sensors

MDPI

Article

A Novel Multi-Objective Trajectory Planning Method for Robots Based on the Multi-Objective Particle Swarm Optimization Algorithm

Jiahui Wang [1], Yongbo Zhang [1,2,*], Shihao Zhu [1] and Junling Wang [3]

[1] School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, China
[2] Aircraft and Propulsion Laboratory, Ningbo Institute of Technology, Beihang University, Ningbo 315100, China
[3] School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China
* Correspondence: zhangyongbo@buaa.edu.cn

Abstract: The three performance indexes of the space robot, travel time, energy consumption, and smoothness, are the key to its important role in space exploration. Therefore, this paper proposes a multi-objective trajectory planning method for robots. Firstly, the kinematics and dynamics of the Puma560 robot are analyzed to lay the foundation for trajectory planning. Secondly, the joint space trajectory of the robot is constructed with fifth-order B-spline functions, realizing the continuous position, velocity, acceleration, and jerk of each joint. Then, the improved multi-objective particle swarm optimization (MOPSO) algorithm is used to optimize the trajectory, and the distribution uniformity, convergence, and diversity of the obtained Pareto front are good. The improved MOPSO algorithm can realize the optimization between multiple objectives and obtain the trajectory that meets the actual engineering requirements. Finally, this paper implements the visualization of the robot's joints moving according to the optimal trajectory.

Keywords: Puma560 robot; multi-objective trajectory planning; MOPSO; B-spline

Citation: Wang, J.; Zhang, Y.; Zhu, S.; Wang, J. A Novel Multi-Objective Trajectory Planning Method for Robots Based on the Multi-Objective Particle Swarm Optimization Algorithm. *Sensors* **2024**, *24*, 7663. https://doi.org/10.3390/s24237663

Academic Editor: Jesús Ureña

Received: 8 November 2024
Revised: 25 November 2024
Accepted: 28 November 2024
Published: 29 November 2024

1. Introduction

With the rapid development of science and technology in today's world, humankind's space exploration has shown unprecedented momentum, and space technology has gradually risen to the focus of public attention. Among them, space robots have become indispensable to space activities due to their powerful functions and high adaptability to the space environment. A series of challenging tasks, such as assembling satellite parts, capturing space targets, and monitoring alien spacecraft, cannot be realized without the support of space robots. These tasks and harsh working environments also set higher requirements for space robots' travel time, energy consumption, and smoothness.

The primary method to improve each performance index is to design and plan the robot's trajectory rationally. Trajectory planning can be performed both in Cartesian space and in joint space. The latter plans the trajectory of each joint of the robot, which has a small amount of calculation and enables the real-time control of the robot. Trajectory planning can usually be divided into two steps. The first step is to interpolate between given path points using interpolation algorithms to obtain a trajectory-time sequence. The second step is to optimize the trajectory in terms of single or multiple performance indexes within the constraints of the kinematics and dynamics of the robot [1].

The mainstream interpolation algorithms in joint space are polynomial interpolation and spline curve interpolation, and the former is mainly used in early research [2–5]. To obtain the robot's trajectory, Ref. [6] used a cubic polynomial to connect the path points. This method is simple to calculate, but the acceleration curve obtained is not continuous, the smoothness could be better, and it tends to cause rigid impacts. Ref. [7] used quintic

polynomial interpolation to ensure the continuity of acceleration, but it increased the amount of calculation, and there was still a problem of easy distortion. Compared to the fifth-degree polynomial, the seventh-degree polynomial adds constraints to the jerk at the start and termination points, realizing the continuity of the jerk. However, the eight boundary conditions increase the difficulty of the solution, and the high-order polynomial interpolation may cause the Runge phenomenon [8]. With the deepening of research, some scholars have applied spline curves to the trajectory planning of robots [9–14]. Ref. [15] used the fifth-order B-spline curves to interpolate the joint space trajectory, which realized the continuity of the jerk and set the velocity and acceleration at the start and stop time to be 0. When interpolating with seventh-order B-spline curves, it is possible to specify the acceleration at the start and stop time, but the calculation process is complicated [1].

Trajectory optimization mostly takes a single performance index as the optimization objective. Robot efficiency, the shortest time required by the robot to perform a task, was the earliest goal of trajectory planning [16–19]. Ref. [20] constructed the trajectory with a quintic polynomial and reduced the robot's travel time by 75.35% through the improved MOPSO algorithm under the constraints of each joint's angles, velocities, and accelerations. The time-optimal trajectory often leads to a large impact on the robot, affecting its motion accuracy and shortening the service life of the robot structure. Many scholars have solved this problem by optimizing the jerk. Ref. [7] effectively increased the smoothness of the robot by optimizing the maximum value of joint jerks. Ref. [21] combined the PSO algorithm with K-means clustering to achieve a fast solution for joint trajectories with minimal shocks. Energy consumption optimization is also an important issue for robots working in unique environments such as oceans, deserts, and space [22–25]. Ref. [26] obtained a parameterized dynamic robot model through identification experiments and used a sequential quadratic programming solver to minimize a mechanical energy-based cost function under consideration of physical constraints. These three performance indexes all play an important role in the motion of the space robot, and the single optimization objective ignores the intricate balance between them. Therefore, there are studies on the comprehensive optimization of multiple objectives [27–29]. Ref. [30] realized the comprehensive optimization of time, energy, and smoothness by a differential evolution algorithm. Ref. [31] used the NSGA-II algorithm to optimize the same three objectives and obtained Pareto optimal solution sets, thus obtaining the high-order continuous optimal trajectories.

There are few studies on multi-objective optimization problems, so this paper proposes a trajectory planning method that can make the robot's travel time, energy consumption, and smoothness achieve the integrated optimal state when performing the task. In this paper, continuous and smooth joint space trajectories are constructed using fifth-order B-spline functions, which also realize the specification of the velocity and acceleration of the robot at the start/stop moment. The trajectories are then optimized using the improved MOPSO algorithm to obtain the Pareto optimal solution sets, from which suitable solutions are selected according to practical needs.

The rest of the paper is organized as follows. Section 2 analyzes the kinematics and dynamics of the Puma 560 robot manufactured by Unimation, USA. Section 3 uses fifth-order B-spline curves to construct the joint space trajectories of the robot, builds a mathematical model for the multi-objective optimization problem based on Section 2, and solves the model with the improved MOPSO algorithm. Section 4 performs simulation experiments on multi-objective trajectory planning. Section 5 summarizes this article.

2. Kinematics and Dynamics Analysis

2.1. Kinematics Analysis

This section gives the kinematics model of the Puma560 robot and analyzes its forward and inverse kinematics. The MDH (Modified Denavit–Hartenberg) coordinate system [32] shown in Figure 1 is established on the Puma560 robot with six rotary joints. The link parameters from the MATLAB R2020b Robot Toolbox are shown in Table 1.

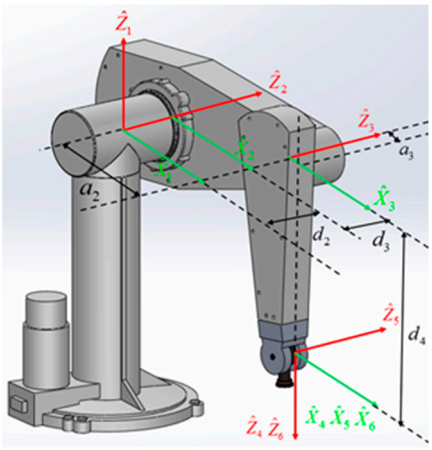


Figure 1. The MDH coordinate system of the Puma560 robot.

Table 1. Link parameters of the Puma560 robot.

Link i	α_{i-1} (rad)	a_{i-1} (m)	d_i (m)	θ_i (rad)
1	0	0	0	θ_1
2	−1.5708	0	0.2435	θ_2
3	0	0.4318	−0.0934	θ_3
4	1.5708	−0.0203	0.4331	θ_4
5	−1.5708	0	0	θ_5
6	1.5708	0	0	θ_6

The simulation model of the Puma560 robot is established by using the Robot Toolbox in MATLAB, as shown in Figure 2.

