Received 11 November 2023, accepted 27 November 2023, date of publication 1 December 2023,
date of current version 14 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3338742

RESEARCH ARTICLE

Research on Motion Control Strategy of Flexible Manipulator Based on Swarm Intelligence Optimization

SONGHUA HU[1], YUHANG HAN[2], MINGXIAN LIU[3], ZHENHUA XU[4], AND HONGYU XIANG[2]
[1]Baoshan Power Supply Bureau of Yunnan Power Grid Company Ltd., Baoshan, Yunnan 678008, China
[2]Faculty of Civil Aviation and Aeronautics, Kunming University of Science and Technology, Kunming, Yunnan 650500, China
[3]Lijiang Power Supply Bureau of Yunnan Power Grid Company Ltd., Lijiang, Yunnan 674100, China
[4]Shanghai ULS Robotics Intelligent Technology Company Ltd., Shanghai 200433, China

Corresponding author: Hongyu Xiang (348321715@qq.com)

This work was supported by the Science and Technology Project of China Southern Power Grid Company Ltd., through the Research and Development of Key Technologies and Equipment for Safety Protection of Hazardous Environments Based on the Bionic Mechanism of Human Movement under Grant YNKJXM20220216.

ABSTRACT The requirement for motion control of robotic arms in industrial settings is a dynamic field. This study examines the principles and derivation of the kinematics of the robotic arm based on the D-H parameter model. Additionally, the introduction of the seventh joint is proposed as a faster solution for solving the inverse kinematics of the robotic arm. Swarm intelligence optimization for path planning is currently advancing, and our proposed improved algorithm, the Golden Eagle search algorithm, enhances the traditional Golden Eagle search algorithm Jining by integrating a stochastic gradient descent strategy and Cauchy mutation strategy. We compare our IGEO algorithm with various other algorithms, and the findings demonstrate that the robotic arm can adeptly circumnavigate obstacles while walking seamlessly through environments with multiple obstacles. The IGEO algorithm is adept at navigating paths obstructed by multiple obstacles. It improves the accuracy by 15.38% as compared to the conventional algorithm and also improves it a lot as compared to other optimisation algorithms by up to 29.88%. It provides a solution to the path planning problem of robotic arms with excellent robustness and accuracy in finding the shortest collision-free path.

INDEX TERMS Six-degree-of-freedom robot, kinematics, D-H parametric model, golden eagle search algorithm, path planning.

I. INTRODUCTION

Industrial robots have seen widespread use in Industrial Manufacturing in recent years. The remote control of human-computer interaction can take over the performance of certain dangerous and difficult tasks that would otherwise require humans. The control of robotic arms plays a vital role in the development of robotic technology, as well as the advancement of various technologies. To enhance and study the performance of the robotic arm, it is imperative to refine the motion control of the arm [1]. This encompasses motion trajectory control, trajectory recognition and prediction, and hazardous action intervention.

The integration of humanoid robots into the real environment has been a longstanding focus of research. To enable flexible use of robotic arms in various scenarios, there is ongoing investigation of robotic arms that possess varying degrees of freedom. In the simulation of trajectory planning for a six-degree-of-freedom robotic arm, it proves challenging to visually confirm the correctness of the kinematic algorithm and the trajectory planning's effectiveness. Therefore, Cheng et al. [2], with proper mathematical modelling of the robotic arm, focused primarily on analyzing the arm within joint space. To establish the six-degree-of-freedom robotic arm's simulation model, they employed the D-H

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad Alshabi.

algorithm and simulated the trajectory planning using the quantum ant colony algorithm. Deng et al. [3] employ a single-class support vector machine model to classify poses of humanoids, and utilize the redundancy characteristics of a 7-degrees-of-freedom manipulator arm through a linear regression model to enhance the search for humanoid poses. Ekrem et al. [4] apply particle swarm optimization (PSO) to the robot arm's trajectory planning, enabling the manipulator to move from the starting point to the target point while avoiding obstacles and selecting the most direct path.

Trajectory planning forms the foundation for the robot arm's movements, and significantly impacts the quality of the completed operation. Dai et al. [5] provide an overview of the current state of spatial obstacle avoidance trajectory planning and motion trajectory planning, discussing the basic principles and practical applications of trajectory planning methods for spatial manipulators. Du et al. [6] presented a method for time-optimal trajectory planning during manipulator motion. The method involves a piecewise polynomial interpolation function based on a local chaotic particle swarm optimization (LCPSO) algorithm. The authors obtained time-optimal and smooth motion trajectories for each joint in the joint space through simulation experiments, demonstrating the method's effectiveness in reducing the running time of robot manipulators while ensuring motion stability. Ni et al. [7] incorporate penalty terms as constraints in the trajectory optimization problem and suggest a novel strategy for increasing the penalty factor. This approach attains the objective of balancing punishment and search capability when dealing with multiple constraints. Wang et al. [8] address the incomplete traits of free-floating space robots and employ a constrained PSO algorithm with stagnant processing tactics to execute the coordinated trajectory structure of free-floating space robots. While the algorithm for optimizing the trajectory of robotic arms is still in development, numerous scholars have conducted extensive research on the kinematic model of robotic arms. In their study, Peng et al. [9] constructed a closed-chain kinematics and dynamic model for multi-arm continuous space robots. They also proposed a collaborative planning strategy for said robots before and after target capture, and developed a compliance control framework.

The increasing development and application of intelligent optimization algorithms have shown outstanding performance in path planning [10]. Scholars have gradually explored the optimization characteristics of conventional intelligent optimization algorithms. Based on this breakthrough, more researchers have shifted their focus towards studying fusion intelligent algorithms. Wang et al. [11] combined the particle swarm algorithm with the artificial fish swarm algorithm. They initially created two subgroups and then iteratively optimized them using both algorithms. By optimizing information sharing, they ultimately derived the PSO-AFSA hybrid algorithm, which exhibited better performance. Further researchers [12] have implemented the intelligent optimization algorithm for robotic arm control and incorporated a range of optimization algorithms through enhancement of the standard intelligent optimization algorithm [13]. As a result, the precision of the robotic arm's operation has been significantly augmented.

Nasrollahy et al. [14] have developed a path planning method for mobile robots which is based on PSO to ensure the shortest path and time while avoiding static and dynamic objects. Further research has been conducted on the path planning method of robot arms. In their study, Lopez-Franco et al. [15] compared the simulation results of eight different path planning optimisation algorithms. Rafal and colleagues [16] investigated the capability of robotic arm palletisation to select the correct item while managing multiple production lines. They employed artificial bee colony algorithms backed by DEB rules to increase productivity and fulfil specific demands. Meanwhile, Arup et al. [17] put forward a Q-learning triggered firefly algorithm (FA) that learns the optimal parameter values of each firefly in the population during the learning phase and performs the path planning of the robot manipulator while dealing with different obstacles.

II. KINEMATIC MODEL

A. ROBOTIC ARM COFIGURATION ANALYSIS

The flexible robot arm examined in this research is designed to clean the cargo area. Given the complexity of the cargo's structure but relative fixity of its placement, the number of joints impact the robotic arm's dynamic performance. Figure 1 displays the configuration of the robot arm. Hence, we equipped the robot with six modular joints and connecting rods, providing six degrees of freedom.

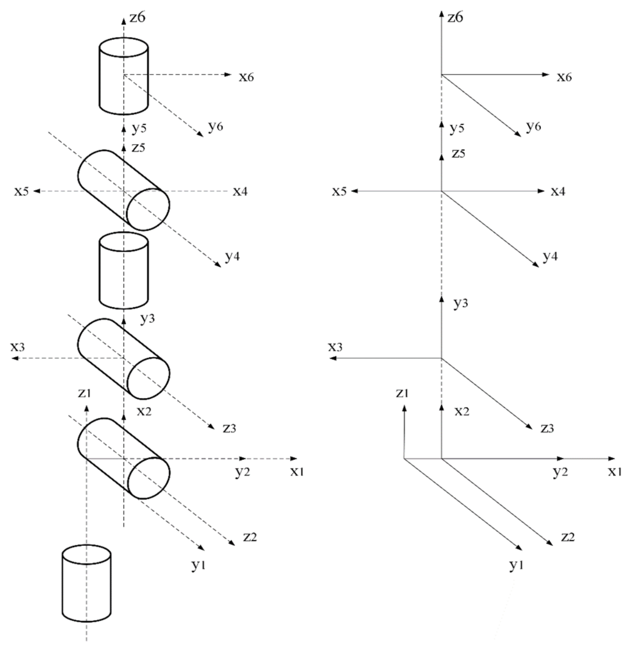

FIGURE 1. 6-DOF robotic arm configuration.

