Proceedings of CONF-MLA 2024 Workshop: Mastering the Art of GANs: Unleashing Creativity with Generative Adversarial Networks
DOI: 10.54254/2755-2721/111/2024.CH17888

Kinematic Analysis and Trajectory Planning Based on a Six-Axis Robotic Arm

Jiayi Li[1], Quan Zhang[2,3,*]

[1]School of International Engineering, Changsha University of Science and Technology, Changsha, 410000, China
[2]School of Mechanical Engineering, Hefei University of Technology, Hefei, 230000, China

[3]2022211851@mail.hfut.edu.cn
*corresponding author

Abstract. Kinematic analysis and time-optimal trajectory planning are applied to the robot in order to enhance its operational efficiency and preserve its continuous, smooth running trajectory. In this paper, the robot kinematics model is established based on the D-H parametric method with the Daqi Elfin series E05 six-axis robot as the research object. After that, forward and inverse kinematic analysis is carried out. And the segmented polynomial interpolation trajectory planning model of robotic arm is optimized in time based on the improved particle swarm algorithm. And MATLAB robot toolbox is used to build the robotic arm simulation model. From there, the trajectory planning simulation experiments are performed. As well as the experimental results show that the improved particle swarm algorithm effectively reduces the trajectory planning time. In the simulation experiment the time is reduced by 40.78% compared to the initial setting. The effectiveness of the robotic arm is greatly increased by this research.

Keywords: D-H parametric method, particle swarm algorithm, trajectory planning.

1. Introduction

Robotics is becoming a more significant part of modern manufacturing due to the quick growth of intelligent manufacturing and industrial automation. Six-axis robotic arm has become an important representative among industrial robots due to its flexibility and wide range of applications. It is capable of performing complex operational tasks such as welding, assembly and handling, and has shown broad application prospects in the medical, service and research fields.

Robot kinematics is the foundation for robots to be able to accomplish tasks smoothly, accurately and efficiently. The D-H parametric method proposed by SUN et al. has been widely used as a generalized method to solve robot kinematics problems [1], which solves the forward and inverse kinematics problems by mathematical analytical method [2]. However, with the complexity of the robotic arm structure and the diversification of application scenarios, the traditional analytical method has gradually exposed the problems of high computational complexity and poor solution stability. For this reason, scholars have begun to explore more efficient and accurate solution methods, such as neural networks, genetic algorithms and other intelligent optimization algorithms in inverse kinematics solutions, and have made positive progress [3]. Although the numerical method has advantages in

© 2024 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/111/2024.CH17888

applicability and scalability, it increases the complexity of computation and reduces the computational efficiency due to its need for iterative computation and strict requirements on algorithm design and convergence conditions. On the contrary, the analytical method establishes a corresponding mathematical model for a specific robot, which can directly derive an analytical solution with high accuracy and fast computational speed, and is better suited for resolving the UR5 robotic arm's kinematics [4].

Trajectory planning is crucial for realizing safe, efficient, precise and autonomous actions of robots and is one of the key aspects in robotics. In order to improve the efficiency, it is necessary to optimize the time solution for robotic arm trajectory planning. Commonly used time optimization algorithms include particle swarm algorithms, sparrow search algorithms, and genetic algorithms. Each of these algorithms has its own advantages, and the parameters of the algorithms often rely on empirical determination.

Thus, the kinematic inverse solution is finished in this work after the E05 robotic arm's kinematic model is established using the D-H parametric technique. Furthermore, a time-optimal segmented polynomial interpolation based on an improved particle swarm method is proposed for trajectory planning in order to realize the robotic arm's time-optimal trajectory planning [5]. The traditional particle swarm algorithm's poor convergence time and tendency toward local optimization are further improved by this method. The 6-degree-of-freedom robotic arm's motion process is simulated and analyzed with the use of the toolbox in MATLAB finally[6]. Thus, the accuracy of the kinematic positive inverse solution and the shortest time consumed on the basis of ensuring the smooth operation of the robotic arm are further verified. The robotic arm has achieved its time-optimal trajectory planning and is now ready to use it [7].