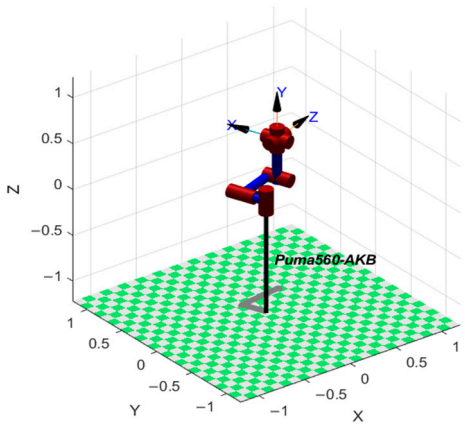


Figure 2. Robot model in MATLAB.

2.1.1. Forward Kinematics Analysis

Forward kinematics analysis refers to obtaining the end-effector pose relative to the base according to the angle of each robot joint. The transformation matrix between neighboring links, that is, the coordinate system $\{i\}$ relative to the coordinate system $\{i-1\}$, can be represented by

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

Substituting the parameters in Table 1 into (1), the six homogeneous transformation matrixes of the Puma560 robot can be obtained by

$$
\begin{gathered}
{}^{0}_{1}T = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -\sin\theta_2 & -\cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}_{3}T = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^{3}_{4}T = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & a_3 \\ 0 & 0 & -1 & -d_4 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{4}_{5}T = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_5 & -\cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}_{6}T = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_6 & \cos\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{gathered} \tag{2}
$$

The transformation matrixes in (2) can be multiplied together to find the transformation matrix of the end-effector coordinate system {6} relative to the base coordinate system {0}

$$
{}^{0}_{6}T = {}^{0}_{1}T{}^{1}_{2}T{}^{2}_{3}T{}^{3}_{4}T{}^{4}_{5}T{}^{5}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}
$$

where p_x, p_y, p_z represent the position of the end-effector, and r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33} represent the orientation of the end-effector. These 12 elements are calculated by the joint angles.

2.1.2. Inverse Kinematics Analysis

Inverse kinematics analysis refers to the inverse solution of the angle of each joint by using the end-effector pose relative to the base. Analyzing the structural characteristics of the Puma560 robot, it is easy to know that its last three axes intersect at one point, and the six joints of the robot are all rotary joints. So, the Pieper method [33] can solve the inverse kinematics of the Puma560 robot. When $\sin(\theta_5) \neq 0$, the joint angles can be obtained by

$$
\theta_1 = \mathrm{A}\tan 2\left(\frac{g_1 y - g_2 x}{g_1^2 + g_2^2}, \frac{g_1 x + g_2 y}{g_1^2 + g_2^2}\right) \tag{4}
$$

$$
\theta_2 = \mathrm{A}\tan 2\left(\frac{-z}{\rho_2}, \pm\sqrt{1 - \frac{z^2}{\rho_2^2}}\right) - \mathrm{A}\tan 2(f_2, f_1) \tag{5}
$$

$$
\theta_3 = \mathrm{A}\tan 2\left(\frac{C - r}{\rho_3}, \pm\sqrt{1 - \frac{(C - r)^2}{\rho_3^2}}\right) - \mathrm{A}\tan 2(a_3, d_4) \tag{6}
$$

$$
\theta_4 = \mathrm{A}\tan 2\left(\frac{x_{33}}{\sin(\theta_5)}, \frac{x_{13}}{\sin(\theta_5)}\right) \tag{7}
$$

$$
\theta_5 = \mathrm{A}\tan 2\left(\pm\sqrt{x_{21}^2 + x_{22}^2}, -x_{23}\right) \tag{8}
$$

$$
\theta_6 = \mathrm{A}\tan 2\left(\frac{-x_{22}}{\sin(\theta_5)}, \frac{x_{21}}{\sin(\theta_5)}\right) \tag{9}
$$

where A tan 2 is predefined in many programming language libraries; its function is to judge the quadrant of the angle according to the positive and negative of x and y while calculating $\tan^{-1}\left(\frac{y}{x}\right)$. Unknown in (4)–(9), such as g_1, g_2, are the intermediate quantities in the derivation process of the Pieper method, which are defined as

$$
f_1 = a_2 + a_3 c_3 + d_4 s_3,\ f_2 = a_3 s_3 - d_4 c_3,\ f_3 = d_3 \tag{10}
$$

$$
g_1 = f_1 c_2 - f_2 s_2,\ g_2 = f_3 + d_2,\ g_3 = -f_1 s_2 - f_2 c_2 \tag{11}
$$

$$r = f_1^2 + f_2^2 + f_3^2 + d_2^2 + 2f_3d_2,\ C = r - \left(a_2^2 + a_3^2 + d_4^2 + d_2^2 + d_3^2 + 2d_2d_3\right) \quad (12)$$

$$x = (f_1c_2 - f_2s_2)c_1 - (f_3 + d_2)s_1,\ y = (f_1c_2 - f_2s_2)s_1 + (f_3 + d_2)c_1,\ z = -f_1s_2 - f_2c_2 \quad (13)$$

$$\rho_2 = \sqrt{f_1^2 + f_2^2},\ \rho_3 = 2a_2\sqrt{a_3^2 + d_4^2} \quad (14)$$

$$\phi_2 = \mathrm{A}\tan 2(f_2, f_1),\ \phi_3 = \mathrm{A}\tan 2(a_3, d_4) \quad (15)$$

$$x_{13} = r_{13}c_{23}c_1 - r_{33}s_{23} + r_{23}c_{23}s_1,\ x_{21} = -r_{31}c_{23} - r_{11}c_1s_{23} - r_{21}s_1s_{23} \quad (16)$$

$$x_{22} = -r_{32}c_{23} - r_{12}c_1s_{23} - r_{22}s_1s_{23},\ x_{23} = -r_{33}c_{23} - r_{13}c_1s_{23} - r_{23}s_1s_{23},\ x_{33} = r_{23}c_1 - r_{13}s_1 \quad (17)$$

where $c_i = \cos\theta_i$, $s_i = \sin\theta_i$, $s_{ij} = \sin(\theta_i + \theta_j)$, $c_{ij} = \cos(\theta_i + \theta_j)$, $i, j = [1, 2, 3, 4, 5, 6]$, and $i \neq j$.

If $\sin(\theta_5) \neq 0$ and $\theta_5 = 0$, then the solutions of $\theta_1, \theta_2, \theta_3$ do not change, and θ_4, θ_5, θ_6 become

$$\theta_4 = 0 \quad (18)$$

$$\theta_5 = 0 \quad (19)$$

$$\theta_6 = \mathrm{A}\tan 2(-x_{12}, x_{11}) \quad (20)$$

where $x_{11} = r_{11}c_{23}c_1 - r_{31}s_{23} + r_{21}c_{23}s_1$, $x_{12} = r_{12}c_{23}c_1 - r_{32}s_{23} + r_{22}c_{23}s_1$.

If $\sin(\theta_5) \neq 0$ and $\theta_5 = \pi$, then the solutions of $\theta_1, \theta_2, \theta_3$ do not change, and θ_4, θ_5, θ_6 become

$$\theta_4 = 0 \quad (21)$$

$$\theta_5 = \pi \quad (22)$$

$$\theta_6 = \mathrm{A}\tan 2(x_{12}, -x_{11}) \quad (23)$$

2.2. Dynamics Analysis

The iterative Newton–Euler dynamics algorithm [34] is computationally efficient, suitable for real-time control, and commonly used for modeling robot dynamics. The algorithm is composed of two parts. First, link velocities and accelerations are iteratively calculated from the base. Second, starting from the end-effector, the force and torque of each link are calculated in reverse. The specific iterative calculation process is as follows.

