applied sciences

MDPI

Article

A Trajectory Planning Method for Polishing Optical Elements Based on a Non-Uniform Rational B-Spline Curve

Dong Zhao and Hao Guo *

Key Laboratory of Mechanism Theory and Equipment Design of The State Ministry of Education, Tianjin University, Tianjin 300072, China; dongzhao@tju.edu.cn

* Correspondence: guohaohalo@163.com; Tel.: +86-156-2090-9866

Received: 10 July 2018; Accepted: 10 August 2018; Published: 12 August 2018

Abstract: Optical polishing can accurately correct the surface error through controlling the dwell time of the polishing tool on the element surface. Thus, the precision of the trajectory and the dwell time (the runtime of the trajectory) are important factors affecting the polishing quality. This study introduces a systematic interpolation method for optical polishing using a non-uniform rational B-spline (NURBS). A numerical method for solving all the control points of NURBS was proposed with the help of a successive over relaxation (SOR) iterative theory, to overcome the problem of large computation. Then, an optimisation algorithm was applied to smooth the NURBS by taking the shear jerk as the evaluation index. Finally, a trajectory interpolation scheme was investigated for guaranteeing the precision of the trajectory runtime. The experiments on a prototype showed that, compared to the linear interpolation method, there was an order of magnitude improvement in interpolation, and runtime, errors. Correspondingly, the convergence rate of the surface error of elements improved from 37.59% to 44.44%.

Keywords: hybrid robot; curve fitting; fair optimisation; trajectory interpolation

1. Introduction

With the rapid development of astronomy, space exploration, and advanced optical instruments, optical elements are being increasingly widely used. The application demands for high-quality and high-efficiency elements present distinct higher requirements for the process technology used in such elements [1]. Computer-controlled optical surfacing (CCOS) has been successfully applied in industrial production, as it can precisely correct the surface error by converting the dwell time of the polishing tool into the feed-rate along the polishing trajectory. Therefore, a trajectory planning method is the key factor affecting high-quality, high-efficiency polishing.

Despite the extent of research on path planning in some fields [2–4], investigation of optical polishing has been rather limited. Although the problem of discontinuous surfaces between the adjacent mm-sized short line segments has attracted wide concern when using parametric curve theory [5–7], frequent acceleration and deceleration result in poor realisation precision of dwell time, i.e., the runtime of trajectory (hereafter referred to as runtime), which further influences the polishing quality of elements and the convergence rate of surfaces [8]. The main focus of this paper is to investigate an interpolation scheme to overcome the above problem, which has two main progressive aspects: fitting and interpolation of parametric curves.

The fitting of parametric curves is a process of converting discrete short line segments into parametric curves. At present, many scholars have carried out research based on the dominant points using a non-uniform rational B-spline (NURBS). Park [9,10] proposed a method for determining the

dominant points according to the discrete curvature. Zhou [11] and Xu [12] improved this method by taking concave-convex turning points and extreme points on the curvature curve as dominant points. To improve the fitting precision, Zhao [13] proposed curve fitting taking squared distance minimisation (SDM) as the evaluation index. Although the dominant points based method is easy with regard to calculation and interpolation, it may result in the loss of runtime at non-dominant points. Thus, only the global fitting method is suitable to the optical polishing. To do so, Yang [14] proposed an optimisation algorithm by establishing the evaluation function for deviation of the fitted distance. Li [15] and Lin [16] classified the trajectory into the different forms of NURBS and then employed a piecewise fitting method for real-time implementation. Based on Gaussian elimination and the continuous short block (CSB) look-ahead algorithm, Tsai [17] and Wang [18] realised the on-line transformation from short line segments to NURBS: however, for the global fitting method, the simplified strategy that setting all the weight factors as 1 eliminates the regulating effect on the fairness of curves and easily causes curvature saltation on the trajectory. Therefore, generating a fair trajectory based on NURBS is the first problem facing the polishing trajectory planning technique.

The interpolation of parametric curves is a process that discretises the NURBS into numerical control (NC) commands. Speed planning, as the critical step in the interpolation process, has been an area of research for numerous scholars: this can be classified into the time-optimal approach and the non-time-optimal approach. The time-optimal approach deals with the planning problem by taking the minimisation of the motion time as the objective to promote manufacturing efficiency [19,20]. For example, Timar [21] proposed a speed planning scheme for NC interpolation by the use of the optimal control theory. On this basis, Sencer [22] and Lu [23] considered the driving capacity constraint and trajectory precision constraint in the interpolation, respectively. The non-time-optimal approach usually deals with the planning problem by taking the minimisation of speed fluctuations as the objective [24]. Various methods, such as the feed-rate evolutionary algorithm [25], the equidistance quaternion method [26], and the improved Adams-Malton algorithm [27] are employed to decrease speed fluctuations and guarantee steady, continuous-trajectory operation. Although the effectiveness of the method has been validated experimentally, is cannot be applied directly to optical polishing because the effect of acceleration and deceleration on the realisation precision of runtime is not considered.

Driven by the practical needs to improve the quality of optical polishing, this paper presents a systematic trajectory planning method that particularly enhances the realisation precision of runtime. Following this introduction, Section 2 calculates all the control points of NURBS through numerical solution to overcome the problem of calculation efficiency. In Section 3, an optimisation algorithm of fairing NURBS is established by taking the shear jerk of trajectory as an evaluation index. Section 4 then proposes an interpolation scheme with which to minimise the realisation error of runtime by planning the feed-rate of trajectory according to the given runtime between adjacent NC codes. Section 5 reports experiments on a prototype machine which shows that the proposed trajectory planning method is more accurate than the linear interpolation method. Conclusions are drawn in Section 6.

2. Trajectory Fitting Based on the NURBS Curve

In this section, according to the given NC codes, all the control points of NURBS are solved, which provides the necessary mathematical model for the interpolation of parametric curves. The basic settings are as follows:

(1) Considering the stability, ease of use and calculation efficiency, cubic NURBS is employed as the fitting tool. (2) The uniform parametric method is used for trajectory fitting, because the polishing elements have a large radius of curvature and the chord lengths between NC codes are distributed uniformly. (3) All weight factors are set to 1. In this case, the NURBS can be treated as a cubic B-spline to simplify the calculation process.

