Iso-parametric tool-path planning for point clouds

Qiang Zou a,b, Jibin Zhao a,∗

a Shenyang Institute of Automation, Chinese Academy of Sciences, Liaoning, 110016, China
b University of Chinese Academy of Sciences, Beijing, 100049, China

ABSTRACT

Due to the compute-intensiveness and the lack of robustness of the algorithms for reconstruction of meshes and spline surfaces from point clouds, there is a need for further research in the topic of direct tool-path planning based on point clouds. In this paper, a novel approach for planning iso-parametric tool-path from a point cloud is presented. Since such planning falls into the iso-parametric category, it intrinsically depends on the parameterization of point clouds. Accordingly, a point-based conformal map is employed to build the parameterization. Based on it, formulas of computing path parameters are derived, which are much simpler than the conventional ones. By regularizing parameter domain and on the basis of the previous formulas, boundary conformed tool-path can be generated with forward and side step calculated against specified chord deviation and scallop height, respectively. Experimental results are given to illustrate the effectiveness of the proposed methods.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

Free-form surfaces (e.g., aero-parts and molds) are widely used in manufacturing industries. They are often machined by computer numerical control (CNC) machine tools that move cutter or table along a specified trajectory. In the most common cases, this trajectory is so-called tool-path which constitutes the core of computer-aided manufacturing (CAM). The automatic generation of tool-path for free-form surfaces is a fundamental issue in modern CAD/CAM systems.

Planning tool-path is a compromising between precision and efficiency, which mainly involves two aspects, path pattern and path parameters. The former is about which shape the tool-path is. More specifically, there are three path patterns so far: direction parallel, contour parallel and spiral. The latter concerns geometric parameters of tool-path, i.e., forward step and side step bounding chord deviation and scallop height, respectively. In this paper, we shall use the terminology interval to refer to offsetting distance on surfaces and the step is used to refer to parametric offsetting distance. Fig. 1 shows the three path patterns and Fig. 2 describes the two path parameters. The basis of tool-path planning was laid in the 1990s. For instance, iso-parametric method [1], iso-planar method [2] and iso-scallop method [3–5] are some typical approaches. Several developments are, e.g., iso-phote [6] and C-space [7] methods with planning tool orientation taken into account. The former in fact proposed another method of parameterization and the latter introduced the classical C-space method for robotics into tool-path planning. What should be noted is the work [8] by G. W. Vickers et al. providing a mathematical method to determine the interval between the consecutive paths. And [2] presented a true machining error calculation method with which accurate forward step can be determined. Based on these pioneering and fundamental works, some recent developments are iso-scallop [9], boundary conformed [10,11], curvilinear [12], iso-conic [13], steepest ascent approach [14] and multiresolution method [15], to name a few. Surveys of much more work about tool-path planning research can be referred in [16,17].

Tool-path planning on a surface is closely related to its representations among which the point cloud, as a direct description of surfaces, has been receiving a growing attention since the pioneering work of Marc Alexa et al. [18]. In fact, a considerable part of meshes and spline surfaces encountered in CAD/CAM are reconstructed from point clouds through approximation. However, this kind of process is complicated and compute-intensive. What
is worse, it is non-robust especially when the point cloud is a non-uniform sampling. Therefore, direct planning of tool-path for point clouds is of great significance. Compared with other representations, the lack of topological information simplifies the representation and storage of surfaces. Yet, when it comes to geometric processing, things become rather tough. That is why although there is enormous literature focusing on tool-path planning, those specified for point clouds are a few. The earlier methods of directly planning on point clouds resorted to resampling, which can regarded as an extension of traditional iso-planar ones. Lin et al. [19] employed a uniform rectangular grid (i.e., the Z-map constructed from an original point cloud) to generate tool-path for milling surfaces slice-by-slice. Rows of the grid on each slice (or level) were picked out as tool-path for the slice. Another relevant method is the work [20] by Feng and Teng. They adaptively computed the forward and side step by constructing a so-called CL-net which in fact is a variant of the Z-map. The iso-scallop method [5] of spline surfaces was extended to point clouds by Wei et al. [21]. Apart from those, S. C. Park et al. [22] generated contour parallel tool-path for pocket milling from a data structure point of view (i.e., PSC-map). The preceding review is carried out regardless of those based on local or global surface reconstruction, which in fact steps backward. To review as far as we can, the extension of traditional iso-parametric tool-path planning method, which can avoid complicated surface–surface intersection and surface offsetting in iso-planar and iso-scallop methods, to point clouds seems to remain blank.