Outward iterations: $i : 0 \to n_l - 1$

$$^{i+1}w_{i+1} = {}^{i+1}_{i}R\,{}^{i}w_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \quad (24)$$

$$^{i+1}\dot{w}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{w}_i + {}^{i+1}_{i}R\,{}^{i}w_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \quad (25)$$

$$^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\left[{}^{i}\dot{w}_i \times {}^{i}P_{i+1} + {}^{i}w_i \times \left({}^{i}w_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right] \quad (26)$$

$$^{i}\dot{v}_{C_i} = {}^{i}\dot{w}_i \times {}^{i}P_{C_i} + {}^{i}w_i \times \left({}^{i}w_i \times {}^{i}P_{C_i}\right) + {}^{i}\dot{v}_i \quad (27)$$

$$F_i = m\dot{v}_{C_i} \quad (28)$$

$$N_i = {}^{C_i}I\dot{w}_i + w_i \times {}^{C_i}Iw_i \quad (29)$$

where n_l is the number of links, $^{i+1}\hat{Z}_{i+1}$ is the unit vector of the coordinate system $\{i+1\}$ on the Z axis, F_i and N_i are, respectively, the inertia force and torque acting on the mass center of the link i.

Inward iterations: $i : n_l \to 1$

$$^{i}f_i = {}^{i}_{i+1}R^{i+1}f_{i+1} + {}^{i}F_i \quad (30)$$

$$^{i}n_i = {}^{i}N_i + {}^{i}_{i+1}R^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}^{i}_{i+1}R^{i+1}f_{i+1} \quad (31)$$

$$\tau_i = {}^{i}n_i^{T\,i}\hat{Z}_i \quad (32)$$

where ${}^{i}f_i$, ${}^{i}n_i$ are, respectively, the force and torque acting on the link i, and τ_i is the driving force of the joint motor.

3. Multi-Objective Trajectory Planning

3.1. Construction of Joint Space Trajectory

A kth-degree B-spline curve [35] is defined by

$$p(u) = \sum_{i=0}^{n} d_i N_{i,k}(u) \tag{33}$$

where d_i is the control point, n+1 is its number, $p(u)$ is the path point at node u, $N_{i,k}(u)$ is the kth-degree B-spline basis function, and its specific definition is

$$\begin{cases} N_{i,0}(u) = \begin{cases} 1, & u_i \le u < u_{i+1} \\ 0, & \text{others} \end{cases} \\ N_{i,k}(u) = \frac{u-u_i}{u_{i+k}-u_i} N_{i,k-1}(u) + \frac{u_{i+k+1}-u}{u_{i+k+1}-u_{i+1}} N_{i+1,k-1}(u) \\ \frac{0}{0} = 0. \end{cases} \tag{34}$$

The interval of $N_{i,k}(u)$, $u \in [u_i, u_{i+k+1}]$, contains $k+1$ node intervals. It can be seen from (34) that for any node $u \in [u_i, u_{i+k+1}]$ on the parameter axis, there are only up to $k+1$ nonzero basis functions $N_{r,k}(u)(r = i-k, i-k+1, \ldots, i)$. This is the local support property of the B-spline curve. Therefore, the B-spline curve can also be expressed as

$$p(u) = \sum_{r=i-k}^{i} d_r N_{r,k}(u) \tag{35}$$

In this paper, the joint space trajectory of the robot is obtained by fifth-order B-spline curve interpolation, so $k = 5$. Assuming that the position-time series of a certain joint is $P = (p_j, t_j), j = 0, 1, \ldots, m$, then the node vector is $U = [u_0, u_1, \ldots, u_{m+2k}]$, and $n = m + k - 1$.

In order to make the B-spline curve pass through the first and end position points of the joint, the node repetition degree of these two positions needs to be defined as

$$u_0 = u_1 = \ldots = u_5 = 0 \tag{36}$$

$$u_{m+5} = u_{n+6} = \ldots = u_{m+10} = 1 \tag{37}$$

Moreover, the accumulative chord length parameterization method normalizes the remaining $m-1$ inner nodes

$$u_i = u_{i-1} + \frac{|\Delta t_{i-6}|}{\sum_{r=0}^{m-1} |\Delta t_r|}, i = 6, 7, \ldots, m+4 \tag{38}$$

The $n+1$ equations are needed to solve $n+1$ control points, where $m+1$ equations can be given by

$$p(u_{i+5}) = \sum_{r=i}^{i+5} d_r N_{r,5}(u_{i+5}) = p_i, \quad u_{i+5} \in [u_5, u_{m+5}], i = 0, 1, \ldots, m \tag{39}$$

Additional conditions determine the other $k-1$ equations. For the fifth-order B-spline curve, specifying the velocity and acceleration of the joint at the start and end points can add four additional equations

$$\begin{cases} p'(u)\big|_{u=u_5} = v_s, \quad p'(u)\big|_{u=u_{m+5}} = v_e \\ p''(u)\big|_{u=u_5} = a_s, \quad p''(u)\big|_{u=u_{m+5}} = a_e \end{cases} \tag{40}$$

where $p'(u)$, $p''(u)$ are, respectively, the first and second derivatives of the B-spline curve, representing the velocity and acceleration of the joint. The deBoor–Cox recurrence formula can calculate the lth derivative of the B-spline curve

$$\begin{cases} p^l(u) = \sum\limits_{r=i-k+l}^{i} d_r^l N_{r,k-l}(u), \ u_i < u < u_{i+1} \\ d_r^l = \begin{cases} d_j, \ l = 0 \\ (k+1-l)\left(d_r^{l-1} - d_{r-1}^{l-1}\right)/(u_{r+k+1-l} - u_r), \ l = 1, 2, \ldots, r. \end{cases} \end{cases} \tag{41}$$

Expression (39) is combined with (40) to obtain

$$A_n d = p \tag{42}$$

where $d = [d_0, d_1, \ldots, d_{n-1}, d_n]^T$, $p = [p_0, p_1, \ldots, p_m, v_s, v_e, a_s, a_e]^T$, and the coefficient matrix is

$$A_n = \begin{bmatrix} 1 & & & & & & & & & \\ & N_{1,5}(u_6) & N_{2,5}(u_6) & \cdots & N_{5,5}(u_6) & & & & & \\ & & N_{2,5}(u_7) & N_{3,5}(u_7) & \cdots & N_{6,5}(u_7) & & & & \\ & & \ddots & & & \ddots & & & & \\ & & & N_{m-2,5}(u_{m+3}) & N_{m-1,5}(u_{m+3}) & \cdots & N_{m+2,5}(u_{m+3}) & & & \\ & & & & N_{m-1,5}(u_{m+4}) & N_{m,5}(u_{m+4}) & \cdots & N_{m+3,5}(u_{m+4}) & & \\ c_{s1} & c_{s2} & & & & & & & & 1 \\ & & & & & & & & c_{e1} & c_{e2} \\ a_{s1} & a_{s2} & & & & & & & & \\ & & & & & & a_{e1} & & a_{e2} & a_{e3} \end{bmatrix}$$

Some parameters in the coefficient matrix are defined by

$$\begin{gathered} c_{s1} = -5/(u_6 - u_1) \\ c_{s2} = 5/(u_6 - u_1) \\ c_{e1} = -5/(u_{m+9} - u_{m+4}) \\ c_{e2} = 5/(u_{m+9} - u_{m+4}) \\ a_{s1} = 20/[(u_6 - u_2)(u_6 - u_1)] \\ a_{s2} = -20\{1/[(u_6 - u_2)(u_6 - u_1)] + 1/[(u_6 - u_2)(u_7 - u_2)]\} \\ a_{s3} = 20/[(u_6 - u_2)(u_7 - u_2)] \\ a_{e1} = 20/[(u_{m+8} - u_{m+4})(u_{m+8} - u_{m+3})] \\ a_{e2} = -20\{1/[(u_{m+8} - u_{m+4})(u_{m+8} - u_{m+3})] + 1/[(u_{m+8} - u_{m+4})(u_{m+9} - u_{m+4})]\} \\ a_{e3} = 20/[(u_{m+8} - u_{m+4})(u_{m+9} - u_{m+4})]. \end{gathered} \tag{43}$$

From (42), all the control points can be obtained by

$$d = A_n^{-1} p \tag{44}$$

Bringing the control points back to (35), the angular displacement curve of each robot joint can be obtained. Then, the angular velocity, angular acceleration, and angular jerk curves of each joint are obtained by (41).

