S-Velocity Profile of Industrial Robot Based on NURBS Curve and Slerp Interpolation

Authors:

Guirong Wang, Fei Xu, Kun Zhou, Zhihui Pang

Date Submitted: 2023-02-20

Keywords: NURBS curve, quaternion, Slerp interpolation, S-velocity planning

Abstract:

This paper presents a novel algorithm for industrial robot trajectory planning based on the NURBS(Non-Uniform Rational B-Spline) curve and Slerp interpolation aiming at the problems that the trajectory of a six-axis industrial robot is not smooth enough in the operation process, the posture planning process is non-uniform, and the six-axis industrial robot starts and stops frequently. Firstly, aiming at the first problem, the trajectory planning algorithm based on the NURBS curve is presented to improve the smoothness of the trajectory curve. Combined with Slerp posture planning based on quaternion description, which realizes the uniform change of posture on the robot's end-effector. Secondly, aiming at the second problem, the S-velocity planning algorithm is presented in the interpolation interval of the robot, which realizes the operation process of complex curves continuously, and improves the operation quality. Finally, this paper uses Bernoulli's lemniscate as the incentive trajectory, and the contrast experiment of trajectory planning between two incentive profiles is designed, which are the NURBS curve and the five-order polynomial curve. The result of the experiment indicates that the planning algorithm proposed in this paper could effectively improve the smoothness of trajectory in a Cartesian workspace, decrease the impact and tremulous in a Cartesian workspace, and effectively improve the performance of the robot working process. The results drawn from this paper lay a certain foundation for the future high-precision control of industrial robots.

Record Type: Published Article

Submitted To: LAPSE (Living Archive for Process Systems Engineering)

Citation (overall record, always the latest version):	LAPSE:2023.0713
Citation (this specific file, latest version):	LAPSE:2023.0713-1
Citation (this specific file, this version):	LAPSE:2023.0713-1v1

DOI of Published Version: https://doi.org/10.3390/pr10112195

processes

MDPI

Article

S-Velocity Profile of Industrial Robot Based on NURBS Curve and Slerp Interpolation

Guirong Wang [1,*], Fei Xu [2], Kun Zhou [1] and Zhihui Pang [1]

[1] School of Mechanical and Electrical Engineering, China Jiliang University, Hangzhou 310018, China
[2] Henan Power Transmission and Transformation Construction Co., Ltd., Zhengzhou 450003, China
* Correspondence: lilygrwang@cjlu.edu.cn; Tel.: +86-137-5814-9518

Abstract: This paper presents a novel algorithm for industrial robot trajectory planning based on the NURBS(Non-Uniform Rational B-Spline) curve and *Slerp* interpolation aiming at the problems that the trajectory of a six-axis industrial robot is not smooth enough in the operation process, the posture planning process is non-uniform, and the six-axis industrial robot starts and stops frequently. Firstly, aiming at the first problem, the trajectory planning algorithm based on the NURBS curve is presented to improve the smoothness of the trajectory curve. Combined with *Slerp* posture planning based on quaternion description, which realizes the uniform change of posture on the robot's end-effector. Secondly, aiming at the second problem, the S-velocity planning algorithm is presented in the interpolation interval of the robot, which realizes the operation process of complex curves continuously, and improves the operation quality. Finally, this paper uses Bernoulli's lemniscate as the incentive trajectory, and the contrast experiment of trajectory planning between two incentive profiles is designed, which are the NURBS curve and the five-order polynomial curve. The result of the experiment indicates that the planning algorithm proposed in this paper could effectively improve the smoothness of trajectory in a Cartesian workspace, decrease the impact and tremulous in a Cartesian workspace, and effectively improve the performance of the robot working process. The results drawn from this paper lay a certain foundation for the future high-precision control of industrial robots.

Keywords: NURBS curve; quaternion; *Slerp* interpolation; S-velocity planning

Citation: Wang, G.; Xu, F.; Zhou, K.; Pang, Z. S-Velocity Profile of Industrial Robot Based on NURBS Curve and Slerp Interpolation. *Processes* **2022**, *10*, 2195. https://doi.org/10.3390/pr10112195

Received: 7 September 2022
Accepted: 21 October 2022
Published: 26 October 2022

1. Introduction

In the working process of industrial robots, the smoothness of the end-effector's trajectory is one of the important indexes to evaluate the robot working quality [1,2]. In order to make the robot move along the given trajectory, generally speaking, it usually goes through three stages: trajectory planning, velocity planning and real-time interpolation [3].

The end-effector's trajectory of industrial robots refers to the coordinate position and posture of the robot's end-effector, which is the function of the working process time respectively; Therefore, the trajectory planning of the robot's end-effector includes the position and posture planning [4]. The position planning of the robot's end-effector could be taken in two steps. Firstly, determine the trajectory curve, i.e., the geometric equation, which is a three-dimensional curve in Cartesian space. Then determine the function between the components of the x, y and z axes and time t in the Cartesian coordinate frame [5]. When it comes to the end-effector posture planning, the posture planning algorithm should be selected corresponding to the way of posture description [6].

For the trajectory planning of the robot's end-effector, determine all the critical points first, then construct the trajectory curve through real-time interpolation [7]. Generally speaking, the basic interpolation algorithm of the robot includes linear interpolation and arc interpolation [8], and other interpolation algorithms are based on the two basic interpolation algorithms above. Among the interpolation algorithm above, the way of connecting

adjacent critical points with tiny straight line segments [9] is simple, namely the form of a "straight line and transition curve" iteratively. However, at the intersection point of two adjacent line segments, there exist abrupt changes in direction, which leads to many unsmooth points in the whole working process, thus affecting the quality of the whole process. Based on the interpolation algorithms above, the form of parabolic transition adopted near the critical points [10] could ensure the smoothness of the trajectory at every trajectory point. Furthermore, polynomial curves with good smoothness could be applied near critical points [11], such as arc transition [12] and high-order polynomial transition [13], which could also ensure the smoothness of the trajectory at every trajectory point. However, such a local smoothing algorithm does not have a unified form for the description of the entire working trajectory, the algorithm solution is complicated, and the computation is large.

Different from the above local smoothing algorithm, the global smoothing algorithm adopts a smooth curve to construct the trajectory according to all the trajectory points. The commonly seen global smoothing curves include the Bézier curve [14], PH curve [15] and NURBS curve [16,17]. The common feature of the above global smoothing curves is that they are all structural curves, which could be constructed according to the boundary conditions, i.e., position, velocity and acceleration at trajectory points. Among the above global smoothing curves, the NURBS curve proposed in the literature [16] could be used as control vertices based on a series of critical points, and different weight factors corresponding to different points could construct a unified curve model. Meanwhile, due to the differentiability of the NURBS curve, it meets the requirement of end-effector trajectory smoothness in the robot working process [18].

The above interpolation algorithms are mainly aimed at the situation where the end-effector posture remains unchanged during the whole working process. If the end-effector posture changes, the corresponding planning algorithm should be selected corresponding to the way of the end-effector posture description [19]. The Robot's end-effector posture could be described with *RPY* angle [20], universal rotation transformation algorithm [21], quaternion [22,23] and other description algorithms. Among the end-effector posture description algorithm above, the *RPY* angle describes the rotation in a simple way, but there exists coupling between *RPY* angles, which may lead to the problem of universal joint locking, and the rotation around x, y and z axes may lose a degree of freedom. As for the universal rotation transformation algorithm, when the rotation angle around any vector in the Cartesian space is close to 0° or 180°, it is difficult to determine the rotation axis, and the transition discontinuity of the posture will occur when the adjacent posture is rotated around the fixed axis while using quaternion to describe the posture rotation could effectively avoid the above problems.

The above trajectory planning algorithms in the robot Cartesian space solve the problem of which trajectory the end-effector moves along with, while the velocity planning algorithm solves the problem of how to move along the planned trajectory in the working process of the robot. The velocity planning algorithm could be categorized as non-flexible and flexible velocity planning. The trapezoidal velocity planning [24], as a typical non-flexible velocity planning algorithm, could be applied to a variety of working conditions due to its simple model and easy implementation. However, the velocity change is not smooth enough, and there exists a mutation of the acceleration during the motor start and stop processes, which is likely to lead to the impact of the joint motor, and it is hard to meet the requirements of high accuracy in working conditions, while the flexible velocity planning could avoid the impact of joint motor and improve the flexibility of the working process. The commonly used flexible velocity planning algorithms include exponential velocity profile [25], trigonometric velocity profile [26], polynomial acceleration and deceleration velocity profile [27], S-Velocity planning [28,29], etc. Among the above flexible velocity planning algorithm, S-Velocity planning ensures the continuity of acceleration profile, which effectively reduces the impact of joint motor, and the algorithm complexity is moderate, could be taken into account the real-time requirements of the system, and is widely used. In the meanwhile, the velocity profile of exponential velocity planning is

smooth and has high-order derivatives, but the algorithm complexity is high, and there are still some mutations at the two endpoints. Trigonometric velocity planning uses a single sinusoidal curve to construct the velocity profile. During the acceleration and deceleration process, there only exists one point which could reach the maximum value of acceleration or jerk, and the abrupt change of jerk will occur at the end of the acceleration and deceleration sections. Polynomial velocity planning further improves the smoothness of the S-velocity profile, but the algorithm is complicated and difficult to meet real-time requirements. Theoretically, exponential velocity planning, trigonometric velocity planning and high-order polynomial velocity planning have higher derivatives and better compliance, but the complexity of such algorithms cannot guarantee the real-time performance of the algorithm, and the application in engineering is affected to a certain extent.

