Adaptive NC Path Generation From Massive Point Data With Bounded Error

Dongdong Zhang

Pinghai Yang

Xiaoping Qian[1]
e-mail: qian@iit.edu

Department of Mechanical, Materials and Aerospace Engineering,
Illinois Institute of Technology,
Chicago, IL 60616

This paper presents an approach for generating curvature-adaptive finishing tool paths with bounded error directly from massive point data in three-axis computer numerical control (CNC) milling. This approach uses the moving least-squares (MLS) surface as the underlying surface representation. A closed-form formula for normal curvature computation is derived from the implicit form of MLS surfaces. It enables the generation of curvature-adaptive tool paths from massive point data that is critical for balancing the trade-off between machining accuracy and speed. To ensure the path accuracy and robustness for arbitrary surfaces where there might be an abrupt curvature change, a novel guidance field algorithm is introduced. It overcomes potential excessive locality of curvature-adaptive paths by examining the neighboring points' curvature within a self-updating search bound. Our results affirm that the combination of curvature-adaptive path generation and the guidance field algorithm produces high-quality numerical control (NC) paths from a variety of point cloud data with bounded error.
[DOI: 10.1115/1.3010710]

Keywords: NC path generation, point-set surface, moving least-squares surface, curvature

1 Introduction

Rapid development of 3D sensing technology has led to the growing use of massive point data in product development, such as reverse engineering, rapid prototyping, manufacturing, and metrology. Such wide use of massive point data necessitates the research on direct processing of massive point data into suitable geometric models that can be used in downstream product design, analysis, and manufacturing. This paper presents an approach for generating adaptive finishing tool paths with bounded error directly from massive point data in three-axis CNC milling.

Existing approaches to path generation from massive point data face a dilemma: better quality (curvature-adaptive) paths but with painstaking computer-aided design (CAD) model reconstruction or easy (direct) path generation but without guarantee on accuracy or efficiency. More specifically,

1. Existing approaches to NC path generation typically involve a reverse engineering process, i.e., reconstructing the CAD models or sculptured surfaces (parametric surfaces, such as Beziers or nonuniform rational B-spline (NURBS) patches) from point data [1–3], and the traditional CAD model based NC tool path generation methods could then be used. Such a reverse engineering process contains various processing steps, such as smoothing, segmentation, and surface fitting, and it is laborious and far from automatic. Though some commercial CAD software, such as CATIA and IMAGEWARE, support the surface reconstruction from point data, high-quality surface reconstruction still requires the interactions from the users with advanced knowledge in surface modeling. Objects of complex geometry or noisy point cloud still pose critical challenges for automatic segmentation in reverse engineering.
2. Alternative approaches [4–8] that can directly generate tool paths from massive point data without CAD model reconstruction, due to the absence of an underlying surface model, produce NC tool paths strictly based on the discrete points that result in two problems. (1) Without a continuous surface, it is difficult to achieve the adaptive forward steps and path intervals, leading to poor machining efficiency. (2) It is difficult to control the machining accuracy when the data are noisy.

Therefore, in this paper, we present an approach that produces high-quality NC tool paths directly from massive point data without laborious CAD model reconstruction. More specifically, this approach uses the moving least-squares (MLS) surface as an underlying representation for the point-set surface. A closed-form formula for normal curvature computation is derived from the implicit form of MLS surfaces. It enables the generation of curvature-adaptive tool paths from massive point data that is critical for balancing the trade-off between machining accuracy and speed. To ensure the accuracy and robustness of the resulting paths for arbitrary surfaces where there might be an abrupt curvature change, a novel guidance field algorithm is introduced. It overcomes potential excessive locality of curvature-adaptive paths by examining the neighboring guidance points' curvature within a self-updating search bound.

Our point-set surface based approach represents many advantages over the existing approaches for generating NC paths from massive point data. Our approach inherits the MLS's intrinsic ability to handle noisy point data. The availability of an underlying surface definition, MLS surfaces, enables our approach to generate curvature-adaptive tool paths with bounded error, leading to efficient and accurate NC machining without the painstaking CAD model reconstruction.

Our main contributions in this paper are as follows.

- *Closed-form formula for computing normal curvatures in MLS surfaces.* Based on the implicit definition of MLS sur-

[1]Corresponding author.

Contributed by the Manufacturing Engineering Division of ASME for publication in the JOURNAL OF MANUFACTURING SCIENCE AND ENGINEERING. Manuscript received April 28, 2008; final manuscript received August 29, 2008; published online December 4, 2008. Review conducted by Burak Ozdoganlar. Paper presented at the ASME 2008 Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2008), Brooklyn, NY, July 6–August 3, 2008.

faces, we develop the closed-form formula for computing normal curvatures in MLS surfaces.

