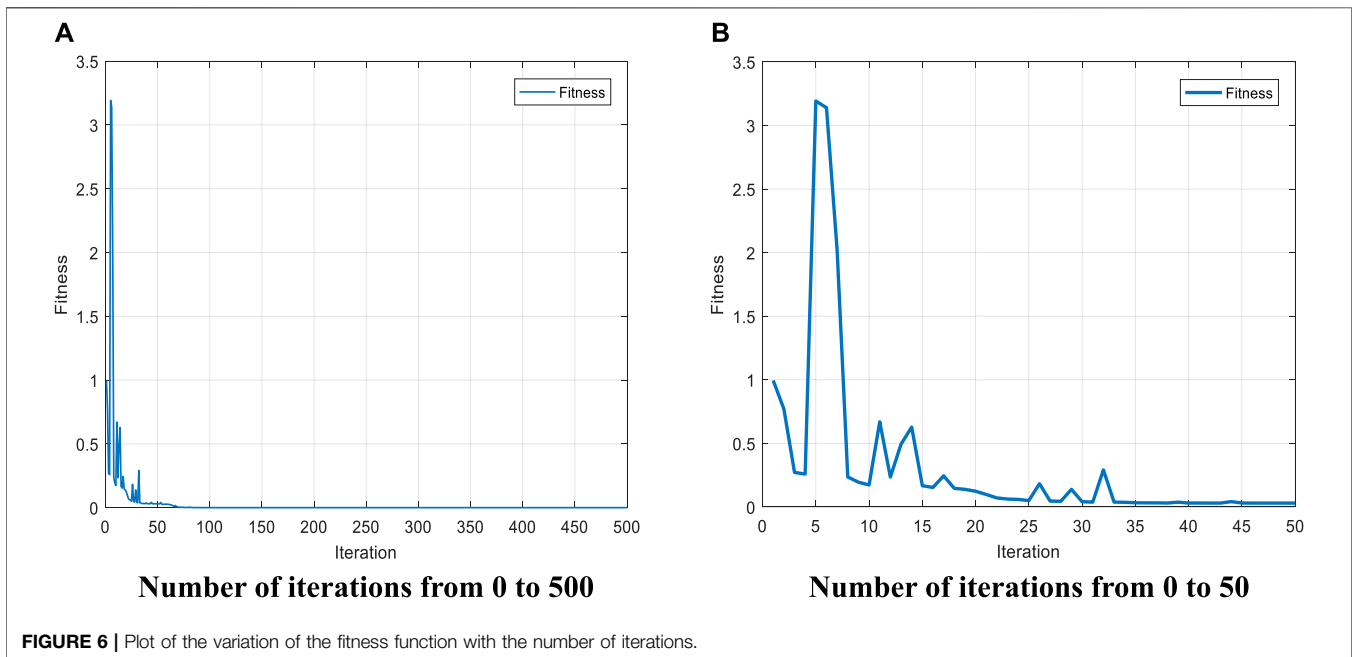
The general industrial six-degree-of-freedom robotic arm is studied above, and its positive kinematic model is established based on the rotating body theory. In Experiment 1, the relevant parameter values and the ranges of each joint angle of the general industrial six-degree-of-freedom robotic arm are given in **Table 1**. And the initial position of the end-effector of the robotic arm and the Euler angles of rotation along  $x, y, z$  axes are set, as shown in **Table 2**. Also, the actual impact points of the four sets of robotic arm end-effectors are set, and the positions of

the end-effectors at the points and the Euler angles of rotation along the  $x, y, z$  axes are given in **Table 3**. The robotic arm end-effector moves from the initial position to the impact point in the process of the robotic arm realizes the change from one position attitude to another position attitude.

**Tables 1–3** show the conditions.  $a_1, a_2, a_3, a_6, d_1, d_4, d_6$  in **Table 1** indicates the parameters of a general six-degree-of-freedom industrial robotic arm, and  $\theta_i$  indicates the joint angle  $i$  ( $i = 1-6$ ). In **Table 2**,  $x_0, y_0, z_0$  denotes the initial position of the robotic arm end-effector,  $\alpha_0, \beta_0, \eta_0$  denotes the initial Euler angles of the robotic arm end-effector rotated along the  $x, y,$  and  $z$  axes, respectively. In **Table 3**,  ${}^0X_E, {}^0Y_E, {}^0Z_E$  denotes the new position of the robotic arm end-effector after moving from the initial position, and  ${}^0\alpha_E, {}^0\beta_E, {}^0\eta_E$  denotes the Euler angles of the robotic arm end-effector rotating along the  $x, y, z$  axes after moving from the initial position, respectively. The initial position of the end-effector of the robot arm and the Euler angles of rotation along the  $x,y,z$  axes set in **Table 2** are known, and the position of the end-effector of the robot arm after moving and the Euler angles of rotation along the  $x, y, z$  axes can be obtained by combining the positive kinematic **Eq. 9**. According to **Table 3**, the specific position of the end-effector of the robot arm after moving and the Euler angles of rotation along  $x, y, z$  axes can be obtained. The error expression, i.e., the fitness function, is obtained by converting the two previous parts into a matrix and making a difference. By solving the minimum value of the fitness function, the algorithm finally obtains the joint angle corresponding to the smallest error and the specific position of the robot arm end-effector after moving and the Euler angle of rotation along the  $x, y, z$  axes. The algorithm can find out the six joint angles corresponding to the end-effector impact point out after the robot arm moves, as



**FIGURE 5** | Schematic diagram of different joint angles of the robotic arm end-effector in the same state.



**FIGURE 6** | Plot of the variation of the fitness function with the number of iterations.

shown in **Table 4**. By comparing the actual position of the end-effector and the Euler angles of rotation along the  $x, y, z$  axes with the position of the given point and the Euler angles of rotation along the  $x, y, z$  axes, it can be shown to some extent that the algorithm can guarantee the position accuracy and direction accuracy. The errors of the position at the impact point and the Euler angles rotated along the  $x, y, z$  axes are shown in **Table 5**.

**Tables 4** and **5** are the results. In **Table 4**,  $\theta_1 \sim \theta_6$  indicates the joint angle 1~joint angles 6 that satisfy the condition when the end-effector reaches the impact point after the robot arm moves. In **Table 5**,  ${}^0X_E, {}^0Y_E, {}^0Z_E$  indicates the new position of the robot arm end-effector after moving from the initial position.  ${}^0\alpha_E, {}^0\beta_E, {}^0\eta_E$  indicates the Euler angles of rotation along the  $x, y, z$  axes after the robot arm end-effector moves from the initial position, respectively. The four sets of joint angles in **Table 4** correspond to the positions and directions of the four sets of impact points in **Table 3**. Since there are often multiple sets of joint angles satisfying the conditions when the state of the end-effector of the robotic arm is certain, as shown in (a), (b), and (c)

in **Figure 5** (where the states of the end-effector of the robotic arm in (a), (b), and (c) are the same). Therefore, among the multiple sets of joint angles solved, a set of joint angles satisfying the range of joint angle values was selected as the final solution. By observing the position error of the robotic arm end-effector and the Euler angular error along the  $x, y, z$  axis rotation in the four sets of data in **Table 5**, we can find that the position error is always kept as 0, and the Euler angular error along the  $x, y, z$  axis rotation is between 10–11 and 10–9. Based on the error accuracy, we can initially judge that the algorithm can guarantee the position accuracy and orientation accuracy of the solution.