3.2. Establishing the Multi-Objective Optimization Model

This model consists of objective functions and constraint conditions. Firstly, the specific expressions of travel time, energy consumption, and smoothness are shown in (45)–(47). It is assumed that the trajectory of each joint of the Puma560 is divided into m segments, which means that each joint is assigned $m + 1$ angle values in turn.

The total travel time of the robot is the sum of the travel time of each trajectory, so its expression is

$$S_1 = T = \sum_{j=1}^{m} \Delta t_j = \sum_{j=1}^{m} (t_j - t_{j-1}) \tag{45}$$

where t_{j-1}, t_j, Δt_j are, respectively, the starting time, ending time, and travel time of the jth trajectory, and T is the total travel time.

The average accelerations of joints represent the energy consumption

$$S_2 = \sum_{i=1}^{6} \sqrt{\frac{1}{T} \int_0^T \ddot{\theta}_i^2 dt} \tag{46}$$

where $\ddot{\theta}_i$ is the acceleration of the ith joint.

The average jerks of joints measure the smoothness

$$S_3 = \sum_{i=1}^{6} \sqrt{\frac{1}{T} \int_0^T \dddot{\theta}_i^2 dt} \tag{47}$$

where $\dddot{\theta}_i$ is the jerk of the ith joint.

Constraints of kinematics and dynamics of the robot need to be considered when the robot performs tasks. Kinematic constraints mainly include the limitation of joint angle, velocity, and acceleration. Dynamic constraints mostly refer to the restriction of the joint torque. Therefore, the multi-objective optimization model can be established as

$$\begin{aligned}
&\text{Minimize} \\
&\quad S = [S_1, S_2, S_3]^T \\
&\text{Subject to} \\
&g_{i,1} = \theta_i^{\min} - \min(\theta_i(t)) \le 0 \\
&g_{i,2} = \max(\theta_i(t)) - \theta_i^{\max} \le 0 \\
&g_{i,3} = \dot{\theta}_i^{\min} - \min\left(\dot{\theta}_i(t)\right) \le 0 \\
&g_{i,4} = \max\left(\dot{\theta}_i(t)\right) - \dot{\theta}_i^{\max} \le 0 \\
&g_{i,5} = \ddot{\theta}_i^{\min} - \min\left(\ddot{\theta}_i(t)\right) \le 0 \\
&g_{i,6} = \max\left(\ddot{\theta}_i(t)\right) - \ddot{\theta}_i^{\max} \le 0 \\
&g_{i,7} = \tau_i^{\min} - \min(\tau_i(t)) \le 0 \\
&g_{i,8} = \max(\tau_i(t)) - \tau_i^{\max} \le 0.
\end{aligned} \tag{48}$$

where $\theta_i(t)$, $\dot{\theta}_i(t)$, $\ddot{\theta}_i(t)$, $\tau_i(t)$ are, respectively, the angle, velocity, acceleration, and driving torque of the ith joint at t time, and $\theta_i^{\max}$, $\theta_i^{\min}$, $\dot{\theta}_i^{\max}$, $\dot{\theta}_i^{\min}$, $\ddot{\theta}_i^{\max}$, $\ddot{\theta}_i^{\min}$, $\tau_i^{\max}$, $\tau_i^{\min}$ are, respectively, the upper and lower limits of the angle, velocity, acceleration, and driving torque of the ith joint.

3.3. Solving the Multi-Objective Optimization Model

The three optimization objectives conflict with each other, and there is a complex balance between them, so they cannot achieve the best solution simultaneously. When the improved MOPSO algorithm is used to solve the previous model, the result is no longer a single optimal solution, but an optimal solution set called the Pareto optimal solution set [36]. There is no good or bad solution in the Pareto optimal solution set, and the appropriate solution can be selected according to the actual engineering needs.

The MOPSO algorithm [37] originated from the study of bird foraging behavior and has the advantages of fast convergence speed, strong global search capability, and a wide

range of application. In this algorithm, each particle's position represents a solution to the problem. Under the influence of the individual optimal position and group optimal position, particles update their velocity and position

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1 \left(p_{id,pbest}^{k} - x_{id}^{k} \right) + c_2 r_2 \left(p_{d,gbest}^{k} - x_{id}^{k} \right) \tag{49}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{50}$$

where w is the inertia weight, c_1 and c_2 are the individual and group learning factors, r_1 and r_2 are random numbers in the range of [0,1], k is the current iteration number, d is the vector's dimension number, x_{id}^{k} and v_{id}^{k} are the position and velocity of particle i, and $p_{id,pbest}^{k}$ and $p_{d,gbest}^{k}$ are the optimal position of individual and group.

Furthermore, the algorithm determines the dominance relationship between particles by comparing the objective function values of particles. To better manage and preserve the non-dominated solution, it is saved to the external archive.

The traditional MOPSO algorithm determines the global leader and deletes the redundant particles in the external archive by random selection. The convergence, distribution uniformity, and accuracy of the Pareto front are not good. When facing complex problems, the algorithm easily falls into the local optimum.

In order to solve these problems, this paper uses the adaptive grid technology and roulette strategy to change the selection of the global leader and redundant particles in the external archive, applies the adaptive mutation technique to the position of particles, and makes the inertia weight, individual, and group learning factors change nonlinearly with the iterations number. The workflow flow chart of the improved MOPSO algorithm is show in Figure 3.

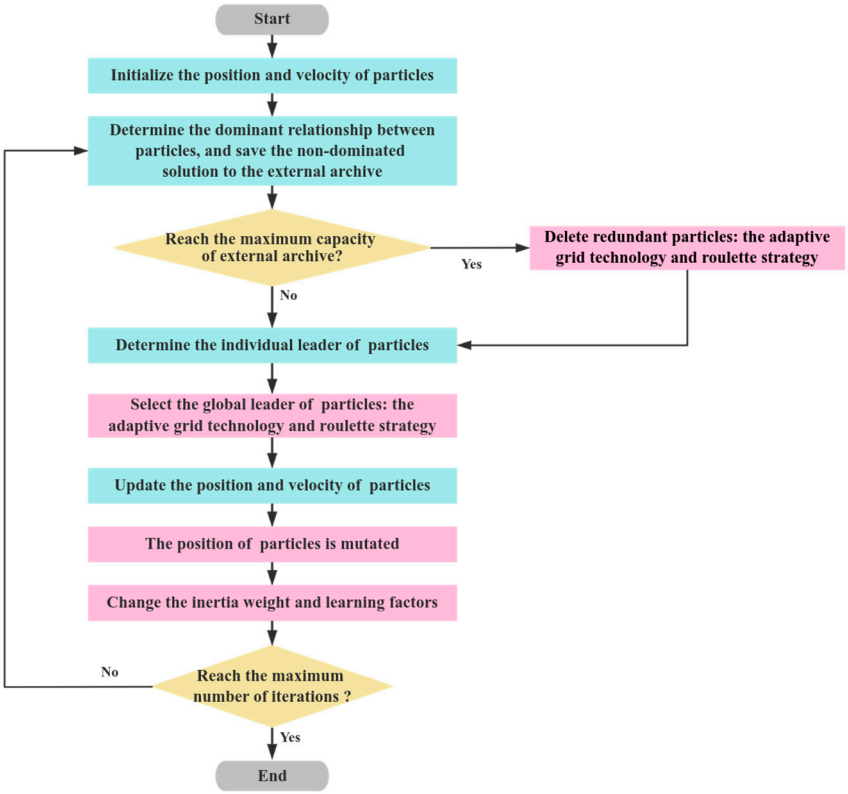


Figure 3. The improved MOPSO algorithm.