- *Curvature-adaptive path generation from massive point data.* The analytical equations of the normal curvature in MLS surfaces support the adaptive tool path generation, which greatly improves both machining efficiency and accuracy.
- *Novel guidance field algorithm with the self-updating search bound.* The introduction of a guidance field allows the shrinkage of forward steps when a larger normal curvature occurs in the forward machining region. Meanwhile, the side steps can also be adjusted by this guidance field, so the resulting curvature-adaptive paths are guaranteed to have bounded error even for surfaces with abrupt curvature changes.

Our results affirm that the combination of the curvature-adaptive path generation and the guidance field algorithm produces high-quality NC tool paths from a variety of point cloud data with bounded error.

The remainder of this paper is organized as follows. Section 2 introduces the related work in NC path generation from point data. Section 3 presents an overview of our approach. Section 4 derives the closed-form formula for computing normal curvatures in an MLS surface. Section 5 introduces our novel guidance field algorithm and the self-updating search bound. Implementation and illustrative examples are given in Sec. 6. Section 7 discusses potential issues in this approach. Finally, conclusions and future work are presented in Sec. 8.

2 Related Work

The study on NC tool path planning has had a very long history. Most of the research related to the sculptured surface machining assumes some exact mathematical representations [9–11]. However, thus far, there is limited work on tool path generation from measured point data. The prevalent approach is through reverse engineering where a sculpture surface representation is reconstructed. Approaches aiming to directly generate NC tool paths from point data without CAD model reconstruction are briefly reviewed below.

Direct NC path generation from point data. Lin and Liu [4] introduced a tool path generation approach directly from point data with a Z-map method where a constant Z-level point model was set up. A linear interpolation method was used to create the regular grid points linked up by line segments. In this method, a large memory was needed to store the grids, and how to control the machining accuracy was not discussed.

Feng and Teng [5] proposed a method where a cutter location-net (CL-net) was established by identifying the forward steps and side steps, and the cutter location points were the weighted averages of the related CL-net nodes. However, due to the absence of a mathematical representation of a continuous surface, the cusp height was approximated.

Chui and Lee [6] proposed a method where the spheres of the cutter radius rolled across the irregular nodal points and Boolean operations were utilized to trace the cutter locations. However, the machining accuracy was approximated and the adaptive tool paths could not be achieved.

Park and Chung [7] developed a 2D algorithm to calculate the finishing tool paths. However, it could only deal with organized point data.

With the algorithmic advancement of triangulating massive point data and with the wide usage of STL file [12], triangle mesh based methods have been used to generate tool paths. In Ref. [8], a mesh offset method for tool path generation was proposed. However, the triangulation process is still a complex task because prefiltering and postprocessing operations are still needed. The multitude of operations makes it difficult to maintain an error bound. Adaptive path generation on triangular mesh would be difficult since the curvature is not directly available on a polygonal object and curvature computing thus involves further approximation.

MLS surface. MLS surfaces have proven to be powerful and convenient in point based geometric processing and graphics applications [13–20]. Salient features of MLS surfaces include the ability of handling noise, up-sampling, down-sampling, and so on. Moreover, based on a more general definition of a projection process [15,16], a mathematical proof of the convergence of the projection procedure was presented [18,19]. The resulting MLS surface was proven to be isotopic to the original sampled surface.

Guidance field algorithm. Our guidance field approach differs from the one in Ref. [21] in two important aspects: we sample dense points on the planar paths instead of projecting the original input points onto the MLS surface; and we use a self-updating search bound to maintain both the curvature-adaptivity and machining accuracy.

3 Overview

In this paper, we use the isoplanar method to generate the finishing tool paths for three-axis milling with a ball-end tool, although other NC path generation strategies can also be used. Specifically, a series of parallel planes, called drive planes, are intersected with the MLS surface defined by the massive point cloud. The intersection curves are called cutter contact paths (CC paths), and the points on the CC paths are called cutter contact points (CC points), shown in Fig. 1. The cutter location paths (CL paths) are those that connect a series of cutter location points (CL points), which can be converted from CC points. Therefore, the CC point generation is the most important component in our approach.

Two critical objectives in NC machining are machining efficiency and machining accuracy. Machining efficiency is primarily determined by the number of tool motions, which consist of two ingredients: forward step and side step. Forward step stands for the line segment between two adjacent CC points, and the maximum allowable chordal deviation is called machining tolerance τ, shown in Fig. 2(*a*). Side step is the distance between two adjacent tool paths, and the maximum allowable error is called the cusp height η, shown in Fig. 2(*b*). The distance between two adjacent drive planes (called the tool path interval) is determined by the side step.