2. Robot Modeling

Robot modeling is the basis for realizing efficient control, this paper takes the Daqi Elfin series E05 six-axis robot as an example, and adopts the standard D-H parameter method to obtain the simplified robotic arm linkage structure and linkage coordinate system, as shown in Figure 1.

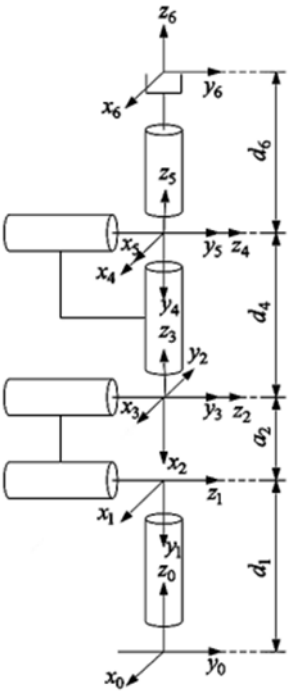


Figure 1. D-H parametric linkage coordinate system.

DOI: 10.54254/2755-2721/111/2024.CH17888

By parameterizing each joint, the following Table 1 was determined:

Table 1. D-H Parameter List

Cartilage □	Linkage offset d_i	Connecting rod length a_i	Joint angle θ_i	Connecting rod torsion angle a_i	Movement range
1	0.25	0	0	-90	-150-150
2	0	-0.4	0	0	-180~180
3	0	0	0	-90	-100~100
4	0.5	0	0	-90	-180~180
5	0	0	0	90	-180~180
6	0.2	0	0	0	-180~180

With the above parameters, the transformation matrix between neighboring connecting rods is calculated, and the corresponding transformation matrix for each joint is obtained T_i^{i+1}:

$$T_i^{i+1} = \begin{bmatrix} c_i & -s_i c_{\alpha_i} & s_i s_{\alpha_i} & a_i c_i \\ s_i & c_i c_{\alpha_i} & -c_i s_{\alpha_i} & a_i s_i \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Calculate the transformation matrix for each joint one by one:

1.Joint 1.transformation matrix :

$$T_0^1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0.25c_1 \\ s_1 & 0 & c_1 & 0.25s_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.Joint 2.transformation matrix:

$$T_1^2 = \begin{bmatrix} c_2 & -s_2 & 0 & -0.4c_2 \\ s_2 & c_2 & 0 & -0.4s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Joint 3. transformation matrix :

$$T_2^3 = \begin{bmatrix} c_3 & 0 & s_3 & 0 \\ s_3 & 0 & -c_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Joint 4. transformation matrix :

$$T_3^4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0.5c_4 \\ s_4 & 0 & c_4 & 0.5s_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. Joint 5. transformation matrix :

DOI: 10.54254/2755-2721/111/2024.CH17888

$$T_4^5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6.Joint 6.transformation matrix :

$$T_5^6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0.2c_6 \\ s_6 & c_6 & 0 & 0.2s_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The total transformation matrix is calculated by matrix multiplication :

$$T_0^6 = T_0^1 \cdot T_1^2 \cdot T_2^3 \cdot T_3^4 \cdot T_4^5 \cdot T_5^6$$

3. Kinematic analysis

3.1. Positive kinematic analysis

In positive kinematic analysis, the end-effector's position is calculated by taking into account the angles (θ) of each joint. Multiplying the transformation matrices of each joint one by one can yield the total transformation matrix of the end-effector.

3.1.1. Construction of transformation matrices

Make surethat each joint has a transformation matrix that matches the data in the D-H parameter table. The specific construction process is as follows:

1.Transformation matrix for joint 1 :

$$T_0^1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0.25 \cdot c_1 \\ s_1 & 0 & c_1 & 0.25 \cdot s_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where, c_1=cos(θ_1), s_1=sin(θ_1).