According to the basic theory of the NURBS, a segment of NURBS $\boldsymbol{C}(u)(0 \leq u \leq 1)$ can be determined based on the four adjacent control points $\boldsymbol{d}_k(k = 0, 1, 2, 3)$. As shown in Figure 1, $\boldsymbol{C}(0)$

and $\boldsymbol{C}(1)$ separately refer to the start point and the end point of the NURBS segment. Based on the aforementioned setting, the NURBS segment can be directly written as:

$$\boldsymbol{C}(u) = \frac{1}{6}\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{d}_0 \\ \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \\ \boldsymbol{d}_3 \end{bmatrix} \tag{1}$$

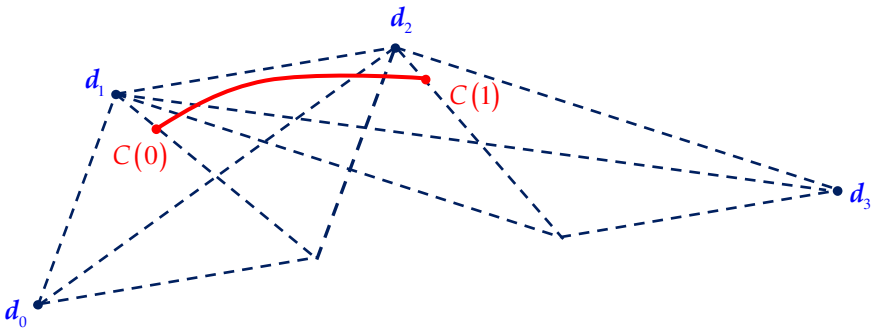

Figure 1. Schematic diagram of the parametric curve.

Thus, point $\boldsymbol{C}_i$ can be expressed as:

$$\boldsymbol{C}_i = \frac{1}{6}(\boldsymbol{d}_{i-1} + 4\boldsymbol{d}_i + \boldsymbol{d}_{i+1}) \tag{2}$$

where $\boldsymbol{C}_i$ denotes the ith NC code in the $n+1$ lines of NC codes and also is taken as the start point of the ith NURBS segment. $\boldsymbol{d}_{i-1}, \boldsymbol{d}_i, \boldsymbol{d}_{i+1}$ respectively denote the three control points corresponding to $\boldsymbol{C}_i$. To calculate $\boldsymbol{C}_1$ and $\boldsymbol{C}_{n+1}$, it is defined that on the condition that $i < 1$, then $\boldsymbol{d}_i = \boldsymbol{d}_1$ and on the condition that $i > n+1$, then $\boldsymbol{d}_i = \boldsymbol{d}_{n+1}$. Equations (3) and (4) can thus be obtained:

$$\boldsymbol{C}_1 = \frac{1}{6}(\boldsymbol{d}_1 + 4\boldsymbol{d}_1 + \boldsymbol{d}_2) = \frac{5}{6}\boldsymbol{d}_1 + \frac{1}{6}\boldsymbol{d}_2 \tag{3}$$

$$\boldsymbol{C}_{n+1} = \frac{1}{6}(\boldsymbol{d}_n + 4\boldsymbol{d}_{n+1} + \boldsymbol{d}_{n+1}) = \frac{1}{6}\boldsymbol{d}_n + \frac{5}{6}\boldsymbol{d}_{n+1} \tag{4}$$

Rewriting Equations (2)–(4) in matrix notation yields:

$$\boldsymbol{A}\boldsymbol{d} = \boldsymbol{C}$$

$$\boldsymbol{A} = \begin{bmatrix} 5 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 4 & 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 5 \end{bmatrix}, \boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \\ \boldsymbol{d}_3 \\ \vdots \\ \boldsymbol{d}_n \\ \boldsymbol{d}_{n+1} \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 6\boldsymbol{C}_1 \\ 6\boldsymbol{C}_2 \\ 6\boldsymbol{C}_3 \\ \vdots \\ 6\boldsymbol{C}_n \\ 6\boldsymbol{C}_{n+1} \end{bmatrix} \tag{5}$$

If the traditional solution methods such as Gauss elimination method and LU decomposition are directly applied to the Equation (5), it may lead to some undesirable phenomena (such as excessive calculation time) due to the large quantity of NC codes for polishing. Hence, the SOR iterative algorithm [28] is used to find stable numerical solutions of Equation (5) as described below.

The coefficient matrix $\boldsymbol{A}$ can be divided into:

$$\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U} \tag{6}$$

where $\boldsymbol{A} = \mathrm{diag}(a_{11}, a_{22}, \ldots, a_{n+1n+1})$, $-\boldsymbol{L}$ denotes a strictly lower triangular matrix, whose elements below the principal diagonal are corresponding elements of $\boldsymbol{A}$. $-\boldsymbol{U}$ denotes a strictly upper triangular matrix, whose elements above the principal diagonal are corresponding elements of $\boldsymbol{A}$. Owing to the matrix $\boldsymbol{D}$ being invertible, Equation (5) can be modified to:

$$\boldsymbol{d} = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})\boldsymbol{d} + \boldsymbol{D}^{-1}\boldsymbol{C} \tag{7}$$

In this case, corner marks (k) and $(k+1)$ are added in Equation (7) to identify the number of iterations. Then, the elementary iterative scheme of $\boldsymbol{d}$ can be written as:

$$\boldsymbol{d}^{(k+1)} = \boldsymbol{M}_1\boldsymbol{d}^{(k)} + \boldsymbol{D}^{-1}\boldsymbol{C} \tag{8}$$

where $\boldsymbol{M}_1 = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})$. The component form of Equation (8) can be expressed as:

$$\boldsymbol{d}_i^{(k+1)} = \frac{1}{a_{ii}}\left(- \sum_{\substack{j=1 \\ j \neq i}}^{n+1} a_{ij}\boldsymbol{d}_j^{(k)} + \boldsymbol{C}_i \right),\ i = 1, 2, \ldots, n+1 \tag{9}$$

which is also known as the Jacobi iteration scheme, noting that, before calculating $\boldsymbol{d}_i^{(k+1)}$, the iterative values of the first $i-1$ components in $\boldsymbol{d}^{(k+1)}$ have been generated, which are more approximate to the true value than the results obtained by the previous iteration. Therefore, $\boldsymbol{d}_1^{(k)}, \boldsymbol{d}_2^{(k)}, \ldots, \boldsymbol{d}_{i-1}^{(k)}$ can be replaced by $\boldsymbol{d}_1^{(k+1)}, \boldsymbol{d}_2^{(k+1)}, \ldots, \boldsymbol{d}_{i-1}^{(k+1)}$ to make $\boldsymbol{d}_i^{(k+1)}$ closer to the true value. To improve the convergence rate of the iteration further, a proper parameter μ is selected to conduct the weighted averaging on the aforementioned iterative scheme:

$$\boldsymbol{d}_i^{(k+1)} = \mu\widetilde{\boldsymbol{d}}_i^{(k+1)} + (1-\mu)\boldsymbol{d}_i^{(k)}, i = 1, 2, \ldots n+1 \tag{10}$$

with:

$$\widetilde{\boldsymbol{d}}_i^{(k+1)} = \frac{1}{a_{ii}}\left(-\sum_{j=1}^{i-1} a_{ij}\boldsymbol{d}_j^{(k+1)} - \sum_{j=i+1}^{n+1} a_{ij}\boldsymbol{d}_j^{(k)} + \boldsymbol{C}_i \right) \tag{11}$$