Curvature-adaptive path generation. In general, in order to achieve the high-efficiency NC machining, the forward steps and side steps need to be maximized while the machining accuracy

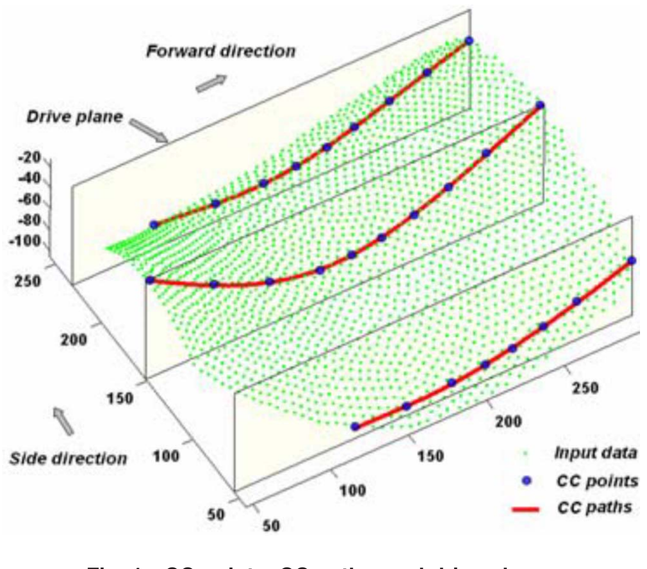


Fig. 1 CC points, CC paths, and drive planes

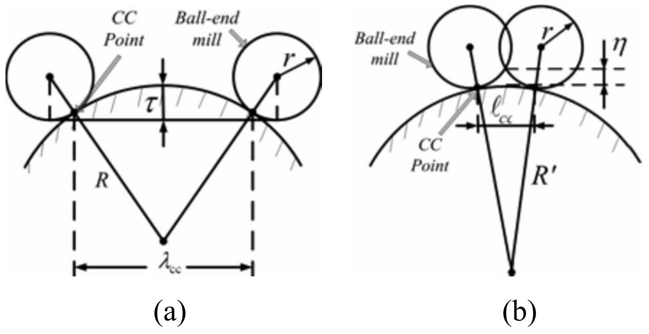


Fig. 2 Machining error: (*a*) tolerance τ: *R* is the normal curvature in the forward direction and λ_{cc} is the forward step; and (*b*) cusp height η: R' is the normal curvature in the side direction and ℓ_{cc} is the side step

needs to be maintained. Hence the curvature-adaptive forward steps and side steps are preferred.

Normal curvatures of CC points along two directions, forward direction (+*X* direction) and side direction (+*Y* direction), are the critical factors in determining forward steps and side steps in sculptured surface machining [1,9]. Based on the osculating circle assumption and the machining accuracy requirements, the forward step can be determined by the normal curvature along the forward direction (called the forward normal curvature) [9], shown as Eqs. (1) and (2).

For the convex case,

$$\lambda_{CC} = 2R\{1 - [(R + r - \tau)/(R + r)]^2\}^{1/2} \tag{1}$$

For the concave case,

$$\lambda_{CC} = 2R\{1 - [(R - r - \tau)/(R - r)^2]\}^{1/2} \tag{2}$$

In Eqs. (1) and (2), $R=1/k^N_{\text{forward}}$, where k^N_{forward} is the forward normal curvature, *r* is the radius of the mill, and τ is the machining tolerance.

Similarly, the side step can be determined by the normal curvature along the side direction (called the side normal curvature) [1], shown in Eqs. (3) and (4).

For the convex case,

$$\ell_{cc} = \frac{R'}{(R' + \eta)(R' + r)} \times \sqrt{2((R' + r)^2 + r^2)(R' + \eta)^2 - ((R' + r)^2 - r^2)^2 - (R' + \eta)^4} \tag{3}$$

For the concave case,

$$\ell_{cc} = \frac{R'}{(R' - \eta)(R' - r)} \times \sqrt{2((R' - r)^2 + r^2)(R' - \eta)^2 - ((R' - r)^2 - r^2)^2 - (R' - \eta)^4} \tag{4}$$

In Eqs. (3) and (4), $R'=1/k^N_{\text{side}}$, where k^N_{side} stands for the side normal curvature, *r* stands for the radius of the ball-end mill, and η stands for the cusp height. The tool path interval depends on both the side normal curvature and the slope in the side direction [2]. We show in Sec. 4 how we derive normal curvature on MLS surfaces.

Guidance field for overcoming extreme locality of the curvature effect. Although curvature-adaptive NC paths effectively resolve the trade-off between machining efficiency and accuracy, the locality of curvature may lead to biased paths when the surface geometry has an abrupt curvature change. For example, as shown in Fig. 3(*a*), the forward normal curvature is small at point A, leading to a long forward step to arrive at point B′. However, between A and B′ there exist regions with a much larger normal curvature, and the machining accuracy thus cannot be maintained.

