Zhang *et al.*
Chinese Journal of Mechanical Engineering (2023) 36:59
https://doi.org/10.1186/s10033-023-00881-x

Chinese Journal of Mechanical Engineering

ORIGINAL ARTICLE

Open Access

A Real-time Look-ahead Trajectory Planning Methodology for Multi Small Line Segments Path

Sai Zhang[1,2], Xinjun Liu[1*], Bingkai Yan[2], Jie Bi[2] and Xiangdong Han[2]

Abstract

When a robot is required to machine a complex curved workpiece with high precision and speed, the tool path is typically dispersed into a series of points and transmitted to the robot. The conventional trajectory planning method requires frequent starts and stops at each dispersed point to complete the task. This method not only reduces precision but also causes damage to the motors and robot. A real-time look-ahead algorithm is proposed in this paper to improve precision and minimize damage. The proposed algorithm includes a path-smoothing algorithm, a trajectory planning method, and a bidirectional scanning module. The path-smoothing method inserts a quintic Bezier curve between small adjacent line segments to achieve G^2 continuity at the junctions. The trajectory planning method utilizes a quartic polynomial and a double-quartic polynomial that can achieve a constant velocity at the velocity limitation. The bidirectional scanning module calculates the velocity at each trajectory planning segment point, simplifying calculation complexity and can be run in real time. The feasibility of the proposed algorithm is verified through simulations and experiments, which can be run in real time. In addition, high machining precision can be achieved by adjusting the relevant parameters.

Keywords Look-ahead, Bidirectional scanning, Transition scheme, Bezier curve, G^2 continuity

1 Introduction

With the increasing demand for the high-precision and high-speed machining of complex curved workpieces, the processing ability of computer numerical control (CNC) and industrial robots that machine small line segments has become critical [1, 2]. The keys are the path-smoothing algorithm, trajectory planning method, calculation of the velocity at each point, and the conditions must be satisfied to reach the specified position at the specified speed, acceleration, and time. Path-smoothing and look-ahead were introduced as effective methods for enhancing the precision and efficiency in machining [3–5].

Curve fitting and curve transition methods are two major approaches that have been proposed in previous studies to achieve smoothness of small line segments. Curve fitting methods connect all points in a sequence to form a curve, including curve fitting algorithms [6], curve interpolation, curve derivation, and curve-arc length algorithms [7, 8], which have been applied in robot controllers and advanced CNC systems [9]. However, this method has several disadvantages. First, the calculation for fitting a curve through all the assigned points is cumbersome to be performed in real time. Second, the range of the curve parametric u is 0 to 1, which incurs increased calculation costs and interpolation difficulty. Curve transition methods, circular arcs, and spline curves are generally employed for insertion between adjacent line segments. The advantage of this method is that the

*Correspondence:
Xinjun Liu
xinjunliu@mail.tsinghua.edu.cn
[1] Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China
[2] Robotphoenix Automation Technology Co., Ltd., Jinan 250100, China

following trajectory can be calculated when machining the previous trajectory.

There are two continuous types, which are the G^2 continuity and C^2 continuity. The definition of C^2 continuity is that the 0th, first, and second derivatives are continuous. In the G^2 continuity, the curves share a common center of curvature at the joining point. Because the acceleration at the connection points of lines and curves should be continuous, the G^2 continuity should be ensured. The curve transition technique has been proposed for constructing the G^2 continuous path. The theory of this technique is that sharp corners between adjacent line segments are replaced with a spline curve, and the adjacent lines and curves have G^2 continuity, which has already been adopted in the robot domain [10, 11]. Jouaneh et al. [12] inserted a circular arc between the adjacent line segments for rapid cornering. However, the circular arc exhibits continuous motion transition only to the velocity, except for acceleration transition. This technique can achieve real-time performance in the NC system because of the minimal computation of circular arc. Because accelerations at junctions between the line segments and circular arcs are noncontinuous, the order of the blending curve must be increased while controlling the fitting tolerance and corner geometry. Pateloup et al. [13] proposed a two-dimensional curve transition algorithm that utilizes cubic B-splines constructed by eight control points to round the pocket contour. Similarly, Zhang et al. [14] employed double cubic B-splines. Walton and Meek [15] proposed utilizing Bezier or PH quintic curves as the transition curves. Bi et al. [16] proposed a cubic Bezier curve with two identical control points in the middle and achieved continuous-curvature path-smoothing.

This becomes a G^2 continuity path, which is constructed using multiple small line segments and blended with spline curves. The next problem is trajectory planning along a path. Because of the existence of spline curves, the velocity at the point on the curves must be lowered such that the limitations of acceleration and jerk are not exceeded in the curved sections. The kinematic limitations of machine tools have been fully considered to obtain a time-optimal feed rate curve [17]. Various of acceleration and deceleration trajectory planning algorithms, such as polynomial, trapezoidal, and S-shaped jerk-limited methods [18–20] and the jerk-continuous methods [21–23], are available, and the S-shaped acceleration and deceleration algorithm is widely used [24–26]. After the path-smoothing and trajectory planning methods are selected, the look-ahead function that calculates the velocity at each point should be determined. Bidirectional scanning algorithms can be adopted in the look-ahead function to constrain the maximum velocity, acceleration, and jerk during the motion process [27–29].

In this study, a method for constructing the G^2 continuity transition curve between adjacent line segments and a jerk-limited look-ahead trajectory planning method were developed. The remainder of this paper is organized as follows. The G^2 continuity transition curve that Bezier curves is presented in Section 2. A jerk-limited real-time bidirectional look-ahead algorithm and the proposed double-quartic polynomial trajectory planning are presented in Section 3. Section 4 describes the designed simulations and experiments. In Section 5, the conclusions are presented.

2 Transition Scheme

A quintic Bezier curve is inserted between adjacent line segments to improve the machining efficiency and precision (Figure 1), which can be defined as [30]

$$\begin{aligned} B(u) &= \sum_{i=0}^{5} C_5^i u^i (1-u)^{5-i} P_i \\ &= (1-u)^5 P_0 + 5u(1-u)^4 P_1 + 10u^2(1-u)^3 P_2 \\ &\quad + 10u^3(1-u)^2 P_3 + 5u^4(1-u) P_4 + u^5 P_5, u \in [0,1], \end{aligned} \tag{1}$$

where P_0, P_1, …, P_5 are the control points of the quintic Bezier curve, and u is the parameter of the Bezier curve.

2.1 Construction of Quintic Bezier Curve

The quintic Bezier curve is inserted between the two adjacent line segments, as shown in Figure 1. For example, P_{start} is the start point of the k^{th} line segment, P_{trans} is the endpoint of the k^{th} line segment and the start point of the $(k+1)^{th}$ line segment, P_{end} is the endpoint of the $(k+1)^{th}$ line segment, and θ is the angle between the two line segments. Because the order of the inserted Bezier curve is quintic, the continuity of G^2 is ensured, and two adjacent line segments have the same normal and tangent directions with the inserted Bezier curve at the connection points. Differentiating Eq. (1) at the

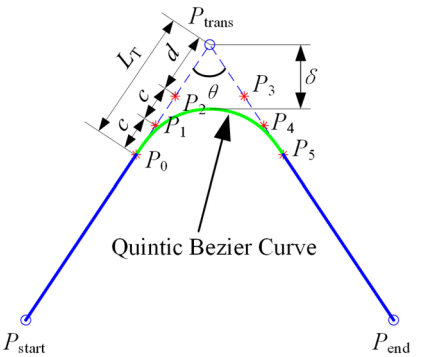


Figure 1 Quintic Bezier curve between two adjacent line segments

parameter u yields the first and second derivatives of the Bezier curve:

$$\begin{aligned} B_{\mathrm{u}} &= \frac{\mathrm{d}B(u)}{\mathrm{d}u} \\ &= 5(1-u)^4(P_1-P_0) + 20u(1-u)^3(P_2-P_1) \\ &\quad + 30u^2(1-u)^2(P_3-P_2) + 20u^3(1-u)(P_4-P_3) \\ &\quad + 5u^4(P_5-P_4), \end{aligned} \tag{2}$$

$$\begin{aligned} B_{\mathrm{uu}} = \frac{\mathrm{d}^2B(u)}{\mathrm{d}u^2} &= 20(1-u)^3(P_2-2P_1+P_0) \\ &\quad + 60u(1-u)^2(P_3-2P_4+P_5). \end{aligned} \tag{3}$$