Set the initial coordinates of the robotic arm to (x_1, y_1, z_1), and then calculate the coordinates of subsequent joint points accordingly. The D-H coordinate parameters of the robot arm are based on the Denavit Hartenberg model [18], with the specific parameters determined by the size of the robot arm's structure and used to determine its basic properties. Technical terminology abbreviations will be always explained when first used. The D-H parameters serve as a foundation for analyzing the forward and inverse kinematics of manipulators. Additionally, the complexity of kinematic models varies depending on the D-H parameters utilized. Table 1 displays the parameters for the provided ur-6-DOF manipulator in accordance with the meaning of the D-H parameter table [19].

TABLE 1. **Robotic arm D-H parameter table.**

Joint	θ_n	d_n	a_{n-1}	α_{n-1}
1	0	144	0	Pi/2
2	Pi/2	0	264	0
3	0	0	236	0
4	-Pi/2	106	0	Pi/2
5	0	114	0	-Pi/2
6	0	68	0	0
7	Pi/2	0	0	Pi/2

where, θ_n represents the joint rotation angle, d_n represents the joint deflection, a_n represents the connecting rod length, and α_n represents the connecting rod rotation angle.

B. POSITIVE KINEMATIC MODEL

The composite transformation of the three-dimensional coordinate system can be decomposed into the translation and rotation of multiple two-dimensional coordinate systems, that is, the transformation of the coordinate system $X-Y$, $X-Z$, and $Y-Z$. Taking the coordinate system $X-Z$ transformation as an example, between n and $n+1$ coordinates, assume that the existing coordinate system is $X_n - Z_n$ and the coordinate system after the composite transformation is $X_{n+1} - Z_{n+1}$.

(1) Rotate the X_n axis around the Z_n axis by θ_{n+1} so that X_n and X_{n+1} are parallel.

(2) Shift the d_{n+1} distance along the Z_n axis so that X_n and X_{n+1} are collinear.

In (1), X_n and X_{n+1} are parallel and both are already perpendicular to the Z_n axis, so translating X_n along the Z_n axis causes X_n and X_{n+1} to coincide, as shown in Figure 3.

(3) Translate the a_{n+1} distance along the X_n axis so that the origins of X_n and X_{n+1} coincide.

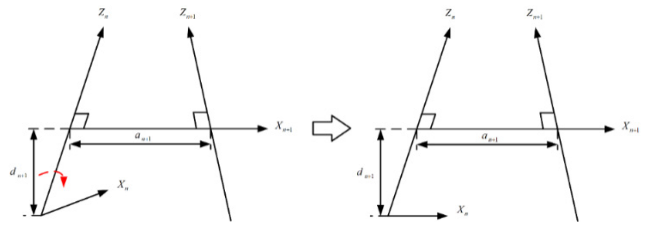

FIGURE 2. **The first rotation transform.**

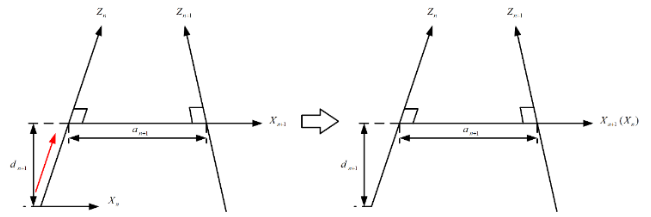

FIGURE 3. **The first translation transformation.**

Translate the a_{n+1} distance along the X_n axis, when the origin of the two coordinate systems n and $n+1$ will be in the same position, and the translation process is shown in Figure 4.

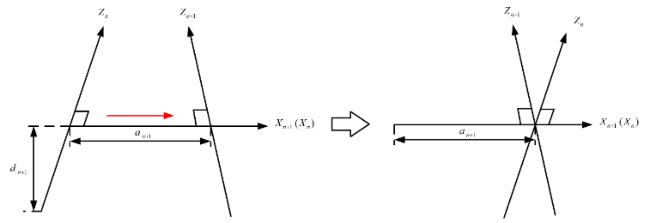

FIGURE 4. **The second translation transformation.**

(4) Rotate the Z_n axis around the X_{n+1} axis α_{n+1} angle so that the Z_n axis and Z_{n+1} axis coincide.

Rotate the Z_n axis around the X_{n+1} axis α_{n+1} angle, the two coordinate systems n and $n+1$ are exactly the same, and the rotation process is shown in Figure 5.

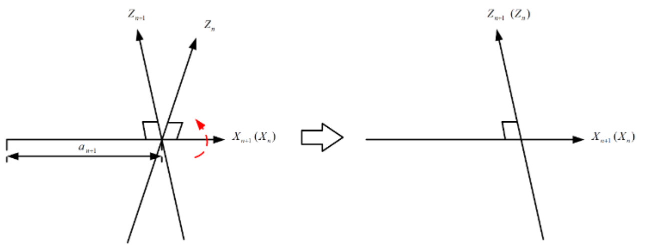

FIGURE 5. **The second rotation transform.**

According to the translation rotation law, the transformation matrix can be obtained as:

$$
{}^{n}_{n+1}T = A_{n+1} = Rot(z, \theta_{n+1}) \times Tra(0, 0, d_{n+1}) \times Tra(a_{n+1}, 0, 0) \times Rot(x, a_{n+1}) \tag{1}
$$

The D-H model is the positive kinematics model of the robotic arm, which describes the process of solving the terminal pose by knowing the joint angle θ during the motion control of the robotic arm. Therefore, the positive kinematic equation of the six-degree-of-freedom robotic arm is:

$$ {}_6^0T = A_1A_2A_3A_4A_5A_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

where,

$$\begin{cases} n_x = -s_6\left(c_4s_1 - s_4\left(c_1s_2s_3 - c_1c_2c_3\right)\right) \\ \qquad -c_6\left(c_5\left(s_1s_4 + c_4 \times \left(c_1s_2s_3 - c_1c_2c_3\right)\right)\right. \\ \qquad \left. +s_5\left(c_1c_2s_3 + c_1c_3s_2\right)\right) \\ n_y = s_6\left(c_1c_4 + s_4\left(s_1s_2s_3 - c_2c_3s_1\right)\right) \\ \qquad +c_6\left(c_5\left(c_1s_4 - c_4\left(s_1s_2s_3 - c_2c_3s_1\right)\right)\right. \\ \qquad \left. -s_5\left(c_2s_1s_3 + c_3s_1s_2\right)\right) \\ n_z = c_6\left(c_{23}s_5 + s_{23}c_4c_5\right) - s_{23}s_4s_6 \end{cases}$$

$$\begin{cases} o_x = s_6\left(c_5\left(s_1s_4 + c_4\left(c_1s_2s_3 - c_1c_2c_3\right)\right)\right. \\ \qquad \left. +s_5\left(c_1c_2s_3 + c_1c_3s_2\right)\right) \\ \qquad -c_6\left(c_4s_1 - s_4\left(c_1s_2s_3 - c_1c_2c_3\right)\right) \\ o_y = c_6\left(c_1c_4 + s_4\left(s_1s_2s_3 - c_2c_3s_1\right)\right) \\ \qquad -s_6\left(c_5\left(c_1s_4 - c_4\left(s_1s_2s_3 - c_2c_3s_1\right)\right)\right. \\ \qquad \left. -s_5\left(c_2s_1s_3 + c_3s_1s_2\right)\right) \\ o_z = -s_6\left(c_{23}s_5 + s_{23}c_4c_5\right) - s_{23}c_6s_4 \end{cases}$$

$$\begin{cases} a_x = s_5\left(s_1s_4 + c_4\left(c_1s_2s_3 - c_1c_2c_3\right)\right) \\ \qquad -c_5\left(c_1c_2s_3 + c_1c_3s_2\right) \\ a_y = -s_5\left(c_1s_4 - c_4\left(s_1s_2s_3 - c_2c_3s_1\right)\right) \\ \qquad -c_5\left(c_2s_1s_3 + c_3s_1s_2\right) \\ a_z = c_{23}c_5 - s_{23}c_4s_5 \end{cases}$$

$$\begin{cases} p_x = a_2c_1c_2 - d_4s_{23}c_1 - d_6\left(s_{23}c_1c_5 + s_1s_4s_5\right. \\ \quad \left. -c_1c_2c_3c_4s_5 + c_1c_4s_2s_3s_5\right) \\ p_y = a_2c_2s_1 - d_4s_{23}s_1 - d_6\left(s_{23}c_5s_1 - c_1s_4s_5\right. \\ \qquad \left. -c_2c_3c_4s_1s_5 + c_4s_1s_2s_3s_5\right) \\ p_z = d_1 + d_4c_{23} + a_2s_2 - \left(d_6s_{23}s_{45}\right)/2 + d_6c_{23}c_5 \\ \qquad + \left(d_6s_{-45}s_{23}\right)/2 \end{cases}$$

C. INVERSE KINEMATIC MODEL

The inverse kinematics problem for a robotic arm involves solving for the angles of the rotational joints based on the known end attitude of the robotic arm, i.e., using the positive kinematics equations to solve $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$. The solution process of manipulator inverse motion is very complex, and there are many solution methods, mainly closed solution and numerical solution [20], this paper takes the analytical method in the closed solution form as an example to analyze and solve the problem.