Therefore, in this paper, we introduce a novel guidance field to proactively identify locations with excessive curvature change that may bias the paths. The guidance field provides a scalar field description of curvature distribution within the object surface. A guidance field algorithm would then intelligently search within a self-updating bound for the correct forward step. As shown in Fig. 3(*b*), the adjusted forward step from the guidance field algorithm is shorter than the initial forward step in the sharp curvature change area by querying the guidance field around point A. Detailed procedures are illustrated in Sec. 5.

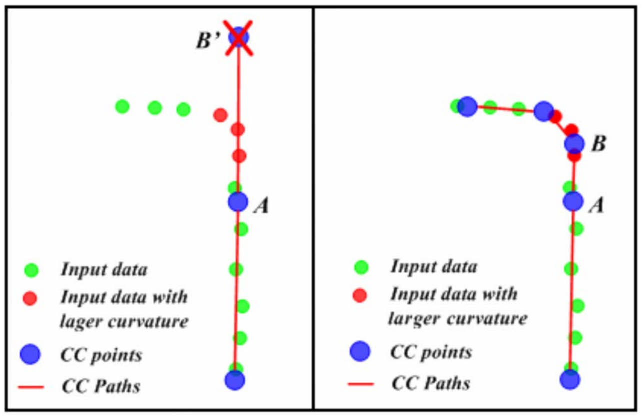


Fig. 3 Guidance field algorithm overcoming the excessive locality of the curvature effect: (*a*) abrupt changes of the curvature at point A leads to the next CC point at B′ resulting in the path bias; and (*b*) the guidance field algorithm checks the curvature in the front region and shrinks the forward step accordingly

4 Closed-Form Formula for Normal Curvature Computing in MLS Surfaces

The calculation of the normal curvature from massive point data is the key issue in our curvature-adaptive approach. In this paper, we use an MLS surface as the underlying surface, and the normal curvature is calculated based on the MLS surface.

4.1 Projection Based MLS Surfaces. The MLS surface was first introduced by Levin [13,14]. Based on Levin's work, Amenta and Kil [15,16] defined MLS surfaces as the local minimum of an energy function $e(\mathbf{y},\hat{\mathbf{n}}(\mathbf{x}))$ along the directions given by a vector field $\hat{\mathbf{n}}(\mathbf{x})$. The MLS surface is defined as

$$g(\mathbf{x}) = \hat{\mathbf{n}}(\mathbf{x})^T\left(\left.\frac{\partial e(\mathbf{y},\hat{\mathbf{n}}(\mathbf{x}))}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{x}}\right) = 0 \tag{5}$$

where **x** and **y** represent the coordinates of the spatial points, each defined as (x,y,z). The normalized $\hat{\mathbf{n}}(\mathbf{x})$ is determined by the normal vector $\hat{\mathbf{v}}_i$ of the nearby input point $\mathbf{q}_i \in Q$. If $\hat{\mathbf{v}}_i$ is unavailable, the covariance matrix can be established to calculate $\hat{\mathbf{v}}_i$ [22] so $\hat{\mathbf{n}}(\mathbf{x})$ can be defined as

$$\hat{\mathbf{n}}(\mathbf{x}) = \frac{\sum_{\mathbf{q}_i \in \mathbf{Q}} \hat{\mathbf{v}}_i G(\mathbf{x},\mathbf{q}_i)}{\|\sum_{\mathbf{q}_i \in \mathbf{Q}} \hat{\mathbf{v}}_i G(\mathbf{x},\mathbf{q}_i)\|}$$

where $G(\mathbf{x},\mathbf{q}_i)=e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}$ is a Gaussian weighting function, in which *h* is the scale factor that determines the width of the Gaussian kernel, discussed extensively in Refs. [17,18,23]. The energy function $e(\mathbf{y},\hat{\mathbf{n}}(\mathbf{x}))$ is defined as

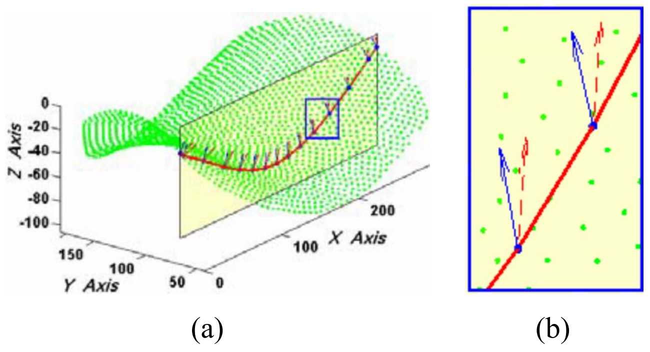


Fig. 4 Surface normal and curve normal. The dense points stand for the input data. The curve is the CC path and points on the CC path are the CC points. The dashed arrows represent the surface normal vectors of the CC points, and the solid ones stand for the curve normal vectors of the CC points.

$$e(\mathbf{y},\hat{\mathbf{n}}(\mathbf{x}_j)) = \sum\nolimits_{\mathbf{q}_i \in \mathbf{Q}} ((\mathbf{y}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}_j))^2 G(\mathbf{y},\mathbf{q}_i)$$

Therefore, by tracing the minimum of the local energy, we can project a spatial point onto the MLS surface. For details of this projection process, please refer to Refs. [15,16].