The following relationship can be derived by analyzing Eqs. (1)–(3) at the start, $u=0$, and the end, $u=1$, of the Bezier curve:

$$\begin{cases} B|_{u=0} = P_0, \\ B_{\mathrm{u}}|_{u=0} = 5(P_1-P_0), \\ B_{\mathrm{uu}}|_{u=0} = 20(P_2-2P_1+P_0), \\ B|_{u=1} = P_5, \\ B_{\mathrm{u}}|_{u=1} = 5(P_5-P_4), \\ B_{\mathrm{uu}}|_{u=0} = 20(P_5-2P_4+P_3). \end{cases} \tag{4}$$

As is well known, control points P_0 and P_5 should be the points from the line segment to transition curve, and the transition curve to the next line segment, respectively, if the G^0 continuity should be ensured. If G^1 continuity is ensured, the unit vector corresponding to $B_{\mathrm{u}}|_{u=0}$ should be equal to the unit direction vector of the first line, as should that of the second line corresponding to $B_{\mathrm{u}}|_{u=1}$. From this point and Eq. (4), P_1 and P_4 should be on the line segments $P_{\mathrm{start}}P_{\mathrm{trans}}$ and $P_{\mathrm{trans}}P_{\mathrm{end}}$, respectively. Because the second derivative of a line segment is a zero vector, $B_{\mathrm{uu}}|_{u=0}$ and $B_{\mathrm{uu}}|_{u=1}$ are zero vectors, another relationship between the control points can be derived:

$$\begin{cases} P_2 - P_1 = P_1 - P_0, \\ P_5 - P_4 = P_4 - P_3. \end{cases} \tag{5}$$

The Bezier curve is determined when the parameters c, d, and L_{T} have been calculated.

2.2 Curvature Optimization and Cornering Errors

The normal acceleration and jerk are calculated by the curvature, k, and the corresponding velocity, v, at this point. Thus, v_{limit} used for look-ahead planning can be calculated using the maximum normal acceleration ${}^{\mathrm{n}}a_{\max}$, maximum jerk $j_{\max}$, and curvature k, which can be expressed as:

$${}^{\mathrm{a}}v_{\mathrm{limit}} = \sqrt{\frac{{}^{\mathrm{n}}a_{\max}}{k}}, \tag{6}$$

$${}^{\mathrm{j}}v_{\mathrm{limit}} = \sqrt[3]{\frac{j_{\max}}{k^2}}, \tag{7}$$

where ${}^{\mathrm{a}}v_{\mathrm{limit}}$ and ${}^{\mathrm{j}}v_{\mathrm{limit}}$ are the velocity limitations corresponding to the maximum normal acceleration and maximum jerk, respectively. v_{limit} is assigned to the minimum of ${}^{\mathrm{a}}v_{\mathrm{limit}}$, ${}^{\mathrm{j}}v_{\mathrm{limit}}$, and $v_{\max}$. The higher the value of ${}^{\mathrm{n}}a_{\max}$ and $j_{\max}$, and the smaller the value of curvature k, the better to run at high speeds for robots.

The curvature $k(u_i)$ of the Bezier curve $B(u)$ at parameter u_i is expressed as:

$$k(u_i) = \frac{\|B_{\mathrm{u}}(u_i) \times B_{\mathrm{uu}}(u_i)\|}{\|B_{\mathrm{u}}(u_i)\|^3}. \tag{8}$$

As shown in Figure 1, the corner distance, L_{T}, is the only scale factor in the construction of the transition curve:

$$L_{\mathrm{T}} = 2c + d. \tag{9}$$

Typically, the range of L_{T} is the minimum length of two adjacent line segments multiplied by a factor range in $(0, 0.5)$. We set n as:

$$n = c/d. \tag{10}$$

After the parameter n is determined, the Bezier curve and its maximum curvature are confirmed. A simple power regression is applied [31] to reduce the maximum curvature and obtain the optimal curvature value k for cornering angles θ:

$$n(\theta) = \frac{1}{2.0769}\theta^{0.9927}. \tag{11}$$

δ is the cornering error between the midpoint of the transition curve and the point of intersection of the adjacent line segments. The smaller the cornering error, the closer to the original trajectory, and the better the machining effect. The values of c and d can be obtained using the technique adopted in Ref. [32] to constrain the cornering error δ within a specific range for a curvature optimum curve:

$$\begin{cases} c = \dfrac{32\delta}{(7n+16)\sqrt{2+2\cos(\theta)}}n, \\ d = \dfrac{32\delta}{(7n+16)\sqrt{2+2\cos(\theta)}}. \end{cases} \tag{12}$$

2.3 Adaptive Interpolation Design

After path-smoothing based on the approach presented in Sections 2.1 and 2.2, the original path, consisting of many line segments, becomes a G^2 continuity path comprising curves and lines. The displacement at each millisecond can be calculated using the motion-planning module, and the interpolation module calculates the point on the line or curve using the displacement. The interpolation module is simple for line segments:

$$P_{\text{current}} = \frac{s_i - s_{\text{current}}}{s_i} P_i + \frac{s_{\text{current}}}{s_i} P_{i+1}, \tag{13}$$

where P_{current} is the current interpolation point, s_i is the length of the current line, s_{current} is the displacement on this line, and P_i and P_{i+1} are the start and end points of the line.

The interpolation module is more complicated for curved segments. The second-order Taylor's expansion is a good method for precisely obtaining the interpolated points on the curve segments:

$$u(t_{i+1}) = u(t_i) + \left.\frac{\mathrm{d}u}{\mathrm{d}t}\right|_{t=t_i} \Delta t + \frac{1}{2}\left.\frac{\mathrm{d}^2u}{\mathrm{d}t^2}\right|_{t=t_i} \Delta t^2 + o(h), \tag{14}$$

where $u(t_i)$ is the curve parameter u at last time t_i, $\frac{\mathrm{d}u}{\mathrm{d}t}$, and $\frac{\mathrm{d}^2u}{\mathrm{d}t^2}$ are the first and second derivatives for u with respect to t. $\frac{\mathrm{d}u}{\mathrm{d}t}$ can be expressed as:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{\mathrm{d}u}{\mathrm{d}s}\frac{\mathrm{d}s}{\mathrm{d}t} = \frac{v}{\mathrm{d}s/\mathrm{d}u}. \tag{15}$$

The calculation of $\frac{\mathrm{d}^2u}{\mathrm{d}t^2}$ depends on acceleration a:

$$\begin{aligned} a = \frac{\mathrm{d}^2s}{\mathrm{d}t^2} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}s}{\mathrm{d}u}\frac{\mathrm{d}u}{\mathrm{d}t}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}s}{\mathrm{d}u}\right)\frac{\mathrm{d}u}{\mathrm{d}t} + \frac{\mathrm{d}^2u}{\mathrm{d}t^2}\frac{\mathrm{d}s}{\mathrm{d}u} \\ &= \frac{\mathrm{d}^2s}{\mathrm{d}u^2}\left(\frac{\mathrm{d}u}{\mathrm{d}t}\right)^2 + \frac{\mathrm{d}^2u}{\mathrm{d}t^2}\frac{\mathrm{d}s}{\mathrm{d}u}. \end{aligned} \tag{16}$$

Therefore, $\frac{\mathrm{d}^2u}{\mathrm{d}t^2}$ can be expressed by

$$\frac{\mathrm{d}^2u}{\mathrm{d}t^2} = \left(a - \frac{\mathrm{d}^2s}{\mathrm{d}u^2}\frac{v^2}{(\mathrm{d}s/\mathrm{d}u)^2}\right) \Big/ (\mathrm{d}s/\mathrm{d}u), \tag{17}$$

where $\frac{\mathrm{d}s}{\mathrm{d}u}$ and $\frac{\mathrm{d}^2s}{\mathrm{d}u^2}$ are the first and second derivatives of arc length s with respect to parameter u, which are the norms of Eqs. (2) and (3), respectively.