2.Transformation matrix for joint 2:

$$T_1^2 = \begin{bmatrix} c_2 & -s_2 & 0 & -0.4 \cdot c_2 \\ s_2 & c_2 & 0 & -0.4 \cdot s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.Transformation matrix for joint 3:

$$T_2^3 = \begin{bmatrix} c_3 & 0 & s_3 & 0 \\ s_3 & 0 & -c_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.Transformation matrix for joint 4:

DOI: 10.54254/2755-2721/111/2024.CH17888

$$T_3^4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0.45 \cdot c_4 \\ s_4 & 0 & c_4 & 0.45 \cdot s_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.Transformation matrix for joint 5:

$$T_4^5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6.Transformation matrix for joint 6:

$$T_5^6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0.2 \cdot c_6 \\ s_6 & c_6 & 0 & 0.2 \cdot s_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.1.2. Calculating the total transformation matrix

The transformation matrices of the individual joints are multiplied together to obtain the total transformation matrix of the end-effector with respect to the base:

$$T_0^6 = T_0^1 \cdot T_1^2 \cdot T_2^3 \cdot T_3^4 \cdot T_4^5 \cdot T_5^6$$

By stepwise multiplication, the final total transformation matrix is obtained in the form:

$$T_0^6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.1.3. Extracting positional information

The position and attitude information of the end-effector is extracted from the total transformation matrix:

Position:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} T_{06_{14}} \\ T_{06_{24}} \\ T_{06_{34}} \end{bmatrix}$$

Attitude:

The attitude of the end-effector can be represented by the direction cosine matrix, usually in Euler angle or quaternion representation. The orientation cosine matrix is extracted as follows:

$$\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}$$

3.2. Inverse kinematic analysis

Inverting the angles (θ) of the joints of the robotic arm is the goal of the inverse kinematic analysis based on the known end-effector's position in Cartesian space.

DOI: 10.54254/2755-2721/111/2024.CH17888

3.2.1. Perform an inverse kinematic solution

First, the target attitude of the end-effector is determined, the target position $\left(p_x, p_y, p_z\right)$ and attitude of the end-effector are known (directional cosine or Euler angle), and then the joint angles are solved step-by-step, with inverse kinematics solved by the following steps.

1.To solve θ_1 :

Calculation of the angle of the first joint from p_x and p_y of the end-effector:

$$\theta_1 = \mathrm{Atan}\left(\frac{p_y}{p_x}\right)$$

2.To solve r：

Calculate the horizontal distance of the end-effector;

$$r = \sqrt{p_x^2 + p_y^2}$$

3.To solve the effective height z_{eff}：

Calculating the effective height is achieved by utilizing the target height of the end effector:

$$z_{eff} = p_z - d_1$$

4.To solve θ_3 :

Using trigonometric functions and geometric relationships, calculate the angle of the third joint. Let and be the lengths of the second and third joints. Let L_2 and L_3 be the lengths of the second and third connecting rods:

$$\theta_3 = \mathrm{Atan}\left(\frac{z_{eff}}{r}\right)$$

5.To solve θ_2 :

Calculate the angle of the second joint through geometric relationships:

$$\theta_2 = \mathrm{Atan}\left(\frac{h}{L}\right)$$

Where h is the height of the end-effector and L is the horizontal distance to the second joint.

6.To solve θ_4 :

The angle of the fourth joint is calculated based on the attitude of the end-effector and the joint configuration:

$$\theta_4 = \mathrm{Atan2}\left(o_y, n_y\right)$$

3.2.2. Selection of inverse kinematic solutions

Since there may be multiple solutions for inverse kinematics, it is crucial to select the appropriate solution. The selection criteria include the principle of continuity, which prioritizes solutions that are close to the current joint angles to minimize abrupt changes during motion. The principle of minimum energy consumption, to select the solution that makes the joint travel the shortest, usually selecting the solution with less change in the joint angle close to the base, as well as ensuring that the selected solution is within the workspace of the robot to avoid the joints exceeding their motion limitations. Attention is also paid to task prioritization, and the solution that best meets the operational requirements is selected based on the needs of the specific task.