If the endpoints of the three axes on the robot converge at a single point, an analytical solution must exist for the robot. Hence, for this six-degree-of-freedom robotic arm, the analytical approach is the quickest and most precise method, and the subsequent steps are outlined below.

From equation (2) we get: (3), as shown at the bottom of the next page.

To enable smooth computation, Table 1 includes an additional connecting rod which is fixedly attached to connecting rod 6 and remains stationary during rotation. Cause $d_6 = 0$, $d_7 = d_6$. Then the coordinate system of the connecting rod 7 coincides exactly with the connecting rod 6, and the homogeneous matrix of the connecting rod 7 is:

$$ {}_7^6T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}_7^6T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4} $$

The homogeneous transformation matrix before and after adding the connecting rod 7 remains unchanged, that is, ${}_6^0T_{old}$ before adding the connecting rod 7 is equal to ${}_7^0T$ after adding the connecting rod 7, so that $d_7 = 0$ then there is:

$$ {}_6^0T_{new} = {}_6^0T_{old} \times {}_7^6T^{-1} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times {}_7^6T^{-1} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

At the same time, since ${}_6^1T_{new} = {}_1^0T^{-1} \times {}_6^0T_{new} = {}_1^0T^{-1} \times {}_6^0T_{old} \times {}_7^6T^{-1}$ makes $d_7 = d_6$, i.e.: (6), as shown at the bottom of the next page.

Since the coordinate systems of connecting rod 6 and connecting rod 7 coincide exactly, it is obtained:

$$ \begin{aligned} {}_6^1T &= {}_1^0T^{-1} \times {}_6^0T = \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ s_1 & -c_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_1n_x + s_1n_y & c_1o_x + s_1o_y & c_1a_x + s_1a_y & c_1p_x + s_1p_y \\ n_z & o_z & a_z & p_z - d_1 \\ s_1n_x - c_1n_y & s_1o_x - c_1o_y & s_1a_x - c_1a_y & s_1p_x - c_1p_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{7} $$

By making the rows and columns equal, the inverse kinematic model can be solved.

$$ \theta_1 = a\tan 2(p_y, p_x) - a\tan 2(-d_4, \pm\sqrt{p_x^2 + p_y^2 - d_4^2}) \tag{8} $$

$$ \theta_5 = a\tan 2(s_5, s_1a_x - c_1a_y) \tag{9} $$

$$ \theta_6 = a\tan 2\left(\frac{-s_1o_x + c_1o_y}{\pm\sqrt{\left(s_1n_x - c_1n_y\right)^2 + \left(s_1o_x - c_1o_y\right)^2}}, \frac{-s_1n_x + c_1n_y}{\pm\sqrt{\left(s_1n_x - c_1n_y\right)^2 + \left(s_1o_x - c_1o_y\right)^2}}\right) \tag{10} $$

$$\theta_2 = a\tan 2(B, A) - a\tan 2(C, \pm\sqrt{A^2 + B^2 - C^2}) \tag{11}$$

$$\theta_4 = \theta_{234} - \theta_{23} \tag{12}$$

$$\theta_3 = \theta_{23} - \theta_2 \tag{13}$$

III. ALGORITHM PRINCIPLE

Path planning is a prominent research area in many fields for efficiently and securely determining the optimal or sub-optimal path between the start and destination points in a simulated environment model based on performance indicator requirements [21]. In recent years, both domestic and international scholars have engaged in research on these algorithms, including A* [22], [23], ant colony algorithm [24], genetic algorithm [25], and golden eagle search algorithm [26]. A* algorithms are implemented using heuristics, but discovering the appropriate heuristics can be time-consuming in larger settings. Ant colony algorithms, on the other hand, are based on positive feedback mechanisms that ants create when foraging for food. While the ant colony algorithm is efficient, forming positive feedback at the start of the search takes considerable time, resulting in slower convergence speeds [27]. Genetic algorithms exhibit efficient parallelism and global search capabilities, however, they may converge prematurely and become fixated on local optimal solutions [28].

On the basis of the theoretical derivation of the study and solution of the positive and negative kinematics model of the manipulator in the first chapter, combined with the research of many scholars applying optimization algorithms to robotic arm path planning, this chapter investigates and simulates the robotic path planning algorithm.

A. TRADITIONAL GEO ALGORITHM

Golden Eagle Optimizer (GEO) is a bionic optimisation algorithm. Its main idea is to hunt according to the golden eagle's continuous adjustment of speed in the spiral stage, dividing the hunt into two stages, the initial stage and the final stage, and constantly adjusting the behaviour mode to achieve the best position close to the target.

1) PREY SELECTION

In every cycle, every golden eagle in the population needs to select prey to exhibit its behaviour. The target is named as a congregation of golden eagles, and each individual golden eagle position illustrates the most excellent positioning mechanism currently identified. As cycles occur, every agent hunts for a target prey in every memory of the population of golden eagles. The vectors of attack and cruise for every Golden Eagle with respect to the chosen prey are subsequently computed. If the calculated new position is superior to the previous memory position, the Golden Eagle's position is updated. Each Golden Eagle selects prey only from its own memory, without interfering with each other, and utilizes a random one-to-one mapping scheme. The selected prey may not necessarily be the farthest or closest, but the selection result is computed based on the attack vector and cruise vector.

2) OFFENSIVE BEHAVIOR

The attack behavior is the calculation of the attack vector, starting from the current position of the golden eagle and ending with the position of the prey in memory, using a vector instead. That is, the attack vector formula is:

$$\overrightarrow{A_i} = \overrightarrow{X_f^*} - \overrightarrow{X_i} \tag{14}$$

In Equation (14), $\overrightarrow{A_i}$ is the attack vector of the i Golden Eagle, $\overrightarrow{X_f^*}$ is the position of the prey, and $\overrightarrow{X_i}$ is the current position of the i Golden Eagle.

3) CRUISING BEHAVIOR

The cruising behavior is the calculation of the cruising vector, and first according to the result of the attack vector, the three-dimensional cruise vector is located in the hyperplane tangent to the circle where the attack vector is located. When cruising, the dimensionality needs to be specified, and the three-dimensional space scalar form in the hyperplane is as follows:

$$h_1x_1 + h_2x_2 + \ldots + h_nx_n = d \Rightarrow \sum_{j=1}^{n} h_jx_j = d \tag{15}$$