This paper presents the trajectory planning algorithm based on the NURBS curve with well-smoothness to improve the smoothness of the trajectory curve. Combined with *Slerp* posture planning based on quaternion description, which realizes the uniform change of posture on the robot's end-effector. Then the S-Velocity planning algorithm is presented in the interpolation interval of the robot, which realizes the operation process of complex curves continuously, and improves the operation quality. Finally, this paper uses Bernoulli's lemniscate as the incentive trajectory, and the contrast experiment of trajectory planning between two incentive profiles is designed, which are the NURBS curve and the five-order polynomial curve. Through the analysis and comparison between the two incentive profile, the velocity profile with the planning algorithm proposed in this paper becomes more smooth, and the acceleration won't change dramatically, which indicates that the planning algorithm proposed in this paper could effectively improve the smoothness of trajectory in a Cartesian workspace, decrease the impact and tremulous in a Cartesian workspace, and effectively improve the performance of robot working process.

The main contributions of this paper could be summarized as follows:

(1) Aiming at the problem that the trajectory is not smooth enough in the traditional working process, a NURBS curve planning algorithm based on a unified curve model is proposed in this paper, and the global smoothing of the trajectory curve is realized; At the same time, the *Slerp* posture planning based on quaternion description is combined to achieve uniform posture change in the planning process;

(2) Aiming at the problem of frequent start and stop in the traditional working process, this paper proposes an S-Velocity planning algorithm in the interpolation interval of the robot based on the trajectory planning algorithm in (1), which realizes the continuous working process of complex curves and improves the quality of robot working process;

(3) Bernoulli's lemniscate is used as the incentive trajectory, and the contrast experiment of trajectory planning between two incentive profiles is designed, which are the NURBS curve and the five-order polynomial curve. Through the analysis and comparison between the two incentive profile, the velocity profile with the planning algorithm proposed in this paper becomes more smooth, and the acceleration won't change dramatically, which indicates that the planning algorithm proposed in this paper could effectively improve the smoothness of trajectory in a Cartesian workspace, decrease the impact and tremulous in a Cartesian workspace, and effectively improve the performance of robot working process.

This paper is organized as follows. In Section 2, the NURBS curve planning algorithm based on the unified curve model and *Slerp* posture planning based on the quaternion description is proposed. In Section 3, based on the above trajectory planning algorithm, this paper proposes an S-Velocity planning algorithm in the interpolation interval of the robot. In Section 4, based on the trajectory planning and velocity planning algorithms, the implementation process of the algorithm is proposed in this paper, and the implementation process of robot operation is designed. In Section 5, a comparison experiment based on the proposed programming algorithm and the five-order polynomial planning algorithm is designed to demonstrate the

effectiveness of the proposed programming algorithm in improving the working quality. In Section 6, this paper arrives at some discussion and conclusions.

2. Construction of Robot's End-Effector Trajectory

2.1. NURBS Curve

Generally speaking, a k-order NURBS curve could be expressed in Formula (1):

$$p(u) = \frac{\sum_{i=0}^{n} \omega_i d_i N_{i,k}(u)}{\sum_{i=0}^{n} \omega_i N_{i,k}(u)} \tag{1}$$

where d_i stands for $n + 1$ control points, i.e., all the critical points, $i = 0, 1, \ldots, n$; ω_i is the weight factor corresponding to the control point, $\omega_0 > 0$, $\omega_n > 0$, the rest $\omega_i \geq 0$ $\omega_0 > 0$; When the ω_i values are not equal, it is the inhomogeneity of the NURBS curve. $U = [u_0, u_1, \ldots, u_{n+k+1}]$ is the node vector, and all the u_i values do not decrease; $0 \leq u \leq 1 \in$ is the normalization factor, $u_0 = u_1 = \ldots = u_k = 0$, $u_{n+1} = u_{n+2} = \ldots = u_{n+k+1} = 1$; the step size between the rest of the u_i values is $1/(n + 1 - k)$, i.e., $u_{k+1} = 1/(n + 1 - k)$, $u_{k+2} = 2/(n + 1 - k)$, $\ldots$ $u_n = (n - k)/(n + 1 - k)$; $N_{i,k}(u)$ is the k-order B-spline basis function, which is defined as in the following Formula (2):

$$\begin{cases} N_{i,0}(u) = \begin{cases} 1, \ u_i \leq u \leq u_{i+1} \\ 0, \ else \end{cases} \\ N_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} N_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \\ define \frac{0}{0} = 0 \end{cases} \tag{2}$$

Formula (2) is the Cox-de-Boor recursive formula described in the literature [17].

Remark 1. *For the trajectory of the robot's end-effector, d_i could be the RPY point to describe the position and posture of the end-effector, but in order to simplify the calculation, d_i is taken as the three-dimensional coordinate point of the end position. Each control vertex d_i corresponds to a k-order B-spline basis function.*

Let the numerator and denominator in Formula (1), respectively, be:

$$A(u) = \sum_{i=0}^{n} \omega_i d_i N_{i,k}(u) \tag{3}$$

$$w(u) = \sum_{i=0}^{n} \omega_i N_{i,k}(u) \tag{4}$$

$$p(u) = \frac{A(u)}{w(u)} \tag{5}$$

Apply the Leibniz formula to calculate the nth-derivative of $A(u)$:

$$\begin{aligned} A^{(n)}(u) &= [w(u)p(u)]^{(n)} = \sum_{i=0}^{n} C_n^i w^{(i)}(u) p^{(n-i)}(u) \\ &= w(u)p^{(n)}(u) + \sum_{i=1}^{n} C_n^i w^{(i)}(u) p^{(n-i)}(u) \end{aligned} \tag{6}$$

It could be obtained from Formula (6) that:

$$p^{(n)}(u) = \frac{A^{(n)}(u) - \sum_{i=1}^{n} C_n^i w^{(i)}(u) p^{(n-i)}(u)}{w(u)} \tag{7}$$

It could be obtained from Formulas (3) and (4) that:

$$A^{(n)}(u) = \sum_{i=0}^{n} \omega_i d_i N_{i,k}^{(n)}(u) \tag{8}$$

$$w^{(n)}(u) = \sum_{i=0}^{n} \omega_i N_{i,k}^{(n)}(u) \tag{9}$$

It could be obtained from the literature [17] that:

$$N_{i,k}^{(n)}(u) = k\left[\frac{N_{i,k-1}^{(n-1)}(u)}{u_{i+k} - u_i} - \frac{N_{i+1,k-1}^{(n-1)}(u)}{u_{i+k+1} - u_{i+1}}\right] \tag{10}$$

The first and second derivatives of the NURBS curve could be obtained from Formulas (6)–(10) as follows:

$$p'(u) = \frac{A'(u) - w'(u)p(u)}{w(u)} \tag{11}$$

$$p''(u) = \frac{A''(u) - 2w'(u)p'(u) - w''(u)p(u)}{w(u)} \tag{12}$$

In addition, the NURBS curve in Formula (1) could be expressed as:

$$p(u) = x(u)\vec{i} + y(u)\vec{j} + z(u)\vec{k},\ 0 \le u \le 1 \tag{13}$$

Therefore, for the first and second derivative formulas of the NURBS curve described in Formulas (11) and (12), corresponding to the NURBS curve form in Formula (13), the velocity and acceleration of three component directions at every point could be obtained.

When the NURBS curve is used for interpolation, it is necessary to know the arc length at the current moment to carry out velocity planning. NURBS curve length L from point $p(u_1)$ to point $p(u_2)$ is:

$$\begin{aligned} L &= \int_{u_1}^{u_2} \left\| \frac{dp(u)}{du} \right\| du = \int_{u_1}^{u_2} f(u)du \\ &= \int_{u_1}^{u_2} \sqrt{\left(\frac{dx(u)}{du}\right)^2 + \left(\frac{dy(u)}{du}\right)^2 + \left(\frac{dz(u)}{du}\right)^2} du \end{aligned} \tag{14}$$

Generally speaking, it is difficult to get the primitive function of the integrand function in Formula (14), so the numerical integration is used to calculate the length of the NURBS curve. Among them, Simpson adaptive integration algorithm could adjust the step size of numerical integration according to the change rate of the integrated function in the integral interval, and the principle is as follows:

$$L = \int_{u_1}^{u_2} f(u)du = S(u_1, u_2) - \frac{u_2 - u_1}{180}\left(\frac{h}{2}\right)^4 f^{(4)}(\eta),\ \eta \in (u_1, u_2) \tag{15}$$

$$\begin{cases} h = u_2 - u_1 \\ S(u_1, u_2) = \dfrac{h}{6}\left[f(u_1) + 4f(\dfrac{u_1 + u_2}{2}) + f(u_2)\right] \end{cases} \tag{16}$$

$$R[f] = -\frac{u_2 - u_1}{180}\left(\frac{h}{2}\right)^4 f^{(4)}(\eta),\ \eta \in (u_1, u_2) \tag{17}$$

Formula (17) is the remainder of the Simpson formula.

2.2. Robot's End-Effector Posture Planning Based on Quaternion

As shown in Section 2, the robot's end-effector posture could be described by *RPY* angle, universal rotation transformation algorithm, quaternion and other algorithms. *RPY* angle posture description algorithm is first rotated by angle γ around the x-axis, then by angle β around the y-axis, and then by angle α around the z-axis to obtain the transformation matrix described in Formula (18):

$$RPY(\alpha,\beta,\gamma) = Rot(z,\alpha)Rot(y,\beta)Rot(x,\gamma) = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & 0 \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & 0 \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

When $\beta = \pi/2$, there exists that:

$$RPY\left(\alpha,\frac{\pi}{2},\gamma\right) = \begin{bmatrix} 0 & -\sin(\alpha-\gamma) & \cos(\alpha-\gamma) & 0 \\ 0 & \sin(\alpha+\gamma) & \sin(\alpha-\gamma) & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

At this point, the rotation angle γ around the x-axis is equivalent to the rotation angle -γ around the z-axis, which results in the loss of degrees of freedom. This is the universal lock described in the literature [21].