To evaluate the algorithm more comprehensively, images of the fitness function values and the number of iterations for the first set of joint angles were selected and are shown in **Figures 6A, B**, respectively. **Figure 6A** represents the fitness function values of the first set of joint angles with the number of iterations from 0 to 500. from **Figure 6A**, it can be seen that the fitness function reaches convergence when the number of iterations reaches about 100. **Figure 6B** represents the variation

**TABLE 6 |** Initial condition setting of PUMA560 robot arm.

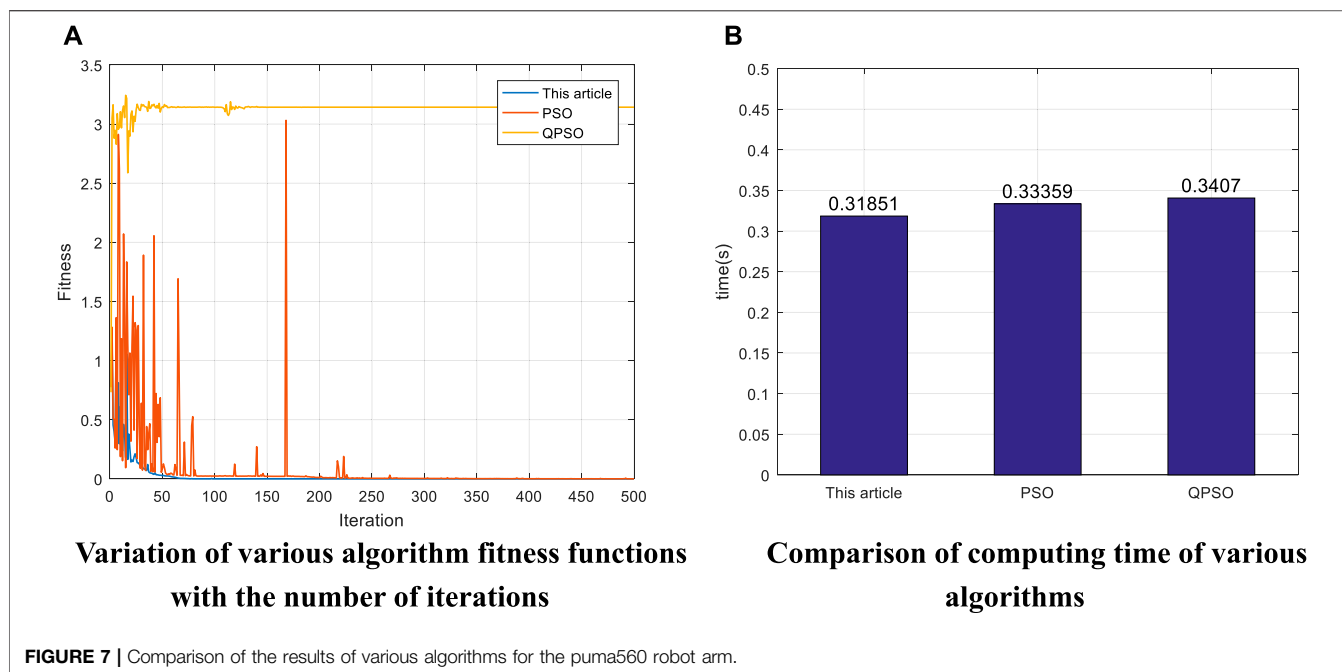
Initial Conditions	Joint i	$\alpha_{i-1}$ (°)	$a_{i-1}$ (m)	$d_i$ (m)	$\theta_i$ (°)	Range of $\theta_i$ (°)	
DH parameter	1	0	0	0	$\theta_1$ (°)	-160-160	
	2	-pi/2	0	0.1491	$\theta_2$ (°)	-110-110	
	3	0	0.4318	0	$\theta_3$ (°)	-135-135	
	4	-pi/2	0.0203	0.4331	$\theta_4$ (°)	-266-266	
	5	pi/2	0	0	$\theta_5$ (°)	-100-100	
	6	-pi/2	0	0	$\theta_6$ (°)	-266-266	
Initial position and Euler angles	<b>Initial position of the end-effector</b>			<b>Initial Euler angles of the end-effector</b>			
	$x_0$ (m)	$y_0$ (m)	$z_0$ (m)	$\alpha_0$ (rad)	$\beta_0$ (rad)	$\eta_0$ (rad)	
Given the position and the Euler angle	N°	<b>Position of end impact point</b>			<b>Euler angles of end impact point</b>		
		${}^0x_E$ (m)	${}^0y_E$ (m)	${}^0x_E$ (m)	${}^0y_E$ (m)	${}^0x_E$ (m)	${}^0y_E$ (m)
Algorithm parameter	Parameters	<b>This paper</b>			<b>PSO</b>	<b>APSO</b>	<b>QPSO</b>
		$c_1$	1.4	1.4	1.4	1.2	-
		$c_2$	1.4	1.4	1.4	1.2	-
		$\omega_{max}$	0.9	0.9	0.9	0.9	-
		$\omega_{min}$	0.7	-	-	0.7	-
		$\beta_0$ and $\beta_1$	-	-	-	-	0.5 and 1

**TABLE 7 |** The six joint angles corresponding to the impact points obtained by different algorithms.

Algorithm	N°	$\theta_1$ (°)	$\theta_2$ (°)	$\theta_3$ (°)	$\theta_4$ (°)	$\theta_5$ (°)	$\theta_6$ (°)
This paper	1	139.147,212	-78.742,965	2.298,743	223.577,471	-47.319,473	-89.781,627
	2	-153.871,884	-72.479,104	44.314,641	-102.235,380	-13.594,398	229.782,832
	3	-133.020650	-37.399,877	59.160,356	176.205,721	42.374,612	-74.592,745
	4	-64.557,618	-59.510,032	75.432,529	-148.731,249	49.034576	-81.522,956
PSO	1	139.145,321	-78.729,959	2.302,142	-136.418,272	-47.307,830	-89.791,711
	2	-153.873,831	-72.478,187	44.301,555	-102.268,096	-13.599,806	229.818,471
	3	-133.017595	-37.394,194	59.156,632	-183.792,886	42.377,145	-74.591,502
	4	-73.090507	-69.438,932	78.661,376	27.906,941	-44.210,988	91.724,240
QPSO	1	9.607,712	-20.610,209	69.598,100	-235.293,713	12.944,303	212.995,170
	2	-118.0051	-40.9856	49.9457	-2.6864	-39.7447	-204.2855
	3	113.9081	73.6642	34.8149	167.6001	90.6805	161.9327
	4	124.0405	45.9792	77.2999	146.3037	98.7763	134.9092

**TABLE 8 |** The errors in position and Euler angles of impact points by different algorithms.

Algorithm	N°	Position Error of the End-Effector			Euler Angles Error of the End-Effector		
		${}^0x_E$ (m)	${}^0y_E$ (m)	${}^0z_E$ (m)	${}^0\alpha_E$ (rad)	${}^0\beta_E$ (rad)	${}^0\eta_E$ (rad)
This paper	1	0	0	0	$-1.70 \times 10^{-9}$	$-1.76 \times 10^{-9}$	$2.03 \times 10^{-9}$
	2	0	0	0	$2.07 \times 10^{-9}$	$-6.62 \times 10^{-9}$	$5.09 \times 10^{-11}$
	3	0	0	0	$2.19 \times 10^{-10}$	$1.29 \times 10^{-8}$	$4.57 \times 10^{-10}$
	4	0	0	0	$-2.35 \times 10^{-9}$	$8.56 \times 10^{-9}$	$1.58 \times 10^{-9}$
PSO	1	$-3.4 \times 10^{-4}$	$2.83 \times 10^{-5}$	$-1.46 \times 10^{-4}$	$2.24 \times 10^{-5}$	$-1.01 \times 10^{-4}$	$-1.90 \times 10^{-4}$
	2	$-7.40 \times 10^{-4}$	$-1.64 \times 10^{-5}$	$1.53 \times 10^{-4}$	$-2.49 \times 10^{-5}$	$2.89 \times 10^{-4}$	$7.41 \times 10^{-6}$
	3	$-5.78 \times 10^{-5}$	$-2.05 \times 10^{-5}$	$-2.01 \times 10^{-4}$	$-4.03 \times 10^{-6}$	$1.45 \times 10^{-5}$	$4.50 \times 10^{-6}$
	4	0.065	-0.019	0.097	$-8.09 \times 10^{-4}$	0.013	-0.0019
QPSO	1	4.66	-1.00	-4.02	-1.00	-1.00	-1.02
	2	17.72	-1.02	-5.41	-1.00	-0.99	-1.02
	3	-11.46	-0.93	-6.45	-1.00	-1.24	-1.00
	4	-6.80	-0.82	-4.35	-1.00	2.13	-0.98



of the fitness function values for the first set of joint angles with the number of iterations from 0 to 50. It can be seen from **Figure 6B** that the fitness function value fluctuates up and down when the number of iterations is small, indicating that the algorithm has a strong search capability, while the fitness function value gradually converges while fluctuating, indicating that the fitness function value gradually approaches the global optimal solution. The combination of **Figure 6A** and **Figure 6B** shows that the algorithm has a certain search ability while converging.