In equation (15), $H = [h_1, h_2, \ldots, h_n]$ is the normal vector, $X = [x_1, x_2, \ldots, x_n]$ is the variable vector, and any point on the hyperplane is $\overrightarrow{P} = [p_1, p_2, \ldots, p_n]$, then there are:

$$d = \overrightarrow{H}\,\overrightarrow{P} = \sum\nolimits_{i=1}^{n} h_jp_j \tag{16}$$

$$ {}_6^1T = \begin{bmatrix} c_{234}c_5c_6 - s_{234}s_6 & -s_{234}c_6 - c_{234}c_5s_6 & -c_{234}s_5 & a_3c_{23} - (d_6s_{2345})/2 + s_2c_2 + (d_6s_{234-5})/2 + d_5s_{234} \\ c_{234}s_6 + s_{234}c_5c_6 & c_{234}c_6 - s_{234}c_5s_6 & -s_{234}s_5 & d_5(s_{23}s_4 - c_{23}c_4) + a_3s_{23} + a_2s_2 - d_6s_5(c_{23}s_4 + s_{23}c_4) \\ -c_6s_5 & -s_5s_6 & c_5 & d_4 + d_6c_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$ {}_6^1T_{new} = \begin{bmatrix} c_{234}c_5c_6 - s_{234}s_6 & -s_{234}c_6 - c_{234}c_5s_6 & -c_{234}s_5 & a_3c_{23} + a_2c_2 + d_5s_{234} \\ c_{234}s_6 + s_{234}c_5c_6 & c_{234}c_6 - s_{234}c_5s_6 & -s_{234}s_5 & a_3s_{23} + a_2s_2 - d_5c_{234} \\ c_6s_5 & -s_5s_6 & c_5 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Since the cruise vector is the tangent vector that produces the circle perpendicular to the attack vector, considering the $\overrightarrow{A_i}$ attack vector as the normal of the hyperplane, the cruise vector can be expressed as the hyperplane position where the Golden Eagle is currently located:

$$\sum_{j=1}^{n} a_j x_j = \sum_{j=1}^{n} a_j^t x_j^* \tag{17}$$

In Equation (17), $A = [a_1, a_2, \ldots, a_n]$ is the attack vector, $X = [x_1, x_2, \ldots, x_n]$ is the decision variable vector, and $X^* = [x_1^*, x_2^*, \ldots, x_n^*]$ is the prey position.

As the cruise vector's starting point is the current position of the Golden Eagle and its position transfer necessitates a random destination, we create a random vector on the cruise hyperplane. We then reduce the dimension of the hyperplane by one, resulting in a new degree of freedom for the reduced dimension that is determined by the hyperplane form (formula (17)). This task can be accomplished by setting a free vector and a fixed vector to determine the destination within a random dimension on the Golden Eagle cruise hyperplane. The steps to achieve this are as follows:

Step1: Randomly select a variable from the variables as a fixed vector;

Step2: Randomly assign values to all variables except the k variable;

Step3: Calculate the value of a fixed vector:

$$c_k = \frac{d - \sum_{j,j\neq k} a_j}{a_k} \tag{18}$$

In Equation (18), c_k is the k element of the target point C and a_j is the first j element of the attack vector $\overrightarrow{A_i}$. At the same time, k is the number of bits of the fixed vector.

Step4: General representation of the point of destination on the cruising hyperplane: (19), as shown at the bottom of the next page.

4) NEW LOCATION TRANSFER

The step size of the Golden Eagle iteration is defined as:

$$\Delta x_i = \vec{r_1} p_a \frac{\overrightarrow{A_i}}{\left\|\overrightarrow{A_i}\right\|} + \vec{r_1} p_c \frac{\overrightarrow{C_i}}{\left\|\overrightarrow{C_i}\right\|} \tag{20}$$

where, $\left\|\overrightarrow{A_i}\right\| = \sqrt{\sum_j^n a_j^2}$, $\left\|\overrightarrow{C_i}\right\| = \sqrt{\sum_j^n c_j^2}$.

Then, superimpose the step vector in formula (20) on the position in the iteration, which is the position of the Golden Eagle individual in the iteration:

$$x^{t+1} = x^t + \Delta x_i^t \tag{21}$$

In Equation (21), x^{t+1} is the position of the golden eagle of the $t+1$ order, x^t is the t position of the golden eagle, and Δx_i^t is the step size of the golden eagle's movement.

If the new position is more suitable than the position stored in memory, the memory is updated accordingly. Otherwise, the original memory position is kept, although the golden eagle is also moved to the new position. During each iteration, every Golden Eagle picks a random Golden Eagle from the population, and rotates around its optimal position to calculate the attack and cruise vectors. Finally, the step size and new position for the next iteration is calculated until all termination conditions are satisfied. The attack and cruise coefficients are outlined below:

$$\begin{cases} p_a = p_a^0 + \frac{t}{T}\left|p_a^T - p_a^0\right| \\ p_c = p_c^0 + \frac{t}{T}\left|p_c^T - p_c^0\right| \end{cases} \tag{22}$$

In Equation (22), p_a^0 and p_a^T are the initial and final values of p_a, and p_c^0 and p_c^T are the initial and final values of p_c.

B. IMPRIVED GEO ALGORITHM

The Golden Eagle algorithm displays outstanding performance, speedy convergence, and robust optimisation capability. However, during the process of interstellar exploration, Golden Eagle adopts a random one-to-one mapping scheme. When operating the robotic arm for long-distance spatial movement, trajectory data experiences significant spatial uncertainty. We enhance the optimization effect by employing a non-convex function in stochastic gradient descent to further improve the purposeful random search performance. During the process of Golden Eagle position iteration, as indicated in formula (23), the position update method is relatively uncomplicated with no clear specification. Therefore, we opted to utilize the Cauchy inverse cumulative integral distribution function to mutate it. Our testing showed a significant improvement in search performance, and the median value during the process was superior.

1) STOCHASTIC GRADIENT DESCENT

The stochastic gradient descent method is mainly used for rapid learning and evolution, so we only extract the non-convex function in it for improvement, and its characteristics of selecting only one sample at a time match the way the Golden Eagle selects its prey, and in the position update of each Golden Eagle iteration, its update formula is:

$$\theta = \theta - \eta \nabla_\theta J(\theta, x_i) \tag{23}$$

2) CAUCHY VARIATION

The initial selection of the rotating golden eagle in the Golden Eagle algorithm is random, thus making it difficult to discover the global optimal solution. To address this issue, we utilized the Cauchy inverse cumulative distribution function to mutate the golden eagle, resulting in a wider range of rotating golden eagles selected by the population. Simultaneously, the mutation approach enhances the properties of the population of golden eagles by randomly selecting an eagle for rotation. This improves their capability to locally optimize while also preventing blind mutations. Equation (24) depicts the Cauchy inverse cumulative distribution function. Equation (25) shows the search formula derived when the golden eagle population

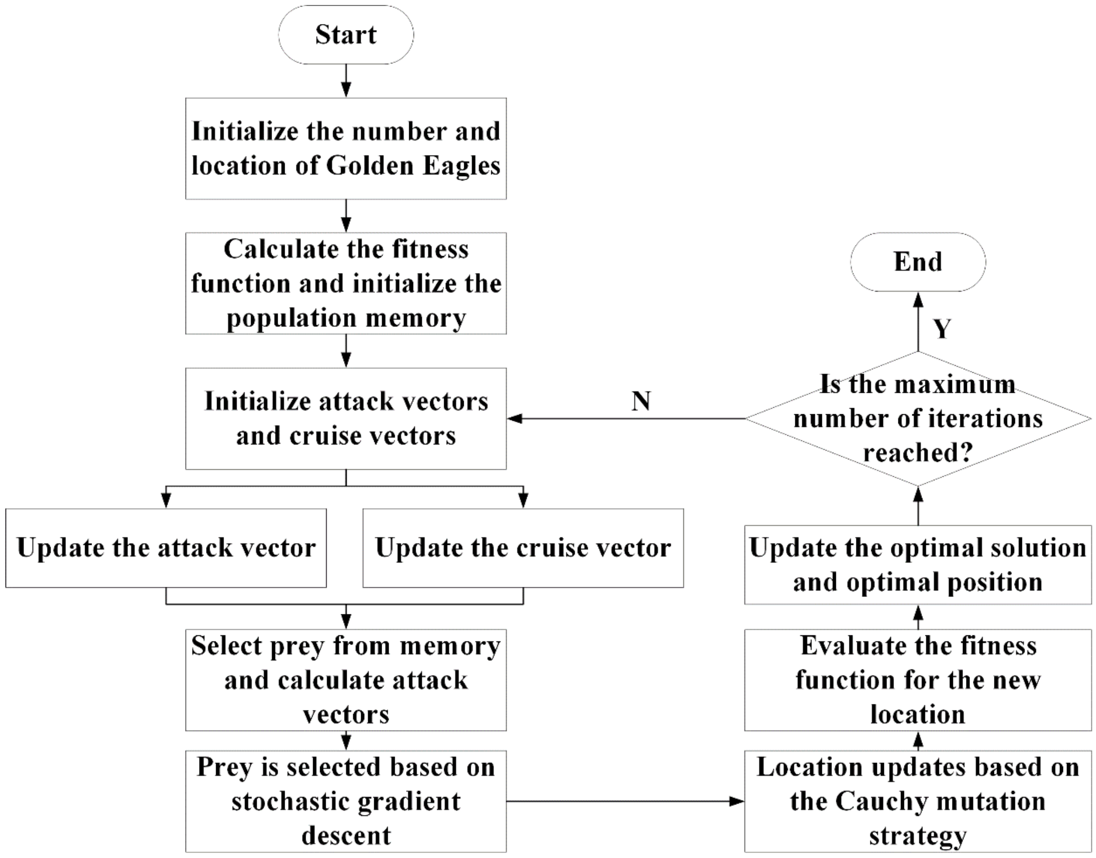
Start
Initialize the number and location of Golden Eagles
Calculate the fitness function and initialize the population memory
Initialize attack vectors and cruise vectors
Update the attack vector
Update the cruise vector
Select prey from memory and calculate attack vectors
Prey is selected based on stochastic gradient descent
Location updates based on the Cauchy mutation strategy
Evaluate the fitness function for the new location
Update the optimal solution and optimal position
Is the maximum number of iterations reached?
N
Y
End