3.3. Examples of detailed calculations

Assuming that the target position of the end-effector is (0.5,0.5,0.3), the angle of each joint is calculated by inverse kinematics.

Count θ_1:

DOI: 10.54254/2755-2721/111/2024.CH17888

$$\theta_1 = \text{Atan}\left(\frac{0.5}{0.5}\right) = 45^\circ$$

Count r: Count Effective height z_{eff}:

$$z_{eff} = 0.3 - 0.25 = 0.05$$

Count θ_3: θ_3 =A $\tan\left(\frac{0}{.}050.707\right) \approx 4.04^\circ$

Count θ_2: assuming $L_2 = 0.4$ and $L_3 = 0.45$,calculate h:

$$h = \sqrt{0.4^2 + 0.45^2} \approx 0.583;$$

$$\theta_2 = \text{A}\tan\left(\frac{0}{.}050.583\right) \approx 4.86^\circ$$

Count θ_4: Assuming that the attitude of the end-effector is given as

$$o_y = 0.1 \quad \text{and} \quad n_y = 0.9:$$

$$\theta_4 = \text{A}\tan2(0.1,0.9) \approx 6.34^\circ$$

Countθ_5 and θ_6: Assuming that the pose information is given as

$$s_5 = 0.2 \text{and} c_5 = 0.8:$$

$$\theta_5 = \text{A}\tan2(0.2,0.8) \approx 14.04^\circ$$

$$\text{Assuming } s_6 = 0.3 \text{ and } c_6 = 0.7: \quad \theta_6 = \text{A}\tan2(0.3,0.7) \approx 23.57^\circ$$

Through detailed forward kinematics and inverse kinematics analysis, this paper establishes an effective kinematic model for the XYZ series robots. The realization of the forward kinematics provides the basis for the position calculation of the end effector. Practical applications require the necessary support for the robot's path planning and control, which is provided by the solution of inverse kinematics. Specific computational examples demonstrate how the application of theory to practical problems lays the foundation for the subsequent design of robot control tubing methods.

4. Trajectory planning

To meet the requirements of trajectory planning, multiple interpolation functions are usually constructed in the joint space based on the operational requirements and constraints to guarantee the consistency and efficiency of the joint motion parameters [8]. In this paper, the advantages of cubic and quintic polynomials are combined in robot trajectory planning. A 3-5-3 polynomial interpolation is used. This method can obtain better trajectory fitting results and computational complexity. In addition, this method can maintain the smoothness and continuity of the trajectory curve and avoid oscillations and vibrations of the robot arm due to sudden speed changes at the interpolation points [9].

4.1. 3-5-3 Piecewise polynomial interpolation function construction

Now, the time interpolation function θ(t) is used to represent the functional relationship between the angle of each joint of the robot arm and time. The 3-5-3 piecewise polynomial interpolation function is expressed as follows:

$$\begin{gathered} \theta_{i1}(t_1) = a_{i13}t_1^3 + a_{i12}t_1^2 + a_{i11}t_1 + a_{i10} \\ \theta_{i2}(t_2) = a_{i2}t_2^5 + a_{i24}t_2^4 + a_{i23}t_2^3 + a_{i22}t_2^2 + a_{i21}t_2 + a_{i20} \\ \theta_{i3}(t_3) = a_{i33}t_3^3 + a_{i32}t_3^2 + a_{i31}t_3 + a_{i30} \end{gathered} \tag{1}$$

Where $\theta_{i1}(t_1)$、$\theta_{i2}(t_2)$、$\theta_{i3}(t_3)$ are the three-segment polynomial interpolation functions of joint i, respectively. The first and last segments are interpolated using a third-order polynomial, and the middle

DOI: 10.54254/2755-2721/111/2024.CH17888

segment is interpolated using a fifth-order polynomial.t_1、t_2、t_3 are the three interpolated trajectory times of joint respectively.