In this paper, a method of iso-parametric tool-path planning for point clouds is presented. Surface parameterization, closely related to machine learning and computer graphics [23,24], is crucial for planning tool-path iso-parametrically. Yang et al. [10] and Sun et al. [25] employed harmonic map to parameterize spline surfaces and meshes respectively, of which free-boundary property was exploited to plan boundary conformal paths. The property means that the boundary of parameter domain can be defined arbitrary. Thus, by mapping spatial boundaries to regular planar boundaries (i.e., rectangles and circles), boundary conformal tool-path can be generated. These methods as well as their developments are heavily dependent on the topological information and hence unlikely to be applied to unstructured point clouds. By locally triangulating, Floater et al. [26] were able to extend their parameterization method to point clouds and this method has been extended to point clouds of spherical topology by Gotsman et al. [27]. In the meshless domain where topological reconstruction is excluded, conformal parameterization is based on the spectral theory. There are mainly two ways to construct point-based Laplacian, heat diffusion scheme [28] and optimal scheme [29]. For heat diffusion scheme, choosing a proper time parameter is non-trivial, and thus the optimal scheme is preferred. Once point-based Laplacian constructed, harmonic map can be exploited to parameterizing point clouds conformally. It is obvious that such parameterization method holds the free-boundary property too. Therefore, similar to the preceding mapping-based family of tool-path planning methods [10,25], boundary conformed tool-path can be generated. However, in addition to the free-boundary property, there is another more exciting property that their works did not cover, i.e., the conformal property. The advantage of conformal parameterization over conventional ones is the conformality (i.e., angle preserving) with which the computation of side and forward step can be simplified significantly. As known, a single path on a free-form surface is offset from a previous one along the direction orthogonal to forward direction. As the conformal parameterization is angle preserving, direction on a surface is consistent to that on its parameter domain. Therefore, for iso-parametric tool-path, its forward direction is the u-direction (or v-direction) and the offsetting direction is the u-direction (or v-direction). What is more, the step size calculation is avoided if the parameterization is conformal, since for any point on the surface, its local shape is similar to its parametric image, with respect to a factor. Namely, a small interval on the surface and its corresponding step in the parameter domain are proportional. In fact, these presented approaches can also be exploited to generate iso-scallop tool-path for point clouds as an extension of the works [3,4].

The remainder of this paper is organized as follows. Section 2 introduces the conformal parameterization of point clouds; in Section 3, we derive the formulas for calculating path parameters, which is a significant simplification of the conventional ones; Section 4 describes the proposed iso-parametric tool-path planning method; Section 5 shows the experimental results; Finally, we conclude the whole paper in Section 6.

2. Conformal point cloud parameterization

Surface parameterization is to find a bijective mapping between two surfaces with similar topology. If both surfaces are discretized
as point clouds, the problem of computing such point-wise mapping is known as point cloud parameterization. Thus, for a point cloud, its parameterization is to find a planar counterpart that represents the point cloud. It is well known that harmonic map is an alternative of conformal parameterization. It satisfies the following Laplace equation 

\[ \nabla^2 f = 0, \]

with respect to the Dirichlet boundary condition: \( f : \partial \sum_2 \to \partial \sum_1 \) is homeomorphism, where \( \nabla^2 \) donates the Laplacian and \( f \) is a map over one surface \( \sum_1 \) to the other \( \sum_2 \). If \( \partial \sum_2 \) is planar, the Laplace equation has a unique solution and hence its inverse is a conformal parameterization for the surface \( \sum_2 \). Similarly, in discrete cases, such parameterization can be obtained by constructing discrete Laplacian and subsequently solving discrete Laplace equation.

2.1. Discrete Laplace equation

The discrete Laplacian \( L \) for a point cloud \( S = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^3 \) is a linear operator which takes a map on \( S \) as input and another map on \( S \) as output. If the size of \( S \) is \( n \), a map on \( S \) is an \( n \)-dimensional vector. Therefore, \( L \) is a linear map between two \( n \)-dimensional vector spaces, which can be represented by an \( n \times n \) matrix. Generally, most methods of constructing such matrix fall into two categories, the Graph-based methods and the Finite Element Method (FEM) based methods.