The combination of the adaptive grid technology and roulette strategy enables the algorithm to select particles in a suitable sub-grid. For each iteration, the maximum value $f_{i\max}$ and minimum value $f_{i\min}$ of the ith objective function on all particles are found as the

upper and lower bounds of the initial grid. In order to cover the boundary particles, the range of the grid is expanded according to

$$LB_i = f_{i\min} - \alpha(f_{i\max} - f_{i\min}), UB_i = f_{i\max} + \alpha(f_{i\max} - f_{i\min}) \tag{51}$$

where LB_i, UB_i are the new upper and lower bounds of the grid, and α is the expansion ratio.

The number of grids for each dimension is set to n_d, and the grid is equally divided into n_d^e sub-grids, where e is the number of objective function. Figure 4 shows the general process of adaptive grid technology.

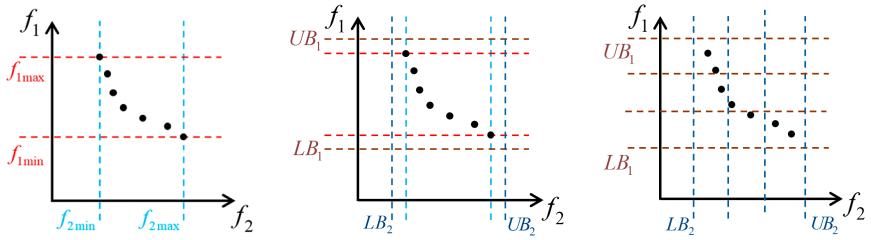


Figure 4. The adaptive grid technology in two-dimensional case. (**a**) The minimum and maximum values of each objective function; (**b**) the enlarged grid range; (**c**) the uniform distribution of grids when $n_d = 3$.

Suppose that there are n_s sub-grids containing particles, and the number of particles in the ith sub-grid is N_i. When selecting the global leader of particles, the probability that the ith sub-grid is selected is

$$P_{i,1} = \frac{e^{-\beta N_i}}{e^{-\beta N_1} + e^{-\beta N_2} + \cdots + e^{-\beta N_b}} \tag{52}$$

where $i = 1, 2, \ldots, n_s$, and β is a non-negative leader selection pressure parameter. It can be seen from (52) that the global leader of particles is more likely to come from sub-grids with fewer particles, which encourages the algorithm to explore areas that have been searched less before and increase the diversity of solutions.

When the number of particles in the external archive is greater than the set number, it is necessary to delete the redundant particles. At this time, the probability that the ith sub-grid is selected is

$$P_{i,2} = \frac{e^{\sigma N_i}}{e^{\sigma N_1} + e^{\sigma N_2} + \cdots + e^{\sigma N_b}} \tag{53}$$

where σ is a non-negative delete selection pressure parameter. It can be seen from (63) that the deleted redundant particles are more likely to come from the sub-grids with more particles, promoting the uniform distribution of the solutions.

Expressions (52)–(53) are the individual selection probabilities in the roulette strategy, and the cumulative probabilities of each grid are defined as

$$Q_{i,1} = \sum_{j=1}^{i} P_{j,1}, \; Q_{i,2} = \sum_{j=1}^{i} P_{j,2}. \tag{54}$$

The individual selection strategy is to generate a random number $r \in [0, 1]$, compare it with $Q_{i,1}$ or $Q_{i,2}$, find the first cumulative probability exceeding r, and select its sub-grid.

When the particle's velocity and position are updated, the particle's position is adaptively mutated. The mutation rate in the kth iteration is

$$pm_k = \left(1 - \frac{k-1}{M-1}\right)^{\frac{1}{h}} \tag{55}$$

where k is the current iteration number, M is the maximum iteration number, and h is a given constant.

The mutation step of the particle i is defined as

$$\Delta x_i^k = pm_k \cdot (x_{\max} - x_{\min}) \tag{56}$$

where $x_{\max}, x_{\min}$ are the limit values of particle position. When pm_k decreases nonlinearly with the increase in the iterations number, Δx_i^k also decreases.

Now, it is randomly specified that the dth dimensional component x_{id}^k of the position vector is mutated. The upper and lower bounds of the range of mutation are

$$lb = x_{id}^k - \Delta x_i^k, ub = x_{id}^k + \Delta x_i^k \tag{57}$$

A random number is generated in the continuous distribution of lb and ub as the value of the position vector after mutation in this dimension. This makes the algorithm jump out of the local optimum in the early stage and converge to the global optimal solution better in the later stage.

The relationship between the inertia weight and the number of iterations is

$$w = w_{\max} - (w_{\max} - w_{\min}) \left(\frac{k}{M} \right)^2 \tag{58}$$

where $w_{\max}, w_{\min}$ are the upper and lower limits of w. As the number of iterations k increases, the inertia weight decreases nonlinearly.

The individual and group learning factors are defined as

$$c_1 = c_{1s} + (c_{1e} - c_{1s}) \sin\left(\frac{\pi k}{2M} \right), c_2 = c_{2s} + (c_{2e} - c_{2s}) \sin\left(\frac{\pi k}{2M} \right) \tag{59}$$

where c_{1s} and c_{2s} are, respectively, the initial values of c_1 and c_2, and c_{1e} and c_{2e} are, respectively, the final values of c_1 and c_2. As the number of iterations increases, c_1 decreases from large to small, and c_2 is the opposite.

These give the particle a strong global search capability at the beginning of the algorithm iterations to avoid falling into local optimum and a strong local search capability at the later stages to improve convergence accuracy.

4. Simulation

In this section, the multi-objective trajectory planning simulation of the Puma560 robot is carried out in MATLAB. The constraints of each joint are shown in Table 2.

Table 2. Kinematic and dynamic constraints.

Constraints	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Angle/(rad)	3.100	3.100	3.100	3.100	3.100	3.100
Velocity/$(\text{rad} \cdot \text{s}^{-1})$	0.876	0.876	1.598	0.876	0.926	0.926
Acceleration/$(\text{rad} \cdot \text{s}^{-2})$	0.725	0.725	2.378	0.725	1.450	1.450
Torque/$(\text{N} \cdot \text{m})$	44.940	44.940	8.866	44.940	0.050	0.050

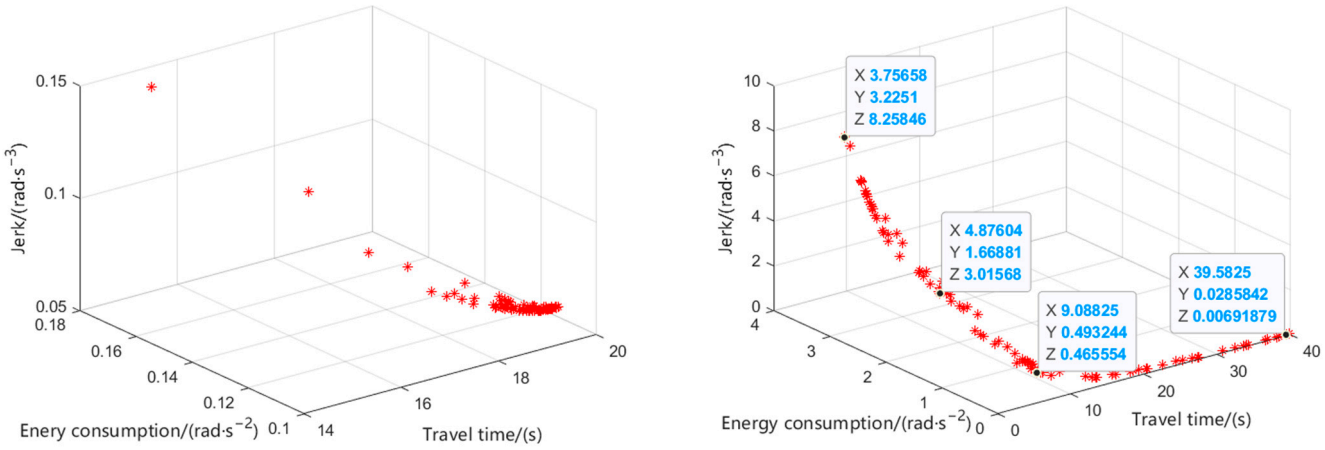
The target captured by the robot is a small ball moving at a constant speed of 0.5 m/s along the Z-axis, and its trajectory is known. Six key positions of each joint in the process of catching the ball are given, as shown in Table 3.