## 6.2 Results Obtained for PUMA560 Robotic Arm

In order to better highlight the advantages of the present algorithm in terms of fast convergence and short time consumption while maintaining accuracy, the second experiment is conducted with the PUMA560 robotic arm as the research object and the paper (Deng and Xie, 2021; Yu et al., 2019) as the reference, by setting the same initial conditions for comparison experiments. The DH parameters of the PUMA560 robotic arm refer to the paper (Deng and Xie, 2021; Tian et al., 2020), and the specific parameter values are shown in **Table 6**. Experiment 2 was the same as experiment 1, and the initial position of the robotic arm end-effector as well as the Euler angles of rotation along the  $x, y, z$  axes were set, as shown in **Table 6**. The actual impact points of four robotic arm end-effectors were also set, and the positions of the end-effectors at this point as well as the Euler angles of rotation along the  $x, y, z$  axes were given, as shown in **Table 6**. The algorithms compared in Experiment 2 contain the algorithm proposed in this paper, the PSO algorithm and the QPSO algorithm. The

relevant parameter settings of the different algorithms are shown in **Table 6**.

In **Table 6**,  $x_0, y_0, z_0$  denotes the initial position of the robotic arm end-effector,  $\alpha_0, \beta_0, \eta_0$  denotes the initial Euler angles of the robotic arm end-effector rotated along the  $x, y,$  and  $z$  axes, respectively.  ${}^0X_E, {}^0Y_E, {}^0Z_E$  denotes the new position of the robotic arm end-effector after moving from the initial position, and  ${}^0\alpha_E, {}^0\beta_E, {}^0\eta_E$  denotes the Euler angles of the robotic arm end-effector rotating along the  $x, y, z$  axes after moving from the initial position, respectively. **Table 7** show the six joint angles corresponding to the impact point obtained by the algorithm proposed in this paper, the PSO algorithm and the QPSO algorithm, respectively. The errors of different algorithms regarding the position and Euler angles of rotation along the  $x, y, z$  axes of the impact point are shown in **Table 8**. Considering that the differences in the initial conditions of different algorithms affect the fairness of the results, the number of iterations is set to 500, and the number of particle swarms is 150.

**Tables 7** and **8** represent the results. In **Table 7**,  $\theta_1 \sim \theta_6$  indicates the joint angle 1~joint angle 6 that satisfy the condition when the end-effector reaches the impact point after the robot arm moves. In **Table 8**,  ${}^0X_E, {}^0Y_E, {}^0Z_E$  indicates the new position of the robot arm end-effector after moving from the initial position.  ${}^0\alpha_E, {}^0\beta_E, {}^0\eta_E$  indicates the Euler angles of rotation along the  $x, y, z$  axes after the robot arm end-effector moves from the initial position, respectively. By comparing the error in **Table 8**, it can be found that the position error of the algorithm proposed in this paper is 0, and the orientation error is between  $5.09 \times 10^{-11}$  and  $1.29 \times 10^{-8}$ . The position error of the conventional PSO algorithm is between -0.019 and 0.097, and the orientation error is between -0.019 and 0.013. The position error of the QPSO algorithm is between -11.46 and 17.72, and the orientation error is between -1.24 and 2.13. The

**TABLE 9 |** Initial condition setting of Seven degrees of freedom robot arm.

Initial Conditions	Joint i	$\alpha_{i-1}$ (°)	$a_{i-1}$ (m)	$d_i$ (m)	$\theta_i$ (°)	Range of $\theta_i$ (°)
DH parameter	1	-pi/2	0	0.5	$\theta_1$ (°)	-180-180
	2	pi/2	0.2	0	$\theta_2$ (°)	-90-30
	3	-pi/2	0.25	0	$\theta_3$ (°)	-90-120
	4	pi/2	0.3	0	$\theta_4$ (°)	-90-90
	5	-pi/2	0.2	0	$\theta_5$ (°)	-90-90
	6	0	0.2	0	$\theta_6$ (°)	-90-90
	7	0	0.1	0.05	$\theta_7$ (°)	-30-90
Initial position and Euler angles	<b>Initial position of the end-effector</b>			<b>Initial Euler angles of the end-effector</b>		
	$x_0$ (m)	$y_0$ (m)	$z_0$ (m)	$\alpha_0$ (rad)	${}^0\beta_E$ (rad)	$\eta_0$ (rad)
	1.250	0.050	0.500	0	0.0175	0
Given the position and the Euler angle	<b>Position of end impact point</b>			<b>Euler angles of end impact point</b>		
	${}^0x_E$ (m)	${}^0y_E$ (m)	${}^0z_E$ (m)	${}^0\alpha_E$ (rad)	${}^0\beta_E$ (rad)	${}^0\eta_E$ (rad)
	-0.5839	-0.7154	-0.5027	0.5977	0.7025	-1.69

**TABLE 10 |** Joint angles according to different algorithms.

Algorithm	$\theta_1$ (°)	$\theta_2$ (°)	$\theta_3$ (°)	$\theta_4$ (°)	$\theta_5$ (°)	$\theta_6$ (°)	$\theta_7$ (°)
This article	41.2034	5.3333	48.9559	49.5050	48.3672	43.5106	42.0882
PSO	64.2282	4.7456	23.8832	48.4828	51.2608	62.2661	23.7729
QPSO	69.7373	2.6296	32.7903	57.5889	42.0689	15.3571	63.6031

**TABLE 11 |** Position and orientation errors according to different algorithms.

Algorithm	Position Error of End-Effector			Euler Error of the End-Effector		
	${}^0x_E$ (m)	${}^0y_E$ (m)	${}^0z_E$ (m)	${}^0\alpha_E$ (rad)	${}^0\beta_E$ (rad)	${}^0\eta_E$ (rad)
This article	0	0	0	0	0	0
PSO	$7.53 \times 10^{-4}$	0.0010	-0.0025	-0.0010	$5.69 \times 10^{-4}$	$-2.96 \times 10^{-4}$
QPSO	-1.77	-2.42	-1.42	2.78	-0.58	-1.62

comparison shows that the proposed algorithm can guarantee higher position and orientation accuracy compared with the traditional PSO algorithm and QPSO algorithm. The joint angles obtained in **Table 7** are all within the range of values, so it can be determined that there is a set of joint angles that allow the end-effector to reach the impact point, i.e., the error comparison of each algorithm is meaningful. To further illustrate the advantages of the algorithms in terms of convergence and operation speed, the first set of data in **Table 6** was selected in Experiment 2 to compare the variation of the fitness function with the number of iterations and the operation time in each algorithm, and the results are shown in **Figures 7A,B**, respectively.