The graph Laplacian \( L \) is defined as

\[
L_g = \begin{cases} \deg(p_i) & i = j \\
-1 & i \neq j \text{ and } p_i \text{ neighbor to } p_j \\
0 & \text{otherwise,}
\end{cases}
\]

where \( \deg(p_i) \) donates the degree of point \( p_i \). And subsequently, the Laplacian \( L(f(p_i)) \) of a map \( f \) at a point \( p_i \) can be expressed as a local sum

\[
(Lf)(i) = \sum_{p_j \in N(p_i)} \left[ f(p_j) - f(p_i) \right],
\]

where \( N(p_i) \) is a set of points neighbor to \( p_i \), e.g., the \( K \) nearest neighborhood (KNN). As for the FEM based methods, it is defined as

\[
(Lf)(i) = \sum_{p_j \in N(p_i)} w_{ij} \left[ f(p_j) - f(p_i) \right],
\]

where \( w_{ij} \) donates introduced weights. In the literature of parameterization, there have been several weights proposed for meshes. For example, the famous cotangent scheme [30] and the mean-value scheme [31]. Since these weights were derived on triangular meshes, it seems to be hard for them to be applied to point clouds directly. However, a weight scheme specified for point clouds has recently been proved to converge to the continuous Laplacian as sampling getting denser by Belkin and Niyogi [28]. The weights are as follows

\[
w_{ij} = e^{-\frac{||p_i - p_j||}{\epsilon}}, \quad t \in \mathbb{R}^+ \text{ and } p_i \in N(p_i),
\]

where \( \epsilon \) is constant for a point cloud. However, it is still unknown how to choose an optimal \( \epsilon \) for a real model [28]. There is another weight scheme (optimal weights) coinciding with Belkin’s. Schmidt and Singh [29] parameterized point clouds conformally with such optimal weights which minimizes a quadric error

\[
\varepsilon = \left\| p_i - \sum w_{ij} p_j' \right\|^2, \quad p_i \in N(p_i),
\]

subjecting to \( \sum w_{ij} = 1 \), where \( p_j' \) is the projected point of \( p_j \) on the tangent plane at \( p_i \). The error \( \varepsilon \) can also be written as

\[
\varepsilon = \left\| \sum w_{ij} (p_i - p_j') \right\|^2.
\]

Then, minimizing the error becomes a least-square problem. Re- 

donate the indices of neighbor points as \( \{1, m\} \), where \( m \) is the size of neighborhood. The solution is to solve a linear system

\[
CW = 1, \quad c_{ik} = (p_i - p_k)'(p_i - p_k),
\]

where \( I \) is the one-vector, and then recalculate the weights so that their sum is 1.

Once the discrete Laplacian for a point cloud constructed, the Laplace equation for a point cloud becomes \( Lf = 0 \). Suppose that \( \partial S = \{p_{1\times}, \ldots, p_n\} \), \( \partial D = \{q_1, \ldots, q_n\} \) and the boundary map \( f : p_i \mapsto q_i \). Then the conformal parameterization problem comes to find the interior mapping with respect to the boundary map, i.e., to solve a linear system

\[
Lf = 0 \Rightarrow AU^T = -BU^T,
\]

where

\[
\begin{bmatrix} A_{n\times r} & B_{rx(n-r)} \\ \rho_{n\times r} & C_{(n-r)xn} \end{bmatrix} = L.
\]

This can also be rewritten as

\[
q_i = - \sum_{p_j \in N(p_i) \setminus S} w_{ij} q_j + \sum_{p_j \in N(p_i) \setminus D} w_{ij} q_j, \quad i = 1, 2, \ldots, r.
\]

Note that \( q_i = [u_i, v_i]^T \) and Eq. (1) is solved twice, one for \( u \) and the other for \( v \) coordinate.

2.2. Boundary points mapping

The pre-step for constructing conformal parameterization is to define boundary map, with mainly two aspects involved: shape of planar boundary and distribution of boundary points. When planning contour parallel tool-path, the boundary is mapped to a circle in a Polar coordinates system. Thus iso-curves corresponding to concentric circles in parameter domain are selected as paths, which are free from intersection detection and trimming, as opposed to the conventional offsetting methods. Paths are generated in a manner as they morph inward from boundary and get smoother along the morphing. When planning direction parallel tool-path, boundary is mapped to a rectangle in a Cartesian coordinates system, so that paths can be simply generated by selecting segments parallel to either of the two edge pairs. This regular domain can help to avoid paths of small size (i.e., not boundary conformed), if the underlying surface is trimmed, as opposed to the conventional iso-planar methods. This merit was shown by Yang et al. [10]. As for distribution, there are two procedures involved, ordering points and assigning them.