FIGURE 6. IGEO algorithm flowchart.

selects the rotating golden eagle.

$$F^{-1}(p; x_0; \gamma) = x_0 + \gamma \cdot \tan(\pi \cdot (p - 1/2)) \tag{24}$$

$$\overrightarrow{X}(t+1) = x_{ij} + \overrightarrow{A} \cdot \tan(\pi \cdot (r - 1/2)) \tag{25}$$

In equations (24) and (25), F^{-1} is the inverse integral function of the Cauchy variant, x_{ij} is the j position of the i golden eagle before the mutation, and the uniform distribution of $\gamma \in \overrightarrow{A}$, $r \in [0, 1]$.

The flow of the Golden Eagle search algorithm based on gradient optimization and Cauchy variation is as follows:

Step1; initialize the population size and location of the Golden Eagle;

Step2: Calculate the fitness function and initialize the population memory location;

Step3: Initialize attack vector $\overrightarrow{A_i}$ and cruise vector c_k;

Step4: Update the Golden Eagle position according to Equation (25), Update the Attack Vector and Cruise Vector according to Equation (14) and Equation (18);

Step5: Based on the memory position of the population, calculate the attack vector and select the prey according to the formula (23);

Step6: Calculate the cruise vector, step vector, update the position and calculate the fitness function of the new position;

Step7: Update the optimal solution and optimal position;

Step8: If the maximum number of iterations is reached, the optimal golden eagle position and global optimal solution are output; Otherwise, go back to step 4.

IV. SIMULATION EXPERIMENTS

Taking the 6DOF robotic arm as an example, this paper first uses the Monte Carlo method to obtain the motion space of the robotic arm [29], takes the sample number $N = 30000$.

The path planning of the robotic arm seeks the shortest path while avoiding obstacles, so this paper uses the two indicators of successful obstacle avoidance and path length of the robotic arm to evaluate the fitness function, and the

$$\overrightarrow{C_i} = \left\{ c_1 = random, c_2 = random, \ldots, c_k = \frac{d - \sum_{j, j \neq k} a_j}{a_k}, \ldots, c_n = random \right\} \tag{19}$$

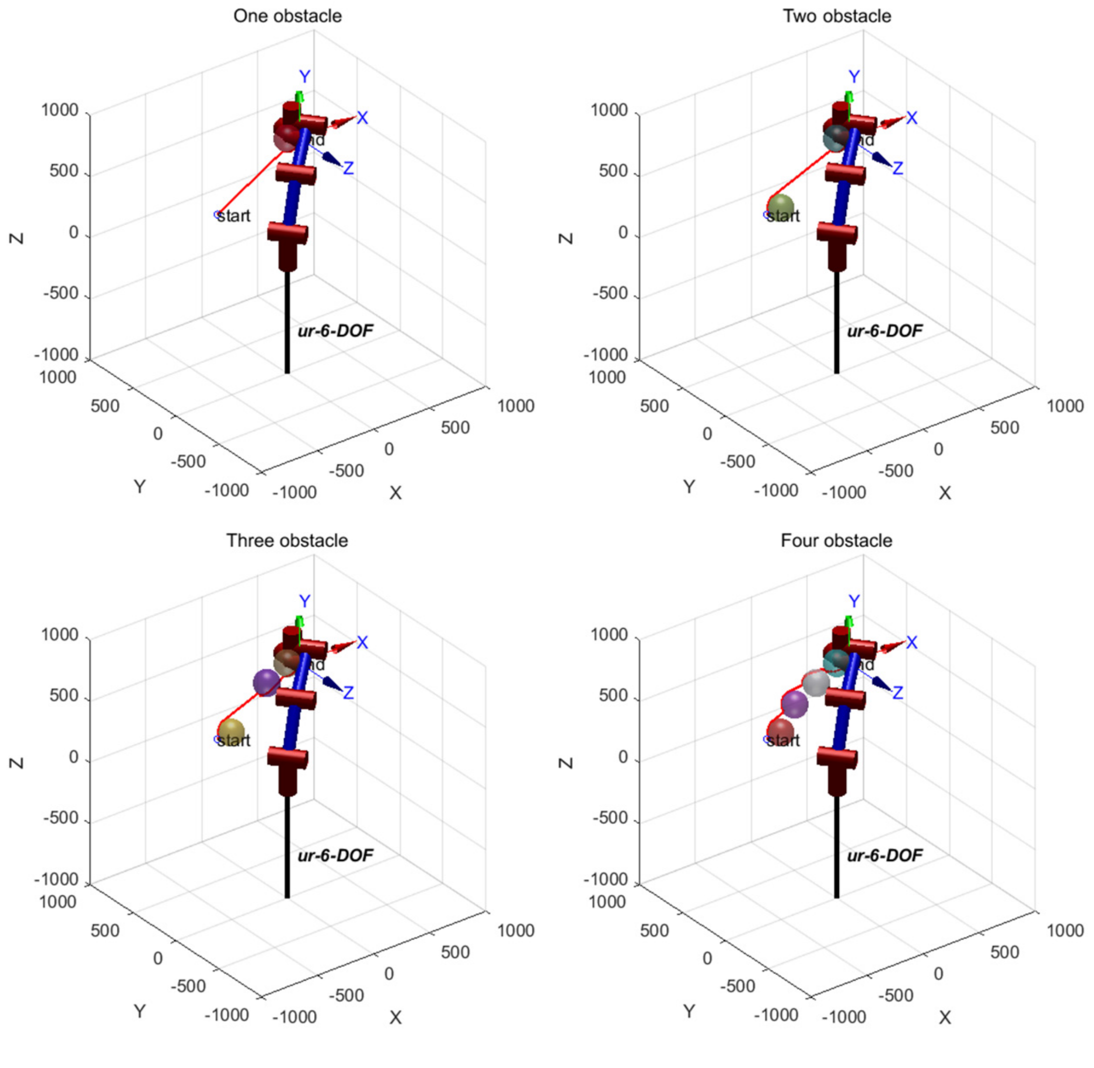

FIGURE 7. **Obstacle avoidance results when the number of obstacle is 1, 2, 3, 4.**

length of a path in the workspace is:

$$L_i = \sum_{j=0}^{n+1}\left(\sqrt{\left(x_i^{j+1}-x_i^j\right)^2+\left(y_i^{j+1}-y_i^j\right)^2+\left(z_i^{j+1}-z_i^j\right)^2}\right) \tag{26}$$

In Equation (26), n is the number of interpolation points searched by Golden Eagle i, and x_i^j, y_i^j, and z_i^j represent the three-dimensional coordinates of the j interpolation point of Golden Eagle i on the path, respectively. The collision loss coefficient for simultaneous collision detection is defined as:

$$collision = \begin{cases} 0, & \text{No collosions} \\ 100, & \text{else} \end{cases} \tag{27}$$

Then the objective function can be defined as:

$$fitness = L_i + collision * L_i \tag{28}$$

To construct an obstacle ball with a radius of 100, randomly generate the starting point of (−400, 300, 300) and the generation target point of (350,360,630), and test the obstacle avoidance performance of the robotic arm when the number of obstacle balls is 1, 2, 3, and 4, respectively, and the obstacle avoidance results are shown in Figure 7. As can be seen from the figure, with the change of the number of obstacle balls, the robotic arm can still successfully avoid obstacles.