Table 3. Position sequence of each joint.

Node	Joint 1/(rad)	Joint 2/(rad)	Joint 3/(rad)	Joint 4/(rad)	Joint 5/(rad)	Joint 6/(rad)
1	0.5821	−0.3805	−0.8168	0.6283	−0.9390	0.2531
2	0.4829	−0.3735	−0.7981	0.6299	−0.9245	0.2621
3	0.0383	−0.1212	0.0608	0.4289	−0.4005	0.6502
4	−0.5872	0.1770	1.0702	0.1995	0.2189	1.1091
5	−1.0317	0.3890	1.7877	0.0364	0.6592	1.4352
6	−1.1310	0.4363	1.9478	0	0.7547	1.5080

The MOPSO algorithm takes the time interval of each trajectory Δt_j as the decision variable, which is in the range of [0.75, 7]. The population size and the maximum iteration number of the traditional and improved MOPSO algorithms are both 200, and the size of the Pareto optimal solution set is 100. In addition, this paper sets n_d in the improved MOPSO algorithm to 5, β to 2, and σ to 2.

The Pareto front obtained by the traditional MOPSO algorithm is shown in Figure 5a. It falls into the local optimum, and the distribution and convergence of the Pareto front are also poor. The Pareto front obtained by the improved MOPSO algorithm is shown in Figure 5b. It jumps out of the local optimum, and the convergence and distribution are significantly improved, which proves the effectiveness of the improved MOPSO algorithm.



(**a**) (**b**)

Figure 5. The Pareto front. (**a**) The traditional MOPSO algorithm and (**b**) the improved MOPSO algorithm.

Four points are taken on the Pareto front, which, from top to bottom, are A, B, C, and D. The closer to A, the shorter the travel time, the more the energy consumption, and the greater the jerk; the closer to D, the less the energy consumption, the smaller the jerk, and the longer the travel time. It follows that smoothness is positively correlated with energy consumption, while they are negatively correlated with travel time. The values of the three objective functions for A, B, C, and D are shown in Table 4.

Table 4. The partial optimum solution.

Solution	Travel Time/(s)	Energy Consummation/(rad·s^{-2})	Jerk/(rad·s^{-3})
A	3.7566	3.2251	8.2585
B	4.8760	1.6688	3.0157
C	9.0883	0.4932	0.4656
D	39.5825	0.0286	0.0069

Taking B and C points as examples, the travel time of point B is 4.8760 s, 46.35% less than that of point C. The energy consumption of point C is 0.4932 rad · s^{-2}, 70.45% lower than that of point B. The jerk of point C is 0.4932 rad · s^{-3}, 70.45% lower than that of point B.

C is selected as the actual solution of the project, and its time series is [0, 1.1990, 3.6445, 5.3612, 7.2749, 9.0883]. As shown in Figure 6, the curves of joint angles, velocities, accelerations, and jerks varying with time can be obtained by interpolating fifth-order B-spline curves.

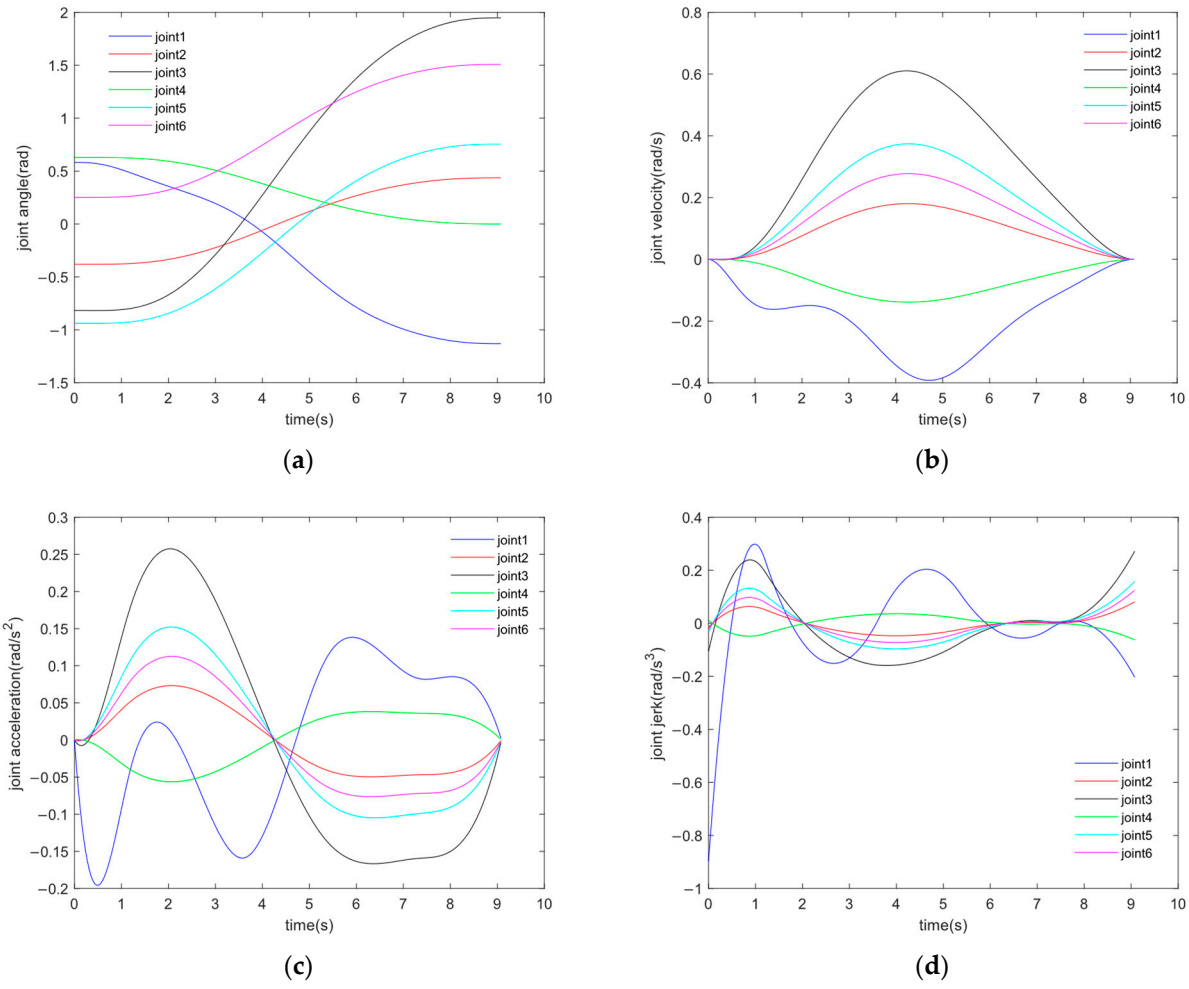


Figure 6. (**a**–**d**) are the angle, velocity, acceleration, and jerk curves of the joints under solution C.

A dominant solution E is randomly selected outside the Pareto optimal solution set as the time series [0, 1.3, 2.4, 5.3, 8.4, 10.4] before the trajectory optimization. Under this time series, the three performance indexes of the robot are shown in Table 5.

Table 5. Comparison before and after optimization.

Solution	Travel Time/(s)	Energy Consumption/(rad·s^{-2})	Jerk/(rad·s^{-3})
C	9.0883	0.4932	0.4656
E	10.4000	1.1457	1.8733

According to the data in Table 5, the travel time of point C is 12.61% higher than that of E, the energy consumption is increased by 56.95%, and the jerk is increased by 75.15%. The three objective function values of point C are better than the results before optimization, improving the robot's comprehensive performance.

Taking robot joint 2 as an example, Figure 7 shows that the trajectories after optimization are smoother and more continuous than before optimization, especially the curves of joint velocity, acceleration, and jerk.