Substituting Equation (11) into Equation (10) leads to the SOR iteration scheme:

$$a_{ii}\boldsymbol{d}_i^{(k+1)} + \mu\sum_{j=1}^{i-1} a_{ij}\boldsymbol{d}_j^{(k+1)} = a_{ii}\boldsymbol{d}_i^{(k)} - \mu\sum_{j=i}^{n+1} a_{ij}\boldsymbol{d}_j^{(k)} + \mu\boldsymbol{C}_i\,,\ i = 1, 2, \ldots, n+1 \tag{12}$$

Note that $\boldsymbol{A}$ is a tridiagonal positive definite matrix, so the optimal value μ_{opt} of the relaxation factor can be expressed as [29]:

$$\mu_{opt} = \frac{2}{1 + \sqrt{1 - [\rho(\boldsymbol{M}_1)]^2}} \tag{13}$$

where $\rho(\boldsymbol{M}_1)$ denotes the spectral radius of $\boldsymbol{M}_1$.

It is worth noting that, for a flat element, the aforementioned method can be directly applied to calculate the NURBS trajectory, but for a curved element, the polishing shaft should always lie along the normal direction of the element. In this case, it is necessary to solve two trajectories of end-point and reference-point on the polishing shaft to determine the polishing attitude (for more details, please see [21]).

3. Fairing of the NURBS

Although the constructed cubic NURBS by the above method satisfies the G2 continuity characteristics, i.e., the second-order derivative functions of the trajectory is continuous, the motion stability of the trajectory is still influenced by curvature saltation. In this section, a fairing optimisation method is proposed by adjusting the weight factors of NURBS.

The fairing optimisation can be classified into two categories, i.e., global and local fairing according to the number of the adjusted point on NURBS. Considering the significant computational burden of global fairing caused by the large quantity of NC codes for polishing, it is more reasonable to modify the outlier points with curvature saltation by the use of local fairing, which are selected from all NC codes [30]. A filtering process is needed to eliminate the influence of curvature fluctuations caused by discrete calculation.

In the fairing optimisation, the shear jerk of the outlier point is taken as the evaluation index:

$$D_j = \frac{\kappa_{next1} - \kappa_j}{\|\boldsymbol{C}_{next1} - \boldsymbol{C}_j\|} - \frac{\kappa_j - \kappa_{last1}}{\|\boldsymbol{C}_j - \boldsymbol{C}_{last1}\|} \tag{14}$$

where κ_j denotes the curvature at the jth outlier point $\boldsymbol{C}_j$, κ_{next1} and κ_{last1} denote the curvature of the $\boldsymbol{C}_{next1}$ and $\boldsymbol{C}_{last1}$, which are on two adjacent sides of the outlier point. The index indicates the curvature changes of the adjacent outlier points.

To generate the weight factors of the various outlier points, the objective function is defined as:

$$L = \sum_{j=1}^{k} D_j^2 + \left(\boldsymbol{C}_j - \boldsymbol{C}_{j,0}\right)^2 \tag{15}$$

where $\boldsymbol{C}_{j,0}$ denotes the jth outlier point before optimisation. It can be seen from Equation (15) that the objective function is composed of two parts: part one is the curvature changes of the outlier point after optimisation, and part two is the adjustment amplitude of outlier points before, and after, optimisation. Thus, the objective function means that fairing optimisation is performed on the premise of modifying the NURBS as little as possible.

According to affine invariant principle of NURBS [31], the four-dimensional (4D) space constructed by the control points and weight factors can be expressed as:

$$\boldsymbol{C}^{\omega}(u) = \sum_{i=1}^{n+1} N_{i,3}(u) \begin{bmatrix} \omega_i \boldsymbol{d}_i \\ \omega_i \end{bmatrix} = \sum_{i=1}^{n+1} N_{i,3}(u) \boldsymbol{d}_i^{\omega} \tag{16}$$

Then, the NURBS defined by Equation (1) can be regarded as the projection of the curve $\boldsymbol{C}^{\omega}(u)$ in 4D space on the centre of the hyperplane $\omega = 1$. Based on Equation (2), it can be seen that:

$$\boldsymbol{C}_j^{\omega} = \frac{1}{6}\boldsymbol{d}_{j,last1}^{\omega} + \frac{2}{3}\boldsymbol{d}_j^{\omega} + \frac{1}{6}\boldsymbol{d}_{j,next1}^{\omega} \tag{17}$$

$$\boldsymbol{C}_{j,last1}^{\omega} = \frac{1}{6}\boldsymbol{d}_{j,last2}^{\omega} + \frac{2}{3}\boldsymbol{d}_{j,last1}^{\omega} + \frac{1}{6}\boldsymbol{d}_j^{\omega} \tag{18}$$

$$\boldsymbol{C}_{j,next1}^{\omega} = \frac{1}{6}\boldsymbol{d}_j^{\omega} + \frac{2}{3}\boldsymbol{d}_{next1}^{\omega} + \frac{1}{6}\boldsymbol{d}_{next2}^{\omega} \tag{19}$$

where $\boldsymbol{d}_j^{\omega}$ denotes the control point corresponding to the jth outlier point and $\boldsymbol{d}_{next1}^{\omega}$, $\boldsymbol{d}_{next2}^{\omega}$ and $\boldsymbol{d}_{last1}^{\omega}$, $\boldsymbol{d}_{last2}^{\omega}$ denote the control points on the two adjacent sides of the control point $\boldsymbol{d}_j^{\omega}$, respectively.

Furthermore, Equation (15) can be written in 4D space as:

$$L^{\omega} = \sum_{j=1}^{k} \left(D_j^{\omega}\right)^2 + \left(\boldsymbol{C}_j^{\omega} - \boldsymbol{C}_{j,0}^{\omega}\right)^2 \tag{20}$$

where $\boldsymbol{C}_j^{\omega} - \boldsymbol{C}_{j,0}^{\omega}$ can be equivalently simplified as $\boldsymbol{d}_j^{\omega} - \boldsymbol{d}_{j,0}^{\omega}$ and D_j^{ω} can be approximately expressed as [32]:

$$D_j^{\omega} \approx \frac{\left(\boldsymbol{C}_{j,next1}^{\omega}\right)'' - \left(\boldsymbol{C}_j^{\omega}\right)''}{l_{j,next1}^{\omega}} - \frac{\left(\boldsymbol{C}_j^{\omega}\right)'' - \left(\boldsymbol{C}_{j,last1}^{\omega}\right)''}{l_{j,last1}^{\omega}} \tag{21}$$

where $\left(\boldsymbol{C}_j^{\omega}\right)''$ denotes the second derivative of the NURBS at $\boldsymbol{C}_j^{\omega}$, $l_{j,next1}^{\omega} = \|\boldsymbol{C}_{j,next1,0}^{\omega} - \boldsymbol{C}_{j,0}^{\omega}\|$ and $l_{j,last1}^{\omega} = \|\boldsymbol{C}_{j,0}^{\omega} - \boldsymbol{C}_{j,last1,0}^{\omega}\|$. Substituting Equation (21) into Equation (20) yields:

$$L^{\omega} = \sum_{j=1}^{k} \left(\frac{\left(\boldsymbol{C}_{j,next1}^{\omega}\right)'' - \left(\boldsymbol{C}_j^{\omega}\right)''}{l_{j,next1}^{\omega}} - \frac{\left(\boldsymbol{C}_j^{\omega}\right)'' - \left(\boldsymbol{C}_{j,last1}^{\omega}\right)''}{l_{j,last1}^{\omega}} \right) + \left(\boldsymbol{d}_j^{\omega} - \boldsymbol{d}_{j,0}^{\omega}\right)^2 \tag{22}$$