The universal rotation transformation algorithm is to rotate θ angle around any unit vector $f = f_x i + f_y i + f_z k$ in Cartesian space, and obtain the rotation matrix described in Formula (20):

$$Rot(f,\theta) = \begin{bmatrix} f_x f_x(1-\cos\theta)+\cos\theta & f_x f_y(1-\cos\theta)-f_z\sin\theta & f_x f_z(1-\cos\theta)+f_y\sin\theta & 0 \\ f_y f_x(1-\cos\theta)+f_z\sin\theta & f_y f_y(1-\cos\theta)+\cos\theta & f_y f_z(1-\cos\theta)-f_x\sin\theta & 0 \\ f_z f_x(1-\cos\theta)-f_y\sin\theta & f_z f_y(1-\cos\theta)+f_x\sin\theta & f_z f_z(1-\cos\theta)+\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

In the above rotation process, the counterclockwise rotation angle is positive, and the clockwise angle is negative.

Generally speaking, the robot's end-effector posture could be described by the matrix described in Formula (21):

$$T = \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

The unit quaternions could be described in the following 2 ways:

$$q = s + ai + bj + ck \tag{22}$$

$$q = \cos\frac{\theta}{2} + f_x\sin\frac{\theta}{2}i + f_y\sin\frac{\theta}{2}j + f_z\sin\frac{\theta}{2}k \tag{23}$$

The rotation axis in Formula (23) is $f = f_x i + f_y i + f_z k$ of the universal rotation transformation.

Combining Formula (22) with (23), there is:

$$s = \cos\frac{\theta}{2}, a = f_x\sin\frac{\theta}{2}, b = f_y\sin\frac{\theta}{2}, c = f_z\sin\frac{\theta}{2} \tag{24}$$

The posture matrix corresponding to the quaternion in Formula (22) could be expressed as:

$$T = \begin{bmatrix} 1-2(b^2+c^2) & 2ab-2sc & 2sb+2ac & 0 \\ 2ab+2sc & 1-2(a^2+c^2) & -2sa+2bc & 0 \\ -2sb+2ac & 2sa+2bc & 1-2(a^2+b^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{25}$$

Combining Formula (20) with (21) and (24), there is:

$$\begin{aligned} n_x + o_y + a_z &= (1-\cos\theta)(f_x^2+f_y^2+f_z^2) + 3\cos\theta \\ &= 1 + 2\cos\theta \end{aligned} \tag{26}$$

$$\begin{cases} \cos\theta = \dfrac{1}{2}(n_x + o_y + a_z - 1) \\ 1+\cos\theta = 2\cos^2\frac{\theta}{2} \\ \qquad\quad = \frac{1}{2}(n_x + o_y + a_z + 1) \end{cases} \tag{27}$$

$$s = \cos\frac{\theta}{2} = \pm\frac{1}{2}\sqrt{n_x + o_y + a_z + 1} \tag{28}$$

$$\begin{cases} o_z - a_y = 2f_x \sin\theta, f_x = \frac{o_z - a_y}{2\sin\theta} \\ a = f_x \sin\frac{\theta}{2} = \frac{o_z - a_y}{4\cos\frac{\theta}{2}} = \frac{o_z - a_y}{4s} \end{cases} \tag{29}$$

$$\begin{cases} s = \cos\dfrac{\theta}{2} = \pm\dfrac{1}{2}\sqrt{n_x + o_y + a_z + 1} \\ a = f_x \sin\dfrac{\theta}{2} = \dfrac{o_z - a_y}{4s} \\ b = f_y \sin\dfrac{\theta}{2} = \dfrac{a_x - n_z}{4s} \\ c = f_z \sin\dfrac{\theta}{2} = \dfrac{n_y - o_x}{4s} \end{cases} \tag{30}$$

According to Formula (30), the corresponding quaternion could be directly calculated from the robot's end-effector posture matrix. Since the rotation direction is not specified, it corresponds to two conjugate quaternions. In addition, Formulas (20) and (25) are completely equivalent according to the above derivation process.

Remark 2. *The robot's end-effector trajectory planning based on quaternion could obtain smooth rotation by interpolation, which effectively solves the problem of unsmooth rotation in RPY angle linear interpolation.*

For any 2 postures q_0 and q_1 described by quaternions, generally, there are two interpolation algorithms, *Lerp* and *Slerp* [24]. *Lerp* is linear interpolation, i.e.,

$$Lerp(q_0, q_1, h) = q_0(1-h) + q_1 h \tag{31}$$

where h could be understood as the parameter in the NURBS curve, $0 \le h \le 1$.

In Figure 1, the linear interpolation is interpolated along a line from q_0 to q_1. This interpolation algorithm is along the chord from q_0 to q_1 on the arc of the unit circle, so the quaternion interpolated by linear interpolation is not the unit quaternion. The linear interpolation could be further improved, and the quaternion obtained by interpolation could be normalized as *Nlerp*, normalized linear interpolation. The *Nlerp* algorithm is as follows:

$$Nlerp(q_0, q_1, h) = \frac{q_0(1-h) + q_1 h}{\|q_0(1-h) + q_1 h\|} \tag{32}$$

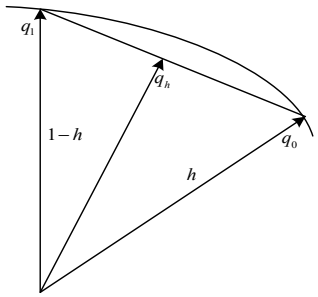


Figure 1. Linear interpolation between 2 postures.

In the above interpolation algorithm, when h is uniformly distributed within the interval [0, 1], the interpolation segments obtained are not uniformly distributed, i.e., linear interpolation only has formally "linear interpolation".

Slerp, i.e., spherical linear interpolation, which is linear interpolation on the sphere, which is a linear interpolation of angles, i.e.,

$$\theta = (1-h)\theta_1 + h\theta_1 \tag{33}$$

In addition, *Slerp* could be expressed as:

$$q_h = Slerp(q_0, q_1, h) = \alpha q_0 + \beta q_1 \tag{34}$$

When Formula (34) is multiplied with q_0 and q_1 (as shown in Figure 2), there are:

$$q_0 q_h = \alpha + \beta q_0 q_1 \tag{35}$$

$$q_1 q_h = \alpha q_1 q_0 + \beta \tag{36}$$

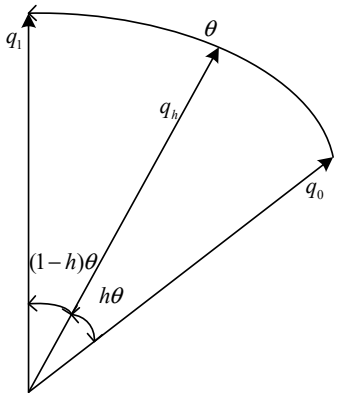


Figure 2. Spherical linear interpolation between 2 postures.

q_0 and q_1 are unit quaternions, then:

$$\cos(h\theta_1) = \alpha + \beta\cos\theta_1 \tag{37}$$

$$\cos((1-h)\theta_1) = \alpha\cos\theta_1 + \beta \tag{38}$$

Combining Formula (37) with (38) gives:

$$\alpha = \frac{\sin((1-h)\theta_1)}{\sin\theta_1}, \beta = \frac{\sin(h\theta_1)}{\sin\theta_1} \tag{39}$$

Combining Formula (34) with (39), there is:

$$Slerp(q_0, q_1, h) = \frac{\sin((1-h)\theta_1)}{\sin\theta_1} q_0 + \frac{\sin(h\theta_1)}{\sin\theta_1} q_1 \tag{40}$$

Remark 3. *Slerp gives the shortest interpolation curve between two points on a sphere described by quaternions, and the rotation angle varies uniformly. In order to improve the efficiency of quaternion posture planning and to avoid redundant posture changes, the Slerp within q_1 or $-q_1$ with a smaller angle between q_0 is selected, i.e.,*

$$Slerp(q_0, q_1, h) = \begin{cases} \frac{\sin((1-h)\theta_1)}{\sin\theta_1} q_0 + \frac{\sin(h\theta_1)}{\sin\theta_1} q_1, if q_0 \cdot q_1 \geq 0 \\ \frac{\sin((1-h)\theta_1)}{\sin\theta_1} q_0 + \frac{\sin(h\theta_1)}{\sin\theta_1} (-q_1), else \end{cases} \tag{41}$$

The first derivative of *Slerp* with respect to h is:

$$\frac{d}{dh} Slerp(q_0, q_1, h) = \begin{cases} -\frac{\cos((1-h)\theta_1)}{\sin\theta_1} \theta_1 q_0 + \frac{\cos(h\theta_1)}{\sin\theta_1} \theta_1 q_1, if q_0 \cdot q_1 \geq 0 \\ -\frac{\cos((1-h)\theta_1)}{\sin\theta_1} \theta_1 q_0 + \frac{\cos(h\theta_1)}{\sin\theta_1} (-\theta_1 q_1), else \end{cases} \tag{42}$$

$$\frac{d^2}{dh^2} Slerp(q_0, q_1, h) = \begin{cases} -\frac{\sin((1-h)\theta_1)}{\sin\theta_1} \theta_1^2 q_0 + \frac{\sin(h\theta_1)}{\sin\theta_1} \theta_1^2 q_1, if q_0 \cdot q_1 \geq 0 \\ -\frac{\sin((1-h)\theta_1)}{\sin\theta_1} \theta_1^2 q_0 - \frac{\sin(h\theta_1)}{\sin\theta_1} (-\theta_1^2 q_1), else \end{cases} \tag{43}$$

$$\frac{d^2}{dh^2} Slerp(q_0, q_1, h) = -\theta_1^2 Slerp(q_0, q_1, h)$$

3. S-Velocity Planning for Industrial Robots

Remark 4. *As described in Section 2, S-velocity planning could ensure the continuity of the acceleration curve and effectively avoid the impact of joint motors in the robot working process. Therefore, the S-velocity profile could be used for velocity planning. The working space of the robot could be categorized as joint space and Cartesian space. In joint space, S-Velocity planning could be carried out for each joint angle respectively. In Cartesian space, S-Velocity planning could be used for six components of the end-effector posture, respectively.*

In the seven-segment S-Velocity planning, there are seven stages of the acceleration and deceleration process, including acceleration, uniform acceleration, deceleration, uniform deceleration, uniform deceleration, acceleration and deceleration segments, as is shown in Figure 3.