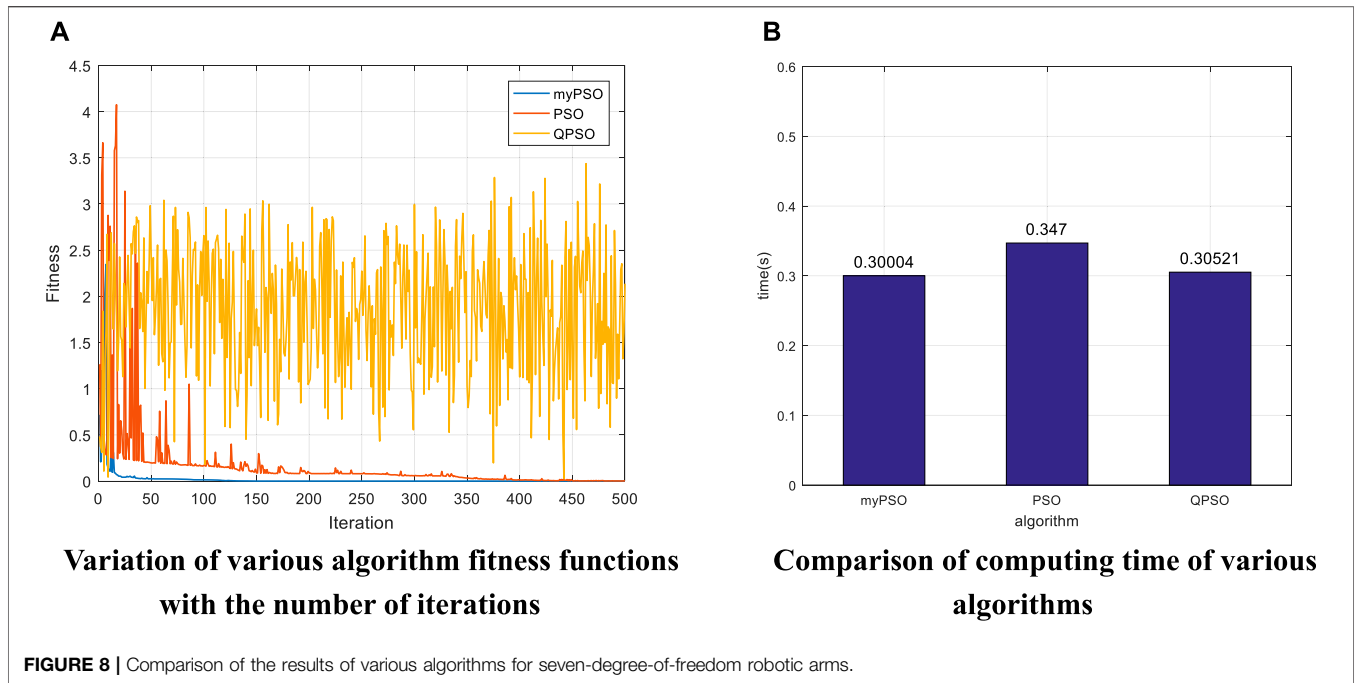
**Figure 7A** summarizes the variation of the fitness function with the number of iterations in various algorithms. It can be seen that the algorithm proposed in this paper converges steadily when the number of iterations reaches about 50. In contrast, the PSO algorithm fluctuates more and does not converge significantly, and the algorithm fluctuates more when it is close to convergence and the number of iterations reaches about 175, indicating that the algorithm is prone to fall into the local optimum and thus misses the

optimal solution, resulting in poor solution accuracy. Compared with the first two algorithms, the QPSO algorithm is unable to maintain convergence. Through comparison, it can be found that the algorithm proposed in this paper converges faster and at the same time ensures stable convergence. **Figure 7B** shows the time required for various algorithms to run the first set of data in **Table 8**. The histogram shows that the algorithm proposed in this paper has the shortest operation time of 0.31851s, while the PSO algorithm and QPSO algorithm have an operation time of 0.33359 and 0.3407s, respectively. By comparing several factors such as end-effector position accuracy, direction accuracy, convergence of the algorithm and operation time, the algorithm proposed in this paper has a good performance.

### 6.3 Results Obtained for the Seven-Degree-of-Freedom Robotic Arm

In order to reflect the wide applicability of the present algorithm, the third experiment is conducted with a seven-degree-of-freedom robotic arm as the object of study, and the paper (Dereli S, Koker R 2019) is used as a reference for comparison experiments by setting the same initial conditions. Where the DH





parameters of the seven-degree-of-freedom robotic arm are set as shown in the paper (Dereli and Koker, 2020; Li et al., 2019a; Li et al., 2019b; Li et al., 2019c). The specific DH parameter table is shown in Table 9. Experiment 3 set up a set of initial positions of the robotic arm end-effectors as well as the Euler angles of rotation along the  $x, y, z$  axes, as shown in Table 9. The actual impact point of a set of robotic arm end-effectors was also set, and the position of the end-effectors at this point as well as the Euler angles of rotation along the  $x, y, z$  axes were given, as shown in Table 9. The algorithms compared in Experiment 3 contain the algorithm proposed in this paper, the PSO algorithm, and the QPSO algorithm.

In Table 9,  $x_0, y_0, z_0$  denotes the initial position of the robotic arm end-effector,  $\alpha_0, \beta_0, \eta_0$  denotes the initial Euler angles of the robotic arm end-effector rotated along the  $x, y,$  and  $z$  axes, respectively.  ${}^0X_E, {}^0Y_E, {}^0Z_E$  denotes the new position of the robotic arm end-effector after moving from the initial position, and  ${}^0\alpha_E, {}^0\beta_E, {}^0\eta_E$  denotes the Euler angles of the robotic arm end-effector rotating along the  $x, y, z$  axes after moving from the initial position, respectively. The six joint angles corresponding to the impact points obtained by different algorithms are shown in Table 10. The errors of different algorithms regarding the position and Euler angles of rotation along the  $x, y, z$  axes of the impact points are shown in Table 11. Considering that the differences in the initial conditions of different algorithms affect the fairness of the results, the number of iterations is set to 500, and the number of particle swarms is 150.

Tables 10 and 11 represent the results. In Table 10,  $\theta_1 \sim \theta_7$  indicates the joint angle 1~joint angles 7 that satisfy the condition when the end-effector reaches the impact point after the robot arm moves.  ${}^0X_E, {}^0Y_E, {}^0Z_E$  indicates the new position of the robot arm end-effector after moving from the initial position.

${}^0\alpha_E, {}^0\beta_E, {}^0\eta_E$  indicates the Euler angles of rotation along the  $x, y, z$  axes after the robot arm end-effector moves from the initial position, respectively. Through the error in Table 11, it can be found that the position error of the algorithm proposed in this paper is 0 and the orientation error is 0. The position error of the conventional PSO algorithm is between  $-0.0025$  and  $0.0010$ , and the orientation error is between  $-0.0010$  and  $5.69 \times 10^{-4}$ . The position error of the QPSO algorithm is between  $-2.42$  and  $-1.42$ , and the orientation error is between  $-1.62$  and  $2.78$ . The comparison shows that when the object is a seven-degree-of-freedom robot arm, the proposed algorithm can still guarantee higher position and orientation accuracy compared with the traditional PSO algorithm and QPSO algorithm. The joint angles obtained in Table 10 are all within the range of values, so it can be determined that the joint angles enable the end-effector to reach the impact point, i.e., the comparison of the errors of each algorithm is meaningful. To further illustrate the advantages of the algorithms in terms of convergence and operation speed, the data in Table 9 were selected for Experiment 3 to compare the variation of the fitness function with the number of iterations and the operation time in each algorithm, and the results are shown in Figures 8A,B, respectively.

Figure 8A summarizes the variation of the fitness function with the number of iterations in various algorithms. It can be seen that the algorithm proposed in this paper converges faster than the other two algorithms, converging at less than 100 iterations. The fact that the value of the fitness function does not fluctuate after convergence indicates that the convergence is more stable. In contrast, the PSO algorithm converges slowly, reaching about 350 iterations before convergence, and there are still small fluctuations when convergence is near. Compared with the first two algorithms, the QPSO algorithm is unable to

**TABLE12** | Comparison with some studies in the literature.

Research	Robotic Arm	Technique Used	Comparative Technique
(Mustafa A and Kerim C 2016)	4DOF	QPSO 6.51e-06 1.96	GA 3.96e-04 17.53
Liu et al. (2021a)	6DOF	PLPSO 2.5918e-15 4.4982	DE 1.0798e-07 8.3929
Dereli and Koker. (2018)	7DOF	Random IW-PSO 6.20e-03 1.6	Global-Local Best IW-PSO 3.64e-03 1.2
(Serkan D, Rasit K, et al., 2019)	7DOF	QPSO 2.775e-17 0.2319	PSO 6.719e-03 0.4498

maintain convergence. The comparison shows that the proposed algorithm can still maintain fast convergence and stable convergence compared to the other two algorithms when the object of study is a seven-degree-of-freedom robot arm. **Figure 8B** shows the running time of various algorithms. The histogram shows that the algorithm proposed in this paper has the shortest operation time of 0.30004s, while the operation time of PSO and QPSO algorithms are 0.347 and 0.30521s respectively. By replacing the experimental object and comparing several factors such as end-effector position accuracy, direction accuracy, convergence of the algorithm and operation time, the algorithm proposed in this paper can still maintain good performance, which can show that the algorithm proposed in this paper has great potential in solving the inverse kinematics problem of multi-degree-of-freedom robotic arm.