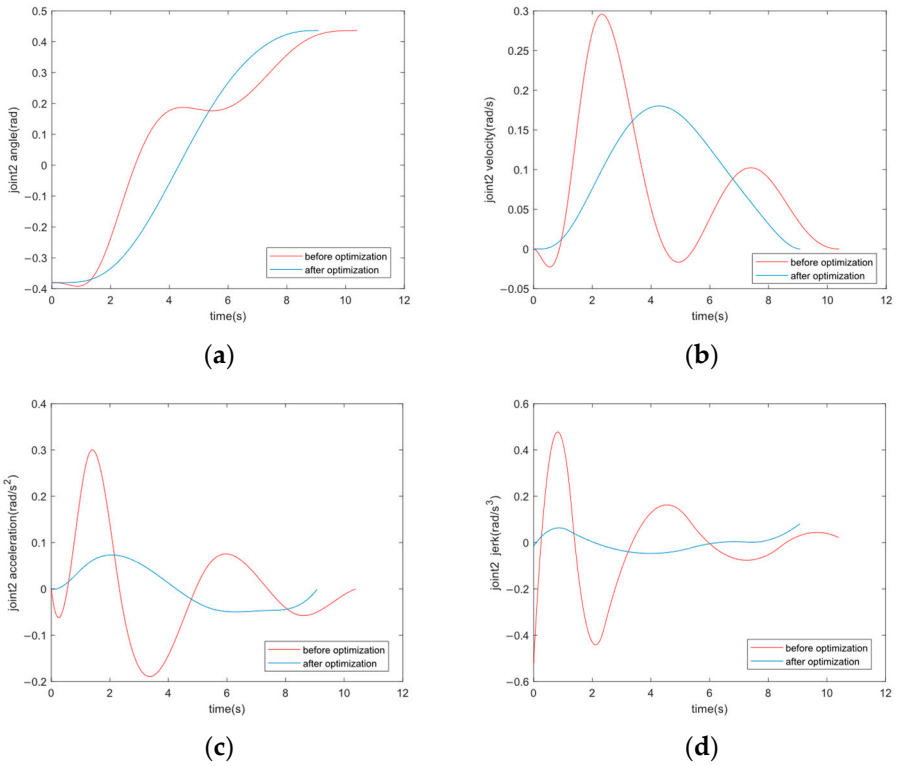


Figure 7. (**a**–**d**) show the curves of angle, velocity, acceleration, and jerk of robot joint 2 before and after optimization.

An optimal solution is randomly selected in the Pareto optimal solution set, called D, and its time series is [0, 1.7255, 4.5821, 6.1810, 7.7798, 10.0000]. The motion of the Puma560 robot under this solution can be visualized by the Robot Toolbox of MATLAB, as shown in Figure 8.

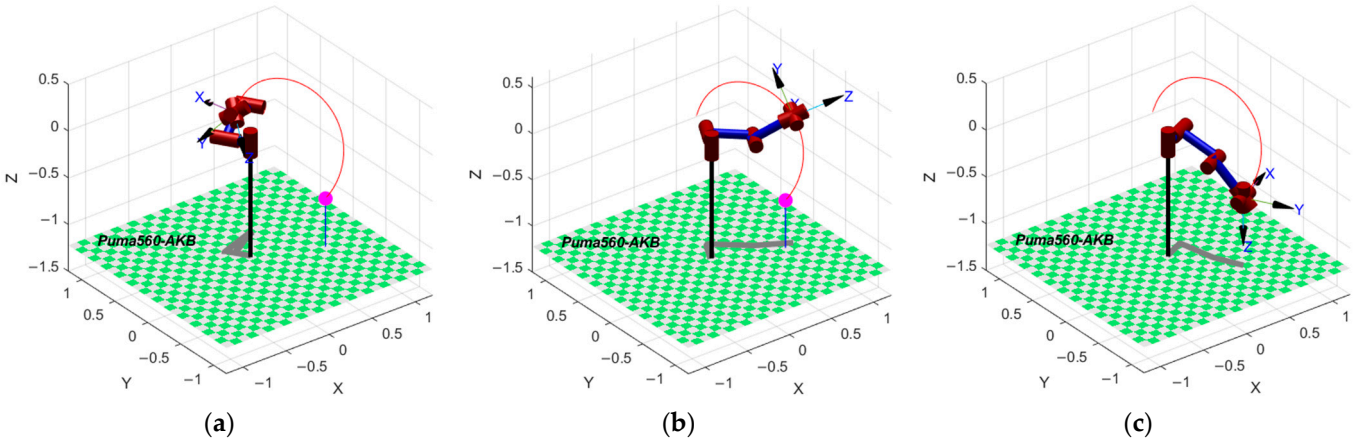


Figure 8. (**a**) The pose of the robot at the start moment; (**b**) the pose of the robot at the middle moment; and (**c**) the pose of the robot at the end moment.

5. Conclusions

This paper investigates a trajectory planning method for a robot which enables it to reach a comprehensive optimal state of travel time, energy consumption, and smooth-

ness when executing a task. In order to fully understand the kinematics and dynamics characteristics of the robot and lay a solid theoretical foundation for follow-up research, this paper first deduces the position and orientation of the end-effector relative to the base and uses the Pieper method to calculate the closed solutions of the inverse kinematics. Finally, the dynamic model of the robot is established by the iterative Newton–Euler dynamics algorithm.

The joint space trajectory of the Puma560 robot is constructed using fifth-order B-spline curves, which has the advantages of continuous jerk and zero velocity and acceleration at the start/stop time. Then, the improved MOPSO algorithm is used to optimize the trajectory of the robot with the time interval between the path points as the decision variable. The convergence and distribution of the Pareto front are good, and the different solutions in the Pareto optimal solution set correspond to different engineering needs. In addition, by comparing the robot's travel time, energy consumption, and smoothness before and after optimization, it can be seen that its three performances have improved. This paper also visualizes the robot movement according to the planned trajectory in the Robot Toolbox of MATLAB.

Author Contributions: Conceptualization, J.W. (Jiahui Wang) and Y.Z.; methodology, J.W. (Jiahui Wang); software, J.W. (Jiahui Wang); validation, J.W. (Jiahui Wang), Y.Z. and S.Z.; formal analysis, J.W. (Jiahui Wang); investigation, J.W. (Jiahui Wang); resources, J.W. (Jiahui Wang) and S.Z.; data curation, J.W. (Jiahui Wang); writing—original draft preparation, J.W. (Jiahui Wang); writing—review and editing, J.W. (Jiahui Wang), Y.Z. and J.W. (Junling Wang); visualization, J.W. (Jiahui Wang); supervision, Y.Z.; project administration, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Science, Technology and Innovation Commission of Shenzhen Municipality, grant number No. JCYJ20200109141201714 ("Research on Flight Test Methods for Aerodynamics and Flight Control of Shipborne UAV").