In Figure 3, the end time of each segment could be recorded as t_1~t_7, every segment lasts T_1~T_7, and the displacement is s, there exists:

$$T_1 = T_3 = T_5 = T_7, T_2 = T_6 \tag{44}$$

Given the following two displacements:

$$s_1 = \frac{Jv_{\max}^2 + v_{\max} a_{\max}^2}{2Ja_{\max}} \tag{45}$$

$$s_2 = \frac{a_{\max}^3}{J^2} \tag{46}$$

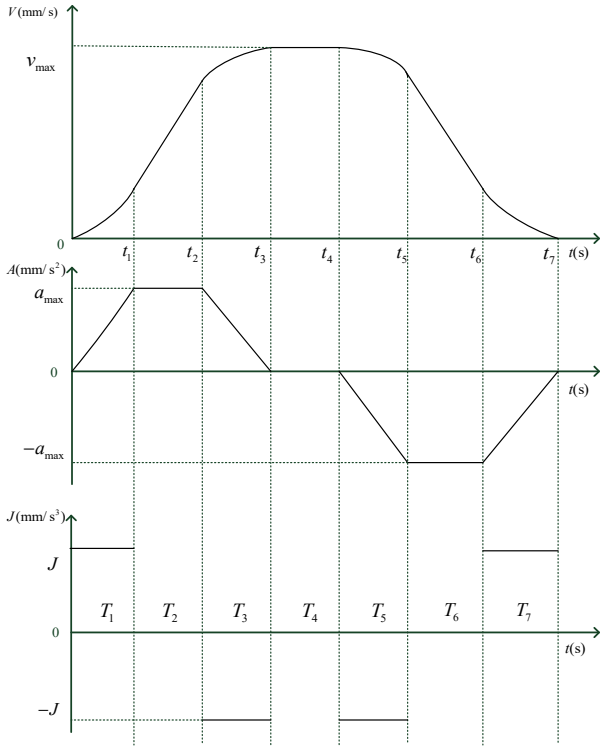


Figure 3. Seven-segment S-Velocity planning.

s_1 and s_2 are, respectively, the displacements achieved by the maximum speed of the acceleration segment, the uniform acceleration segment and the deceleration segment, and the displacements completed by the maximum velocity of the acceleration segment and the deceleration segment.

(1) If:

$$v_{\max} \le \frac{a_{\max}^2}{J} \&\& s > 2v_{\max}\sqrt{\frac{v_{\max}}{J}} \tag{47}$$

Then the maximum velocity, $v_{\max}$, could be achieved within a given distance s, while the maximum acceleration, $a_{\max}$, couldn't be achieved. The velocity planning is in five segments, and there is no uniform acceleration and deceleration segment.

Revised:

$$\begin{cases} a_{\max} = \sqrt{v_{\max} J} \\ T_1 = T_3 = T_5 = T_7 = \dfrac{a_{\max}}{J} \\ T_2 = T_6 = 0 \\ T_4 = \dfrac{s}{v_{\max}} - 2\sqrt{\dfrac{v_{\max}}{J}} \end{cases} \tag{48}$$

(2) If s > 2s1, then the maximum velocity $v_{\max}$ and maximum acceleration $a_{\max}$ could be achieved within a given distance s, and the velocity planning is in seven-segment.

Respectively, the seven segments are:

$$\begin{cases} T_1 = T_3 = T_5 = T_7 = \dfrac{a_{\max}}{J} \\ T_2 = T_6 = \dfrac{v_{\max}}{a_{\max}} - \dfrac{a_{\max}}{J} \\ T_4 = \dfrac{s - 2s_1}{v_{\max}} \end{cases} \tag{49}$$

$$v(t) = \begin{cases} \frac{1}{2}Jt^2, t < t_1 \\ \frac{1}{2}Jt_1^2 + a_{\max}(t - t_1), t_1 < t < \sum_{i=1}^{2} t_i \\ v_{\max} - \frac{1}{2}J(t_1 + t_2 + t_3 - t)^2, \sum_{i=1}^{2} t_i < t < \sum_{i=1}^{3} t_i \\ v_{\max}, \sum_{i=1}^{3} t_i < t < \sum_{i=1}^{4} t_i \\ v_{\max} - \frac{1}{2}J(t - \sum_{i=1}^{4} t_i)^2, \sum_{i=1}^{4} t_i < t < \sum_{i=1}^{5} t_i \\ v_{\max} - \frac{1}{2}Jt_5^2 - a_{\max}(t - \sum_{i=1}^{5} t_i), \sum_{i=1}^{5} t_i < t < \sum_{i=1}^{6} t_i \\ \frac{1}{2}J(\sum_{i=1}^{7} t_i - t)^2, \sum_{i=1}^{6} t_i < t < \sum_{i=1}^{7} t_i \end{cases} \tag{50}$$

$$d(t) = \begin{cases} \frac{Jt^3}{6}, t < t_1 \\ d_1 + \frac{J}{2}t_1^2(t - t_1) + \frac{a_{\max}}{2}(t - t_1)^2, t_1 < t < \sum_{i=1}^{2} t_i \\ d_2 + v_{\max}(t - t_1 - t_2) + \frac{J}{6}(t_1 + t_2 + t_3 - t)^3 - \frac{J}{6}t_3^3, \sum_{i=1}^{2} t_i < t < \sum_{i=1}^{3} t_i \\ d_3 + v_{\max}(t - t_1 - t_2 - t_3), \sum_{i=1}^{3} t_i < t < \sum_{i=1}^{4} t_i \\ d_4 + v_{\max}(t - \sum_{i=1}^{4} t_i) - \frac{1}{6}J(t - \sum_{i=1}^{4} t_i)^3, \sum_{i=1}^{4} t_i < t < \sum_{i=1}^{5} t_i \\ d_5 + (v_{\max} - \frac{1}{2}Jt_5^2)(t - \sum_{i=1}^{5} t_i) - \frac{a_{\max}}{6}(t - \sum_{i=1}^{5} t_i)^2, \sum_{i=1}^{5} t_i < t < \sum_{i=1}^{6} t_i \\ d = s - \frac{1}{6}J(\sum_{i=1}^{7} t_i - t)^3, \sum_{i=1}^{6} t_i < t < \sum_{i=1}^{7} t_i \end{cases} \tag{51}$$

$$\begin{cases} d_1 = \frac{a_{\max}^3}{6J^2} \\ d_2 = d_1 + \frac{J}{2}t_1^2 t_2 + \frac{a_{\max}t_2^2}{2} \\ d_3 = d_2 + \frac{v_{\max}a_{\max}}{J} - \frac{a_{\max}^3}{6J^2} \\ = \frac{v_{\max}^2}{2a_{\max}} + \frac{v_{\max}a_{\max}}{2J} = s_1 \\ d_4 = s - d_3 \\ d_5 = s - d_2 \\ d_6 = s - d_1 \end{cases} \tag{52}$$

If $s \geq 2s_2$ and $s < 2s_1$, then the maximum acceleration, $a_{\max,}$ could be achieved within a given distance s, while the maximum velocity $v_{\max}$ couldn't be achieved.

The velocity planning is in six segments, and there is no constant velocity segment.

Revised:

$$\begin{cases} v_{\max} = \dfrac{\sqrt{a_{\max}^4 + 4J^2 a_{\max} s} - a_{\max}^2}{2J} \\ T_1 = T_3 = T_5 = T_7 = \dfrac{a_{\max}}{J} \\ T_2 = T_6 = \dfrac{v_{\max}}{a_{\max}} - \dfrac{a_{\max}}{J} \\ T_4 = 0 \end{cases} \tag{53}$$

If $s < 2s_1$, then neither the maximum velocity $v_{\max}$ nor the maximum acceleration, $a_{\max}$, could be achieved within a given distance s. The velocity planning is in four segments, and there is no uniform acceleration, uniform velocity and uniform deceleration segment.

Revised:

$$\begin{cases} a_{\max} = \sqrt[3]{\dfrac{sJ^2}{2}}, v_{\max} = JT_1^2 \\ T_1 = T_3 = T_5 = T_7 = \dfrac{a_{\max}}{J} \\ T_2 = T_6 = 0 \\ T_4 = 0 \end{cases} \tag{54}$$

The S-velocity planning algorithm flow chart is shown in Figure 4.