## 6.4 DISCUSSIONS

Robot inverse kinematics solutions are of great importance because they are the basis for the subsequent study of robot path planning and control. The existing robot inverse kinematics solutions, such as analytical and geometric methods, can be applied to a limited number of scenarios, and they are mainly used for robotic arms that satisfy the “Pieper” criterion and have analytical solutions. The intelligent algorithm represented by particle swarm algorithm is becoming a more promising and exploitable means of solving robot inverse kinematics with high accuracy, short time and wide range of application. In this paper, the traditional particle swarm algorithm is used as the basis, and its inertia weights as well as the initial population are mainly optimized. Based on the characteristics that large inertia weights are suitable for global search and small weights are suitable for local search, an adaptive weighting strategy is proposed in this paper. Adjusting the inertia weights according to the change of the fitness function can effectively reduce the probability of the function value falling into the local optimum. According to **Figures 7A, 8A**, it can be seen that when the fitness function is close to convergence, the traditional PSO algorithm still has a large undulation phenomenon, i.e., the situation that the value of the fitness function falls into a local minimum, while by introducing the adaptive weight strategy, the undulation phenomenon at convergence is effectively reduced, as shown in

the blue curve of **Figure 7A**. In this paper, we propose a condition setting based on the limit joints. Firstly, the maximum and minimum values of each joint angle are determined according to the range of joint angles, then the position factor  $k$  is introduced (through several experiments,  $k = 0.5$  is finally taken), and finally the velocity factor is determined by the position factor  $k$  as well as the maximum joint angle. Due to the introduction of the position coefficient  $k$  in the velocity factor, a reasonable reference standard is provided for the initialization of position and velocity. At the same time, the speed of the algorithm is improved by introducing the condition setting of the limit joints before the iteration instead of continuously performing the boundary detection during the iteration. Finally, an exponential product form modeling method (POE) based on spinor theory is chosen. Compared with the traditional DH modeling method, the spinor approach describes the motion of a rigid body as a whole and avoids the singularities that arise when described by a local coordinate system. The above three experiments confirm the advantages of the algorithm proposed in this paper in terms of solution accuracy, operation speed, convergence of the algorithm and applicability. Experiment 1 takes a general six-degree-of-freedom industrial robotic arm as the research object, and sets up four sets of robotic arm end-effector impact point positions and Euler angles of rotation along the  $x$ ,  $y$ ,  $z$  axes to verify the algorithm. **Table 5** shows that the final end-effector position error solved by the algorithm is 0 and the orientation error is  $10^{-11}$ – $10^{-9}$ , which can initially show that the algorithm can guarantee the position accuracy and orientation accuracy of the end-effector. Meanwhile, the variation of the fitness function with the number of iterations in **Figures 6A,B** of Experiment 1 can preliminarily show that the algorithm has convergence. Experiment 2 takes the PUMA560 robotic arm as the research object and compares the algorithm proposed in this paper with a variety of algorithms by replacing the data taking the same verification method as Experiment 1. The error in **Table 8** show that the position error of the proposed algorithm is 0 and the maximum orientation error is  $1.29 \times 10^{-8}$ , while the lowest position error of other algorithms is  $-1.64 \times 10^{-5}$  and the lowest orientation error is  $4.5 \times 10^{-6}$ . The data in **Figure 7B** show that the running time of the proposed algorithm is 0.31851s, while the shortest running time of other algorithms is 0.33359s. The curve in **Figure 7A** shows that the proposed algorithm converges faster and more stably than the other algorithms. Through various comparisons,

it can be found that the proposed algorithm has higher accuracy of position and direction solving, faster operation speed, and faster and more stable convergence than other algorithms. Experiment 3 takes a seven-degree-of-freedom manipulator as the research object and sets up one group of robot arm end-effector impact point positions as well as Euler angles of rotation along the  $x, y, z$  axes to verify the algorithm. The error in **Table 11** shows that the position error of the proposed algorithm is 0 and the orientation error is 0, while the lowest position error of the other algorithms is  $7.53 \times 10^{-4}$  and the lowest orientation error is  $-2.69 \times 10^{-4}$ . The data in **Figure 8B** shows that the running time of the proposed algorithm is 0.30004s, while the shortest running time of the other algorithms is 0.30521s. The curve in **Figure 8A** shows that the proposed algorithm converges faster and more stably than the other algorithms. Through various comparisons, it can be found that the proposed algorithm can maintain its own advantages for different research objects and has wide applicability. This paper also summarizes the improved PSO algorithm applied to the inverse kinematics solution of robotic arm, and compares different algorithms in terms of both position error and solution time, as shown in **Table 12**.

## 7 CONCLUSION

In this paper, the algorithm is verified and compared in terms of solution accuracy, operation time and convergence through three experiments. Experiment 1 takes a general six-degree-of-freedom industrial robotic arm as the research object, and sets up four groups of robotic arm end-effector impact point positions as well as postures. By bringing the relevant parameters of the robotic arm, the range of joint angle values, the initial postures and positions of the robotic arm end-effectors into the algorithm, the joint angles that meet the conditions are obtained, and finally the actual robotic arm end-effector positions and postures are obtained. The convergence of the algorithm is also initially illustrated by the variation of a selected set of fitness functions with the number of iterations. The final position error brought into the algorithm is 0, and the orientation error interval is  $10^{-9} \sim 10^{-11}$ , which preliminarily illustrates that the algorithm can guarantee a certain accuracy of position and orientation solution. Experiment 2 compares the algorithm proposed in this paper with the traditional particle swarm algorithm (PSO) and quantum particle swarm algorithm (QPSO) in terms of solution accuracy, operation time and convergence, using the PUMA560 robotic arm as the research object. In which the experimental approach is consistent with Experiment 1, the position error of the algorithm proposed in this paper is 0, the maximum direction error is  $1.29 \times 10^{-8}$ , and the operation time of a set of data is 0.31851s. The minimum position error of the other two algorithms is  $-1.64 \times 10^{-5}$ , the minimum direction error is  $-4.03 \times 10^{-6}$ , and the operation time of a set of data is 0.33359s. At the same time, by comparing the changes of the fitness function with the number of iterations in a set of data, it can be found that the algorithm proposed in this paper converges faster and more stably than other algorithms. Finally, through various comparisons, it can be found that the algorithm proposed in this paper can guarantee high accuracy of position and direction

solving, faster solving speed, and more stable and faster convergence. Experiment 3 takes a seven-degree-of-freedom robotic arm as the object of study, sets up a group of robotic arm end-effector impact point positions and postures and repeats the operation steps of experiment 2, in which the position error of the proposed algorithm is 0, the orientation error is 0, and the operation time is 0.30004s. The minimum position error of the other two algorithms is  $7.53 \times 10^{-4}$ , the minimum orientation error is  $-2.96 \times 10^{-4}$ , and the minimum operation time is 0.30521s. At the same time, by comparing the changes of the fitness function with the number of iterations, we can find that the algorithm proposed in this paper still maintains the advantages of stable and fast convergence. By replacing different experimental objects and comparing with various algorithms, it can be found that the algorithm proposed in this paper can maintain its advantages in solution accuracy, operation time and convergence while having strong applicability, which indicates that the algorithm has greater potential in solving the inverse kinematics of multi-degree-of-freedom robotic arm. In the future, we will start from improving the stability of the particle swarm algorithm, and strive to find the ideal solution with less number of trials. In addition, the study of robot arm motion control will also be carried out in the follow-up work.

## DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding authors.

## AUTHOR CONTRIBUTIONS

GZ and DJ provided research ideas and plans; XL, XT and YS wrote programs and conducted experiments. BT analyzed and explained the simulation results; JK improved the algorithm. JY co-authored the manuscript, and were responsible for collecting data; YL and ZF revised the manuscript for the corresponding author and approved the final submission.