Therefore, $u(t_{i+1})$ is obtained; however, a parametric truncation error $o(h)$ exists, which is completely ignored, resulting in interpolated points that are not real points. This causes sharp variations in velocity, acceleration, and jerk [31].

Referring to Eq. (14) and assuming that $u'(t_{i+1})$ is $u(t_{i+1})$ but ignores the parametric truncation error:

$$u'(t_{i+1}) = u(t_i) + \left.\frac{\mathrm{d}u}{\mathrm{d}t}\right|_{t=t_i} \Delta t + \frac{1}{2}\left.\frac{\mathrm{d}^2u}{\mathrm{d}t^2}\right|_{t=t_i} \Delta t^2. \tag{18}$$

Eq. (14) can be rewritten as:

$$u(t_{i+1}) = u'(t_{i+1}) + \varepsilon, \tag{19}$$

where ε is the parametric truncation error $o(h)$. The interpolation accuracy is high when ε is compensated for. The next step is to determine ε.

$$v(u(t_{i+1})) = \frac{\sqrt{[X(u(t_{i+1})) - X(u(t_i))]^2 + [Y(u(t_{i+1})) - Y(u(t_i))]^2}}{\Delta t}, \tag{20}$$

where $X(u(t_{i+1}))$ and $Y(u(t_{i+1}))$ are the x- and y- coordinates of the point on the curve, respectively, which can be expressed as:

$$\begin{aligned} X(u(t_{i+1})) &= X\left(u'(t_{i+1}) + \varepsilon\right) \\ &= X\left(u'(t_{i+1})\right) + \frac{\mathrm{d}X\left(u'(t_{i+1})\right)}{\mathrm{d}u}\varepsilon, \end{aligned} \tag{21}$$

$$\begin{aligned} Y(u(t_{i+1})) &= Y\left(u'(t_{i+1}) + \varepsilon\right) \\ &= Y\left(u'(t_{i+1})\right) + \frac{\mathrm{d}Y\left(u'(t_{i+1})\right)}{\mathrm{d}u}\varepsilon. \end{aligned} \tag{22}$$

Assuming that

$$\begin{cases} \Delta X'(u(t_{i+1})) = X\left(u'(t_{i+1})\right) - X(u(t_i)), \\ \Delta Y'(u(t_{i+1})) = Y\left(u'(t_{i+1})\right) - Y(u(t_i)). \end{cases} \tag{23}$$

By substituting $X(u(t_{i+1}))$ and $Y(u(t_{i+1}))$ described by Eqs. (21) and (22), respectively, into Eq. (20) and multiplying Δt on both sides of Eq. (20) simultaneously, and Eq. (20) can be rewritten as:

$$v(u(t_{i+1}))\Delta t = \sqrt{\left[\Delta X'(u(t_{i+1})) + \frac{\mathrm{d}X(u'(t_{i+1}))}{\mathrm{d}u}\varepsilon\right]^2 + \left[\Delta Y'(u(t_{i+1})) + \frac{\mathrm{d}Y(u'(t_{i+1}))}{\mathrm{d}u}\varepsilon\right]^2}. \tag{24}$$

Therefore, the quadratic equation for ε can be expressed as:

$$\left[\left(\frac{\mathrm{d}X\left(u'(t_{i+1})\right)}{\mathrm{d}u}\right)^2+\left(\frac{\mathrm{d}Y\left(u'(t_{i+1})\right)}{\mathrm{d}u}\right)^2\right]\varepsilon^2 + 2\left[\Delta X'(u(t_{i+1}))\frac{\mathrm{d}X\left(u'(t_{i+1})\right)}{\mathrm{d}u}+\Delta Y'(u(t_{i+1}))\frac{\mathrm{d}Y\left(u'(t_{i+1})\right)}{\mathrm{d}u}\right]\varepsilon + \left(\Delta X'(u(t_{i+1}))\right)^2+\left(\Delta Y'(u(t_{i+1}))\right)^2-v^2(u(t_{i+1}))\Delta t^2=0. \tag{25}$$

Assuming that:

$$A=\left(\frac{\mathrm{d}X\left(u'(t_{i+1})\right)}{\mathrm{d}u}\right)^2+\left(\frac{\mathrm{d}Y\left(u'(t_{i+1})\right)}{\mathrm{d}u}\right)^2,$$

$$B=2\left[\Delta X'(u(t_{i+1}))\frac{\mathrm{d}X\left(u'(t_{i+1})\right)}{\mathrm{d}u}+\Delta Y'(u(t_{i+1}))\frac{\mathrm{d}Y\left(u'(t_{i+1})\right)}{\mathrm{d}u}\right],$$

$$C=\left(\Delta X'(u(t_{i+1}))\right)^2+\left(\Delta Y'(u(t_{i+1}))\right)^2-v^2(u(t_{i+1}))\Delta t^2.$$

Eq. (25) can be simplified as:

$$A\varepsilon^2+B\varepsilon+c=0. \tag{26}$$

The parametric truncation error, ε, can be obtained by solving this equation. Eq. (26) has complex roots or only a real root if $B^2-4AC\le 0$. If $B^2-4AC>0$ is satisfied, then the equation has two real roots:

$$\varepsilon_1=\frac{-B+\sqrt{B^2-4AC}}{2A},\varepsilon_2=\frac{-B-\sqrt{B^2-4AC}}{2A}. \tag{27}$$

Because the value of ε_1 is small (close to 0), ε_2 is negative and has a larger absolute value. A smaller value of root ε_1 is desirable to achieve a relatively more accurate compensation during the interpolation process. A more accurate $u(t_{i+1})$ can be obtained by substituting ε_1 into Eq. (19).

3 Look-ahead Function

In the previous section, a path with a continuous curvature and a mixture of small line segments and curves was obtained. This section presents a real-time bidirectional look-ahead algorithm. The look-ahead function consists of two main modules. The first module is a bidirectional scanning module that utilizes the proposed trajectory planning method to calculate the velocities at each trajectory planning segment point. The second module is trajectory planning based on the quartic polynomial, which includes the proposed double-quartic polynomial trajectory planning and quartic polynomial trajectory planning for line segments and curves, respectively.

3.1 Bidirectional Scanning Look-ahead Module

In the first module, the array of $\boldsymbol{\nu}_{\mathrm{limit}}$ should be calculated before running the bidirectional scanning module, as expressed by Eq. (28), and the initial parameters should be transmitted to the bidirectional scanning module. $\boldsymbol{\nu}_{\mathrm{limit}}$ consists of the velocities at the starting and ending points, at the junctions between the line segments and transition curves, and in the middle of the transition curves. The point in the middle of the transition curve was adopted because of the maximum curvature of these curves at the middle points in general. This method can ensure the acceleration and jerk within the limitations in the middle of curves and improve the machining efficiency.

$$\boldsymbol{\nu}_{\mathrm{limit}}(i)=\min\left({}^{\mathrm{a}}\boldsymbol{\nu}_{\mathrm{limit}}(i),{}^{\mathrm{j}}\boldsymbol{\nu}_{\mathrm{limit}}(i),\nu_{\max}\right), \tag{28}$$

where ${}^{\mathrm{a}}\boldsymbol{\nu}_{\mathrm{limit}}$ and ${}^{\mathrm{j}}\boldsymbol{\nu}_{\mathrm{limit}}$ are calculated using Eqs. (6) and (7), respectively.