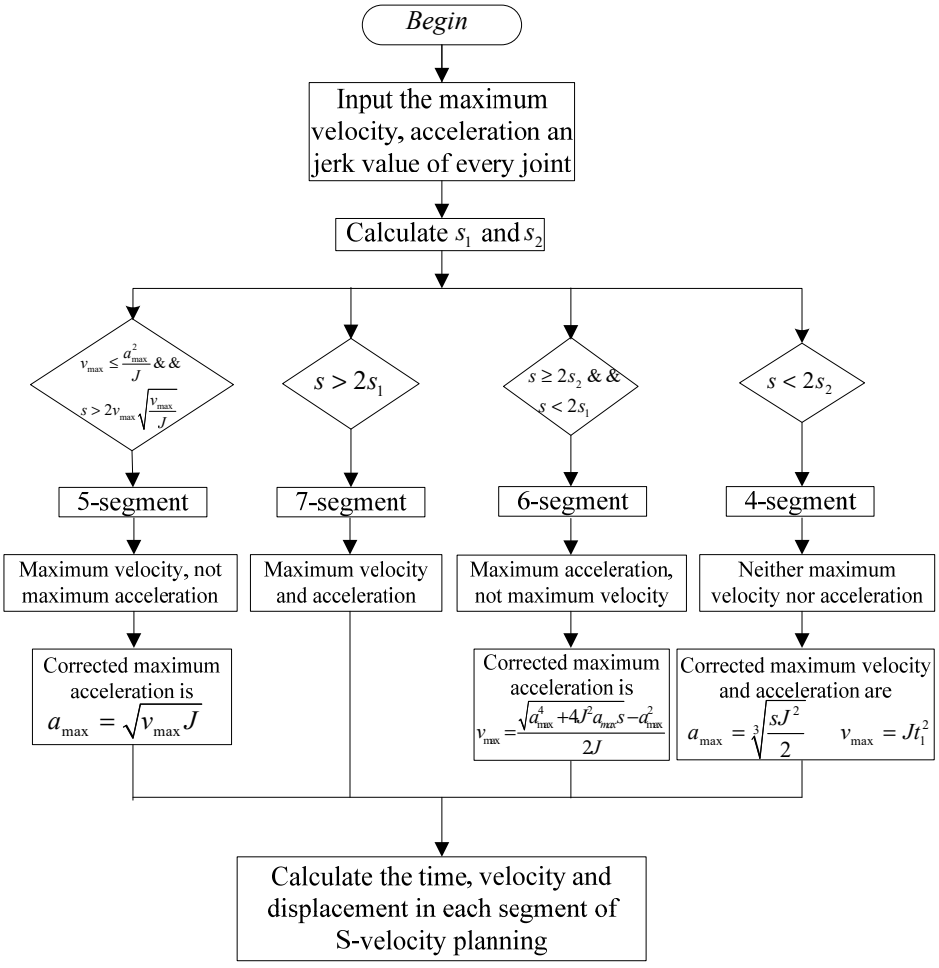


Figure 4. S-velocity planning algorithm flow chart.

4. The Implementation of the Robot Trajectory Planning Algorithm

4.1. The Execution Flow of the Trajectory Planning Algorithm

Remark 5. *The robot trajectory planning algorithm proposed in this paper could be taken as follows.*

Step 1. Construct the end-effector trajectory.

As shown in Section 2, the end-effector trajectory of the robot consists of the coordinate position and posture, which are the function of the operation process time, t, respectively. Therefore, the position and posture could be planned, respectively. According to Section 3, the end-effector position of the robot adopts NURBS curve planning, and the posture adopts quaternion description-based *Slerp* planning.

Step 2. Ensure all the critical points and read point information of critical points.

The robot's end-effector trajectory constructed in Step 1 is divided into n segments according to the robot's interpolation period, and $n + 1$ critical trajectory points are obtained, which contain the position and posture information. All the critical points are entered in the form of instructions. Then read all the trajectory points shown in the teaching pendant as the parameters of the planning function, and read the number of trajectory points n. Next, input the order of NURBS curve k, and calculate the node vector $U = [u_0, u_1, \ldots, u_{n+k+1}]$ according to the definition of the NURBS curve in Section 2.1. Also, the weight vector ω_i is the input parameter, $i = 0, 1, \ldots, n$. According to Formula (2), calculate the k-order B-spline basis functions $N_{i,k}(u)$ at all control vertices. Based on the above input parameters, the k-order NURBS curve $p(u)$ could be calculated according to Formula (1).

Step 3. Calculate the inverse solution corresponding to all the critical points, and select the optimal inverse solution.

The workspace of industrial robots could be categorized as joint space and Cartesian space. In joint space, a group of end-effector positions corresponding to joint angles could be obtained by forward kinematics. On the other hand, the corresponding joint angles could be obtained in Cartesian space through the inverse kinematics of the robot, and the joint angles obtained by the inverse kinematics solution are usually not unique, so the inverse solution needs to be optimized.

It could be obtained by the DH parameter method described in the literature [30] that:

$$\begin{aligned} T_i = {} & Rot(Z_{i-1}, \theta_i) Trans(0, 0, d_i) \\ & Trans(a_i, 0, 0) Rot(X_i, \alpha_i) \end{aligned} \tag{55}$$

where T_i is the link transformation matrix, $i = 1, 2, \ldots, 6$. In Formula (55), a_i is the link length, α_i is the torsional angle, d_i is the link offset, and θ_i is the joint angle.

The forward kinematics equation of the robot is:

$$T = T_1 T_2 T_3 T_4 T_5 T_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{56}$$

Formula (56) could also be expressed as:

$$\mathbf{x} = x(\theta) \tag{57}$$

where $\mathbf{x} = [x\ y\ z\ rx\ ry\ rz]^T$ is the posture at the end-effector of the robot, which could be described by the posture matrix at the right of Formula (56). The forward kinematics equation of the robot is the function of 6 joint angles.

The derivative of Formula (57) with respect to time, t, is:

$$\dot{\mathbf{x}} = \sum_{i=1}^{6}\sum_{j=1}^{n}\frac{\partial x_i}{\partial \theta_j}\dot{\theta}_j = J(\theta)\dot{\theta} \tag{58}$$

$$\begin{cases} J(\theta) = \left[J_{ij}(\theta)\right]_{i\times j} = \left[\frac{\partial x_i}{\partial \theta_j}\right]_{i\times j} \\ i = 1,2,\ldots,6; j = 1,2,\ldots,n \end{cases} \tag{59}$$

For 6-DOF industrial robot, n = 6, and $J(\theta)$ is a 6 × 6 matrix. Therefore, Formula (58) could also be expressed as:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} & \cdots & J_{16} \\ J_{21} & J_{22} & \cdots & J_{26} \\ \vdots & \vdots & \ddots & \vdots \\ J_{61} & J_{62} & \cdots & J_{66} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_6 \end{bmatrix} \tag{60}$$

The derivative of Formula (60) with respect to time t is:

$$\ddot{x} = J(\theta)\ddot{\theta} + \dot{J}(\theta)\dot{\theta} \tag{61}$$

For posture planning, *Slerp* planning based on quaternion description could be adopted, as is described in Section 2.2.

The inverse kinematics of robots could be solved by the KDL [31], IKFAST, and other motion planning libraries. KDL, i.e., the kinematics and dynamics library, uses the Newton–Raphson iterative algorithm [32] to solve the numerical solution of the inverse kinematics. IKFAST, i.e., fast inverse kinematics, solves the analytic solution of the inverse kinematics for the robot that satisfies Pieper's principle [33]. In the process of robot trajectory planning, the solving speed of robot inverse kinematics directly affects the real-time performance of robot trajectory planning. Therefore, it is of great significance to select a fast and accurate motion algorithm library for robot trajectory planning.

When solving the inverse kinematics, because some end-effector positions and postures of the robot correspond to multiple sets of different joint angles, the inverse kinematics usually has multiple solutions. Therefore, the inverse solution needs to be optimized. The optimal principle of inverse solution is usually that the motion variation of each joint is the least; that is, the motion posture variation is the mildest. Here we could define the displacement in joint space by referring to the definition of displacement in Cartesian space, i.e.,

$$\Delta\theta = \sqrt{\sum_{i=1}^{6}(\Delta\theta_i)^2} \tag{62}$$

That is to say, $\Delta\theta$ is the smallest in Formula (62), or $\Delta\theta_i$ is the smallest.

Step 4. Apply S-Velocity planning in the interpolation interval between adjacent trajectory points.

According to Section 2 of this paper, S-velocity planning could ensure the continuity of acceleration and effectively avoid the impact of joint motors in the robot working process. Therefore, the S-Velocity profile could be selected for planning in the end-effector Cartesian space.

Step 5. In all the interpolation intervals, after the interpolation of the ith segment completes, enter the next segment interpolation until the last segment, and store all the interpolation points of all segments in the .txt file.

The flow chart of the above robot's end-effector trajectory planning can be shown in Figure 5.

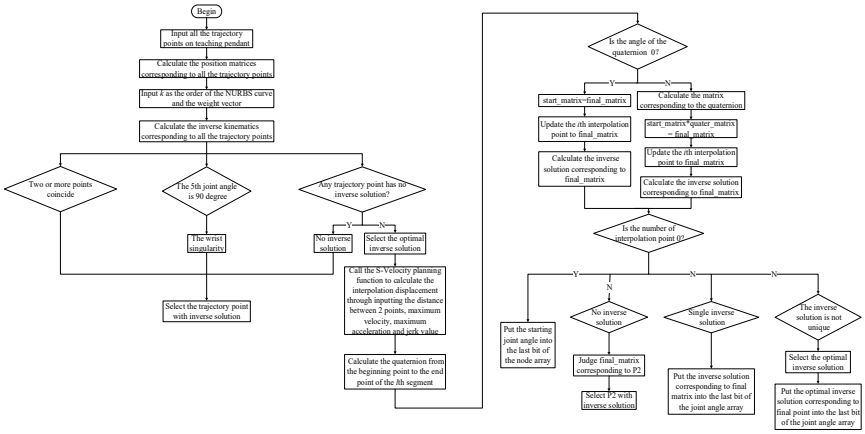


Figure 5. Flow chart of robot's end-effector trajectory planning.