The trajectory constraints are that the four interpolated values x_{i0}、x_{i1}、x_{i2} and x_{i3} of the joint i are known, and the velocities and accelerations of the starting point and end point are 0. The joint displacement, angular velocity, and angular acceleration between the interpolation points must be continuous. According to the trajectory constraints, the solution is obtained by solving the 14 coefficient solutions a in the three polynomial equations.

4.2. Time-optimal trajectory planning based on particle swarm algorithm

4.2.1. Optimizing the objective function

Different interpolation times affect the motion parameters such as displacement, angular velocity and angular acceleration for each segment of the trajectory. Therefore, an optimization algorithm must be used to determine the appropriate interpolation time to choose the best path that satisfies the motion specifications. Next, the shortest time required for every joint of the robotic arm to complete the motion under kinematic constraints is investigated, and its optimization objective function is:

$$f(t) = min \sum_{i=0}^{n} (t_{i1} + t_{i2} + t_{i3}) \tag{2}$$

Where:$t_1 + t_2 + t_3$ is the fitness function.

4.2.2. Particle swarm optimization algorithm

Particle swarm optimization algorithm (PSO) is a meta-heuristic optimization algorithm, whose basic idea is to take the region of the problem to be solved as the target search space, and generate some particles in this space to search continuously. The position of each particle in the search space represents a pre-selected solution, and all the particles will continuously update their positions according to the details of both the population-wide and individual-optimal solutions in the hope of searching for the optimal solution in the end, but the traditional PSO algorithm is easy to be affected by the to-be-optimized function to converge to the local optimum in advance [10]. During each iteration, all particles in the particle swarm must move from their current position to a new position based on their updated velocities. The following formula is used to update the particles' location and velocity:

$$v_{id}(n+1) = \omega v_{id}(n) + c_1 r_1(n)\left[p_{jd}(n) - x_{id}(n)\right] + c_2 r_2(n)\left[p_{gd}(n) - x_{id}(n)\right] \tag{3}$$

$$x_{id}(n+1) = x_{id}(n) + v_{id}(n+1) \tag{4}$$

Where n is the current iteration number;$v_{id}(n)$is the d-dimensional component of the velocity of the ith particle at the nth update;$x_{id}(n)$ is the d-dimensional component of the position of the ith particle at the nth update;p_j is the optimal position that each particle finds for itself during the whole flight process.p_g is the optimal position found by the whole swarm of particles.$r_1(n)$、$r_2(n)$ are randomly generated values between 0 and 1. c_1、c_2 are the self and population learning factors respectively, and ω is the inertia weight.

The parameters in the standard particle swarm algorithm are fixed.ω describes the inertia of the particle. In the pre-evolutionary period ω need to be bigger to guarantee that every particle may fly separately and thoroughly explore the area. ω later on need to be smaller and learn more from other particles.$c_1$and $c_2$are the learning factors for self and population, respectively.$c_1$should be as large as possible in the early stage and $c_2$should be as large as possible in the later stage. This balances the global and local search capabilities of the particle. The 3 parameters together influence the direction of flight of the particles. Supposing other particles find a better position, but the inertia of the current particle is

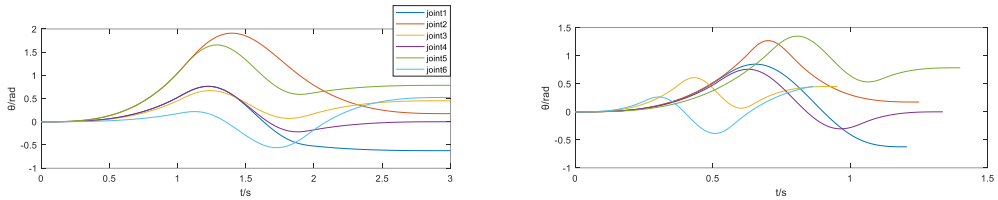
DOI: 10.54254/2755-2721/111/2024.CH17888