The method used to order boundary points is inspired by the material in Floater’s [26]. First, a point cloud is classified into two subsets, interior and boundary. Then, those boundary points are manually broken into several simple parts, as shown in Fig. 3(a). For each part, boundary points are parameterized into the unit interval \([0, 1]\) by the method similar to Section 2.1. Their boundary points are the end points of each part (i.e., the breaking points). And their weights

\[
w_{ij} = 1/\left\| p_i - p_j \right\|, \quad p_j \in N(p_i).
\]

The ordering of parameter values is used to order the 3D boundary points. Finally, all parts are combined into an ordered boundary according to the ordering of breaking points.
Boundary points are assigned along planar boundary according to chord length between the adjacent points on the 3D boundary. Specifically, when planning contour parallel tool-path, boundary points are mapped to a circle in a Polar coordinates system with $f : p_i \mapsto (R_0, \lambda_i \cdot 2\pi)$, $p_i \in \partial S$, where radius $R_0$ can be set arbitrarily and we set it to be $L/(2\pi)$ with $L$ being length of the ordered boundary. $\lambda_i$ is the distribution parameter with $\lambda_{i+1} = 0$ and for $r + 1 < i \leq n$

$$\lambda_i = \sum_{j=r+2}^{i} \|p_j - p_{j-1}\| / L.$$  

Intuitively, points are proportionally mapped around the whole boundary, with respect to the chord length.

When planning direction parallel tool-path, boundary points are mapped to a rectangle. They are first divided into four parts manually, so that four breaking points are consistent to the four planar corner points. Inherently, each part has a local ordering. Then, each part is also mapped to its corresponding planar edge proportionally. Suppose that consecutive lengths of the four parts are $L_1, L_2, L_3$ and $L_4$. Then the lower left point is set to be $(0, 0)$ and the upper right point is $((L_1 + L_3) / 2, (L_2 + L_4) / 2)$. The first part is mapped to $u$-axis with $f : p_i \mapsto (\lambda_i \cdot (L_1 + L_3) / 2, 0)$, $p_i \in \text{first part of} \ \partial S$, where $\lambda_i$ is the assigning parameter with $\lambda_{i+1} = 0$ and for the rest points

$$\lambda_i = \sum_{j=r+2}^{i} \|p_j - p_{j-1}\| / L_1,$$

and correspondingly for the rest three parts.

As the preceding shows, conformal point cloud parameterization can be boiled down to solve a sparse linear system. It can provide some perfect properties, described in Section 3, for tool-path planning. The angle preserving property helps to simplify the computation of path parameters, the free-boundary property provides natural parameter domains for planning tool-path and the shape preserving property makes the 3D shape of paths consistent to that of 2D parametric ones.

### 3. Path parameters calculation

Determining geometric parameters for tool-path (i.e., forward and side step) is closely related to differential properties of a surface. Some basic notions about differential geometry are introduced and it is also shown how to simplify the formulas for calculating forward and side step in this section.

#### 3.1. Differential geometry

There are many curvatures for a surface (e.g., Gaussian curvature, mean curvature and principle curvatures) and the one involved with tool-path planning is normal curvature. Normal curvature is defined as the curvature of a curve that is the intersection between the surface itself and a plane determined by the normal vector $n$ and a tangent vector $e$. Consider a parametric surface $x(u, v) = (u, v) \in \mathbb{R}^2$, the normal curvature is expressed as

$$\kappa = \frac{\|F\|}{\|E\|} = \frac{L du^2 + 2M du dv + N dv^2}{E du^2 + 2F du dv + G dv^2}$$

where $E = r_u \cdot r_u, F = r_u \cdot r_v, G = r_v \cdot r_v$ are the coefficients of the first fundamental form $I$ and $L = n \cdot r_{uu}, M = n \cdot r_{uv}, N = n \cdot r_{vv}$ are coefficients of the second fundamental form II. The pair $(du, dv)$ in fact represents the direction $e$. For instance, the normal curvature along direction $(du, 0)$ is $\kappa = L/E = n \cdot r_{uu} / r_u \cdot r_u$.

When the parameterization is conformal, it is known that $E = G$ and $F = 0$ which implies directions $(0, dv)$ and $(du, 0)$ are orthogonal on the surface. What is more, the local shapes around $p \in P$ and $q \in D$, where $p = r(q)$, are similar with respect to a factor $\sigma$. The factor can be given with the following derivation

$$\sigma = \left| \frac{dr}{dv} \right| = \frac{1}{|dv|} = \frac{\sqrt{r_u \cdot r_u}}{|dv|} = |r_u| = |r_u| .$$

Therefore, a small increment on a surface is $|r_u|$ times of the corresponding parametric increment.