For the global look-ahead module, as shown in Figure 2, the forward scanning look-ahead module is performed first. Velocity v_{start} at the start point $P_1\left(P_{v,1}\right)$ is set to 0 to ensure start running with a zero velocity for robots. $\boldsymbol{\nu}_{\mathrm{limit}}(i)$ and $\boldsymbol{\nu}_{\mathrm{limit}}(i+1)$, as the initial velocities, are transmitted to the trajectory planning module. If the maximum acceleration or jerk exceeds the

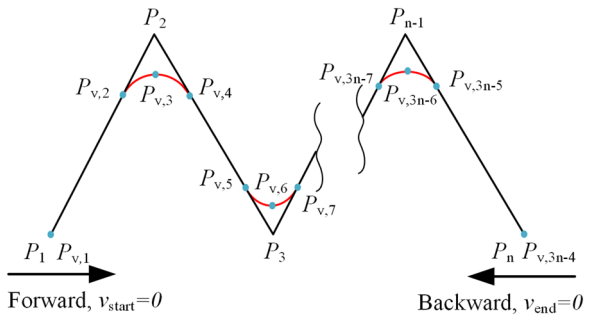


Figure 2 Schematic diagram of look-ahead module

limitations of the acceleration or jerk, an appropriate new $\boldsymbol{v}_{\text{limit}}(i+1)$ that can ensure the acceleration and jerk within the limitations will be calculated by the trajectory planning module to substitute the old value. When the forward scanning is completed, the array of $\boldsymbol{v}_{\text{limit}}$ has preliminary update completed with the start velocity equal to 0, but the end velocity is not usually equal to 0. Hence, in the next step, the backward scanning is performed to ensure that the end velocity is equal to 0.

In the backward scanning look-ahead module, the velocity v_{end} at end point $P_n\left(P_{v,3n-4}\right)$ was set to 0 to ensure that the robot stopped running at a zero velocity. $\boldsymbol{v}_{\text{limit}}(i+1)$ and $\boldsymbol{v}_{\text{limit}}(i)$ in $\boldsymbol{v}_{\text{limit}}$ are transmitted to the trajectory planning module as the initial velocities. If the maximum acceleration or jerk exceeds the limitations of the acceleration and jerk, an appropriate new $\boldsymbol{v}_{\text{limit}}(i)$ that can ensure that the acceleration and jerk are within the limitations is calculated by the trajectory planning module to substitute the old value.

3.2 Quartic Polynomial Trajectory Planning

For smooth start-stop motion, quartic polynomial and double-quartic polynomial trajectory planning are employed in the curve segments and line segments, respectively. Quartic polynomial trajectory planning can achieve acceleration or deceleration, except constant velocity. Double-quartic polynomial trajectory planning has the advantage of a quartic polynomial, and can achieve a constant velocity. The displacement, s, of the quartic polynomial is expressed as:

$$s = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4. \tag{29}$$

Because the displacement at the starting point is 0, the velocity at the starting point is v_0, and the acceleration at the starting point is 0, the partial parameters of the quartic polynomial can be obtained as:

$$\begin{cases} c_0 = 0, \\ c_1 = v_0, \\ c_2 = 0. \end{cases} \tag{30}$$

Based on the quartic polynomials, two critical equations can be derived:

$$c_4 = \frac{-c_3}{2T}, \tag{31}$$

$$T = \frac{2s}{v_0 + v_1}, \tag{32}$$

where T is the total running time, and s is the path length. The maximum acceleration and jerk occur at $\frac{T}{2}$ and 0, respectively.

$$a_{\text{m}} = \frac{6c_3 T}{2} + 12c_4\left(\frac{T}{2}\right)^2, \tag{33}$$

$$j_{\text{m}} = 6c_3. \tag{34}$$

If $a_{\text{m}} > a_{\max}$ or $j_{\text{m}} > j_{\max}$, the end velocity of this segment should be changed. Assuming that the maximum jerk just reaches the jerk limitation, c_3 can be obtained:

$$c_3 = \frac{1}{6} j_{\max}. \tag{35}$$

Substituting Eq. (35) into Eq. (31) yields

$$c_4 = -\frac{j_{\max}}{12T}. \tag{36}$$

Substituting Eqs. (30), (35), and (36) into Eq. (29) gives:

$$\frac{j_{\max}}{6} T^3 + 2v_0 T - 2s = 0. \tag{37}$$

Eq. (37) is a cubic equation, the T can be obtained by solving this equation. The velocity corresponding to the end point can then be obtained:

$$\boldsymbol{v}_{\text{limit}}(i) = \min\left(\boldsymbol{v}_{\text{limit}}(i), c_1 + 2c_2 T + 3c_3 T^2 + 4c_4 T^3\right). \tag{38}$$

3.3 Double-quartic Polynomial Trajectory Planning

The proposed double-quartic polynomial trajectory planning is similar to quartic polynomial trajectory planning. The differences are that this method may have a constant velocity period and consists of two quartic polynomial trajectory planning, which improves the efficiency for machining long line segments. Double-quartic polynomial trajectory planning has seven forms, as shown in Figure 3. It includes all the advantages of quartic polynomial trajectory planning, except that it is more computationally intensive than quartic polynomial trajectory planning.

The flowchart of the double-quartic polynomial trajectory planning is shown in Figure 4. If $v_{\text{start}} = v_{\text{end}} = v_{\max}$, then there exists only a constant velocity period. Otherwise, calculate the length of the current path (s), s_{vm} the length of displacement from v_{start} to $v_{\max}$ plus the length of displacement from $v_{\max}$ to v_{end}, and s_{se} the length of displacement from v_{start} to v_{end}. If $s \geq s_{\text{vm}}$, three cases are possible, each of which includes two of the three of acceleration, deceleration, and constant velocity. Otherwise, there are three cases, including two that can be achieved by quartic polynomial trajectory planning, and the last

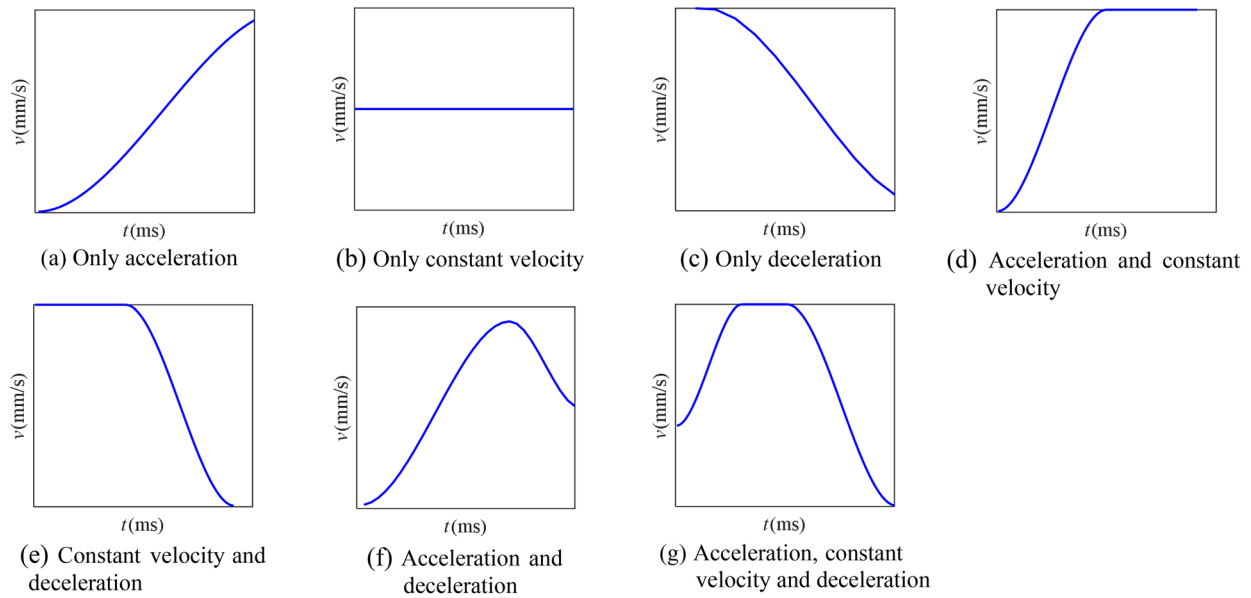


Figure 3 Velocity classification of double-quartic trajectory planning

case that requires double-quartic polynomial trajectory planning.