4.2 Normal Curvature Calculation. In our path generation approach, two normal curvatures for the forward step and the side step are required, as discussed in Sec. 3. In this section, we focus on computing the normal curvature.

We first compute the curve curvature of CC points along the intersection curve. Based on the surface normal and the curve normal of the CC points, we can then compute the corresponding forward normal curvature, as shown in Fig. 4.

In our work, the drive plane is parallel to the XZ plane and is denoted as $y=a$, where a is a constant value for a given drive plane. So the CC path on an MLS surface can be represented as

$$\begin{cases} g(\mathbf{x}) = \hat{\mathbf{n}}(x,y,z)^T\left(\left.\dfrac{\partial e(\mathbf{y},\hat{\mathbf{n}}(x,y,z))}{\partial \mathbf{y}}\right|_{(x,y,z)}\right) = 0 \\ y=a \end{cases} \tag{6}$$

Equation (6) can be simplified as

$$g(x,z) = \hat{\mathbf{n}}((x,a,z))^T\left(\left.\frac{\partial \hat{e}(\mathbf{y},\hat{\mathbf{n}}(x,a,z))}{\partial \mathbf{y}}\right|_{(x,a.z)}\right) = 0 \tag{7}$$

and

$$g(x,z) = \hat{\mathbf{n}}((x,a,z))^T\left(\left.\frac{\partial \hat{e}(\mathbf{y},\hat{\mathbf{n}}(x,a,z))}{\partial \mathbf{y}}\right|_{(x,a.z)}\right)$$
$$= \sum\nolimits_{\mathbf{q}_i \in \mathbf{Q}} 2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(1-\frac{1}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2\right) \cdot (\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}) \tag{8}$$

Hence the curvature on the planar curve, k_{curve}, can be computed [24]. Based on Eq. (8), we can denote the normal curvature of this planar curve as

$$k^N_{\text{forward}} = k_{\text{curve}} \cos\theta = -\frac{T(g(x,z))\cdot H(g(x,z))\cdot T(g(x,z))^T}{\|\nabla g(x,z)\|} \cdot \hat{\mathbf{n}}((x,y,z))\cdot\hat{\mathbf{n}}((x,a,z)) \tag{9}$$

where $-(T(g(x,z))\cdot H(g(x,z))\cdot T(g(x,z))^T/\|\nabla g(x,z)\|)$ represents the curvature of the implicit planar curve, $\hat{\mathbf{n}}((x,y,z))$ represents the unit normal vector of the MLS surface shown as the dashed arrows in Fig. 4, and $\hat{\mathbf{n}}((x,a,z))$ represents the unit normal vector of the implicit curve shown as the solid arrows in Fig. 4, so θ is the angle between $\hat{\mathbf{n}}((x,y,z))$ and $\hat{\mathbf{n}}((x,a,z))$. In Eq. (9),

$$\nabla g(x,z) = \begin{pmatrix} \dfrac{\partial g(x,z)}{\partial x} & \dfrac{\partial g(x,z)}{\partial z} \end{pmatrix}$$

is the gradient of $g(x,z)$, and the Hessian matrix is

$$H(g(x,z)) = \begin{pmatrix} \dfrac{\partial^2 g(x,z)}{\partial x\,\partial x} & \dfrac{\partial^2 g(x,z)}{\partial x\,\partial z} \\ \dfrac{\partial^2 g(x,z)}{\partial z\,\partial x} & \dfrac{\partial^2 g(x,z)}{\partial z\,\partial z} \end{pmatrix}$$

and $T(g(x,z))$ is the unit tangent vector of the implicit curve

$$T(g(x,z)) = \frac{\begin{pmatrix} -\dfrac{\partial g(x,z)}{\partial z} & \dfrac{\partial g(x,z)}{\partial x}\end{pmatrix}}{\left\|\begin{pmatrix} -\dfrac{\partial g(x,z)}{\partial z} & \dfrac{\partial g(x,z)}{\partial x}\end{pmatrix}\right\|} = \frac{\nabla g(x,z)}{\|\nabla g(x,z)\|}\cdot\begin{bmatrix} 0 & 1 \\ -1 & 0\end{bmatrix}$$

Therefore, the critical part is to calculate the gradient matrix $\nabla(g(x,z))$ and the Hessian matrix $H(g(x,z))$. From Eq. (8), we can get

$$\nabla(g(x,z)) = \sum\nolimits_{\mathbf{q}_i \in \mathbf{Q}} 2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(\frac{2}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x})) \cdot\left(\frac{1}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2-1\right)\cdot(\mathbf{x}-\mathbf{q}_i) + \left(1-\frac{3}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2\right)\cdot(\hat{\mathbf{n}}(\mathbf{x})+\nabla^T(\hat{\mathbf{n}}(\mathbf{x})) \cdot(\mathbf{x}-\mathbf{q}_i))\right)$$