In discrete cases, the difficulty is an approximation of the first and second order differential properties and the unit normal. The first idea may be rational spline surface reconstruction and then analytical evaluation. However, this is rather complicated. We next use a well-known difference scheme to approximate the two properties. Consider a planar point $q_0$ and its KNN. Their corresponding spatial points can be obtained with the previous parameterization. Insert two points $q_1$ and $q_2$ into the KNN, as in Fig. 4. Their corresponding spatial points are computed using the method described in Section 3.3 and the difference $\Delta u$ is chosen as the shortest distance between $q_0$ and its KNN. Expand $p_1$ and $p_2$ as the Taylor series

$$p_1 = r(u_0 - \Delta u, v_0) = r(u_0, v_0) - r_u(u_0, v_0) \cdot \Delta u + r_{uu}(u_0, v_0) \cdot \Delta u^2 + O(\Delta u^3);$$

$$p_2 = r(u_0 + \Delta u, v_0) = r(u_0, v_0) + r_u(u_0, v_0) \cdot \Delta u + r_{uu}(u_0, v_0) \cdot \Delta u^2 + O(\Delta u^3).$$

Then

$$r_u(u_0, v_0) = \frac{p_2 - p_1}{2\Delta u} \quad \text{and} \quad r_{uu}(u_0, v_0) = \frac{p_2 + p_1 - 2p_0}{2\Delta u^2} .$$

And correspondingly for $r_v$ and $r_{uv}$. Note that the coefficients $F, M$ are not needed in our method.

As to the unit normal estimation problem, it can be regarded as a local tangent plane approximation problem which in turn becomes a least-square fitting one [18]. Specifically, for a point $p_i$ and its KNN, its local covariance matrix is

$$C = \frac{1}{|N(p_i)|} \sum_{p_j \in N(p_i)} (p_j - c_i)(p_j - c_i)^T$$

where $c_i$ is the barycentric point. And the optimal unit normal is the normalized eigenvector corresponding to the smallest eigenvalue.

### 3.2. Forward and side step

Forward step is responsible for chord deviation. A single tool path is usually discretized to be a series of line or arc segments (this paper focuses on line segments), for the limited capacity of CNC interpolator. The chord deviation is defined to describe the error of approximating a curve segment with a line one, for details see [2]. Maximizing the forward step which determines the length of each segment is significant for machining efficiency.
The forward surface interval for machining can be expressed as
\[ l_f = \sqrt{8eR \cos \theta - 4e^2}, \]
where \( R \) is the radius of normal curvature along forward direction, \( e \) is chord deviation and \( \theta \) donates angle between normal vector of the surface and normal vector (sometimes called curvature vector) of the single path. For iso-parametric tool-path, the radius of forward normal curvature along the direction \((0, dv)\) can be simplified as
\[ R = \left| \frac{1}{k_f} \right| = \left| \frac{1}{N} \right| = \left| \frac{G}{N} \right| = \left| \frac{r_e \cdot r_e}{n \cdot r_{uv}} \right|. \]

To generate a single iso-parametric toolpath, the forward step should be computed. As mentioned, conformal maps preserve shapes infinitesimally, namely, the 3D shape is similar to the 2D counterpart in a small range. Therefore, locally, the forward step is proportional to the forward surface interval with respect to a factor \( \sigma \), implying
\[ \Delta u_i = v_{i+1} - v_i = \frac{l_i}{\sigma_i} = l_i |r_i|. \]

Eventually, the forward step can be iteratively computed with the expression
\[ v_{i+1} = v_i + \frac{l_i}{|r_i|}. \]

3.3. Linear interpolation

The assumption that each point of tool paths should be one of the existing points of a point cloud is unreasonable. Thus, interpolation is inevitable. The one we choose is of linear precision [32], but much simpler than rational spline interpolation. It allows a point to be expressed as a weighted linear combination of its neighbor points. What is more, it is consistent to the conformal parameterization.

Consider a planar point \( q_0 \) with its KNN being \( \{q_1, \ldots, q_n\} \), its corresponding spatial point \( p_0 \) is
\[ p_0 = \sum_{i=1}^{n} w_i q_i, \quad \sum_{i=1}^{n} w_i = 1, \]
where \( \{p_i\} \) are the corresponding spatial points of the planar KNN. Although there are many schemes of the weights, they are chosen as the optimal ones as in the conformal parameterization.

4. Tool-path planning

Planning tool-path is to represent a surface with a series of curves against some error criteria (i.e., chord deviation and scallop height). Iso-parametric tool-path consisting of \( m \) parametric curves \( \{r(u_0, v), \ldots, r(u_m, v)\} \) are generated by keeping one parameter \( e.g., u \) constant and varying others. We next show how to generate such curves on a surface for the two parallel tool-path patterns respectively.

4.1. Direction parallel tool-path

In order to construct row-like curves on surfaces by the iso-parametric method, the parameters \( u \) and \( v \) should be variables of a Cartesian coordinates system. As mentioned, the conformal parameterization has a free-boundary property. Therefore, the parameter domain for planning direction parallel tool-path is set as a planar block, which means that the boundary is mapped to a rectangle. If the lower left point of the rectangle is set to be the original point and up right point be point \((a, b)\), forward direction can be chosen as the positive direction of \( v \)-axis and side direction as the positive direction of \( u \)-axis.