When the double-quartic polynomial trajectory planning is determined and the velocity cannot reach $v_{\max}$, the maximum velocity $v'_{\max}$ reached during running can be calculated using the algorithm (Table 1).

Based on the relationships between v_{start}, v_{end}, $v_{\max}$, and s, the double-quartic polynomial trajectory planning was constructed, and double-quartic polynomial trajectory planning was achieved using the proposed algorithm.

4 Simulation and Experimental Results

The proposed path-smoothing technique and bidirectional look-ahead algorithm were tested through simulations and experiments. First, the proposed method was compared with the B spline and the Blend methods on a regular path in the simulation. Subsequently, an experiment on the drawing horse was conducted using a SCARA robot.

4.1 Simulation Results

For a regular path such as a semicircular outline, the velocity of the look-ahead method gradually increases until it reaches the maximum velocity at the midpoint of the path, and then decreases to 0. The velocity in the Blend method frequently accelerates and decelerates [32]. The velocity of the B spline method [33] may be similar to the velocity of the look-ahead method, but may be less smooth. A simulation was designed to investigate our approach and the proposed algorithms. A regular path with a semicircular outline was selected and divided into 150 line segments. The average length of these small line segments was approximately 1 mm, as shown in Figure 5.

For these three methods, the limitations of velocity, acceleration, and jerk were set to 2×10^3mm/s, 4×10^4mm/s^2, and 1.8×10^7mm/s^3, respectively. The number of look-ahead segments was set to 150, and the blend ratio of the Blend method was set to 50%. The trajectory points, velocity, acceleration, and jerk in the simulation were recorded; the comparison curves are shown in Figures 6–8, respectively.

As shown in Figure 6, the maximum velocity of the look-ahead method is significantly larger than that of the Blend method for a regular path but smaller than that of the B spline method. B spline trajectory planning is the fastest because its trajectory is an entire curve, which is easier to accelerate. However, it is only suitable for offline machining because it uses optimization algorithms, which are computationally intensive and may have no solution. The velocity of look-ahead accelerates to the maximum gradually and then decelerates to 0 gradually. However, the velocity of the Blend method oscillates around a certain value, the maximum value of the Blend method is significantly lower than that of the proposed method. The velocity of the B spline method accelerates to the maximum speed, then is equivalent to a constant speed, and finally decelerates to 0.

As shown in Figures 7 and 8, the acceleration and jerk of the Blend method oscillate around the X-axis, and its maximum acceleration and jerk are significantly larger than those of the proposed method. In addition, the acceleration and jerk of the B spline method sometimes

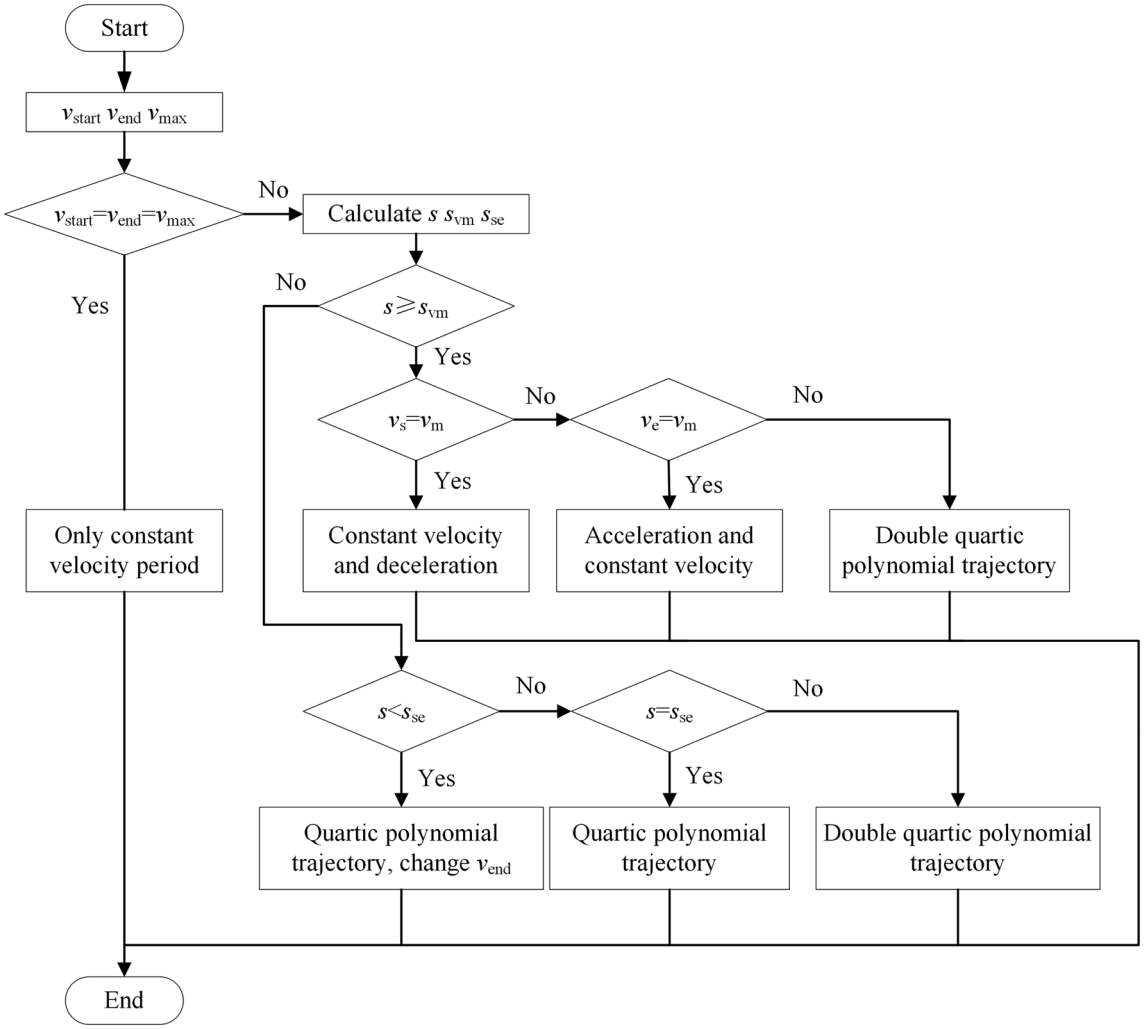


Figure 4 Flowchart of double-quartic polynomial trajectory planning

change sharply and their maximum values are larger than those of the look-ahead method. This simulation proves that the operation of the algorithm satisfies our requirements, verifying the effectiveness of the proposed algorithm.

Table 1 Pseudocode of double-quartic polynomial trajectory planning

Pseudocode
$s_1 = 0$
$v'_{max} = v_{end}$
$i = 0$
While $i < n$
$s_1 = s_{v_{start} \to v'_{max}} + s_{v_{end} \to v'_{max}}$
$v'_{max} = v_{s=\frac{s-s_1}{2}}$
end

4.2 Experimental Results

In the actual experiment, a SCARA robot manufactured by RobotPhoenix Automation Technology Co., Ltd. was