4.2. Trajectory Planning Algorithm Implementation

Remark 6. *The trajectory planning of industrial robots needs strong support from the control system. Generally speaking, the robot system consists of a robot body, a robot electric control cabinet, a teaching device and other parts. Among all the components, the robot body is the servo mechanism, which is composed of six joint motors, through the robot link coordinate system, which could achieve the complex operation of the robot. The robot electric control cabinet is composed of a controller and matching electrical components. As an input device, the teaching pendant undertakes the functions of point input and teaching program analysis in the online teaching process.*

In the robot working process, after teaching all the trajectory points on the teaching pendant, analyze the user program, then execute the user program when the instruction parsing is complete. In the user program execution process, call the trajectory planning function, then we could get the joint angles corresponding to all the interpolation points. Next, send all the joint angles to every joint motor encoder, and the work of the robot is completed. The whole working process could be named as teaching, planning and execution. This is the whole online teaching process of robot operation, as shown in Figure 6.

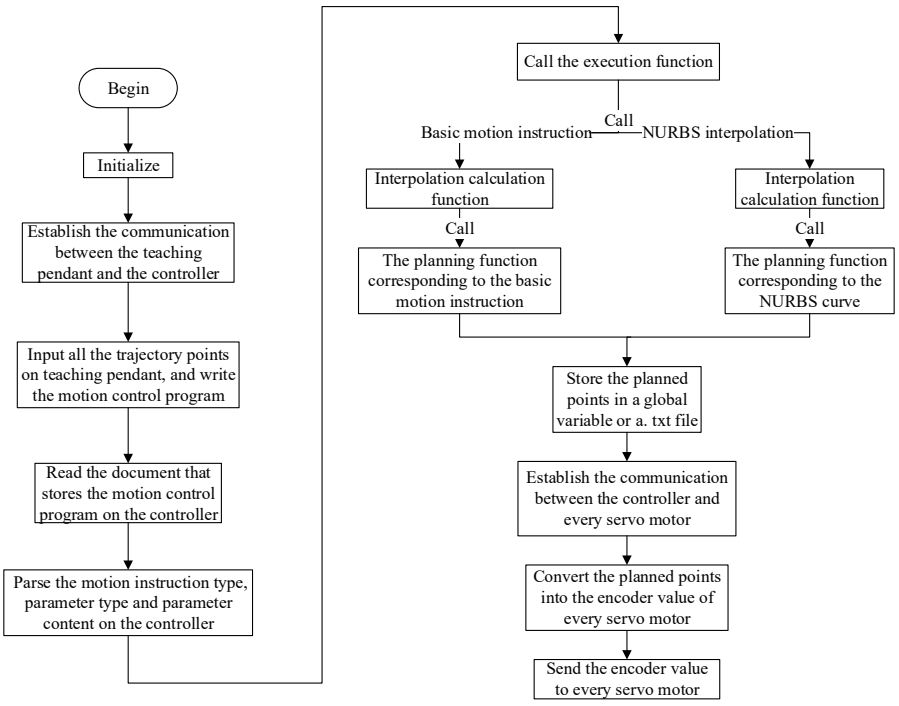


Figure 6. Online teaching flow chart.

5. Analysis of Simulation Experiment Results

5.1. The Kinematic Model of Industrial Robot

To solve the forward and inverse kinematics of the robot, it is necessary to establish the link coordinate frame of the robot, i.e., to solve the DH parameters of the robot first. SR4C robot is selected as the research object in this paper, and its DH parameters are shown in Table 1.

Table 1. The DH parameter of the SR4C robot.

Link	a_i	α_i	d_i	θ_i	Limit (deg)
1	40	90	330	θ_1	−180~180
2	315	0	0	θ_2	−130~80
3	70	90	0	θ_3	−70~160
4	0	−90	310	θ_4	−240~240
5	0	90	0	θ_5	−30~200
6	0	0	70	θ_6	−360~360

In the DH parameter table shown in Table 1, a_i (mm) is the link length, α_i (deg) is the torsional angle, d_i (mm) is the link offset, and θ_i (deg) is the joint angle. The six joints of the SR4C robot are rotational joints, so the joint angle could be arbitrarily set within the limit range.

The link coordinate frame of the SR4C industrial robot could be established according to the DH parameters shown in Table 1, as shown in Figure 7.

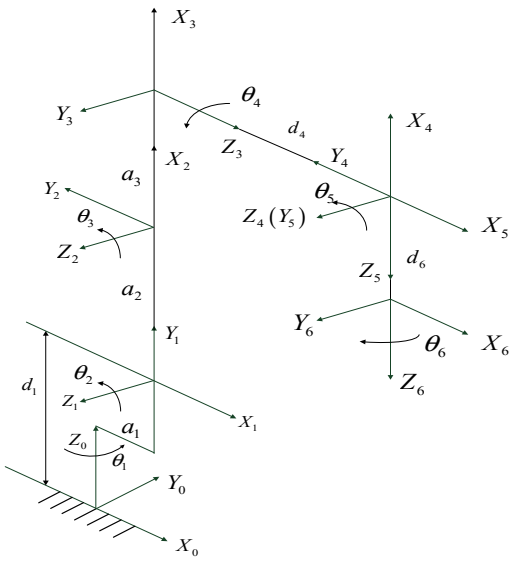


Figure 7. The link coordinate frame of the SR4C robot.

The forward kinematics model of the robot in Formula (56) could be obtained from the DH parameters shown in Table 1.

5.2. Analysis of Simulation Experiment Results

In order to demonstrate the effectiveness of the proposed algorithm, Bernoulli's lemniscate, which is commonly used in robots, is selected as the incentive trajectory to construct the NURBS curve. The equation of Bernoulli's lemniscate is:

$$\left(x^2 + y^2\right)^2 = 2a^2(x^2 - y^2) \tag{63}$$

Let the focus length a =50$\sqrt{2}$ mm; The parametric equation is:

$$\begin{cases} x = a\sqrt{2\cos 2\theta}\cos\theta \\ y = a\sqrt{2\cos 2\theta}\sin\theta \\ \theta \in \left[0, \frac{\pi}{4}\right] \cup \left[\frac{3\pi}{4}, \frac{5\pi}{4}\right] \cup \left[\frac{7\pi}{4}, 2\pi\right] \end{cases} \tag{64}$$

The incentive trajectory is shown in Figure 8.

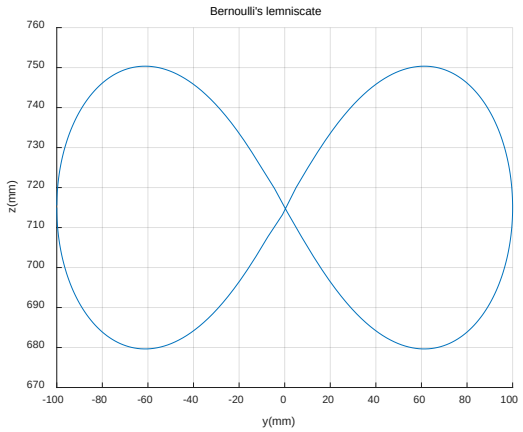


Figure 8. Incentive trajectory.

NURBS curve parameters are as follows: the curve order k = 3; the node vector is U = [0,0,0,0,1/313,2/313, ... ,312/313,1,1,1,1]; The control vertex is determined by Formula (65).

$$\begin{cases} x_i = 420 \\ y_i = -100\cos\theta_i\sqrt{\cos 2\theta_i} \\ z_i = 715 + 100\sin\theta_i\sqrt{\cos 2\theta_i} \\ rx_i = 0 \\ ry_i = \dfrac{90i}{316} \\ rz_i = -\dfrac{90i}{316} \end{cases} \tag{65}$$

where i = 0, 1, ... , 315; The step size of θ_i is 0.01 rad. The control factor is ω = [1,1 ... ,1]; The number of control vertices is n + 1 = 316. The constructed NURBS curve is the coordinate of the robot base coordinate frame when the robot end-effector is working along Bernoulli's lemniscate.

The *Slerp* planning described in Section 2.2 of this paper is adopted for the planning of the robot's end-effector posture, i.e., the rotation angle varies uniformly. The corresponding posture matrix could be obtained from the kinematics of the robot, and the corresponding quaternion could be obtained by combining it with Formula (30).

In this paper, the S-Velocity planning of robots based on NURBS and *Slerp* interpolation could obtain the incentive trajectory during the planning process. The simulation experiment could be taken by referring to the planning process described in Section 3.

In order to demonstrate the effectiveness of the proposed algorithm, the five-order polynomial curve is used for the trajectory planning contrast expriment. Since there are too many trajectory points for manual teaching when teaching path points are input according to Formula (65); therefore, nine representative critical points are selected, and the trajectory point distribution is shown in Figure 9.

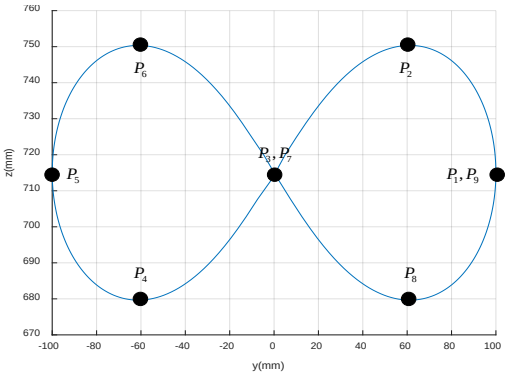


Figure 9. Critical point distribution.