## FUNDING

This work was supported by grants of the National Natural Science Foundation of China (Grant Nos.52,075,530, 51,575,407, 51,505,349, 51,975,324, 61,733,011, 41,906,177); the Grants of Hubei Provincial Department of Education (D20191105); the Grants of National Defense PreResearch Foundation of Wuhan University of Science and Technology (GF201705) and Open Fund of the Key Laboratory for Metallurgical Equipment and Control of Ministry of Education in Wuhan University of Science and Technology (2018B07, 2019B13) and Open Fund of Hubei Key Laboratory of Hydroelectric Machinery Design & Maintenance in Three Gorges University(2020KJX02, 2021KJX13).

## REFERENCES

- Aydilek, İ. B. (2018). A Hybrid Firefly and Particle Swarm Optimization Algorithm for Computationally Expensive Numerical Problems. *Appl. Soft Comput.* 66, 232–249. doi:10.1016/j.asoc.2018.02.025
- Ayyıldız, M., and Çetinkaya, K. (2016). Comparison of Four Different Heuristic Optimization Algorithms for the Inverse Kinematics Solution of a Real 4-DOF Serial Robot Manipulator. *Neural Comput. Appl.* 27 (4), 825–836. doi:10.1007/s00521-015-1898-8
- Bai, D., Sun, Y., Tao, B., Tong, X., Xu, M., Jiang, G., et al. (2022). Improved Single Shot Multibox Detector Target Detection Method Based on Deep Feature Fusion. *Concurrency Comput.* 34 (4), e6614. doi:10.1002/CPE.6614
- Chen, T., Peng, L., Yang, J., Cong, G., and Li, G. (2021a). Evolutionary Game of Multi-Subjects in Live Streaming and Governance Strategies Based on Social Preference Theory during the COVID-19 Pandemic. *Mathematics* 9 (21), 2743. doi:10.3390/math9212743
- Chen, T., Qiu, Y., Wang, B., and Yang, J. (2022). Analysis of Effects on the Dual Circulation Promotion Policy for Cross-Border E-Commerce B2B export Trade Based on System Dynamics during COVID-19. *systems* 10 (1), 13. doi:10.3390/systems10010013
- Chen, T., Yin, X., Yang, J., Cong, G., and Li, G. (2021b). Modeling Multi-Dimensional Public Opinion Process Based on Complex Network Dynamics Model in the Context of Derived Topics. *Axioms* 10 (4), 270. doi:10.3390/axioms10040270
- Chen, Y., Li, L., Peng, H., Xiao, J., Yang, Y., and Shi, Y. (2017). Particle Swarm Optimizer with Two Differential Mutation. *Appl. Soft Comput.* 61, 314–330. doi:10.1016/j.asoc.2017.07.020
- Cheng, Y., Li, G., Li, J., Sun, Y., Jiang, G., Zeng, F., et al. (2020). Visualization of Activated Muscle Area Based on sEMG. *Ijs* 38 (3), 2623–2634. doi:10.3233/JIFS-179549
- Cheng, Y., Li, G., Liu, Y., Liu, Y., Yu, M., and Jiang, D. U. (2021). Gesture Recognition Based on Surface Electromyography-Feature Image. *Concurrency Comput. Pract. Experience* 33 (6), e6051. doi:10.1002/cpe.6051
- Deng, H., and Xie, C. (2021). An Improved Particle Swarm Optimization Algorithm for Inverse Kinematics Solution of Multi-DOF Serial Robotic Manipulators. *Soft Comput.* 25, 13695–13708. doi:10.1007/s00500-021-06007-6
- Dereli, S., and Köker, R. (2018). IW-PSO Approach to the Inverse Kinematics Problem Solution of a 7-DOF Serial Robot Manipulator. *Int. J. Nat. Eng. Sci.* 36 (1), 75–85.
- Dereli, S., and Köker, R. (2020). A Meta-Heuristic Proposal for Inverse Kinematics Solution of 7-DOF Serial Robotic Manipulator: Quantum Behaved Particle Swarm Algorithm. *Artif. Intell. Rev.* 53 (2), 949–964. doi:10.1007/s10462-019-09683-x
- Duan, H., Sun, Y., Cheng, W., Jiang, D., Yun, J., Liu, Y., et al. (2021). Gesture Recognition Based on Multi-modal Feature Weight. *Concurrency Comput. Pract. Experience* 33 (5), e5991. doi:10.1002/cpe.5991
- El-Sherbiny, A., Elhosseini, M. A., and Haikal, A. Y. (2018). A Comparative Study of Soft Computing Methods to Solve Inverse Kinematics Problem. *Ain Shams Eng. J.* 9, 2535–2548. doi:10.1016/j.asej.2017.08.001
- Hao, Z., Wang, Z., Bai, D., Tao, B., Tong, X., and Chen, B. (2021b). Intelligent Detection of Steel Defects Based on Improved Split Attention Networks. *Front. Bioeng. Biotechnol.* 9. doi:10.3389/fbioe.2021.810876
- Hao, Z., Wang, Z., Bai, D., and Zhou, S. (2021a). Towards the Steel Plate Defect Detection: Multidimensional Feature Information Extraction and Fusion. *Concurrency Computat. Pract. Exper* 33 (21), e6384. doi:10.1002/CPE.6384
- Harrison, K. R., Engelbrecht, A. P., and Ombuki-Berman, B. M. (2016). Inertia Weight Control Strategies for Particle Swarm Optimization. *Swarm Intell.* 10 (4), 267–305. doi:10.1007/s11721-016-0128-z
- He, Y., Li, G., Liao, Y., Sun, Y., Kong, J., Jiang, G., et al. (2019). Gesture Recognition Based on an Improved Local Sparse Representation Classification Algorithm. *Cluster Comput.* 22 (Suppl. 5), 10935–10946. doi:10.1007/s10586-017-1237-1
- Huang, L., Fu, Q., He, M., Jiang, D., and Hao, Z. (2021). Detection Algorithm of Safety Helmet Wearing Based on Deep Learning. *Concurrency Computat. Pract. Exper* 33 (13), e6234. doi:10.1002/cpe.6234
- Huang, L., Fu, Q., Li, G., Luo, B., Chen, D., and Yu, H. (2019). Improvement of Maximum Variance Weight Partitioning Particle Filter in Urban Computing and Intelligence. *IEEE Access* 7, 106527–106535. doi:10.1109/ACCESS.2019.2932144