To calculate the minimum value of L^{ω}, the partial derivative of Equation (22) about $\boldsymbol{d}_j^{\omega}$ is set to 0:

$$\begin{aligned} \frac{\partial L^{\omega}}{\partial \boldsymbol{d}_j^{\omega}} = \sum_{j=1}^{k} 2 \Bigg(& \frac{\boldsymbol{d}_{j,next2}^{\omega} - 3\boldsymbol{d}_{j,next1}^{\omega} + 3\boldsymbol{d}_j^{\omega} - \boldsymbol{d}_{j,last1}^{\omega}}{l_{j,next1}^{\omega}} - \frac{\boldsymbol{d}_{j,next1}^{\omega} - 3\boldsymbol{d}_j^{\omega} + 3\boldsymbol{d}_{j,last1}^{\omega} - \boldsymbol{d}_{j,last2}^{\omega}}{l_{j,last1}^{\omega}} \Bigg) \\ & \cdot \left(\frac{3}{l_{j,next1}^{\omega}} + \frac{3}{l_{j,last1}^{\omega}} \right) + 2\left(\boldsymbol{d}_j^{\omega} - \boldsymbol{d}_{j,0}^{\omega}\right) = 0 \end{aligned} \tag{23}$$

Then, the equations for $\boldsymbol{d}_1^{\omega}, \boldsymbol{d}_2^{\omega}, \ldots, \boldsymbol{d}_k^{\omega}$ can be expressed as:

$$\begin{aligned} \boldsymbol{A}^{\omega}\boldsymbol{d}^{\omega} &= \boldsymbol{C}^{\omega} \\ \boldsymbol{A}^{\omega} &= diag\left(\left(\frac{3}{l_{j,next1}^{\omega}} + \frac{3}{l_{j,last1}^{\omega}} \right)^2 + 1 \right) \\ \boldsymbol{d}^{\omega} &= \left(\boldsymbol{d}_1^{\omega}, \boldsymbol{d}_2^{\omega}, \ldots, \boldsymbol{d}_k^{\omega}\right)^{\mathrm{T}} \\ \boldsymbol{C}^{\omega} &= \left(\boldsymbol{C}_1^{\omega}, \boldsymbol{C}_2^{\omega}, \ldots, \boldsymbol{C}_k^{\omega}\right)^{\mathrm{T}} \\ \boldsymbol{C}_j^{\omega} &= \boldsymbol{d}_{j,0}^{\omega} - \left(\frac{\boldsymbol{d}_{j,next2}^{\omega} - 3\boldsymbol{d}_{j,next1}^{\omega} - \boldsymbol{d}_{j,last1}^{\omega}}{l_{j,next1}^{\omega}} - \frac{\boldsymbol{d}_{j,next1}^{\omega} + 3\boldsymbol{d}_{j,last1}^{\omega} - \boldsymbol{d}_{j,last2}^{\omega}}{l_{j,last1}^{\omega}} \right) \left(\frac{3}{l_{j,next1}^{\omega}} + \frac{3}{l_{j,last1}^{\omega}} \right) \end{aligned} \tag{24}$$

According to Equation (24), the optimised control points and weight factors corresponding to the outlier points can then be generated.

4. NURBS Interpolation

The NURBS interpolation is used to discretise the parametric curve to the NC commands based on the planned feed-rate. In this section, an interpolation method for optical polishing is proposed aiming to minimise the realisation error of the trajectory runtime.

4.1. Feed-Rate Planning

Feed-rate planning is the main influencing factor in interpolating the NC commands along the trajectory. For the optical polishing, to guarantee the desired runtime of trajectory, the specific method is displayed as follows:

Step 1: considering that now there is no analytical solution to calculate the length of NURBS, the Simpson formula is used to obtain the estimation of the length through numerical iteration:

$$p = \frac{up - low}{6}(f(low) + 4f(mid) + f(up)) \tag{25}$$

with:

$$f(u) = \frac{ds}{du} = \sqrt{x'(u) + y'(u) + z'(u)} \tag{26}$$

where $x'(u), y'(u), z'(u)$ respectively denote the first-order derivatives of $x(u), y(u), z(u)$, which are the one-dimensional curves of $\boldsymbol{C}(u)$ along x, y, z axes. up and low denote the upper and lower boundaries of u, $mid = (up + low)/2$.

Step 2: the length between the two points corresponding to the parameters up and low is calculated:

$$l = \|\boldsymbol{C}(up) - \boldsymbol{C}(low)\| \tag{27}$$

Step 3: the error between the aforementioned two lengths is calculated:

$$e = |p - l| \tag{28}$$

The convergence threshold $[e]$ is given. If $e > [e]$, let $up = mid$ and repeat Steps 1 and 2. If $e < [e]$, turn to Step 4.

Step 4: the parameter interval $(0, 1)$ was sectioned by the equivalent distance $up - low$ to generate the knot vector $(u_0, u_1, \ldots, u_n)$. Thus, the length of the NURBS is:

$$s \approx \sum_{i=1}^{n} \|\boldsymbol{C}(u_i) - \boldsymbol{C}(u_{i-1})\| \tag{29}$$

Equipped with the length at hand, the S-curve motion law is invoked to guarantee that the feed-rates of the adjacent NURBS segments are changed smoothly. Moreover, the initial sections of the trajectory segments are defined as the feed-rate transition zone. Then, the feed-rates remain constant until the end of the trajectory segments. In this case [33]:

$$s = 2v_{low}t_a + Jt_a^3 + v_{up}t_v \tag{30}$$

$$v_{up} = v_{low} + Jt_a^2 \tag{31}$$

where v_{low} and v_{up} denote the initial and final feed-rates of the trajectory segment. J, $2t_a$ and t_v denote the jerk, acceleration (deceleration) time and the uniform motion time, respectively. Thus, the runtime of the trajectory segment is $t_d = 2t_a + t_v$.