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lan, J.; Xie, Y.; Liu, G.; Cao, M. A Multi-Objective Trajectory Planning Method for Collaborative Robot. *Electronics* **2020**, *9*, 859. [CrossRef]
2. Bailon, W.P.; Cardiel, E.B.; Campos, I.J.; Paz, A.R. Mechanical energy optimization in trajectory planning for six DOF robot manipulators based on eighth-degree polynomial functions and a genetic algorithm. In Proceedings of the 7th International Conference on Electrical Engineering Computing Science and Automatic Control, Tuxtla Gutierrez, Mexico, 8–10 September 2010; pp. 446–451.
3. Liu, J.; Wang, H.; Li, X.; Chen, K.; Li, C. Robotic arm trajectory optimization based on multiverse algorithm. *Math. Biosci. Eng.* **2023**, *20*, 2776–2792. [CrossRef] [PubMed]
4. Machmudah, A.; Parman, S.; Zainuddin, A.; Chacko, S. Polynomial joint angle arm robot motion planning in complex geometrical obstacles. *Appl. Soft Comput.* **2013**, *13*, 1099–1109. [CrossRef]
5. Porawagama, C.D.; Munasinghe, S.R. Reduced jerk joint space trajectory planning method using 5-3-5 spline for robot manipulators. In Proceedings of the 7th International Conference on Information and Automation for Sustainability, Colombo, Sri Lanka, 22–24 December 2014; pp. 1–6.
6. Kim, K.W.; Kim, H.S.; Choi, Y.K.; Park, J.H. Optimization of cubic polynomial joint trajectories and sliding mode controllers for robots using evolution strategy. In Proceedings of the IECON'97 23rd International Conference on Industrial Electronics, Control, and Instrumentation, New Orleans, LA, USA, 14 November 1997; pp. 1444–1447.
7. Lu, S.; Ding, B.; Li, Y. Minimum-jerk trajectory planning pertaining to a translational 3-degree-of-freedom parallel manipulator through piecewise quintic polynomials interpolation. *Adv. Mech. Eng.* **2020**, *12*, 1687814020913667. [CrossRef]
8. Boryga, M.; Grabo, A. Planning of manipulator motion trajectory with higher-degree polynomials use. *Mech. Mach. Theory* **2009**, *44*, 1400–1419. [CrossRef]
9. Chen, D.; Li, S.; Wang, J.; Feng, Y.; Liu, Y. A multi-objective trajectory planning method based on the improved immune clonal selection algorithm. *Robot. Comput. Integr. Manuf.* **2019**, *59*, 431–442. [CrossRef]

10. Gasparetto, A.; Zanotto, V. Optimal trajectory planning for industrial robots. *Adv. Eng. Softw.* **2010**, *41*, 548–556. [CrossRef]
11. Wang, Z.; Li, Y.; Sun, P.; Luo, Y.; Chen, B.; Zhu, W. A multi-objective approach for the trajectory planning of a 7-DOF serial-parallel hybrid humanoid arm. *Mech. Mach. Theory* **2021**, *165*, 104423. [CrossRef]
12. Gao, Y.; Xie, W.; Li, Q.; Li, X.; Hu, M.; Zhao, L. Time-Jerk Optimal Trajectory Planning of Industrial Robot based on Hybrid Particle Swarm Optimization Algorithm. In Proceedings of the 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021; pp. 6327–6331.
13. Hansen, C.; Öltjen, J.; Meike, D.; Ortmaier, T. Enhanced approach for energy-efficient trajectory generation of industrial robots. In Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE), Seoul, Republic of Korea, 20–24 August 2012; pp. 1–7.
14. Shi, B.; Zeng, H. Time-Optimal Trajectory Planning for Industrial Robot based on Improved Hybrid-PSO. In Proceedings of the 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021; pp. 3888–3893.
15. Gasparetto, A.; Zanotto, V. A new method for smooth trajectory planning of robot manipulators. *Mech. Mach. Theory* **2007**, *42*, 455–471. [CrossRef]
16. Yao, J.; Sun, C.; Zhang, L.; Xiao, C.; Yang, M.; Zhang, S. Time optimal trajectory planning based on simulated annealing algorithm for a train uncoupling robot. In Proceedings of the 29th Chinese Control and Decision Conference (CCDC), Chongqing, China, 28–30 May 2017; pp. 5781–5785.
17. Bianco, C.G.L.; Piazzi, A. A genetic/interval approach to optimal trajectory planning of industrial robots under torque constraints. In Proceedings of the 1999 European Control Conference (ECC), Karlsruhe, Germany, 31 August–3 September 1999; pp. 942–947.
18. Mora, P.R. On the Time-optimal Trajectory Planning along Predetermined Geometric Paths and Optimal Control Synthesis for Trajectory Tracking of Robot Manipulators. Ph.D. Thesis, University of California, Berkeley, CA, USA, 2013.
19. Abu-Dakka, F.J.; Assad, I.F.; Alkhdour, R.M. Statistical evaluation of an evolutionary algorithm for minimum time trajectory planning problem for industrial robots. *Int. J. Adv. Manuf. Technol.* **2017**, *89*, 389–406. [CrossRef]
20. Zhang, W.; Fu, S. Time-optimal Trajectory Planning of Dulcimer Music Robot Based on PSO Algorithm. In Proceedings of the 2020 Chinese Control and Decision Conference (CCDC), Hefei, China, 22–24 August 2020; pp. 4769–4774.
21. Lin, H.I. A fast and unified method to find a minimum-jerk robot joint trajectory using particle swarm optimization. *J. Intell. Robot. Syst. Theory Appl.* **2014**, *75*, 379–392. [CrossRef]
22. Zhou, Y.; Han, G.; Wei, Z.; Huang, Z.; Chen, X.; Wu, J. Optimal trajectory planning of robot energy consumption based on improved sparrow search algorithm. *Meas. Control* **2024**, *57*, 1014–1021. [CrossRef]
23. Yokose, Y. Energy-saving trajectory planning for robots using the genetic algorithm with assistant chromosomes. *Artif. Life Robot.* **2020**, *25*, 89–93. [CrossRef]
24. Luo, L.-P.; Yuan, C.; Yan, R.-J.; Yuan, Q.; Wu, J.; Shin, K.-S.; Han, C.-S. Trajectory planning for energy minimization of industry robotic manipulators using the Lagrange interpolation method. *Int. J. Precis. Eng. Manuf.* **2015**, *16*, 911–917. [CrossRef]
25. Mohammed, A.; Schmidt, B.; Wang, L.; Gao, L. Minimizing Energy Consumption for Robot Arm Movement. *Procedia CIRP* **2014**, *25*, 400–405. [CrossRef]
26. Paes, K.; Dewulf, W.; Elst, K.V. Energy efficient trajectories for an industrial ABB robot. *Procedia CIRP* **2014**, *15*, 105–110. [CrossRef]
27. Ye, J.; Hao, L.; Cheng, H. Multi-objective optimal trajectory planning for robot manipulator attention to end-effector path limitation. *Robotica* **2024**, *42*, 1761–1780. [CrossRef]
28. Chen, W.; Wang, H.; Liu, Z.; Jiang, K. Time-energy-jerk optimal trajectory planning for high-speed parallel manipulator based on quantum-behaved particle swarm optimization algorithm and quintic B-spline. *Eng. Appl. Artif. Intell.* **2023**, *126*, 107223. [CrossRef]
29. Cao, X.; Yan, H.; Huang, Z.; Ai, S.; Xu, Y.; Fu, R.; Zou, X. A Multi-Objective Particle Swarm Optimization for Trajectory Planning of Fruit Picking Manipulator. *Agronomy* **2021**, *11*, 2286. [CrossRef]
30. Saravanan, R.; Ramabalan, S. Evolutionary Minimum Cost Trajectory Planning for Industrial Robots. *J. Intell. Robot. Syst.* **2008**, *52*, 45–77. [CrossRef]
31. Shi, X.; Fang, H.; Guo, L. Multi-objective optimal trajectory planning of manipulators based on quintic NURBS. In Proceedings of the 2016 IEEE International Conference on Mechatronics and Automation, Harbin, China, 7–10 August 2016.
32. Craig, J.J. *Introduction to Robotics: Mechanics and Control*, 3rd ed.; Pearson Education, Inc.: London, UK, 2005; pp. 73–76.
33. Pieper, D.L. The Kinematics of Manipulators Under Computer Control. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 1968.
34. Luh, J.Y.S.; Walker, M.W.; Paul, R.P.C. On-Line Computational Scheme for Mechanical Manipulators. *ASME J. Dyn. Sys. Meas. Control* **1980**, *102*, 69–76. [CrossRef]
35. Piegl, L.; Tiller, W. *The Nurbs Book*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 1997; pp. 81–100.
36. Sathiya, V.; Chinnadurai, M. Evolutionary Algorithms-Based Multi-Objective Optimal Mobile Robot Trajectory Planning. *Robotica* **2019**, *37*, 1363–1382. [CrossRef]
37. Coello, C.A.C.; Lechuga, M.S. MOPSO: A proposal for multiple objective particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002.