- Huang, L., He, M., Tan, C., Jiang, D., Li, G., and Yu, H. (2020). Jointly Network Image Processing: Multi-task Image Semantic Segmentation of Indoor Scene Based on CNN. *IET image process* 14 (15), 3689–3697. doi:10.1049/iet-ipr.2020.0088
- Jiang, D., Li, G., Sun, Y., Hu, J., Yun, J., and Liu, Y. (2021a). Manipulator Grabbing Position Detection with Information Fusion of Color Image and Depth Image Using Deep Learning. *J. Ambient Intell. Hum. Comput* 12 (12), 10809–10822. doi:10.1007/s12652-020-02843-w
- Jiang, D., Li, G., Sun, Y., Kong, J., Tao, B., and Chen, D. (2019b). Grip Strength Forecast and Rehabilitative Guidance Based on Adaptive Neural Fuzzy Inference System Using sEMG. *Pers Ubiquit Comput.* doi:10.1007/s00779-019-01268-3
- Jiang, D., Li, G., Sun, Y., Kong, J., and Tao, B. (2019a). Gesture Recognition Based on Skeletonization Algorithm and CNN with ASL Database. *Multimed Tools Appl.* 78 (21), 29953–29970. doi:10.1007/s11042-018-6748-0
- Jiang, D., Li, G., Tan, C., Huang, L., Sun, Y., and Kong, J. (2021b). Semantic Segmentation for Multiscale Target Based on Object Recognition Using the Improved Faster-RCNN Model. *Future Generation Computer Syst.* 123, 94–104. doi:10.1016/j.future.2021.04.019
- Jiang, D., Zheng, Z., Li, G., Sun, Y., Kong, J., Jiang, G., et al. (2019c). Gesture Recognition Based on Binocular Vision. *Cluster Comput.* 22 (Suppl. 6), 13261–13271. doi:10.1007/s10586-018-1844-5
- Kucuk, S., and Bingul, Z. (2014). Inverse Kinematics Solutions for Industrial Robot Manipulators with Offset Wrists. *Appl. Math. Model.* 38 (7-8), 1983–1999. doi:10.1016/j.apm.2013.10.014
- Li, C., Li, G., Jiang, G., Chen, D., and Liu, H. (2020). Surface EMG Data Aggregation Processing for Intelligent Prosthetic Action Recognition. *Neural Comput. Appl.* 32 (22), 16795–16806. doi:10.1007/s00521-018-3909-z
- Li, G., Jiang, D., Zhou, Y., Jiang, G., Kong, J., and Manogaran, G. (2019a). Human Lesion Detection Method Based on Image Information and Brain Signal. *IEEE Access* 7, 11533–11542. doi:10.1109/ACCESS.2019.2891749
- Li, G., Li, J., Ju, Z., Sun, Y., and Kong, J. (2019b). A Novel Feature Extraction Method for Machine Learning Based on Surface Electromyography from Healthy Brain. *Neural Comput. Appl.* 31 (12), 9013–9022. doi:10.1007/s00521-019-04147-3
- Li, G., Zhang, L., Sun, Y., and Kong, J. (2019c). Towards the Semg Hand: Internet of Things Sensors and Haptic Feedback Application. *Multimed Tools Appl.* 78 (21), 29765–29782. doi:10.1007/s11042-018-6293-x
- Li, Z., Li, G., Jiang, G., Fang, Y., Ju, Z., and Liu, H. (2015b). Computation of Grasping and Manipulation for Multi-Fingered Robotic Hands. *J. Comput. Theor. Nanosci* 12 (3), 6192–6197. doi:10.1166/jctn.2015.4655
- Li, Z., Wang, W., Yan, Y., and Li, Z. (2015a). PS-ABC: A Hybrid Algorithm Based on Particle Swarm and Artificial Bee colony for High-Dimensional Optimization Problems. *Expert Syst. Appl.* 42 (22), 8881–8895. doi:10.1016/j.eswa.2015.07.043
- Liao, S., Li, G., Wu, H., Jiang, D., Liu, Y., Yun, J., et al. (2021). Occlusion Gesture Recognition Based on Improved SSD. *Concurrency Comput. Pract. Experience* 33 (6), e6063. doi:10.1002/cpe.6063
- Liao, S., Li, G., Li, J., Jiang, D., Jiang, G., Sun, Y., et al. (2020). Multi-object Intergroup Gesture Recognition Combined with Fusion Feature and KNN Algorithm. *Ijs* 38 (3), 2725–2735. doi:10.3233/JIFS-179558
- Lim, W. H., and Mat Isa, N. A. (2014). An Adaptive Two-Layer Particle Swarm Optimization with Elitist Learning Strategy. *Inf. Sci.* 273, 49–72. doi:10.1016/j.ins.2014.03.031
- Liu, F., Huang, H., Li, B., and Xi, F. (2021a). A Parallel Learning Particle Swarm Optimizer for Inverse Kinematics of Robotic Manipulator. *Int. J. Intell. Syst.* 36, 6101–6132. doi:10.1002/int.22543
- Liu, X., Jiang, D., Tao, B., Jiang, G., Sun, Y., Kong, J., et al. (2021b). Genetic Algorithm-Based Trajectory Optimization for Digital Twin Robots. *Front. Bioeng. Biotechnol.* 9. doi:10.3389/fbioe.2021.793782
- Liu, Y., Xiao, F., Tong, X., Tao, B., Xu, M., Jiang, G., et al. (2022c). Manipulator Trajectory Planning Based on Work Subspace Division. *Concurrency Comput. Pract. Experience* 34 (5), e6710. doi:10.1002/cpe.6710
- Liu, Y., Jiang, D., Duan, H., Sun, Y., Li, G., Tao, B., et al. (2021c). Dynamic Gesture Recognition Algorithm Based on 3D Convolutional Neural Network. *Comput. Intelligence Neurosci.* 2021, 1–12. doi:10.1155/2021/4828102
- Liu, Y., Jiang, D., Tao, B., Qi, J., Jiang, G., Yun, J., et al. (2022a). Grasping Posture of Humanoid Manipulator Based on Target Shape Analysis and Force Closure. *Alexandria Eng. J.* 61 (5), 3959–3969. doi:10.1016/j.aej.2021.09.017
- Liu, Y., Jiang, D., Yun, J., Sun, Y., Li, C., Jiang, G., et al. (2021d). Self-tuning Control of Manipulator Positioning Based on Fuzzy PID and PSO Algorithm. *Front. Bioeng. Biotechnol.* 9. doi:10.3389/fbioe.2021.817723
- Liu, Y., Li, C., Jiang, D., Chen, B., Sun, N., Cao, Y., et al. (2022b). Wrist Angle Prediction under Different Loads Based on GA-ELM Neural Network and Surface Electromyography. *Concurrency Comput.* 34 (3), e6574. doi:10.1002/CPE.6574