The Hessian of $g(x,z)$ can be represented as

$$\begin{aligned} H(g(x,z)) = \nabla(\nabla(g(x,z))) = \sum\nolimits_{\mathbf{q}_i \in \mathbf{Q}} &-\frac{4}{h^2}e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(\frac{2}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))\cdot\left(\frac{1}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2-1\right)\cdot(\mathbf{x}-\mathbf{q}_i)\right. \\ &\left.+\left(1-\frac{3}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2\right)\cdot(\hat{\mathbf{n}}(\mathbf{x})+\nabla^T(\hat{\mathbf{n}}(\mathbf{x}))\cdot(\mathbf{x}-\mathbf{q}_i))\right)\cdot(\mathbf{x}-\mathbf{q}_i)^T + 2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(\frac{6}{h^4}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2-\frac{2}{h^2}\right) \\ &\cdot(\mathbf{x}-\mathbf{q}_i)\cdot(\hat{\mathbf{n}}^T(\mathbf{x})+(\mathbf{x}-\mathbf{q}_i)^T\cdot\nabla(\hat{\mathbf{n}}(\mathbf{x}))) + \frac{4}{h^2}e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))\cdot\left(\frac{1}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2-1\right) \\ &\cdot\mathbf{I} - \frac{12}{h^2}e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))\cdot(\hat{\mathbf{n}}(\mathbf{x})+\nabla^T(\hat{\mathbf{n}}(\mathbf{x}))\cdot(\mathbf{x}-\mathbf{q}_i))\cdot(\hat{\mathbf{n}}^T(\mathbf{x})+(\mathbf{x}-\mathbf{q}_i)^T\cdot\nabla(\hat{\mathbf{n}}(\mathbf{x}))) \end{aligned}$$

$$+ 2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(1-\frac{3}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2\right)\cdot(\nabla(\hat{\mathbf{n}}(\mathbf{x}))+\nabla^T(\hat{\mathbf{n}}(\mathbf{x}))+\nabla^T(\nabla(\hat{\mathbf{n}}(\mathbf{x})))\cdot(\mathbf{x}-\mathbf{q}_i))$$

where **I** is the identity matrix.

Similarly, the intersection curve between the MLS surface and the plane perpendicular to the drive plane can be expressed as

$$\begin{cases} g(\mathbf{x}) = \hat{\mathbf{n}}(x,y,z)^T\left(\left.\dfrac{\partial e(\mathbf{y},\hat{\mathbf{n}}(x,y,z))}{\partial \mathbf{y}}\right|_{(x,y,z)}\right) = 0 \\ x = b \quad (b \text{ is a constant value as to a certain plane}) \end{cases}$$

So the side normal curvature k^N_{side} can be represented as

$$k^N_{\text{side}} = -\frac{T(g(y,z))\cdot H(g(y,z))\cdot T(g(y,z))^T}{\|\nabla g(y,z)\|}\cdot \hat{\mathbf{n}}((x,y,z))\cdot \hat{\mathbf{n}}((b,y,z)) \tag{10}$$

Note that the signs of k^N_{forward} and k^N_{side} have the following explicit physical meanings: if they are negative, the curve is concave; otherwise, it is convex. These correspond to different formulas for the forward step and the side step as described in Sec. 3.

5 Guidance Field Algorithm

The purpose of introducing a guidance field in NC path generation is to maintain the error bound in NC paths, defined as the maximum allowable machining tolerance τ and the cusp height η, by overcoming potential excessive curvature locality for surfaces with abrupt curvature change.

The guidance field is a scalar function, which represents the distribution of the curvature effect over a spatial domain. In our approach, on a given drive plane we associate guidance points P_i (densely sampled points on the CC paths) with two kinds of guidance values: guidance tool path interval $\varsigma(P_i)$ and guidance forward step $\lambda(P_i)$. Guidance points are sampled with a user-defined point density by intersecting the given drive plane with the MLS surface. $\varsigma(P_i)$ represents the tool path interval calculated at every guidance point P_i, and the minimum of $\varsigma(P_i)$ serves as the final tool path interval, which determines the position of the next drive plane. Note that we use guidance points instead of CC points for the interval calculation because the tool path interval calculated only at CC points might miss some smaller tool path intervals calculated at other points on the given drive plane. $\lambda(P_i)$ stands for the forward step calculated at every guidance point P_i based on its forward normal curvature, and is used for determining the forward step at every CC point. We now detail how guidance points are sampled and how a self-updating search bound is used to adjust the forward steps.