In Figure 9, the robot moves according to $P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_9$, working along the desired trajectory, and P_1, P_5, and P_9 coincide with each other. The coordinate information of every critical point is shown in Table 2, and in the second column of Table 2, the unit of the first three measurements is mm, and the unit of the last three measurements is deg.

With the help of the AT403 absolute laser tracker, a target ball is installed at the end-effector of the SR4C robot, and the camera of the laser tracker can track the movement of the robot's end trajectory in real-time.

Remark 7. *The data point tracking and acquisition block diagram based on the laser tracker is shown in Figure 10.*

Table 2. Coordinate information of every critical point.

	Coordinate in Cartesian Space (mm, deg)	Coordinate in Joint Space (deg)
P_1	(420, 100, 715, 0, 0, 0)	(−166.61, 29.83, 129.9, 0, 20.27, −103.39)
P_2	(420, 61.74, 750.4, 0, 11.25, −11.25)	(−171, 26.75, 127.2, 4.79, 36, −112.7)
P_3	(420, 0, 715, 0, 22.52, −22.49)	(−178.5, 22.16, 139.94, 11.4, 38.08, −119.42)
P_4	(420, −61.74, 679.6, 0, 33.78, −33.81)	(174.1, 20.81, 147, 20.16, 40.97, −127.3)
P_5	(420, −100, 715, 0, 44.96, −45.01)	(−9.58, −6.1, −2.61, −152.5, 27.52, −130.24)
P_6	(420, −61.74, 750.4, 0, 56.19, −56.3)	(−1.97, −6.87, 6.63, 50.4, 151.17, 21.98)
P_7	(420, 0, 715, 0, 67.52, −67.45)	(−171.4, 24.37, 129.94, 74.54, 21.5, 160.05)
P_8	(420, 61.74, 679.6, 0, 78.72, −78.81)	(−162.39, 79.05, 27.91, 80.56, 8.92, 107.07)
P_9	(420, 100, 715, 0,90, −90)	(−157.96, 43.32, 94.59, 74.82, −16.17, 135.74)

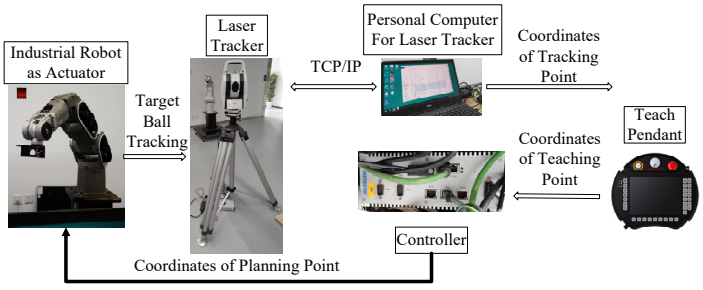


Figure 10. Laser tracker data point acquisition.

The steps of laser tracker data point collection are as follows:

(1) Set the laser tracker, robot controller, and teaching pendant in the same network IP address segment;

(2) Calibrate the position of the laser tracker, and adjust the camera angle to the center of the robot end-effector target ball. Then we could obtain the transformation matrix of the laser tracker to the world coordinate frame;

(3) When the position of the laser tracker is calibrated, Pos_{ideal}, the theoretical position and posture value at the center of the target ball is calculated. Then the robot moves along the trajectory planned according to the teaching points, and the laser tracker obtains the robot's end-effector position and posture value in real-time, namely the measured value Pos_{track}.

(4) When the robot working process is completed, the Pos_{track} of all measured values could be exported, and the trajectory could be drawn in MATLAB.

According to the above step (4), the data derived from the laser tracker are processed to obtain the comparison diagram of the trajectory obtained by the NURBS planning algorithm and the five-order polynomial planning algorithm, as shown in Figure 11.

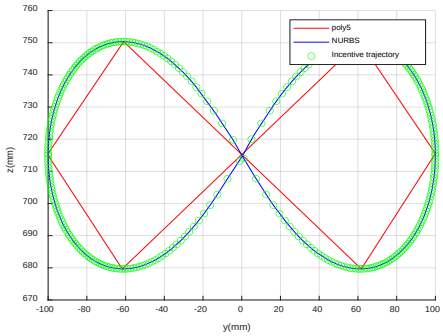
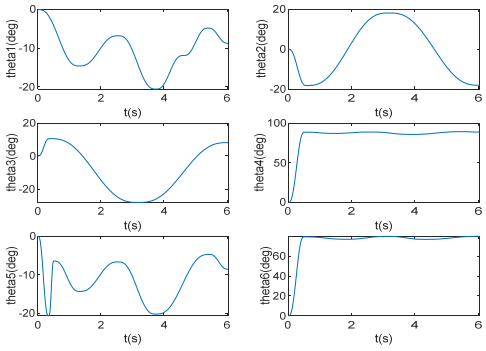
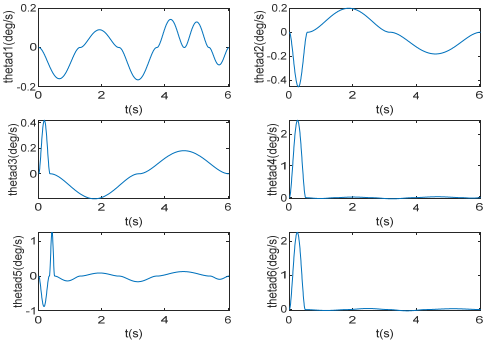


Figure 11. NURBS and five-order polynomial.

According to the trajectory planning process in Figure 5, the displacement, velocity and acceleration curves of each joint planned by the algorithm in this paper could be obtained, as shown in Figure 12.

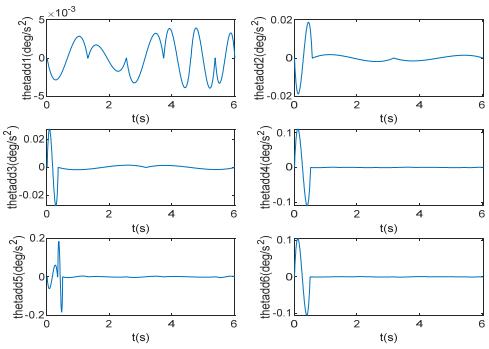


(**a**)



(**b**)

Figure 12. *Cont.*



(**c**)

Figure 12. Joint space curve under NURBS planning; (**a**) Angle curves of six joints under NURBS planning; (**b**) Angular velocity curves of six joints under NURBS planning; (**c**) Angular acceleration curves of six joints under NURBS planning.

Remark 8. *In Figure 12, the angular acceleration of each joint changes continuously with time without mutation, which ensures the smoothness of angular velocity and joint angle curve, reduces the impact during the operation of the joint motor, and extends the service life of joint motors. At the same time, with the improvement of the smoothness of the joint motor angle curve, the quality of the working process is also improved.*

The curvature diagram of the incentive trajectory based on the proposed algorithm and five-order polynomial planning is shown in Figure 13.

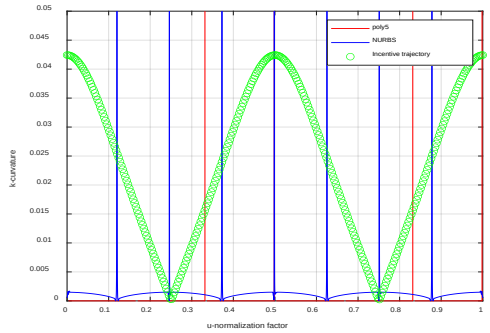


Figure 13. Curvature graph of NURBS and five-degree polynomial planning algorithms.

The calculation of curvature at any point on the NURBS curve could be determined by the following formula:

$$k = \frac{|p'(u) \times p''(u)|}{|p'(u)|^3} = \frac{\sqrt{(y'z'' - y''z')^2 + (x'z'' - x''z')^2 + (x'y'' - x''y')^2}}{\left(\sqrt{x'^2 + y'^2 + z'^2}\right)^3} \quad (66)$$

Remark 9. *It can be seen from Figure 14 that the curvature value of the incentive trajectory obtained by using the five-order polynomial planning is not 0 only at a limited number of points, indicating that the incentive trajectory of the five-order polynomial planning is composed of multiple straight lines, and the smoothness of the incentive trajectory needs to be improved, while the proposed algorithm is based on the NURBS curve construction algorithm. whose curvature of the entire incentive trajectory is not 0, and the curvature value is lower than the original incentive trajectory, that is to say, a relatively gentle NURBS curve is constructed, which effectively improves the smoothness of the incentive trajectory.*

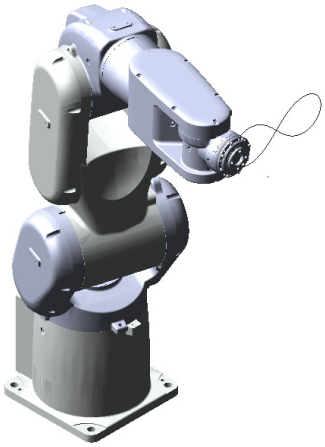


Figure 14. Simulation result.

The data points obtained in the laser tracker are derived as incentive trajectories in Adams and imported into the SR4C simulation model. The results are shown in Figure 14.

In Figure 14, the trajectory of the robot better fits with the designed incentive trajectory, and the robot runs smoothly during the working process. The joint angle, angular velocity and angular acceleration curves shown in Figure 12 fully prove the effectiveness and rationality of the algorithm in Section 5, and improves the working quality.