too large to fly to the better position quickly, which may lead to problems such as the algorithm being prone to premature maturation or the algorithm being slow to converge at a later stage.The values of the inertia weight ω and the learning factors c_1、c_2 in the PSO algorithm will directly affect the algorithm's convergence performance. As opposed to the traditional PSO algorithm, where ω、c_1 and c_2 are constant, this paper uses adaptive inertia weights and a nonlinear learning factor. Adaptive adjustment of inertia weights integrates both the particle's intended value and the number of repeats. Initially, a larger ω enhances the global search capability of the particles. When a particle approaches the optimal solution, the ω is reduced to increase the local search capability of the particle, which greatly improves the results of PSO optimization.Combining the learning factors c_1 and c_2,when c_1 is increased, the individual cognitive ability of the particles is enhanced, but the convergence speed is slow;when c_2 is increased, the social cognitive ability of the particles is enhanced, and the convergence speed is accelerated, but the algorithm is very prone to premature maturity.

In order to ensure the searching capabilities of the PSO algorithm and coordinate the individual and social cognitive abilities of the particles, a method of dynamically adjusting the learning factors is used, which associates the value of the learning factor with the particle's current number of iterations. In the early stages of the search, the search significance of the individual particles is greater than the global search significance, so $c_1>c_2$;In order to obtain a global optimal solution more quickly during the later stages of the search, the communication between the particles and the swarm should be increased, so $c_1<c_2$.

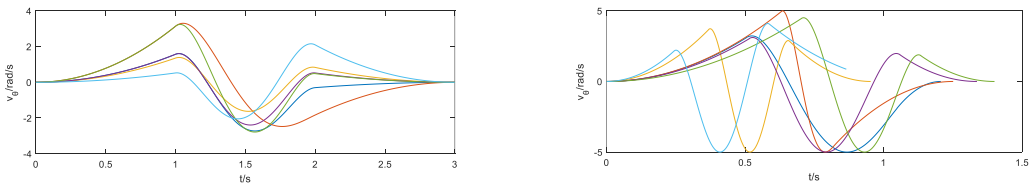
5. Simulation experiment

In order to ensure that the joints of the 6-degree-of-freedom robot arm can move quickly and smoothly to the specified position, two path positions are interpolated between the starting position and the end position, and a 3-5-3 piecewise polynomial is used to connect each segment of the trajectory. The starting point and end point have a speed and acceleration of 0, and the speed and acceleration at the intersection of each segment are equal. From the simulation results, the position, speed and acceleration change curves of each joint of the robot arm are shown in Figure 2, Figure 3 and Figure 4.



(a) Position curve before optimization (b)Optimized position curve

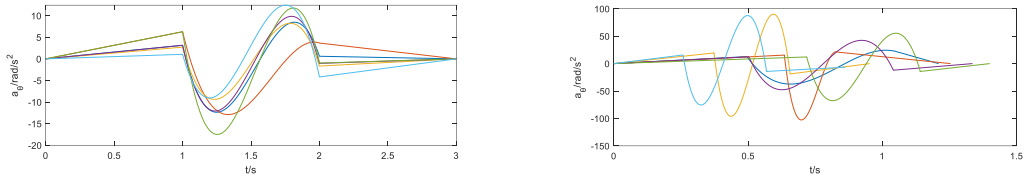
Figure 2. Position curves before and after optimization



(a)Velocity profile before optimization (b)Optimized velocity profile

Figure 3. Velocity profile before and after optimization

DOI: 10.54254/2755-2721/111/2024.CH17888



(a)Acceleration profile before optimization (b)Optimized acceleration profile

Figure 4. Acceleration profile before and after optimization

After the optimization of particle swarm algorithm, the optimal running time of each joint of the robotic arm in the 3-segment polynomial trajectory is obtained under the velocity constraints as shown in Table 2.