5.1 Guidance Point Sampling. Due to the up-sampling ability of the MLS surface, we can sample dense guidance points on the drive plane to guarantee the machining accuracy, as shown in Fig. 5. The denser these points are, the more accurate the result would be. Without loss of generality, we assume by default that the guidance points be sampled at the average density of input point data (the distance between two adjacent guidance points is denoted as ω). The rationale for this is that we assume the input data are dense enough to represent the object. Detailed procedures for sampling points on a CC path on the MLS surface are described below.

Step 1. Define a subset I of input data near the drive plane. Project the subset data onto the drive plane and determine the

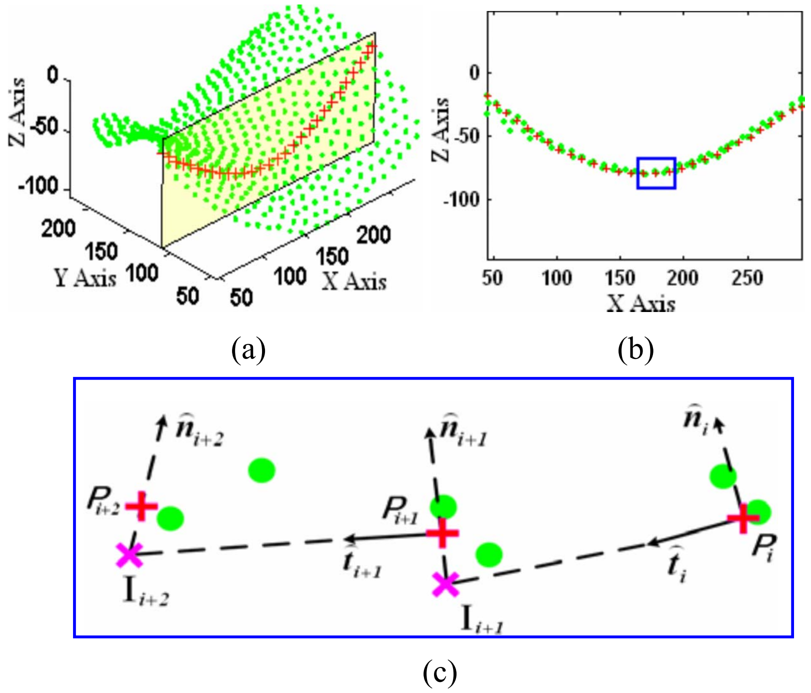


Fig. 5 Algorithm for sampling guidance points. (*a*) Display of guidance points in 3D; (*b*) guidance points on the drive plane; and (*c*) zoom-in of the frame in (*b*). The points stand for the input point data, and the crosses represent the guidance points. The forks are the intermediate points used to generate guidance points.

starting trace point I_S with the minimum X coordinate and the ending trace point I_E with the maximum X coordinate. Project I_S and I_E to the MLS surface and their corresponding projected points are denoted as P_S and P_E, which are the starting and ending guidance points.

Step 2. Trace all the guidance points P_i (i=1…n) from P_S to P_E. If $i \geq 1$ and $\|P_i - P_E\| < \delta$, stop the process and output $P_S, P_1, \ldots, P_n, P_E$ as the resulting guidance points on the given driven plane.

In this algorithm, the most critical issue is to generate the next guidance point P_{i+1} based on the current one P_i. At the current point P_i, we first set up a local coordinate system, whose directional axes are defined as the normal vector $\hat{\mathbf{n}}_i$ and its perpendicular direction vector $\hat{\mathbf{t}}_i$ separately, shown in Fig. 5(c). Then we define an intermediate point I_{i+1} along the direction vector $\hat{\mathbf{t}}_i$.

$$I_{i+1} = P_i + \omega \cdot \hat{\mathbf{t}}_i$$

where ω is the sample distance between two adjacent guidance points. Finally, we define the next guidance point P_{i+1} by projecting I_{i+1} onto the MLS surface.

In Sec. 4.1, the method of projecting a spatial point onto the MLS surface was discussed. Here, we introduce the algorithm for projecting points onto a planar CC path on the MLS surface. Take I_{i+1} as an example. We can define the planar normal vector $\hat{\mathbf{n}}_{i+1}$ of I_{i+1}, shown in Fig. 5(c), then we can trace the MLS point along $\hat{\mathbf{n}}_{i+1}$. Note that any point $\mathbf{x}$ on the MLS surface should project to itself, which is defined as

$$\|\psi_P(\mathbf{x}) - \mathbf{x}\| = 0$$

where $\psi_P(\mathbf{x})$ represents the corresponding projected point of $\mathbf{x}$ [13,14], so the problem is transformed into finding the point that can project to itself along the direction vector $\hat{\mathbf{n}}_{i+1}$. How to find this point along the line becomes the same issue as the line/MLS surface intersection described in Ref. [25].