6. Conclusions

Firstly, The NURBS curve planning algorithm based on the unified curve model proposed is used in this manuscript, and the detailed process is also given. This algorithm realizes the global fairing of the trajectory curve. The Slerp curve equation described by quaternion is deduced in detail, and on this basis, the robot's posture is planned. Then, in the interpolation interval of the robot, the S-type speed planning algorithm is used to realize the speed planning of complex curves. Finally, the data points of the planned excitation trajectory are used as the input parameters of the teaching program and run on the SR4C robot platform. During the operation process, the laser tracker is used to obtain the trajectory points at the end of the robot in real-time. By analyzing the data of the laser tracker, the actual trajectory map is drawn and compared with the five-degree polynomial planning method. The experimental results show that the algorithm proposed in this manuscript can better improve the flexibility of the trajectory, avoid frequent starting and stopping in the operation process, and improve the operation quality compared with the five-degree polynomial programming method. The research results of this manuscript provide practical application value for the high precision and stable operation of industrial robots.

Author Contributions: Conceptualization, G.W.; methodology, G.W.; software, G.W. and K.Z.; validation, Z.P.; formal analysis, F.X.; investigation, F.X.; resources, Z.P.; data curation, K.Z.; writing-original draft preparation, G.W.; writing-review and editing, F.X.; visualization, G.W.; supervision, Z.P.; project administration, G.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of Zhejiang Province, grant number LGG22F030001.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Rodriguez-Linan, A.; Lopez-Juarez, I.; Zalapa-Elias, A.; Chinas-Sanchez, P. An Approach to Acquire Path-Following Skills by Industrial Robots From Human Demonstration. *IEEE Access* **2021**, *9*, 82351–82363. [CrossRef]
2. Wang, G.; Arora, H. Research on Continuous Trajectory Planning of Industrial Welding Robot Based on CAD Technology. *Comput. Aided Des. Appl.* **2022**, *19*, 74–87. [CrossRef]
3. Hua, Y. Review of Trajectory Planning for Industrial Robots. In Proceedings of the 242nd ECS Meeting, Atlanta, GA, USA, 9–13 October 2021.
4. Lu, Y.; Wang, K. Kinematics Analysis and Trajectory Planning of Polishing Six-axis Robot. In Proceedings of the IOP Conference Series: Earth and Environmental Science, Digital Meeting, 30 May–3 June 2021.
5. Gümbel, P.; He, X.; Dröder, K. Precision Optimized Pose and Trajectory Planning for Vertically Articulated Robot Arms. *Procedia CIRP* **2022**, *106*, 185–190. [CrossRef]
6. Tian, S.; Chen, M.; Li, Y. Research on the Trajectory Planning and Control Technology of Industrial Robots Guided by Computer Visualization. In Proceedings of the 242nd ECS Meeting, Atlanta, GA, USA, 9–13 October 2021.
7. Gleeson, D.; Jakobsson, S.; Salman, R.; Ekstedt, F.; Sandgren, N.; Edelvik, F.; Carlson, J.S.; Lennartson, B. Generating Optimized Trajectories for Robotic Spray Painting. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 1380–1391. [CrossRef]
8. Bhardwaj, G.; Mishra, U.A.; Sukavanam, N.; Balasubramanian, R. Planning Adaptive Brachistochrone and Circular Arc Hip Trajectory for a Toe-Foot Bipedal Robot going Downstairs. In Proceedings of the 239th ECS Meeting with IMCS18, Digital Meeting, 30 May–3 June 2021.
9. Wang, S.D.; Luo, X.; Xu, S.J.; Luo, Q.S.; Han, B.L.; Liang, G.H.; Jia, Y. A Planning Method for Multi-Axis Point-to-Point Synchronization Based on Time Constraints. *IEEE Access* **2020**, *8*, 85575–85604. [CrossRef]
10. Dantam, N.; Stilman, M. Spherical Parabolic Blends for Robot Workspace Trajectories. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014.
11. Santina, C.D.; Rus, D. Control Oriented Modeling of Soft Robots: The Polynomial Curvature Case. *IEEE Robot. Autom. Lett.* **2020**, *5*, 290–298. [CrossRef]
12. Xu, J.; Liu, J.; Sheng, J. Arc Path Tracking Algorithm of Dual Differential Driving Automated Guided Vehicle. In Proceedings of the 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, Beijing, China, 13–15 October 2018.
13. Campos, J.A.F.; Flores, J.A.R.; Montufar, C.P. Robot Trajectory Planning for Multiple 3D Moving Objects Interception: A Polynomial Interpolation Approach. In Proceedings of the 2008 Electronics, Robotics and Automotive Mechanics Conference, Cuernavaca, Mexico, 30 September–3 October 2008.
14. Lu, T.C.; Chen, S.L. Real-Time Local Optimal Bézier Corner Smoothing for CNC Machine Tools. *IEEE Access* **2021**, *9*, 152718–152727. [CrossRef]
15. Li, D.; Zhang, L. Corner Smoothing Interpolation Algorithm Based on PH Curve. In Proceedings of the 2019 3rd International Conference on Electronic Information Technology and Computer Engineering, Xiamen, China, 18–20 October 2019.
16. Annoni, M.; Bardine, A.; Campanelli, S.; Foglia, P.; Prete, C.A. A Real-time Configurable NURBS Interpolator with Bounded Acceleration, Jerk and Chord Error. *Comput. Aided Des.* **2012**, *44*, 509–521. [CrossRef]
17. Shi, X.; Fang, H.; Gang, P.; Xu, X.; Chen, H. Time-Energy-Jerk Dynamic Optimal Trajectory Planning for Manipulators Based on Quintic NURBS. In Proceedings of the 2018 3rd International Conference on Robotics and Automation Engineering, Guangzhou, China, 17–19 November 2018.
18. 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.
19. Liu, Y.; Jiang, D.; Tao, B.; Qi, J.; Jiang, G.; Yun, J.; Huange, L.; Tong, X.; Chen, B.; Li, G. Grasping Posture of Humanoid Manipulator Based on Target Shape Analysis and Force Closure. *Alex. Eng. J.* **2022**, *61*, 3959–3969. [CrossRef]
20. Huanga, Q.; Enomoto, R. Hybrid Position, Posture, Force and Moment Control of Robot Manipulators. In Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics, Bangkok, Thailand, 22–25 February 2009.
21. Liu, B.; Zhang, F.; Qu, X.; Shi, X. A Rapid Coordinate Transformation Method Applied in Industrial Robot Calibration Based on Characteristic Line Coincidence. *Sensors* **2016**, *16*, 239. [CrossRef] [PubMed]
22. Jiang, B.; Li, J.; Du, X.; Duan, P. Attitude Interpolation Algorithm for Industrial Robot Based on Quaternion Method. In Proceedings of the 2021 International Conference on Mechanical Engineering Intelligent Manufacturing and Automation Technology, Guilin, China, 15–17 January 2021.
23. Wang, G.; Liu, X.; Han, S. A Method of Robot Base Frame Calibration by Using Dual Quaternion Algebra. *IEEE Access* **2018**, *6*, 74865–74873. [CrossRef]
24. Martínez, J.R.G.; Reséndiz, J.R.; Prado, M.Á.M.; Miguel, E.E.C. Assessment of Jerk Performance S-curve and Trapezoidal Velocity Profiles. In Proceedings of the 2017 XIII International Engineering Congress, Santiago de Queretaro, Mexico, 15–19 May 2017.
25. Rymansaib, Z.; Iravani, P.; Sahinkaya, M.N. Exponential Trajectory Generation for Point to Point Motions. In Proceedings of the 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Wollongong, Australia, 9–12 July 2013.
26. Huang, J.; Zhu, L.M. Feedrate Scheduling for Interpolation of Parametric Tool Path Using the Sine Series Representation of Jerk Profile. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2016**, *231*, 2359–2371. [CrossRef]

27. Wu, C.; Chiu, Z.Y.; Liu, J.; Trabia, M.B. Time-Optimal Trajectory Planning along Parametric Polynomial Lane-Change Curves with Bounded Velocity and Acceleration: Simulations for a Unicycle Based on Numerical Integration. *Model. Simul. Eng.* **2018**, *2018*, 9348907. [CrossRef]
28. Li, J.; Liu, Y.; Li, Y.; Zhong, G. S-Model Speed Planning of NURBS Curve Based on Uniaxial Performance Limitation. *IEEE Access* **2019**, *7*, 60837–60849. [CrossRef]
29. Shi, B.H.; Jiang, T. NURBS Piecewise Interpolation Algorithm Based on Discrete S-Type Velocity Planning. In Proceedings of the 2019 Chinese Control And Decision Conference, Nanchang, China, 3–5 June 2019.
30. Craig, J.J. *Introduction to Robotics: Mechanics and Control*; Pearson Education: New York, NY, USA, 2005; pp. 62–134.
31. Soetens, P. A Software Framework for Real-Time and Distributed Robot and Machine Control. *J. Appl. Phys.* **2006**, *70*, 1991–2000.
32. Lee, S.; Lee, J.; Bang, J.; Lee, J. 7 DOF Manipulator Construction and Inverse Kinematics Calculation and Analysis using Newton-Raphson Method. In Proceedings of the 2021 18th International Conference on Ubiquitous Robots (UR), Gangneung, Korea, 12–14 July 2021.
33. Peiper, D.L. The Kinematics of Manipulators Under Computer Control. Ph.D. Thesis, Stanford University, School of Humanities and Sciences, Stanford, CA, USA, 1968.