- Liu, Y., Xu, M., Jiang, G., Tong, X., Yun, J., Liu, Y., et al. (2022d). Target Localization in Local Dense Mapping Using RGBD SLAM and Object Detection. *Concurrency Comput.* 34 (4), e6655. doi:10.1002/CPE.6655
- Luo, B., Sun, Y., Li, G., Chen, D., and Ju, Z. (2020). Decomposition Algorithm for Depth Image of Human Health Posture Based on Brain Health. *Neural Comput. Applic* 32 (10), 6327–6342. doi:10.1007/s00521-019-04141-9
- Ma, R., Zhang, L., Li, G., Jiang, D., Xu, S., and Chen, D. (2020). Grasping Force Prediction Based on sEMG Signals. *Alexandria Eng. J.* 59 (3), 1135–1147. doi:10.1016/j.aej.2020.01.007
- Mao, B., Xie, Z., Wang, Y., Handroos, H., Wu, H., and Shi, S. (2017). A Hybrid Differential Evolution and Particle Swarm Optimization Algorithm for Numerical Kinematics Solution of Remote Maintenance Manipulators. *Fusion Eng. Des.* 124, 587–590. doi:10.1016/j.fusengdes.2017.03.042
- Netjinda, N., Achalakul, T., and Sirinaovakul, B. (2015). Particle Swarm Optimization Inspired by Starling Flock Behavior. *Appl. Soft Comput.* 35, 411422–422. doi:10.1016/j.asoc.2015.06.052
- Ngo, T. T., Sadollah, A., and Kim, J. H. (2016). A Cooperative Particle Swarm Optimizer with Stochastic Movements for Computationally Expensive Numerical Optimization Problems. *J. Comput. Sci.* 13, 68–82. doi:10.1016/j.jocs.2016.01.004
- Pathak, M., Junco, J., and Ram, R. V. (2019). Inverse Kinematics of mobile Manipulator Using Bidirectional Particle Swarm Optimization by Manipulator Decoupling. *Mechanism Machine Theor.* 131, 385–405.
- Shastri, S., Parvez, Y., and Chauhan, N. R. (2019). Inverse Kinematics for a 3-R Robot Using Artificial Neural Network and Modified Particle Swarm Optimization. *J. Inst. Eng. India Ser. C* 101 (4), 355–363. doi:10.1007/s40032-019-00539-5
- Sun, Y., and Zhao, Z. (2022). Low-illumination Image Enhancement Algorithm Based on Improved Multi-Scale Retinex and ABC Algorithm Optimization. *Front. Bioeng. Biotechnol.* doi:10.3389/fbioe.2022.843020
- Sun, Y., Hu, J., Li, G., Jiang, G., Xiong, H., Tao, B., et al. (2020a). Gear Reducer Optimal Design Based on Computer Multimedia Simulation. *J. Supercomput* 76 (6), 4132–4148. doi:10.1007/s11227-018-2255-3
- Sun, Y., Li, C., Li, G., Jiang, G., Jiang, D., Liu, H., et al. (2018). Gesture Recognition Based on Kinect and sEMG Signal Fusion. *Mobile Netw. Appl.* 23 (4), 797–805. doi:10.1007/s11036-018-1008-0
- Sun, Y., Tian, J., Jiang, D., Tao, B., Liu, Y., Yun, J., et al. (2020b). Numerical Simulation of thermal Insulation and Longevity Performance in New Lightweight Ladle. *Concurrency Computat Pract. Exper* 32 (22), e5830. doi:10.1002/CPE.5830
- Sun, Y., Weng, Y., Luo, B., Li, G., Tao, B., Jiang, D., et al. (2020c). Gesture Recognition Algorithm Based on Multi-scale Feature Fusion in RGB-D Images. *IET image process* 14 (15), 3662–3668. doi:10.1049/iet-ipr.2020.0148
- Sun, Y., Xu, C., Li, G., Xu, W., Kong, J., Jiang, D., et al. (2020d). Intelligent Human Computer Interaction Based on Non Redundant EMG Signal. *Alexandria Eng. J.* 59 (3), 1149–1157. doi:10.1016/j.aej.2020.01.015
- Sun, Y., Yang, Z., Tao, B., Jiang, G., Hao, Z., and Chen, B. (2021). Multiscale Generative Adversarial Network for Real-world Super-resolution. *Concurrency Computat Pract. Exper* 33 (21), e6430. doi:10.1002/CPE.6430
- Taherkhani, M., and Safabakhsh, R. (2016). A Novel Stability-Based Adaptive Inertia Weight for Particle Swarm Optimization. *Appl. Soft Comput.* 38, 281–295. doi:10.1016/j.asoc.2015.10.004
- Tan, C., Sun, Y., Li, G., Jiang, G., Chen, D., and Liu, H. (2020). Research on Gesture Recognition of Smart Data Fusion Features in the IoT. *Neural Comput. Applic* 32 (22), 16917–16929. doi:10.1007/s00521-019-04023-0
- Tanweer, M. R., Suresh, S., and Sundararajan, N. (2016). Dynamic Mentoring and Self-Regulation Based Particle Swarm Optimization Algorithm for Solving Complex Real-World Optimization Problems. *Inf. Sci.* 326, 1–24. doi:10.1016/j.ins.2015.07.035
- Tao, B., Huang, L., Zhao, H., Li, G., and Tong, X. (2021). A Time Sequence Images Matching Method Based on the Siamese Network. *Sensors* 21 (17), 5900. doi:10.3390/s21175900
- Tao, B., Liu, Y., Huang, L., Chen, G., and Chen, B. (2022a). 3D Reconstruction Based on Photoelastic Fringes. *Concurrency Computat Pract. Exper* 34 (1), e6481. doi:10.1002/CPE.6481
- Tao, B., Wang, Y., He, F., Chen, B., and Chen, B. (2022b). Photoelastic Stress Field Recovery Using Deep Convolutional Neural Network. *Front. Bioeng. Biotechnol.* doi:10.3389/fbioe.2022.818112
- Thangaraj, R., Pant, M., Abraham, A., and Bouvry, P. (2011). Particle Swarm Optimization: Hybridization Perspectives and Experimental Illustrations. *Appl. Mathematics Comput.* 217 (12), 5208–5226. doi:10.1016/j.amc.2010.12.053
- Tian, J., Cheng, W., Sun, Y., Li, G., Jiang, D., Jiang, G., et al. (2020). Gesture Recognition Based on Multilevel Multimodal Feature Fusion. *Ijfs* 38 (3), 2539–2550. doi:10.3233/IJFS-179541
- Tong, Y., Liu, J., Liu, Y., and Yuan, Y. (2021). Analytical Inverse Kinematic Computation for 7-DOF Redundant Sliding Manipulators. *Mechanism Machine Theor.* 155 (1), 104006. doi:10.1016/j.mechmachtheory.2020.104006
- Wang, H., Lu, X., Sheng, C., Zhang, Z., Cui, W., and Li, Y. (2018). General Frame for Arbitrary 3R Subproblems Based on the POE Model. *Robotics Autonomous Syst.* 105, 138–145. doi:10.1016/j.robot.2018.04.002
- Wang, Y., Ding, X., Tang, Z., Hu, C., Wei, Q., and Xu, K. (2021). A Novel Analytical Inverse Kinematics Method for SSRMS-type Space Manipulators Based on the POE Formula and the Paden-Kahan Subproblem. *Int. J. Aerospace Eng.* 2021, 20211–20213. doi:10.1155/2021/6690696
- Weng, Y., Sun, Y., Jiang, D., Tao, B., Liu, Y., Yun, J., et al. (2021). Enhancement of Real-Time Grasp Detection by Cascaded Deep Convolutional Neural Networks. *Concurrency Comput. Pract. Experience* 33 (5), e5976. doi:10.1002/cpe.5976
- Wu, X., Jiang, D., Yun, J., Liu, X., Sun, Y., Tao, B., et al. (2022). Attitude Stabilization Control of Autonomous Underwater Vehicle Based on Decoupling Algorithm and PSO-ADRC. *Front. Bioeng. Biotechnol.* doi:10.3389/fbioe.2022.843020
- Xiao, F., Li, G., Jiang, D., Xie, Y., Yun, J., Liu, Y., et al. (2021). An Effective and Unified Method to Derive the Inverse Kinematics Formulas of General Six-DOF Manipulator with Simple Geometry. *Mechanism Machine Theor.* 159, 104265. doi:10.1016/j.mechmachtheory.2021.104265
- Xu, M., Zhang, Y., Wang, S., and Jiang, G. (2022). Genetic-Based Optimization of 3D Burch-Schneider Cage with Functionally Graded Lattice Material. *Front. Bioeng. Biotechnol.* 10. doi:10.3389/fbioe.2022.819005
- Yang, C., Gao, W., Liu, N., and Song, C. (2015). Low-discrepancy Sequence Initialized Particle Swarm Optimization Algorithm with High-Order Nonlinear Time-Varying Inertia Weight. *Appl. Soft Comput.* 29, 386–394. doi:10.1016/j.asoc.2015.01.004
- Yang, Z., Jiang, D., Sun, Y., Tao, B., Tong, X., Jiang, G., et al. (2021). Dynamic Gesture Recognition Using Surface EMG Signals Based on Multi-Stream Residual Network. *Front. Bioeng. Biotechnol.* 9. doi:10.3389/fbioe.2021.779353
- Yiyang, L., Xi, J., Hongfei, B., Zhining, W., and Liangliang, S. (2021). A General Robot Inverse Kinematics Solution Method Based on Improved PSO Algorithm. *IEEE Access* 9 (99), 32341–32350. doi:10.1109/ACCESS.2021.3059714
- Yu, M., Li, G., Jiang, D., Jiang, G., Tao, B., and Chen, D. (2019). Hand Medical Monitoring System Based on Machine Learning and Optimal EMG Feature Set. *Pers Ubiquit Comput.* doi:10.1007/s00779-019-01285-2
- Yu, M., Li, G., Jiang, D., Jiang, G., Zeng, F., Zhao, H., et al. (2020). Application of PSO-RBF Neural Network in Gesture Recognition of Continuous Surface EMG Signals. *Ijfs* 38 (3), 2469–2480. doi:10.3233/IJFS-179535
- Yun, J., Sun, Y., Li, C., Jiang, D., Tao, B., Li, G., et al. (2022). Self-adjusting Force/bit Blending Control Based on Quantitative Factor-Scale Factor Fuzzy-PID Bit Control. *Alexandria Eng. J.* 61 (6), 4389–4397. doi:10.1016/j.aej.2021.09.067
- Zhang, X., Xiao, F., Tong, X., Sun, Y., Liu, Y., Yun, J., et al. (2022). Time Optimal Trajectory Planning Based on Improved Sparrow Search Algorithm. *Front. Bioeng. Biotechnol.* doi:10.3389/fbioe.2022.852408

**Conflict of Interest:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

**Publisher's Note:** All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Copyright © 2022 Zhao, Jiang, Liu, Tong, Sun, Tao, Kong, Yun, Liu and Fang. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.