It is worth noting that previous studies have shown that the feed-rates, when limited by the runtime of the trajectory segments, are much lower than the maximum value which is constrained by the chord error and the driving capacity. Therefore, there is no need to check the feed-rate again.

4.2. NURBS Interpolation

The essence of interpolation is to generate the NC command along the trajectory according to the period t_s. As each interpolation point of the NURBS corresponds to one curve parameter, only the curve parameters need to be solved.

Taking u as the function of t, the second-order Taylor expansion can be expressed as:

$$u_{i+1} = u_i + \left.\frac{du}{dt}\right|_{t=t_i} t_s + \frac{1}{2}\left.\frac{d^2u}{dt^2}\right|_{t=t_i} t_s^2 \tag{32}$$

The feed-rate $v(u)$ can be written as:

$$v(u) = \|\frac{d\boldsymbol{C}(u)}{du}\| = \|\frac{d\boldsymbol{C}(u)}{du}\frac{du}{dt}\| \tag{33}$$

Furthermore:

$$\frac{du}{dt} = \frac{v(u)}{\|\mathbf{C}'(u)\|} \tag{34}$$

Calculating the derivative of Equation (34):

$$\frac{d^2u}{dt^2} = \frac{\frac{dv(u)}{du}\frac{du}{dt}}{\|\mathbf{C}'(u)\|} - v(u)\frac{\frac{d(\|\mathbf{C}'(u)\|)}{du}\frac{du}{dt}}{\|\mathbf{C}'(u)\|^2} \tag{35}$$

where:

$$\frac{d(\|\mathbf{C}'(u)\|)}{du} = \frac{\mathbf{C}''(u)\cdot\mathbf{C}'(u)}{\|\mathbf{C}'(u)\|}$$

Substituting Equations (33)–(35) into Equation (32) gives:

$$u_{i+1} = u_i + \frac{v_C(u_i)}{\|\mathbf{C}'(u_i)\|}t_s + \frac{1}{2}\left[\frac{v'_C(u_i)}{\|\mathbf{C}'(u_i)\|^2}v_C(u_i) - \frac{\mathbf{C}''(u)\cdot\mathbf{C}'(u)}{\|\mathbf{C}'(u)\|^4}v_C^2(u_i)\right]t_s^2 \tag{36}$$

Substituting u_{i+1} into the curve equation obtained through the fairing optimisation described in Section 3, the next interpolation point can be acquired. Repeating this process until:

$$t_i = \text{ceil}\left(\frac{t_d}{t_s}\right) * t_s \tag{37}$$

where ceil$(\cdot)$ denotes an integer that is rounded up. In this way, interpolation of all trajectory segment can be completed.

5. Experiments

Both simulation and experiments were carried out to validate the effectiveness of the presented method on the prototype of the hybrid polishing robot. As shown in Figure 2, it is mainly composed of a 6-DOF (degrees of freedom) hybrid robot, a polishing effector, a magnetic worktable, a column, and a CNC system. The hybrid robot is composed of a 3-DOF (3UPS and UP) parallel mechanism and a 3-DOF wrist. The UP limb and the wrist form a UPS or UPRRR limb. Here, R, U, S, and P represent, respectively, revolute, universal, spherical, and prismatic joints, and the underlined P, S, and R denote the actuated prismatic, spherical, and revolute joints, respectively. The CNC system is built upon an IPC+PMAC open architecture, consisting of a host control computer responsible for reconstruction of the parametric curves, trajectory interpolation, and NC command generation, and a PMAC motion controller for servo-control of the actuated joints.

1. CNC system 2. Column 3. Hybrid robot 4. Polishing effector 5. Magnetic worktable

Figure 2. The prototype of the polishing robot.

Without loss of generality, a segment of NC codes was taken from the polishing trajectory to validate the effectiveness of the proposed interpolation method. According to the SOR iterative scheme mentioned in Section 2, Figure 3 shows the two fitted trajectories. Then, the optimisation algorithm described in Section 3 is used to smooth the curvature of the trajectory. The change threshold of curvature for judging outlier points is given as 0.01. As shown in Table 1, the fairness of the two trajectories is both significantly improved. The maximum curvatures of the two trajectories are reduced by as much as 79.7% and 63.3% and the maximum absolute values of shear jerks are decreased by 91.2% and 90.2%, correspondingly. Considering the optimisation and interpolation methods for the two trajectories are same, experimental results of the end-point trajectory are just shown in the following discussion for the sake of simplicity.

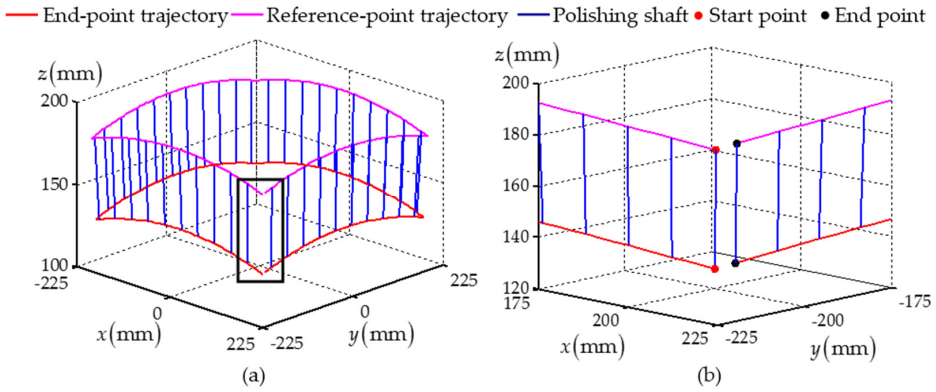

Figure 3. The initial fitted polishing trajectory for fairing optimisation and interpolation (**a**) the fitted polishing trajectory (**b**) the partial enlarged view.

Based on the optimisation trajectory shown in Figure 4, the discrete NC command sequences were generated and sent to the PMAC motion controller, which were mapped into the servo-command of actuated joints through an inverse kinematic model. The PMAC motion controller synchronously gathered the positions and velocities fed back from the servo-motors. The interpolation and sampling periods are 10 and 20 ms, respectively. To validate the effectiveness of the method, a comparison experiment was carried out utilizing the linear interpolation method. Figure 5 shows the experimental result, which is computed by the feedback positions of all the actuated joints. It can be seen from the partial enlarged view that the NC commands generated by the proposed method are closer to the NC codes than those found using linear interpolation, which means that the removal position of polishing process can be reached more accurately. Figures 6 and 7 show the runtime between the adjacent NC nodes. Compared with the desired value, the proposed method is able to realise the runtime more precisely, which indicates that the removal quantity during the polishing process can be more precisely controlled, correspondingly. To evaluate the effect of the interpolation method, the indices are defined as the interpolation error (e_i) and the runtime error of the trajectory (e_t):

$$e_i = \|\boldsymbol{C}_{i0} - \boldsymbol{C}_i\|_2,\ i = 1, 2, \ldots, n \tag{38}$$

$$e_{ti} = |t_{i0} - t_i|,\ i = 1, 2, \ldots, n \tag{39}$$

where $\boldsymbol{C}_{i0}$ and $\boldsymbol{C}_i$ denote the ith desired NC code and actual NC code after interpolation, t_{i0} and t_i denote the desired runtime and the actual runtime between the $i-1$th and ith NC codes. It can be seen from Table 2 that, compared to the linear interpolation method, the interpolation error and the runtime error generated through the proposed method are an order of magnitude smaller. These data further imply that the proposed interpolation method has more precision than the linear interpolation method in the polishing process.