Table 2. Optimal running time for each joint

joint i	t_1	t_2	t_3
1	0.499613	0.553227	0.1554
2	0.63217	0.185387	0.431953
3	0.378519	0.284601	0.28887
4	0.500773	0.539177	0.29544
5	0.712767	0.421543	0.26596
6	0.251536	0.339574	0.51062

The trajectories of the joints of the robotic arm are significantly changed after optimization using the improved PSO algorithm. Compared with the trajectory of the robot arm joints before optimization, the position trajectory and acceleration of the robot arm joints are smoother; the speed is faster; and the time taken to complete the motion trajectory is also significantly reduced. In order to ensure that every joint on the robotic arm has sufficient time to complete the corresponding trajectory, the maximum value of the time used by each joint to complete the same segment of the trajectory is taken to be the running time of that segment of the robotic arm. From Table 2, the three times are 0.712767s, 0.553227s and 0.51062s, respectively. The total time of the three periods added together is 1.776614s, which is 1.223386s less than the initial value of 3s before optimization, and the time taken by the robotic arm to complete the 3-segment trajectory is reduced by about 40.78%.

6. Conclusion

In this paper, a forward and inverse kinematics analysis model and an optimal time trajectory planning algorithm are established for the Dazu E05 collaborative robot. Robotics Toolbox in MATLAB is used to perform motion simulation analysis of the robotic arm. The results of the study show that the mathematical model of the robotic arm established based on the standard D-H parametric method performs well in terms of reasonableness and reliability, and proves that the computational results of the forward-inverse kinematics algorithm are accurate and credible, and that it can accurately predict the trajectory and attitude of the robotic arm in the simulation experiments. The smooth and continuous end motion trajectory of the robotic arm is obtained through the 3-5-3 segmented polynomial interpolation algorithm planning, which improves the working accuracy of the robotic arm and enhances the working smoothness, and proves that the method is a real and feasible trajectory planning method. The improved particle swarm algorithm effectively reduces the trajectory planning time, which is 40.78% shorter than the initial setup in the simulation experiments, and contributes significantly to raising the robotic arm's efficiency. Research on kinematic analysis and trajectory planning of a six-axis robotic arm is still deficient: the model is oversimplified and difficult to predict accurately in practical applications; The trajectory planning algorithm is inefficient and poorly adaptable; the lack of integration of sensors and actuators affects the control accuracy. In future research in this field, more refined models can be used

DOI: 10.54254/2755-2721/111/2024.CH17888

to improve prediction accuracy and study efficient and adaptive trajectory planning algorithms. Attention should also be paid to strengthening the integration of sensors and actuators to improve the overall performance of the system.

Authors Contribution

All the authors contributed equally and their names were listed in alphabetical order.

References

[1] Sun J D Cao G Z Li W B et al 2017 Analytical inverse kinematic solution using the D-H method for a 6-DOF robot 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)

[2] Concepcion R et al 2021 Denavit-Hartenberg-based Analytic Kinematics and Modeling of 6R Degrees of Freedom Robotic Arm for Smart Farming Journal of Computational Innovations and Engineering Applications vol5 no 2 pp1-7

[3] KöKer R I 2013 A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization Information Sciences vol222 pp528-543

[4] Jin M et al 2020 An efficient and accurate inverse kinematics for 7-DOF redundant manipulators based on a hybrid of analytical and numerical method IEEE Access pp16316-16330

[5] Shin S, Shi Y 2022 SONG Jianfeng, et al. Research on trajectory planning of drilling truck manipulator based on improved particle swarm algorithm Industrial and Mining Automation vol 48 no 03 pp71-77+85

[6] Cai J Deng J Zhang W et al 2021 Modeling Method of Autonomous Robot Manipulator Based on D‐H Algorithm Mobile Information Systems vol1 pp4448648

[7] Du Y Chen Y 2022 Time optimal trajectory planning algorithm for robotic manipulator based on locally chaotic particle swarm optimization Chinese Journal of Electronics vol 31 no5 pp906-914

[8] Zhang L Li X 2021 A review of the research status of trajectory planning for industrial robots Mechanical Science and Technology vol 40no 06 pp853-862

[9] Ke W Ping X 2024 Kinematic analysis and trajectory planning based on a six-axis robot Combined Machine Tools and Automation Processing Technology vol 07 pp17-22

[10] Song B Wang Z Zou L 2021 An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve Applied Soft Computing vol 100 no 1 pp106960