For each path, start from an initial point \((u_0, 0)\), iteratively determine the next forward point \((u_i, v_i) \rightarrow (u_{i+1}, v_{i+1})\) by the expression (2) and linear interpolation until \( v \)-coordinate is out of range, i.e., \( v_{i+1} > b \), and set the last \( v_i \) to be \( b \). The first initial point is \((0, 0)\), all other initial points are determined by side step.
For consecutive paths, compute side step for each point on the previous path with expressions (3) or (4), where the expression (3) is for interior points and (4) for points near boundary. Then select the minimal step as side step for the next path. It is a two-stage procedure. First plan 3–5 paths from each side with the expression (4) resulting in a narrowed rectangular parameter domain, which is shown by dotted segments in Fig. 5(a). This procedure is necessary since there is distortion of conformity at points near boundary. Then iteratively determine side step with expressions (3) from \( u_0 \) until \( u \)-coordinate is out of \( u_1 \), as shown by the solid segments in Fig. 5(a). And set the last \( u \) to be \( u_I \).

Note that the expression (4) in Appendix A is the conventional formula for computing path parameters.

### 4.2. Contour parallel tool-path

The difference between contour and direction parallel tool-path is the coordinate system used. For contour parallel tool-path, the parameters \( u \) and \( v \) should be variables of a Polar coordinates system. And the boundary is mapped to a circle with its center being the original point and \( R \) the radius. Forward direction can be chosen as the positive direction of \( \theta \)-axis and side direction as the positive direction of \( \rho \)-axis.

For each path, start from an initial point \((\rho_i, 0)\), iteratively determine next forward point \((\rho_i, \theta_j) \) to \((\rho_i, \theta_{j+1})\) by the forward step computing and linear interpolation until \( \theta \)-coordinate is out of range \( 2\pi \). And then set the last \( \theta_j \) to be \( 2\pi \).

For consecutive paths, similar to direction parallel tool-path, it is divided into two parts. First, plan 3–5 paths from the boundary, which also results in a narrowed circular parameter domain. Then iteratively compute side step for each path from the circle \( R \) until \( \rho \)-coordinate is less than radius of cutter. And set the last \( \rho \) to be the radius of cutter.

### 5. Experimental results

In this section, the proposed method is implemented on real data. Three typical models are chosen to illustrate the effectiveness of it, as in Fig. 6. A cockpit surface (a) is used to plan direction parallel tool-path, a human face (b) is used to generate contour parallel tool-path and a free-form surface (c) with complex boundary is used to show both paths. The former two point clouds were generated by a 3D scanner (a coordinate measuring machine). The latter was generated by the UG software. Their bounding box ranges are \( 221.49 \times 105.83 \times 58.25, 139.57 \times 118.93 \times 32.76 \) and \( 83.50 \times 48.21 \times 18.46 \), respectively.

For planning iso-parametric tool-path, the first thing to do is parameterization. The K-d tree is exploited to quickly search for KNN and the number of neighbor points is 12. When ordering boundaries points the number is 4. The algorithm GMRES is chosen to solve the sparse equations. The algorithm in [33] is adopted to extract boundaries from point clouds. Note that, the former two point clouds are smoothen by the Laplacian fairing method and the Laplacian scheme is consistent to those in Section 2.1. The results of parameterization are presented in Figs. 7–9. Fig. 7(c) shows the angle preserving property of the cockpit point cloud with a rectangular domain. Fig. 8 (a) shows the angle preserving property of the face point cloud with a circular domain. Fig. 9(a) and (b) shows the angle preserving property of the free-form point cloud with both domains.

As point clouds have become parametric ones, a series of forward and side points can be computed using the analytical expressions in Section 3.2. And subsequently, the corresponding tool-path can be generated by linear interpolation. A ball–end cutter with radius \( r = 4 \) mm is chosen to illustrate the path generation so that tool orientation does not matter. The limited scallop height is \( h = 1 \) mm and chord deviation is \( e = 0.01 \) mm. In order to clearly showing tool paths, the error criterion (scallop height) is set to be much greater than those in real cases. Fig. 7 shows the direction parallel paths on the cockpit point cloud. A comparison between the proposed method and the conventional iso-planar method is also given. Fig. 7(a) and (b) shows paths generated by the iso-planar method. Fig. 7(d) and (e) shows corresponding paths by the proposed method. Fig. 8 shows the contour parallel paths on the face point cloud and a comparison between the proposed method and the conventional offsetting method. Fig. 8(b) shows paths generated by the proposed method. Fig. 8(c) shows corresponding paths by the offsetting method. Fig. 9 shows both path patterns planned for the free-form point cloud. Fig. 9(c) shows direction parallel tool-path and Fig. 9(d) shows contour parallel tool-path.