Table 1. Changes in the two trajectories before, and after, fairing optimisation.

Trajectory	Before		After	
	Maximum Curvature	Maximum Shear Jerk	Maximum Curvature	Maximum Shear Jerk
End-point	1.249	0.285	0.253	0.025
Reference-point	1.380	0.325	0.506	0.032

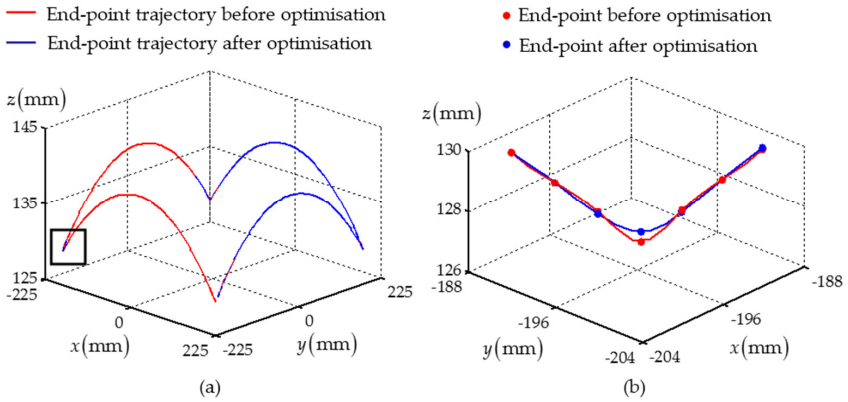

Figure 4. The end-point trajectory before, and after, fairing optimization: (**a**) the end-point trajectory; and (**b**) the partial enlarged view

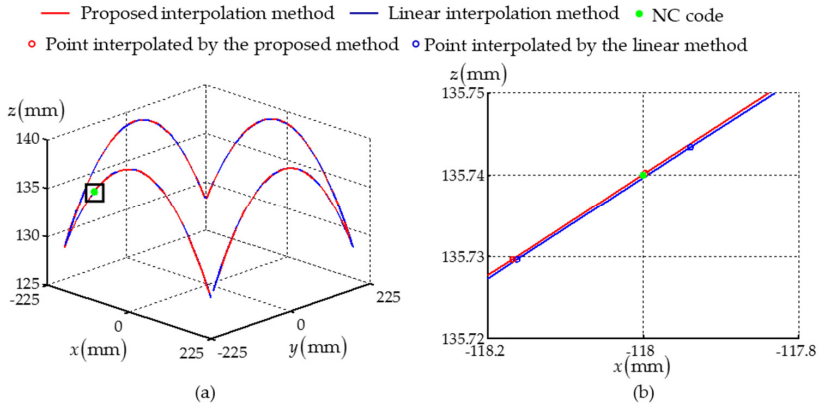

Figure 5. The end-point trajectory interpolated by different methods: (**a**) the end-point trajectory; and (**b**) the partial enlarged view.

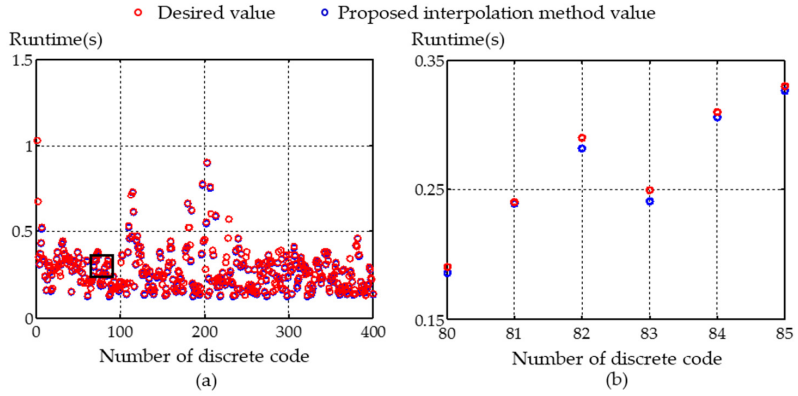

Figure 6. Runtime of the trajectory interpolated by the proposed method: (**a**) runtime of the trajectory; and (**b**) the partial enlarged view.

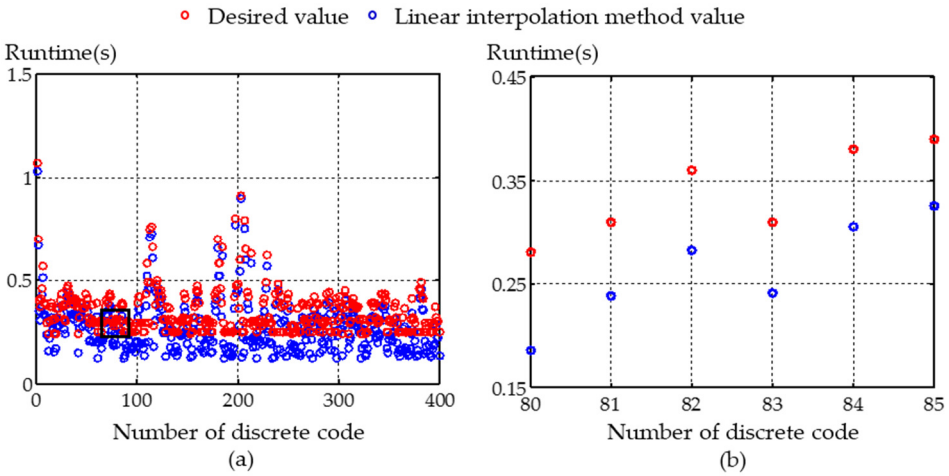

Figure 7. Runtime of the trajectory interpolated by the linear method: (**a**) runtime of the trajectory; and (**b**) the partial enlarged view.

Table 2. Comparison of the effect of the two interpolation methods.