Using the IGEO algorithm, Figure 7 displays the optimized path planning results of the robot arm while taking into account four obstacle balls as an example.

To evaluate the algorithm's performance, this paper compares the proposed IGEO algorithm with the classical DE, GA, and SA algorithms. After conducting 30 simulation experiments, the average fitness function of each algorithm is taken as an indicator. The results are presented in Figure 8 and the path result is shown in Figure 9. The specific fitness value data can be found in table 2 (with two decimal places retained).

According to Figure 8, the IGEO algorithm exhibits greater convergence performance. Additionally, Table 2 data demonstrates that the IGEO algorithm achieves consistent

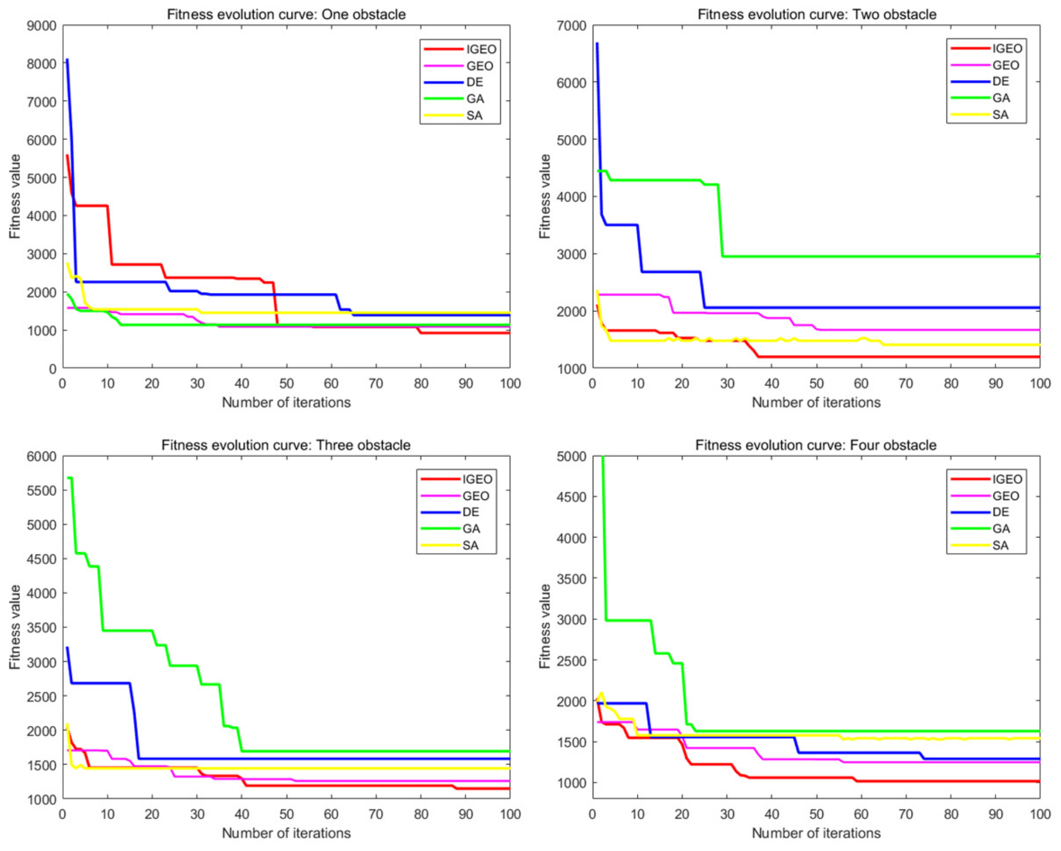

FIGURE 8. Comparison chart of five algorithms.

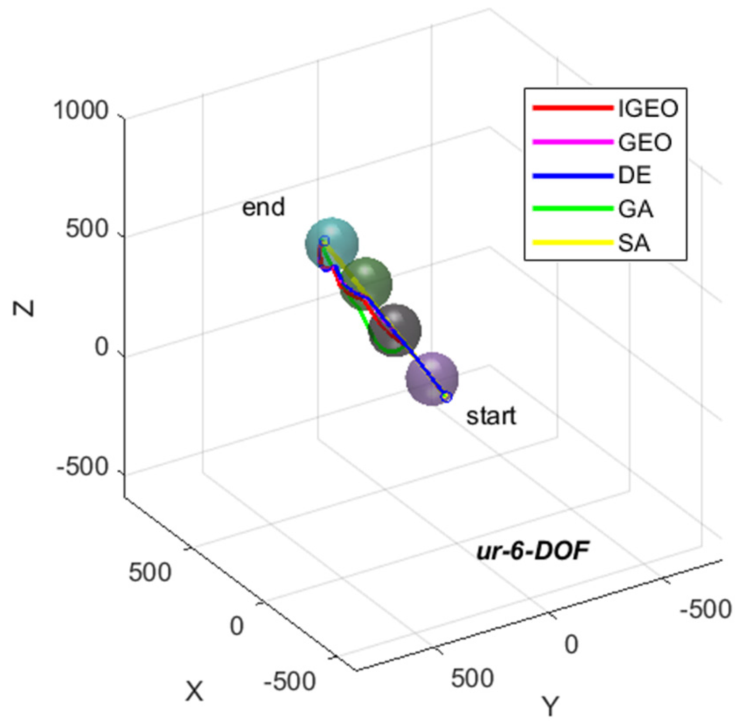

FIGURE 9. Comparison of path results of five algorithms.

TABLE 2. **Comparison table of simulation experimental data.**

Number of obstacles	Algorithm	Optimal fitness value	Worst fitness value	Average fitness value
One	IGEO	945.78	1559.72	1239.37
	GEO	1078.40	1618.60	1344.60
	DE	610.12	2116.82	1354.99
	GA	1023.18	4480.63	1984.59
	SA	991.43	1696.86	1373.73
Two	IGEO	852.15	1512.07	1190.43
	GEO	1084.70	1432.40	1233.70
	DE	558.88	2211.58	1307.48
	GA	1434.75	3969.43	2247.65
	SA	1221.13	1629.56	1453.00
Three	IGEO	818.16	1603.38	1294.86
	GEO	1050.20	1785.30	1326.30
	DE	970.33	2193.45	1315.12
	GA	942.14	3246.67	1818.38
	SA	1196.92	1848.47	1453.21
Four	IGEO	845.29	1686.92	1237.15
	GEO	1034.30	1978.70	1426.90
	DE	870.88	1996.06	1409.35
	GA	949.68	3196.21	1803.80
	SA	1134.97	1733.20	1443.82

optimization performance with smaller numbers of obstacles, yielding the lowest average fitness value. However, the optimal fitness value is slightly higher compared to some other algorithms. However, as the number of obstacles increases, the IGEO algorithm proves to be the most efficient among all tested data, indicating its superior performance in resolving path planning issues for a robotic arm navigating a multi-obstacle environment.

To enable further experimentation, it is essential to investigate both successful and unsuccessful attempts to find the shortest route, leading to a degree of damage to the robotic arm. It is important to note that technical term abbreviations will be explained upon first use. This damage is indicated primarily by the angular displacement, velocity, and acceleration of the arm's joints. Figures 10 to 12 display the angular displacement, velocity, and acceleration for each joint during path planning.

It can be seen from Figures 10 to 12 that when the path planning of the robotic arm is completed, the angle, velocity and acceleration of each joint of the robotic arm are very smooth, and most of the joints are well connected, indicating that the robotic arm can perform smooth motion.