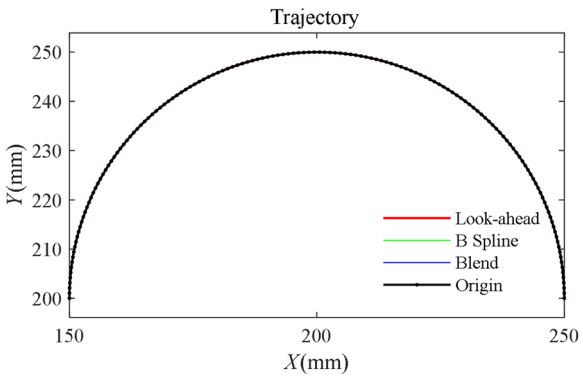


Figure 5 Comparison of semicircular trajectories for the three methods

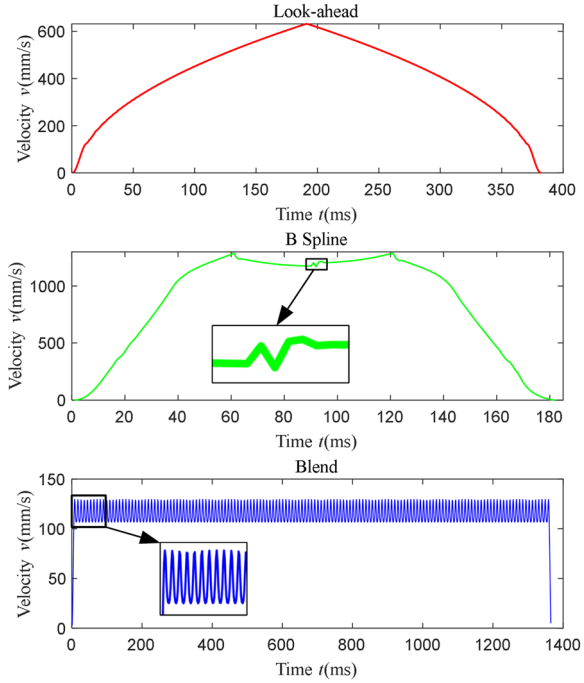


Figure 6 Comparison of velocity values for the three methods

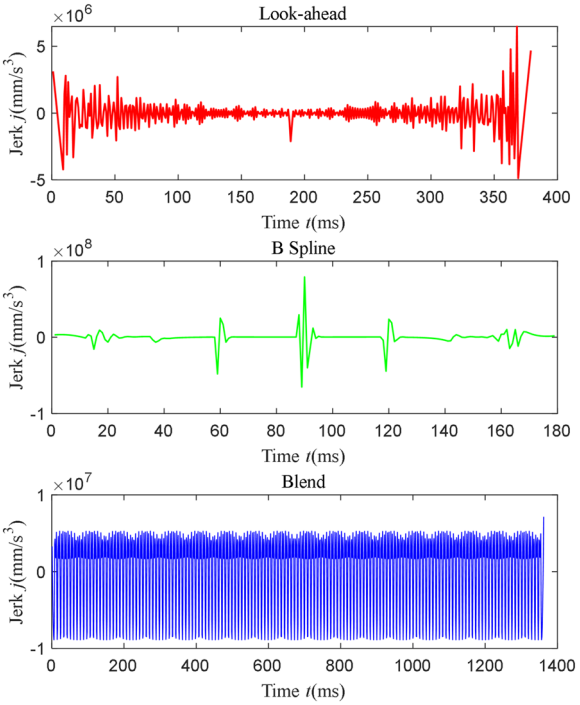


Figure 8 Comparison of jerk values for the three methods

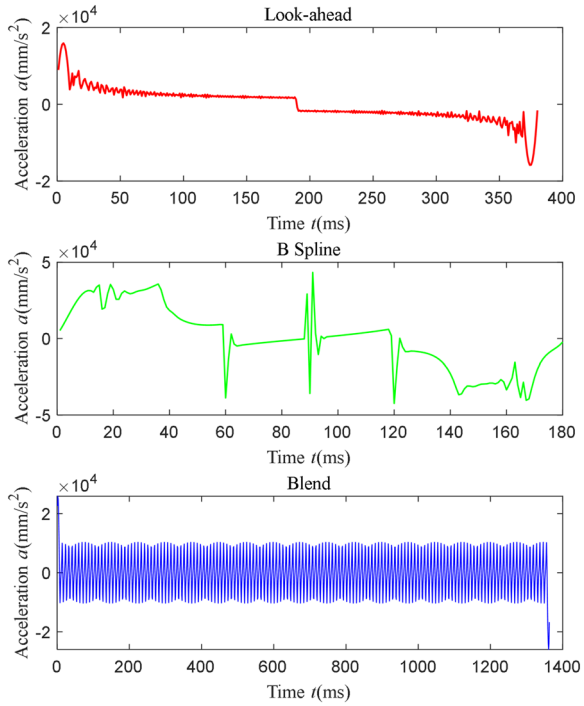


Figure 7 Comparison of acceleration values for the three methods

adopted. A self-developed Gorilla controller was utilized to test the machining ability of the proposed algorithm for a path that consisting of many small continuous line segments. The central processing unit of the Gorilla controller was I5-8500T, and it controlled the motion of the robot using CODESYS Control RTE, which is a real-time software PLC for PC-based industrial controllers in Windows programmable with the IEC 61131-3 development system, CODESYS.

For a horse outline consisting of 760 linear segments with an average length of approximately 1 mm, the robot drew a horse using the algorithm proposed in this study, as shown in Figure 9. As shown in Figure 10, the robot with a pen at the end, the proposed method, and the Blend algorithm were used to draw a picture of the horse. Figure 10(a) and (b) shows that the horse drawn by the look-ahead algorithm is smoother than that drawn by the Blend algorithm. Blend method dithering occurs because the soft tip cannot keep up with the movement of the robot in the case of frequent acceleration and deceleration, demonstrating the effectiveness of the proposed algorithm. The strands in Figure 10(a) do not maintain consistent strand widths because the platform where the paper is located is not parallel to the XY plane of the robot base frame.

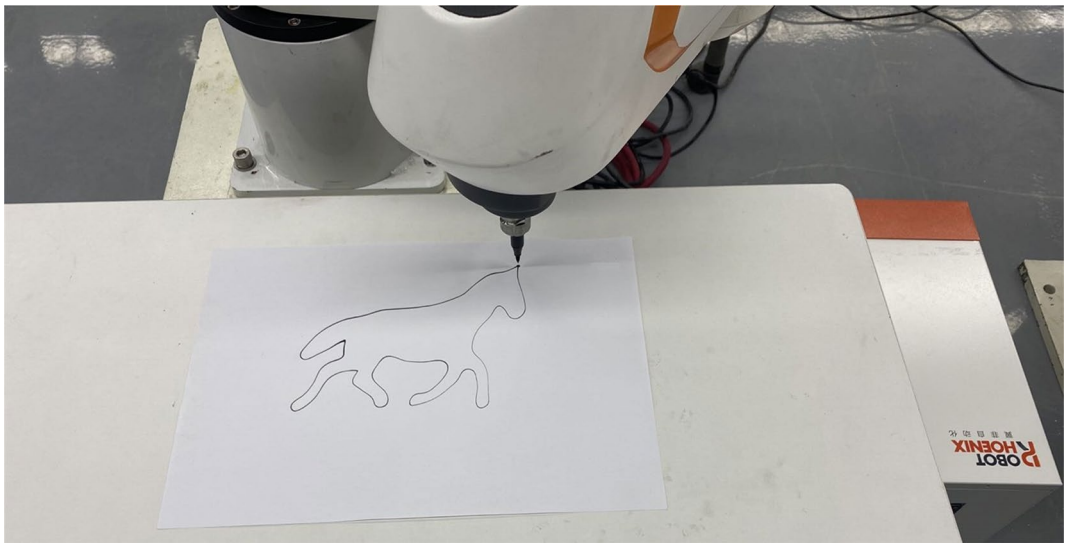
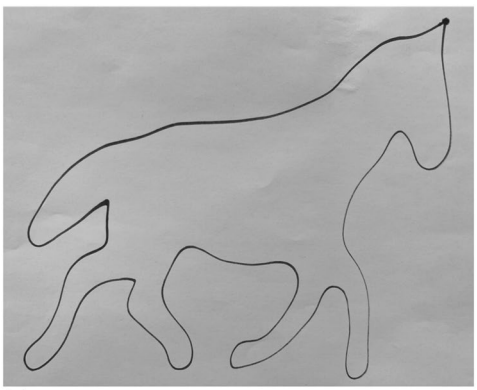
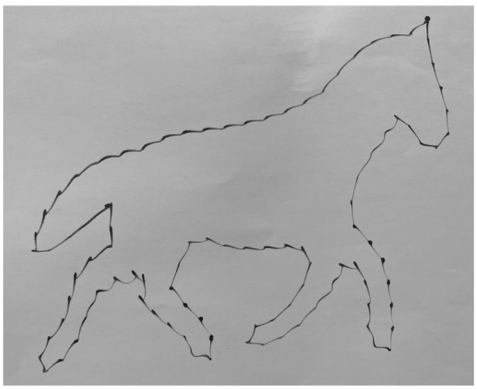