Method	Interpolation Error (mm)		Runtime Error (s)	
	Maximum	Mean	Maximum	Mean
Proposed	0.005	0.002	0.010	0.005
Linear	0.091	0.076	0.130	0.076

To validate the modification effect of the interpolation method on the surface error of optical elements, polishing experiments were carried out on fused silica elements by, respectively, using the linear interpolation method and the proposed interpolation method, as shown in Figure 8. Using the Nanovea contour graph, the polished zone (70 mm $\times$ 70 mm) of the elements was detected. The experimental results obtained using the two interpolation methods are displayed in Figures 9 and 10, respectively. It can be seen, from the figures, that, after conducting linear interpolation-based polishing, the surface error (PV) decreases from $11.894\lambda(\lambda = 633\text{nm})$ to 7.422λ. In contrast, using the proposed interpolation method, the surface error (PV) is reduced from 11.282λ to 6.267λ. The convergence rate of the surface error is increased from 37.59% to 44.44%, which further verifies the effectiveness of the proposed method.

Figure 8. Polishing experiment on fused silica elements.

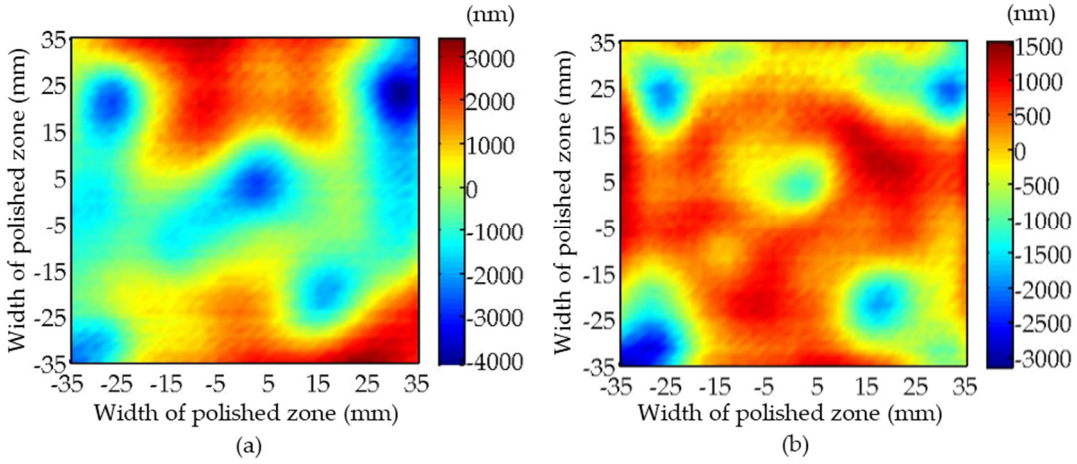

Figure 9. Changes in surface errors (**a**) before and (**b**) after conducting the linear interpolation-based polishing.

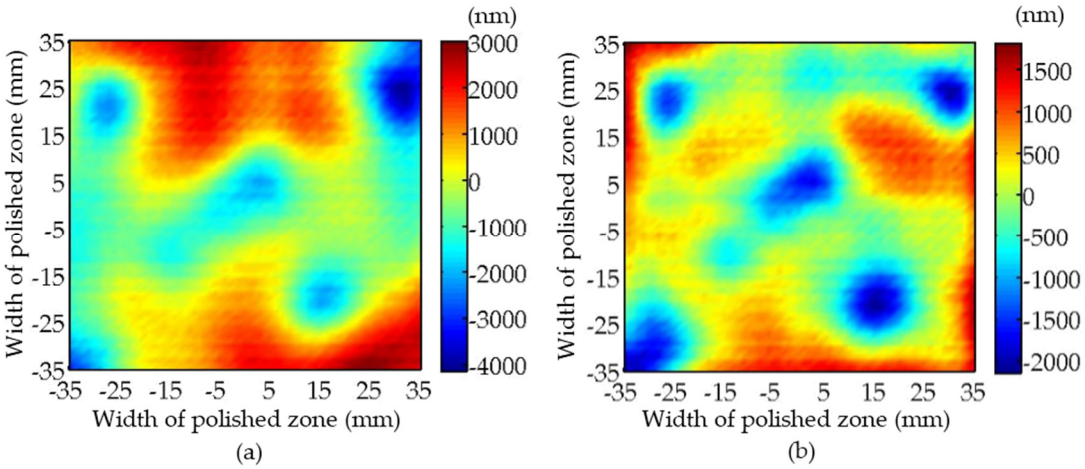

Figure 10. Changes in surface errors (**a**) before and (**b**) after conducting the proposed interpolation-based polishing.

6. Conclusions

A new trajectory planning method for optical polishing is proposed in this paper. It is developed to generate NC commands that can decrease the interpolation error and the runtime error. First, to obtain the NURBS trajectory without loss of the information about the runtime, a global fitting method is presented based on SOR iteration theory to deal with the problem of the computational burden arising from the large quantity of NC codes for polishing. Then, taking the shear jerk of the NURBS as the evaluation index, a fairing optimisation method is carried out to smooth the curvature saltation of the trajectory. Finally, the feed-rate planning method and the path interpolation scheme are proposed to reduce the realisation error of the trajectory runtime.

Simulation results verify that the fairing optimisation proposed in this research can modify the curvature saltation at the expense of trajectory accuracy compared to the given NC codes; however, the curvature saltation only occurs at the turning point of the trajectory and the trajectory error magnitude generated by the optimisation algorithm is consistent with the linear interpolation. Therefore, the trajectory accuracy is not treated as the index with which to evaluate the smoothing effect in Section 5. The effect of the optimisation algorithm on the optical polishing will be investigated in future work.

It can be seen from Figures 5–7, and Table 2, that the runtime error of the proposed interpolation method arises because the trajectory runtime cannot be divided exactly by the interpolation period and

is rounded up to an integer. In contrast, the runtime error of the linear interpolation method arises as a result of the acceleration and deceleration that occurs frequently between adjacent short line segments. Similar to the runtime error, the interpolation error of the proposed interpolation method is a result of the numerical calculation error of the trajectory length and that of the linear interpolation method is due to the transition for the discontinuity of the adjacent short line segments. Then the convergence rate of the surface error is increased with the help of the improvement in the interpolation, and runtime, errors, validating the effectiveness of the proposed interpolation process. For the aforementioned error, their individual effects on the optical polishing need to be further indicated in future work.

Author Contributions: D.Z. wrote the manuscript and performed the experiments, H.G. analysed the data and revised the manuscript. All authors discussed the results and commented on the manuscript.

Funding: This research was partially funded by the National Science and Technology Major Project of China (grant no. 2017ZX04022001-206), the National Natural Science Foundation of China (grant no. 51420105007), and EU H2020-MSCA-RISE 2016 (grant no. 734272).