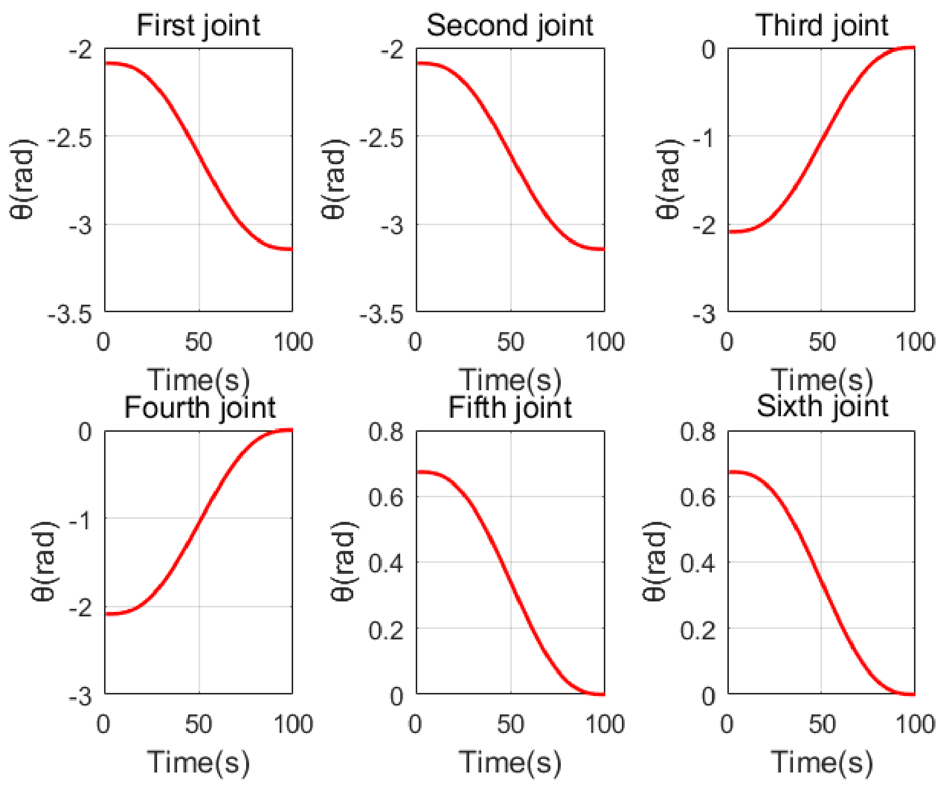

FIGURE 10. Diagram of individual joint angles over time.

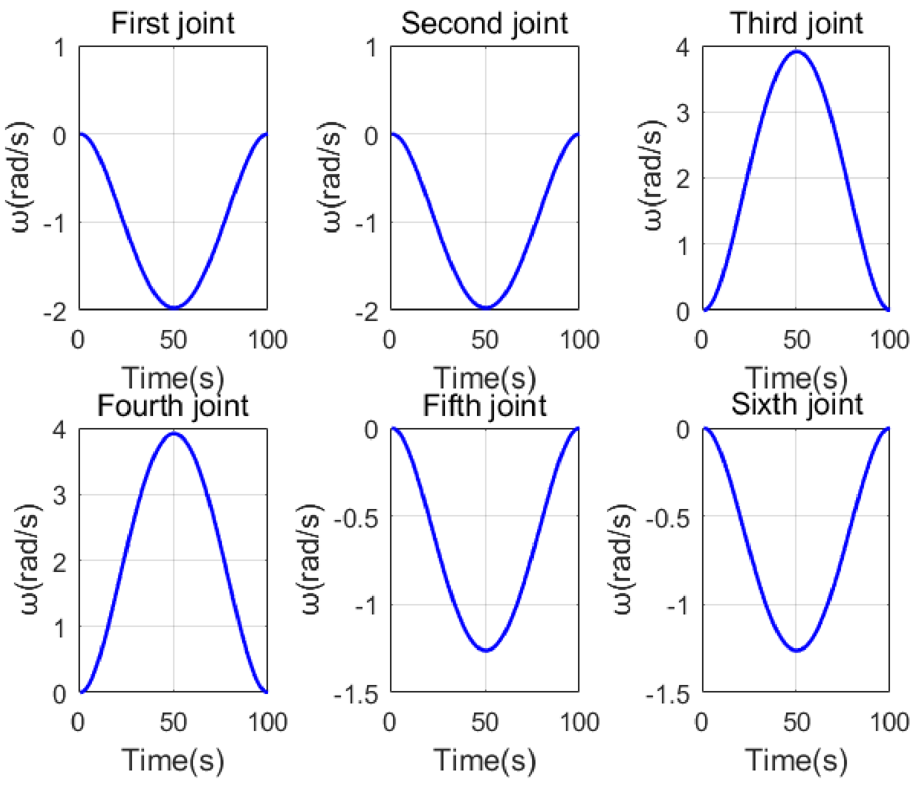

FIGURE 11. Angular velocity of each joint over time.

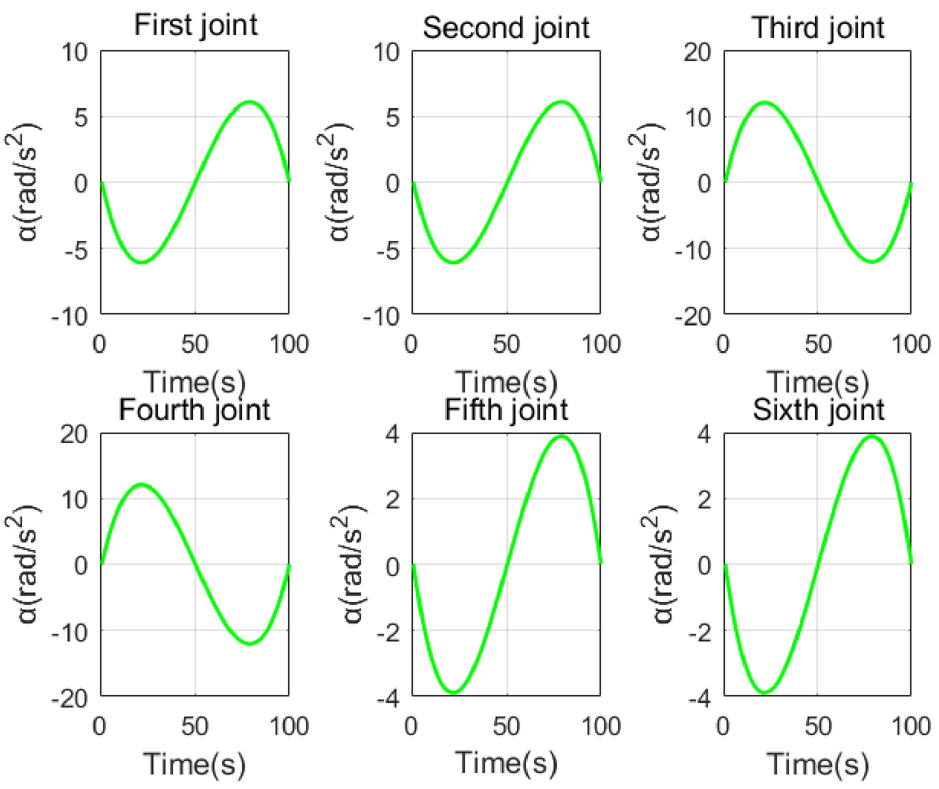

FIGURE 12. Angular acceleration of each joint with time.

V. CONCLUSION

Taking the original 6-DOF manipulator as an example, this paper constructs the D-H parameter model of the manipulator, establishes the coordinate system of the manipulator linkage, and then derives the coordinate change matrix through the formula, and completes the positive kinematics model of the 6-DOF manipulator arm and its solution. The solution of the inverse kinematics of the six-degree-of-freedom manipulator arm is completed by the analytical method.

Following the fundamental principle of the GEO algorithm, two enhancement strategies are presented, resulting in the introduction of the IGEO algorithm. Various simulation experiments were conducted in obstacle-laden environments with an assortment of balls, ultimately leading to the successful path planning of the six-degree-of-freedom robotic arm. The results of the simulation demonstrate that the IGEO algorithm successfully enables the six-degree-of-freedom manipulator to locate the most optimal path, and exhibits strong robustness and accuracy even as the number of obstacles gradually increases.

The improved algorithm proposed in this paper has an optimal fitness value of 845.29 and an average fitness value of 1237.15 for the case of four obstacles. Compared to several other algorithms the maximum improvement is 29.88%, the minimum improvement is 14.16%, and compared to the original algorithm the improvement is 15.38%.

This paper makes a study for the application of population intelligence optimisation algorithms in robotic arms, but this is simulated when the obstacles are static. In real life, robotic arms face more complex situations in motion, and the application of population intelligence optimisation algorithms in these situations will be the focus of the next research that can be carried out.

ACKNOWLEDGMENT

The authors would like to thank China Southern Power Grid Company Ltd., for providing data support and electricity supply bureau for providing the simulation.

REFERENCES

[1] Y. Yang, J. Han, Z. Liu, Z. Zhao, and K.-S. Hong, "Modeling and adaptive neural network control for a soft robotic arm with prescribed motion constraints," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 2, pp. 501–511, Feb. 2023.