As the figures show, for direction parallel tool-path, the lengths of paths are rather even and it is boundary conformed, as opposed to iso-planar methods which often generate uneven paths in terms of length if the initial plane is chosen poorly. And, for contour parallel tool-path, boundary morphs inward gradually making tool-path boundary conformed, as opposed to the conventional offsetting methods needing the post-process of removing intersection between offsetting paths. Another problem of conventional contour parallel tool-path is that they preserve sharp corners of boundaries, which limits the feed–rate when approaching these corners. And thus, machining efficiency and tool wear are reduced. However, the method proposed rounds these corners automatically and gradually.

As mentioned, discrete boundary mapping will cause distortion near boundary. In Fig. 10, the effect of different assignments is shown, one assigns boundary according to chord length between adjacent 3D points (the adopted method) and the other assigns boundary uniformly. It is shown that chord length based method can give better results, especially when the 3D boundary points are irregularly distributed. In Fig. 11, the effect of corners of different angles on tool-path is shown. These corners are mapped to be right angles when planning direction parallel tool-path, which causes local non-conformality. The effect of these corners seems to make side intervals conservative when employing the simplified side step calculation formula, as shown in Fig. 11(c). The model is chosen to illustrate the effect because its radius of normal curvature is much greater than the cutter radius and it has three typical angles, obtuse, acute and right angles. The acute angle is 25.56° and the scallop height is limited to \( h = 0.03 \) mm. The number of paths in (b) is 26 and 30 in (c). Fig. 12 shows side interval approximation error \( \varepsilon(\%) = (l_i - l_i^d) / l_i \times 100 \) where \( l_i^d = \Delta u \cdot \sigma \) and \( l_i \) donates side interval between two adjacent paths of the human face model (Fig. 8). Fig. 12(b) shows error of two interior adjacent paths and (c) shows error of two paths near boundary. The mean value, maximum value and minimum value of (b) are 1.4083, 3.1971 and 0.0380 respectively. The corresponding values of (c) are 2.2442, 4.884 and 0.1945. As shown in Appendix B, the error of scallop height is twice that of side interval. Therefore, the corresponding mean error of scallop height is around 5%.

### 6. Conclusions

The shortcomings of converting point clouds to meshes or spline surfaces are compute-intensiveness and non-robustness. To overcome these, a direct tool-path planning method (iso-parametric) for point clouds is presented. It follows a conformal point cloud parameterization laying a foundation for the planning. Simpler mathematical formulas for determining path parameters analytically are then presented. Finally, tool-path can be generated by iteratively computing the path parameters and the linear
Fig. 5. Iso-parametric tool-path planning. (a) Direction parallel tool-path; (b) Contour parallel tool-path.

Fig. 6. Tested point cloud models. (a) the surface of a cockpit; (b) the surface of a human face; (c) a free-form surface.

Fig. 7. The results of a cockpit surface. (a) (b) tool-path generated by the iso-planar method; (c) angle-preserving property with a rectangular domain; (d) direction parallel tool-path in one direction; (e) direction parallel tool-path in the other direction.

Fig. 8. The results of a face surface. (a) angle-preserving property with a circular domain; (b) contour parallel tool-path; (c) tool-path generated by the conventional offsetting method.
Fig. 9. The results of a free-form surface. (a)/(b) angle-preserving property with a rectangular domain and a circular domain, respectively; (c) direction parallel tool-path; (d) contour parallel tool-path.

Fig. 10. Comparison of different assignments. (a) original point cloud with irregular boundary; (b) parameterization with a rectangular domain using chord length boundary assignment; (c) parameterization with a rectangular domain using uniform boundary assignment; (d) parameterization with a circular domain using chord length boundary assignment; (e) parameterization with a circular domain using uniform boundary assignment.

Fig. 11. The effect of distortion near boundary on tool-path. (a) original point cloud with different angle; (b) paths planned using the accurate expression (4); (c) paths planned using the simplified expression (3).
interpolation. The angle preserving property of the conformal parameterization simplifies the calculation of tool-path parameters as well as the transferring of surface interval to parametric step. Moreover, the free-boundary property helps generate the boundary-conformed tool-path.

However, this work preserves the weakness of iso-parametric methods inherently. That is, the spatial path interval is uneven making it less efficient than iso-scallop ones. Fortunately, when planning iso-scallop tool-path from point clouds, most results in this paper remain valid, e.g., formulas of computing path parameters. The planning for point clouds with arbitrary topology (e.g., a point cloud with holes) is not studied. Yet, by controlling the boundary map, tool-path for them can be generated similar to the cases presented with some modification.