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

References

1. Rupp, W. Conventional optical polishing techniques. *Int. J. Opt.* **1971**, *18*, 1–16. [CrossRef]
2. Kaltsoukalas, K.; Makris, S.; Chryssolouris, G. On generating the motion of industrial robot manipulators. *Rob. Comput. Integr. Manuf.* **2015**, *32*, 65–71. [CrossRef]
3. Makris, S.; Tsarouchi, P.; Matthaiakis, A. Dual arm robot in cooperation with humans for flexible assembly. *CIRP Ann.* **2017**, *66*, 13–16. [CrossRef]
4. Lei, W.; Wang, S. Robust real-time NURBS path interpolators. *Int. J. Mach. Tools Manuf.* **2009**, *49*, 625–633. [CrossRef]
5. Jahanpour, J.; Alizadeh, M. A novel acc-jerk-limited NURBS interpolation enhanced with an optimized S-shaped quintic feedrate scheduling scheme. *Int. J. Adv. Manuf. Technol.* **2015**, *77*, 1889–1905. [CrossRef]
6. Liu, X.; Peng, J.; Si, L. A novel approach for NURBS interpolation through the integration of acc-jerk-continuous-based control method and look-ahead algorithm. *Int. J. Adv. Manuf. Technol.* **2017**, *88*, 961–969.
7. Zhao, J.; Li, L.; Wang, G. Research of NURBS Curve Real-time Interpolation Feed Speed Planning. *Tool Eng.* **2015**, *49*, 7–11.
8. Lin, M.; Tsai, M.; Yau, H. Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm. *Int. J. Mach. Tools Manuf.* **2007**, *47*, 2246–2262. [CrossRef]
9. Park, H.; Lee, J. B-spline curve fitting based on adaptive curve refinement using dominant points. *Comput.-Aided Des.* **2007**, *39*, 439–451. [CrossRef]
10. Park, H. B-spline surface fitting based on adaptive knot placement using dominant columns. *Comput.-Aided Des.* **2011**, *43*, 258–264. [CrossRef]
11. Zhou, H.; Wang, Y.; Liu, Z. On non-uniform rational B-splines curve fitting based on the least control points. *J. Xian Jiaotong Univ.* **2008**, *42*, 73–77.
12. Xu, J. B-spline Curve Approximation Based on Feature Points Automatic Recognition. *J. Mech. Eng.* **2009**, *45*, 212–217. [CrossRef]
13. Zhao, H.; Lu, Y.; Zhu, L. Look-ahead interpolation of short line segments using B-spline curve fitting of dominant points. *P. I. Mech. Eng. B-J. Eng.* **2014**, *229*, 1–13. [CrossRef]
14. Yang, X.; Hu, Z.; Zhong, Z. Research on the NURBS Curve Fitting for Tool Path Generation. *China Mech. Eng.* **2009**, *20*, 983–984.
15. Li, W.; Liu, Y.; Yamazaki, K. The design of a NURBS pre-interpolator for five-axis machining. *Int. J. Adv. Manuf. Technol.* **2008**, *36*, 927–935. [CrossRef]
16. Lin, K.; Ueng, W.; Lai, J. CNC codes conversion from linear and circular paths to NURBS curves. *Int. J. Adv. Manuf. Technol.* **2008**, *39*, 760–773. [CrossRef]
17. Tsai, M.; Nien, H. Development of a real-time look-ahead interpolation methodology with spline-fitting technique for high-speed machining. *Int. J. Adv. Manuf. Technol.* **2010**, *47*, 621–638. [CrossRef]

18. Wang, J.; Yau, H. Real-time NURBS interpolator: application to short linear segments. *Int. J. Adv. Manuf. Technol.* **2009**, *41*, 1169–1185. [CrossRef]
19. Timar, S.; Farouki, R. Time-optimal traversal of curved paths by Cartesian CNC machines under both constant and speed-dependent axis acceleration bounds. *Rob. Comput. Integr. Manuf.* **2007**, *23*, 563–579. [CrossRef]
20. Sencer, B. Smooth Trajectory Generation and Precision Control of 5-Axis CNC Machine Tools. Ph.D. Thesis, The University of British Columbia, Vancouver, BC, Canada, October 2009.
21. Boyadjieff, C.; Farouki, R.; Timar, S. *Smoothing of Time-Optimal Feed rates for Cartesian CNC Machines*; Springer: Berlin, Germany, 2005; pp. 84–101.
22. Sencer, B.; Altintas, Y.; Croft, E. Feed optimization for five-axis CNC machine tools with drive constraints. *Int. J. Mach. Tools Manuf.* **2008**, *48*, 733–745. [CrossRef]
23. Lu, L.; Zhang, L.; Ji, S. An offline predictive feedrate scheduling method for parametric interpolation considering the constraints in trajectory and drive systems. *Int. J. Adv. Manuf. Technol.* **2016**, *83*, 2143–2157. [CrossRef]
24. Zhao, H.; Zhu, L.; Ding, H. A parametric interpolator with minimal feed fluctuation for CNC machine tools using arc-length compensation and feedback correction. *Int. J. Mach. Tools Manuf.* **2013**, *75*, 1–8. [CrossRef]
25. Sun, Y.; Zhao, Y.; Bao, Y. A novel adaptive-feedrate interpolation method for NURBS tool path with drive constraints. *Int. J. Mach. Tools Manuf.* **2013**, *77*, 74–81. [CrossRef]
26. Zhang, J.; Zhang, L.; Zhang, K. Double NURBS trajectory generation and synchronous interpolation for five-axis machining based on dual quaternion algorithm. *Int. J. Adv. Manuf. Technol.* **2016**, *83*, 2015–2025. [CrossRef]
27. Peng, J.; Liu, X.; Si, L. A Novel Approach for NURBS Interpolation with Minimal Feed Rate Fluctuation Based on Improved Adams-Moulton Method. *Math. Probl. Eng.* **2017**, *2017*, 1–10. [CrossRef]
28. Bai, Z.; Parlett, B.; Wang, Z. On generalized successive over relaxation methods for augmented linear systems. *Numer. Math.* **2005**, *102*, 1–38. [CrossRef]
29. Huang, J. Another Version of SOR Iteration and Its Generalization. Master's Thesis, Dalian University of Technology, Dalian, China, June 2013.
30. Li, Y. Study and Realization on Construction, Fairing and Smooth Joining Between Adjacent of the NURBS Surface. Master's Thesis, Xi'an University of Technology, Xi'an, China, March 2010.
31. Surhone, L.; Timpledon, M.; Marseken, S. *Non-Uniform Rational B-Spline*; Betascript Publishing: Montana, MT, USA, 2010.
32. Zhang, H.; Jiang, D.; Ding, Y. A weight-based optimal faring algorithm for planar cubic NURBS curves. *J. Southwest Minzu Univ.* **2005**, *31*, 351–355.
33. Li, X.; Wu, Y.; Leng, H. Research on a New S-curve Acceleration and Deceleration Control Method. *Modul. Mach. Tool. Autom. Manuf. Technol.* **2007**, 50–53.