Figure 9 Robot drawing a horse



(a) Path of proposed method



(b) Path of Blend method

Figure 10 Comparison of paths of proposed and Blend method

The running time of the Blend method and B spline method are equal to that of the proposed method. During the period when the robot drew the picture of a horse, the trajectory points, velocity, and acceleration of the proposed method, B spline method and Blend method were recorded. The trajectory points are shown in Figure 11. Comparisons of the velocity, acceleration,

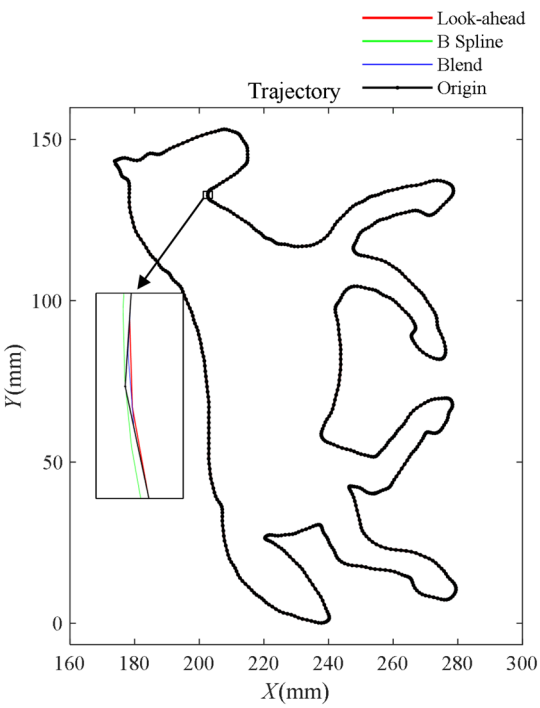


Figure 11 Comparison of trajectory paths plotted using the three methods

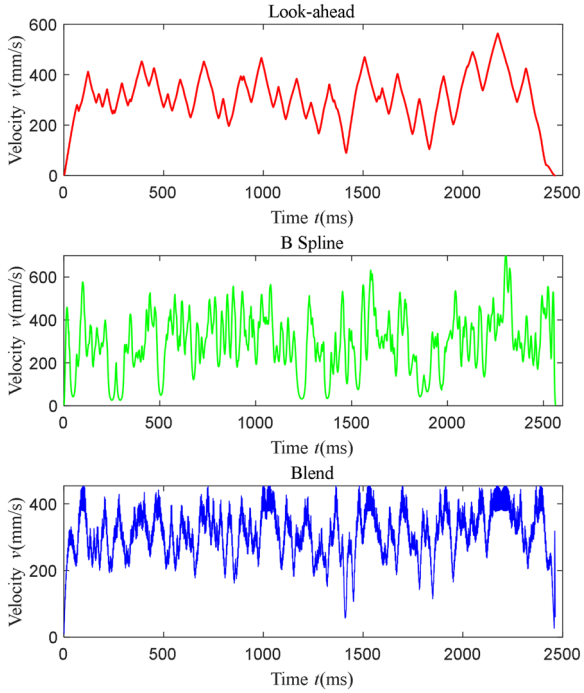


Figure 12 Comparison of velocity values using the three methods

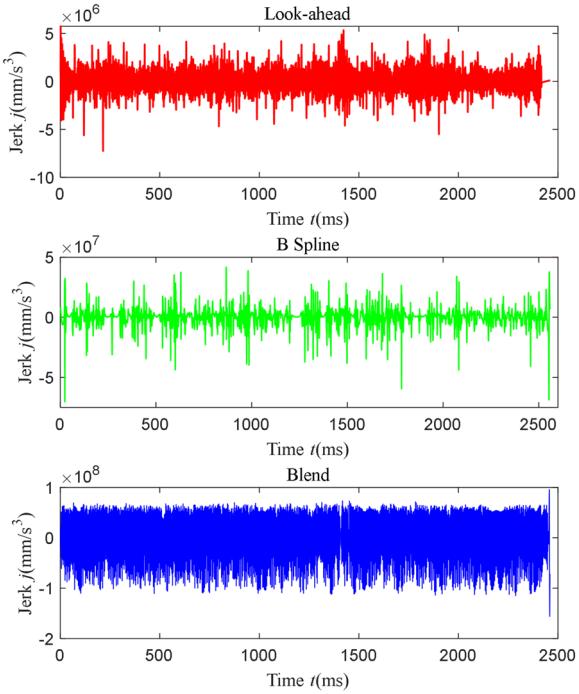


Figure 14 Comparison of jerk values obtained using the three methods

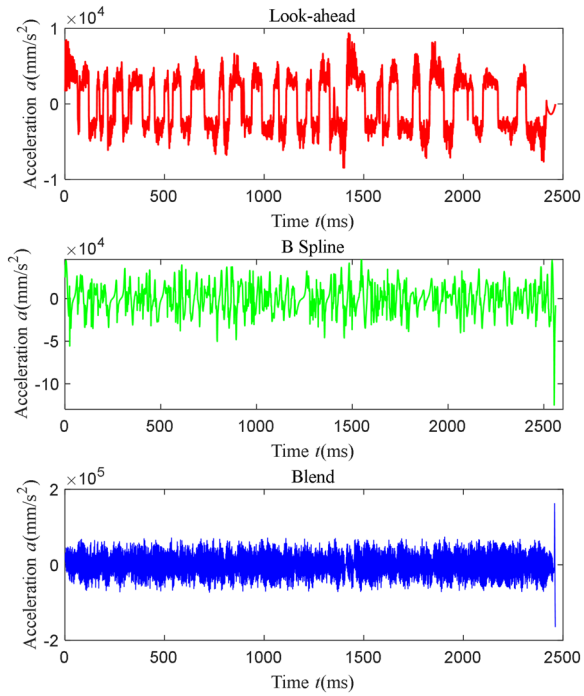


Figure 13 Comparison of acceleration values obtained using the three methods

and jerk of the three methods are shown in Figures 12–14, respectively.

Figure 12 shows the order of the velocity curve compliance as the look-ahead, B spline, and Blend methods. The acceleration and jerk of the Blend method are significantly larger than those of the B spline and look-ahead methods(Figures 13 and 14), proving the efficiency of the proposed methods. The time required to calculate the look-ahead algorithm with respect to this path was 24.016 ms when looking ahead 20 linear segments at a time, and approximately 0.032 ms were required for each linear segment in the look-ahead module.

5 Conclusions

A practical path-smoothing method and look-ahead algorithm are presented in this study. A Quintic Bezier curve was used to blend the line segments as continuous curvature transitions. A bidirectional scanning algorithm and quartic polynomial trajectory planning algorithm can decrease the total running time and minimize vibration. The simulation and experimental results verified the effectiveness of this method for robot machining of multiple small line segments.

Compared with previous methods, the algorithms proposed in this study have advantages:

(1) Realization of G^2 continuity, which mixes line segments and curves.

(2) The bidirectional scanning algorithm efficiently reduces the time spent looking forward and backward, thus achieving the look-ahead function in real time.

(3) The quartic polynomial trajectory planning algorithm can reduce the vibration of the robot at the start and stop.

(4) Compensation of the interpolation algorithm can eliminate truncation error and further enhance movement smoothness.

Acknowledgements
Not applicable.

Author contributions
XL was in charge of the entire trial; SZ wrote and modified the manuscript; BY carried out the simulation and experiment; JB modified the manuscript and assisted with the experiment; XH conducted the experiment and data analysis. All authors read and approved the final manuscript.