[2] X. Cheng and M. Zhao, "Analysis on the trajectory planning and simulation of six degrees of freedom manipulator," in *Proc. 3rd Int. Conf. Mech., Control Comput. Eng. (ICMCCE)*, Sep. 2018, pp. 385–387.

[3] Y.-H. Deng and J.-Y. Chang, "Human-like posture correction for seven-degree-of-freedom robotic arm," *J. Mech. Robot. Trans.*, vol. 14, no. 2, Apr. 2022, Art. no. 024501.

[4] Ö. Ekrem and B. Aksoy, "Trajectory planning for a 6-axis robotic arm with particle swarm optimization algorithm," *Eng. Appl. Artif. Intell.*, vol. 122, Jun. 2023, Art. no. 106099.

[5] Y. Dai, C. Xiang, Y. Zhang, Y. Jiang, W. Qu, and Q. Zhang, "A review of spatial robotic arm trajectory planning," *Aerospace*, vol. 9, no. 7, p. 361, Jul. 2022.

[6] Y. Du and Y. Chen, "Time optimal trajectory planning algorithm for robotic manipulator based on locally chaotic particle swarm optimization," *Chin. J. Electron.*, vol. 31, no. 5, pp. 906–914, Sep. 2022.

[7] S. Ni, W. Chen, H. Ju, and T. Chen, "Coordinated trajectory planning of a dual-arm space robot with multiple avoidance constraints," *Acta Astronautica*, vol. 195, pp. 379–391, Jun. 2022.

[8] M. Wang, J. Luo, J. Yuan, and U. Walter, "Coordinated trajectory planning of dual-arm space robot using constrained particle swarm optimization," *Acta Astronautica*, vol. 146, pp. 259–272, May 2018.

[9] J. Peng, H. Wu, C. Zhang, Q. Chen, D. Meng, and X. Wang, "Modeling, cooperative planning and compliant control of multi-arm space continuous robot for target manipulation," *Appl. Math. Model.*, vol. 121, pp. 690–713, Sep. 2023.

[10] Z. Jiulong and W. Xiaofeng, "Analysis and research of several new intelligent optimization algorithms," *J. Frontiers Comput. Sci. Technol.*, vol. 16, no. 1, pp. 88–105, 2022.

[11] L. G. Wang, Q. H. Shi, and Y. Hong, "Hybrid optimization algorithm of PSO and AFSA," *Comput. Eng.*, vol. 36, no. 5, pp. 176–178, Mar. 2010.

[12] L. T. Zhang, L. Zhang, Q. Song, W. H. Qian, and X. Song, "Research on optimal control method of single flexible manipulator by improved particle swarm optimization algorithm," (in Chinese), *Mech. Sci. Technol. Aerosp. Eng.*, pp. 1–6, Oct. 2023.

[13] D. M. Li, H. Du, Y. L. Zhao, H. L. Cao, T. Wang, Y. Wang, P. W. Ma, and X. H. Luan, "Research on planning and optimization of trajectory for underwater vision welding robot," (in Chinese), *Mach. Tool Hydraul.*, vol. 51, no. 8, pp. 35–41. Apr. 2023.

[14] A. Z. Nasrollahy and H. H. S. Javadi, "Using particle swarm optimization for robot path planning in dynamic environments with moving obstacles and target," in *Proc. 3rd UKSim Eur. Symp. Comput. Modelling Simulation*, Nov. 2009, pp. 60–65.

[15] C. Lopez-Franco, J. Hernandez-Barragan, A. Y. Alanis, and N. Arana-Daniel, "A soft computing approach for inverse kinematics of robot manipulators," *Eng. Appl. Artif. Intell.*, vol. 74, pp. 104–120, Sep. 2018.

[16] R. Szczepanski, K. Erwinski, M. Tejer, A. Bereit, and T. Tarczewski, "Optimal scheduling for palletizing task using robotic arm and artificial bee colony algorithm," *Eng. Appl. Artif. Intell.*, vol. 113, Aug. 2022, Art. no. 104976.

[17] A. K. Sadhu, A. Konar, T. Bhattacharjee, and S. Das, "Synergism of firefly algorithm and Q-learning for robot arm path planning," *Swarm Evol. Comput.*, vol. 43, pp. 50–68, Dec. 2018.

[18] C. Yang, Y. Jiang, J. Na, Z. Li, L. Cheng, and C.-Y. Su, "Finite-time convergence adaptive fuzzy control for dual-arm robot with unknown kinematics and dynamics," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 3, pp. 574–588, Mar. 2019.

[19] T. Chai, C. G. Zhuang, and B. Zhang, "Online motion planning algorithm for 6-DOF manipulator in dynamic environment," *Machinery*, vol. 59, no. 2, pp. 7–16, Feb. 2021.

[20] A. El-Sherbiny, M. A. Elhosseini, and A. Y. Haikal, "A comparative study of soft computing methods to solve inverse kinematics problem," *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 2535–2548, Dec. 2018.

[21] B. Song, Z. Wang, and L. Zou, "On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm," *Cogn. Comput.*, vol. 9, no. 1, pp. 5–17, Feb. 2017.

[22] C. Zammit and E.-J. van Kampen, "Comparison between A* and RRT algorithms for 3D UAV path planning," *Unmanned Syst.*, vol. 10, no. 2, pp. 129–146, Apr. 2022.

[23] L. Zhaoying, S. Ruoling, and Z. Zhao, "A new path planning method based on sparse A* algorithm with map segmentation," *Trans. Inst. Meas. Control*, vol. 44, no. 4, pp. 916–925, Feb. 2022.

[24] Y. Su, J. Liu, X. Xiang, and X. Zhang, "A responsive ant colony optimization for large-scale dynamic vehicle routing problems via pheromone diversity enhancement," *Complex Intell. Syst.*, vol. 7, no. 5, pp. 2543–2558, Jun. 2021.

[25] Y.-I. Joo, "Rescuing path guidance method in a ship using genetic algorithm," *J. Korean Soc. Mar. Eng.*, vol. 43, no. 9, pp. 744–749, Nov. 2019.

[26] A. Mohammadi-Balani, M. Dehghan Nayeri, A. Azar, and M. Taghizadeh-Yazdi, "Golden eagle optimizer: A nature-inspired metaheuristic algorithm," *Comput. Ind. Eng.*, vol. 152, Feb. 2021, Art. no. 107050.

[27] Q. Luo, H. B. Wang, Y. Zheng, and N. Wang, "Research on path planning of mobile robot based on improved ant colony algorithm," *Neural Comput. Appl.*, vol. 32, no. 6, pp. 1555–1566, Dec. 2020.

[28] M. Wang, "Real-time path optimization of mobile robots based on improved genetic algorithm," *Proc. Inst. Mech. Eng. I, J. Syst. Control Eng.*, vol. 235, no. 5, pp. 646–651, May 2021.

[29] Z. Wang, Y. Gan, and X. Dai, "Assembly-oriented task sequence planning for a dual-arm robot," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 8455–8462, Jul. 2022.

SONGHUA HU was born in Anning, Yunnan. He received the Graduate degree in electricity management from the Yunnan Electric Power School, in 1996. Since 1996, he has been a First-Class Top-Notch Skill Expert and a Senior Technician in substation operation with Baoshan Power Supply Bureau. His main research interest includes intelligent application of power system substations.

YUHANG HAN was born in Handan, Hebei. He received the B.S. degree in mechanical engineering from the Kunming University of Science and Technology, Kunming, Yunnan, in 2023. His research interests include swarm intelligence and deep learning.

MINGXIAN LIU was born in Qujing, Yunnan. He received the B.S. degree in mechanical engineering from the Kunming University of Science and Technology, Kunming, Yunnan, in 2012. His research interests include distribution network operation inspection, live operation, and artificial intelligence.

ZHENHUA XU received the Graduate degree from Shanghai Jiao Tong University. He was with the FANUC Robotics and SAIC-GM SGM (Pan-Asia Research and Development Centre) as a Research and Development Engineer. He received the Shanghai Automotive Technology Innovation Award and a number of invention patents; successively participated in the Founding of King Robotics, Fourier Intelligence, and two robotics companies as the Co-Founder and the CTO.

HONGYU XIANG was born in Wuhan, Hubei. He received the B.S. degree in mechanical engineering from the Kunming University of Science and Technology, Kunming, Yunnan, in 2023. His research interests include artificial intelligence and intelligent systems.