Acknowledgments

The authors are grateful for the support provided by National Key Basic Research Project of China (grant # 2011CB302400) and National Natural Science Foundation of China (grants # 50975274, 50975495).

Appendix A. Non-conformality error analysis

According to the Riemann Mapping Theorem, if the boundary shape of parameter domain is given, a conformal parameterization always exists in continuous cases. However, in discrete cases, mapping a complex 3D boundary to either a rectangle or a circle will inevitably cause distortion of conformality at points near boundary, since the Laplacian and boundary map are approximations. But the distortion is limited to points near boundary, as shown in Fig. 9(a) (b). There are some ways to avoid this by computing boundary map as part of the solution, with a trade-off being that the free-boundary property is lost. However, as demonstrated in [10,25], such property is of great significance for planning boundary-conformed tool-path. We next show the effect of such distortion on tool-path.

For a point on the surface, its local shape is linearly related to its image in the parameter domain, with respect to a Jacobian.

Suppose that the side interval vector is \( \mathbf{s} = k_1 r_u + k_2 r_v \). Its corresponding parametric increment is

\[
[\Delta u \Delta v] = J^{-1}(k_1 r_u + k_2 r_v) = [k_1/|r_u| \quad k_2/|r_v|],
\]

where \( J = (r_u, r_v) \) is the Jacobian at the point, and the coefficients are \( k_1 = l_u/\cos (\theta - \pi /2) \), \( k_2 = l_v/\sin (\theta - \pi /2) \). The angle \( \theta = \angle (r_u, r_v) \). Subsequently, take \( u \) direction as side direction, the side step can be computed with

\[
\Delta u = \frac{\sqrt{8hr / (R_u \pm r)}}{\cos (\theta - \pi /2) \cdot |r_u|},
\]

where \( R_u \) is the radius of normal curvature along the direction orthogonal to forward direction. According to the Euler Theorem, the side normal curvature is \( \kappa_s = 2H - k_f \), where \( H \) donates the mean curvature. For a point cloud, it can be easily estimated by the method [33]. This is the original expression for computing side interval. If we still employ the expression (3) at non-conformal points, it gives

\[
\Delta u' = \frac{\sqrt{8hr / (R_u \pm r)}}{|r_u|}.
\]

If \( \Delta u' \leq \Delta u \), the simplified expression will give conservative paths, i.e., the tool-path is less efficient but the precision is guaranteed. Otherwise, the simplified formula will give incorrect steps at the point. Although the Euler theorem can relate \( R_u \) with \( R_v \), the sign of \( \Delta u - \Delta u' \) is undermined. However, if the radius of normal curvature at the point is much greater than the cutter radius (i.e., \( R/R + r \approx 1 \)) or the local surface is isotropic (i.e., the normal curvature is constant at the point), the sign is negative, namely, Formula (3) tends to generate denser tool-path for points near boundary.

The difference between expressions (3) and the original one is a projection procedure

\[
l^s_i = k_1 l_u/\cos (\theta - \pi /2).
\]

Then, for iso-parametric tool-path, the side step is

\[
u_i + 1 = u_i + \frac{\Delta u}{|r_u|}.
\]

Never can all point clouds satisfy the previous conditions, i.e., for a few point clouds, the effect of boundary mapping on tool-path planning is uncertain. But actually, distortion caused by boundary mapping only appears at the points which are very close to the boundary or boundary corners. Thus, in this paper, the expression (4) instead of (3) is employed for paths near boundary. According to the experiments we conducted, the number of these paths is 3–5. This is rather conservative. A better way maybe choose the expression according to a criterion \( |\theta - \pi /2| < \eta \). However, we do not yet know of a principled way to choose the threshold for each point cloud.

Appendix B. Linearity between error of side interval and scallop height

As known, side interval and scallop height are related with the following expression

\[
l = \sqrt{8hr / R \pm r}.
\]

Then, we have

\[
(l_i + \Delta l_i)^2 = 8(h + \Delta h)rR / (R \pm r).
\]

Substitute the first expression into it and eliminate the high order component, having

\[
2l_i \Delta l_i = 8\Delta hrR / (R \pm r).
\]

Divide both side by \( l_i^2 \), giving

\[
|\Delta l_i/l_i| = |\Delta h|h \Rightarrow 2e = e.
\]

Finally, we can get a conclusion that the error of scallop height is twice that of side interval.
References


Qiang Zou is currently a Master’s student at the Chinese Academy of Sciences. His research interests include computational geometry for CAD/CAM, and computer graphics.

Jibin Zhao is a professor at Shenyang Institute of Automation, Chinese Academy of Sciences. He received his Ph.D. from the Chinese Academy of Sciences in 2004. His research focuses on geometric processing and solid modeling, computer graphics and NC machining.