Authors' Information
Sai Zhang, born in 1983, is currently a PhD candidate at *DME*, *Tsinghua University*, *China*.
Xinjun Liu, born in 1971, is currently a professor and a Ph.D. candidate supervisor at *DME*, *Tsinghua University*, *China*. His research interests include robotics, parallel mechanisms, and advanced manufacturing equipment. Tel: +86-10-62796573.
Bingkai Yan, born in 1987, is a production manager at *Robotphoenix Automation Technology Co., Ltd. China*.
Jie Bi, born in 1995, is a robotics control engineer at *Robotphoenix Automation Technology Co., Ltd. China*.
Xiangdong Han, born in 1994, is a robotics control engineer at *Robotphoenix Automation Technology Co., Ltd. China*.

Funding
Supported by National Natural Science Foundation of China (Grant No. 92148301)

Data availability
The datasets used and analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing Interests
The authors declare no competing financial interests.

Received: 19 June 2022 Revised: 28 March 2023 Accepted: 3 April 2023
Published online: 28 April 2023

References

[1] J Yang, A Yuen. An analytical local corner smoothing algorithm for five-axis CNC machining. *International Journal of Machine Tools and Manufacture*, 2017, 123: 22-35.
[2] A I Obukhov, L I Martinova, A B Lyubimov. Developing of the look ahead algorithm for linear and nonlinear laws of control of feedrate in CNC. *Automation and Remote Control*, 2020, 81(2): 380-386.
[3] H Zhao, L M Zhu, H Ding. A real-time look-ahead interpolation methodology with curvature-continuous B-spline transition scheme for CNC machining of short line segments. *International Journal of Machine Tools and Manufacture*, 2013, 65: 88-98.
[4] D N Song, Y G Zhong, J W Ma. Look-ahead-window-based interval adaptive feedrate scheduling for long five-axis spline toolpaths under axial drive constraints. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 2020, 234(13): 1656-1670.
[5] Y Zhang, T Wu, C Li, et al. Investigation on path optimization and look-ahead speed control algorithm during numerical control grinding of dentures of glass ceramics. *International Journal of Advanced Manufacturing Technology*, 2021, 113(7): 1899-1913.
[6] B Xu, Y Ding, W Ji. An interpolation method based on adaptive smooth feedrate scheduling and parameter increment compensation for NURBS curve. *ISA Transactions*, 2022, 128: 633-645.
[7] S J Ji, L G Lei, J Zhao, et al. An adaptive real-time NURBS curve interpolation for 4-axis polishing machine. *Robotics and Computer-Integrated Manufacturing*, 2021, 67: 102025.
[8] J Li, Y Liu, Y Li, et al. S-model speed planning of NURBS curve based on uniaxial performance limitation. *IEEE Access*, 2019, 7: 60837-60849.
[9] Z Yang, L Shen, C Yuan, et al. Curve fitting and optimal interpolation for CNC machining under confined error using quadratic B-splines. *Computer-Aided Design*, 2015, 66: 62-72.
[10] Y Jin, S Zhao, Y Wang. An optimal feed interpolator based on G2 continuous Bézier curves for high-speed machining of linear tool path. *Chinese Journal of Mechanical Engineering*, 2019, 32: 43.
[11] L Xu, M Y Cao, B Y Song. A new approach to smooth path planning of mobile robot based on quartic Bezier transition curve and improved PSO algorithm. *Neurocomputing*, 2022, 473: 98-106.
[12] M K Jouaneh, Z Wang, D A Dornfeld. Trajectory planning for coordinated motion of a robot and a positioning table. *IEEE Transactions on Robotics and Automation*, 1990, 6(6): 735-745.
[13] V Pateloup, E Duc, P Ray. B-spline approximation of circle arc and straight line for pocket machining. *Computer-Aided Design*, 2010, 42(9): 817-827.
[14] L Zhang, Y You, J He, et al. The transition algorithm based on parametric spline curve for high-speed machining of continuous short line segments. *The International Journal of Advanced Manufacturing Technology*, 2011, 52: 245-254.
[15] D J Walton, D S Meek. G2 blends of linear segments with cubics and Pythagorean-hodograph quintics. *International Journal of Computer Mathematics*, 2009, 86(9): 1498-1511.
[16] Q Bi, Y Wang, L Zhu, et al. A practical continuous-curvature Bezier transition algorithm for high-speed machining of linear tool path. *Intelligent Robotics and Applications*, Aachen, Germany, December 6-8, 2011: 465-476.
[17] X Beudaert, S Lavernhe, C Tournier. Feedrate interpolation with axis jerk constraints on 5-axis NURBS and G1 tool path. *International Journal of Machine Tools and Manufacture*, 2012, 57(3): 73-82.
[18] T Zhao, B Zi, S Qian, et al. Algebraic method-based point-to-point trajectory planning of an under-constrained cable-suspended parallel robot with variable angle and height cable mast. *Chinese Journal of Mechanical Engineering*, 2020, 33: 54.
[19] Y Lang, D Yu, W Han, et al. An efficient motion trajectory planning method in CNC system. *15th IEEE Conference on Industrial Electronics and Applications*, Kristiansand, Norway, November 9-13, 2020: 272-277.
[20] H D Liu, X C Xi, W Liang, et al. A look-ahead transition algorithm for jump motions with short line segments in EDM. *The International Journal of Advanced Manufacturing Technology*, 2018, 95(1): 1409-1419.
[21] X Liu, J Peng, L Si. A novel approach for NURBS interpolation through the integration of ACC-jerk-continuous-based control method and look-ahead algorithm. *The International Journal of Advanced Manufacturing Technology*, 2017, 88(1): 961-969.
[22] H Ni, J Yuan, S Ji, et al. Feedrate Scheduling of NURBS interpolation based on a novel jerk-continuous ACC/DEC algorithm. *IEEE Access*, 2018, 6: 66403-66417.
[23] P F Xiao, H H Ju, Q D Li. Point-to-point trajectory planning for space robots based on jerk constraints. *Review of Scientific Instruments*, 2021, 92(9): 094501.
[24] Q Chen, C Zhang, H Ni, et al. Trajectory planning method of robot sorting system based on S-shaped acceleration/deceleration algorithm. *International Journal of Advanced Robotic Systems*, 2018, 15(6): 1729881418813805.

[25] H Ni, T Hu, C Zhang, et al. An optimized federate scheduling method for CNC machining with round-off error compensation. *The International Journal of Advanced Manufacturing Technology*, 2018, 97(5): 2369-2381.
[26] X Guo, S Wu, Q Wang, et al. Trajectory planning of multi-dimensional actuator based on improved s-shaped velocity curve. *Electrotehnica, Electronica, Automatica*, 2020, 68: 66-73.
[27] J Han, Y Jiang, X Q Tian, et al. A local smoothing interpolation method for short line segments to realize continuous motion of tool axis acceleration. *International Journal of Advanced Manufacturing Technology*, 2018, 95(5): 1729-1742.
[28] S Timar, R Farouki, T Smith, et al. Algorithms for time-optimal control of CNC machines along curved tool paths. *Robotics and Computer-Integrated Manufacturing*, 2005, 21(1): 37-53.
[29] Y Zhang, M Y Zhao, P Q Ye, et al. A G4 continuous B-spline transition algorithm for CNC machining with jerk-smooth feedrate scheduling along linear segments. *Computer-Aided Design*, 2019(115): 231-243.
[30] C L Yan, D S Du, C X Li. Design of a real-time adaptive interpolator with parameter compensation. *The International Journal of Advanced Manufacturing Technology*, 2007, 35(1): 169-178.
[31] B Sencer, K Ishizaki, E Shamoto. A curvature optimal sharp corner smoothing algorithm for high-speed feed motion generation of NC systems along linear tool paths. *The International Journal of Advanced Manufacturing Technology*, 2015, 76(9): 1977-1992.
[32] S Tajima, B Sencer. Accurate real-time interpolation of 5-axis tool paths-with local corner smoothing. *International Journal of Machine Tools and Manufacture*, 2019, 142: 1-15.
[33] W Fan, X S Gao, C H Lee, et al. Time-optimal interpolation for five-axis CNC machining along parametric tool path based on linear programming. *The International Journal of Advanced Manufacturing Technology*, 2013, 69(5): 1373-1388.