5.2 Forward Guidance Field. The determination of the forward step is divided into two phases: initial forward step determination and final forward step determination. For an arbitrary CC point Q_j, the initial forward step $\lambda(Q_j)$ is calculated based on its forward normal curvature. Then the guidance field in an updating search bound needs to be checked to adjust the initial forward step. If there is a smaller guidance value λ in the search bound, the initial forward step should shrink to guarantee the machining error bound.

The critical issue in this algorithm is how to define the search bound. When the bound is too small, the guidance field may not be able to overcome the locality of curvature effect. When the bound is too large, it may lead to the loss of the curvature-adaptivity of the tool paths. Therefore, in this paper, we propose a self-updating search bound to guarantee the machining accuracy without affecting the curvature-adaptivity of the tool paths. Detailed procedures are illustrated below.

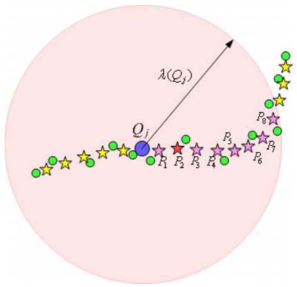
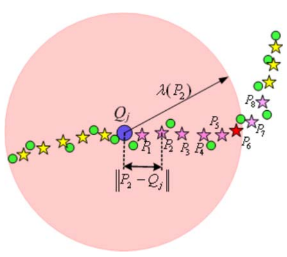
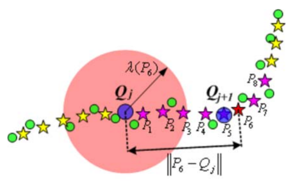
For a CC point Q_j, use the initial forward step $\lambda(Q_j)$ as the first search bound $\chi(Q_j)$ around CC point Q_j, i.e., $\chi(Q_j)=\lambda(Q_j)$.

Step 1: Locating guidance points within the current search bound. In this search bound, sort the guidance points as P_i (i =1…n) based on their Euclidean distance to the CC point Q_j. Their corresponding guidance values are denoted as $\lambda(P_i)$ (i =1…n). Note that we only consider the guidance points in the forward direction.

Step 2: Comparing the search bound and the guidance values within the search bound. Compare $\chi(Q_j)$ with the guidance values for points within the search bound, i.e., from $\lambda(P_1)$ to $\lambda(P_n)$.

Step 2a. If there exists some $\lambda(P_i)$ that is smaller than $\chi(Q_j)$, then this suggests that the current forward step cannot guarantee the accuracy, so we have to adjust the forward step. Pick the first guidance point P_k whose guidance value $\lambda(P_k)$ is smaller than the current search bound $\chi(Q_j)$. Then the search bound $\chi(Q_j)$ is up-

Table 1 Illustration of the guidance field algorithm

Diagram	Steps
	**STEP1**: Find the guidance points $P_1 \ldots P_8$ within the search bound $\chi(Q_j)=\lambda(Q_j)$. The initial search bound $\chi(Q_j)$ is calculated based on the forward normal curvature at current CC point Q_i from Eq.(1) or Eq.(2). **STEP2.a**: Assume that among the guidance points $P_1 \ldots P_8$, P_2 is the first point with the guidance value $\lambda(P_2)$ smaller than the initial search bound $\chi(Q_j)$, so we set $\lambda(P_2)$ as the new search bound $\chi(Q_j)=\lambda(P_2)$.
	**STEP3**: Assume that $\chi(Q_j)=\lambda(P_2)>\|P_2-Q_j\|$, so the final forward step could be longer than $\|P_2-Q_j\|$. Then we use the search bound $\chi(Q_j)=\lambda(P_2)$ to repeat STEP 2.a. **STEP2.a** Assume that among the guidance points $P_1 \ldots P_8$, P_6 is the first one with the guidance value $\lambda(P_6)$ smaller than the search bound $\chi(Q_j)=\lambda(P_2)$. Hence the new search bound $\chi(Q_j)$ is updated as $\lambda(P_6)$ and we set $\chi(Q_j)=\lambda(P_6)$.
	**STEP3**: We assume that $\chi(Q_j)=\lambda(P_6)<\|P_6-Q_j\|$, so we use $\|P_5-Q_j\|$ as the final forward step and we obtain the next CC point Q_{j+1}.

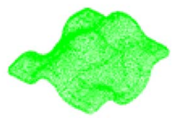
dated to $\lambda(P_k)$. Go to Step 3.

Step 2b. If there is no $\lambda(P_i)$ smaller than $\chi(Q_j)$, then this suggests that all the points in the search bound have a smaller forward normal curvature than that at Q_j. Therefore, the initial forward step $\lambda(Q_j)$ is fine and the final forward step $\lambda_{\text{final}}=\lambda(Q_j)$. We thus can find the next CC point Q_{j+1}.

Step 3: Comparing the search bound and distances to the previously obtained guidance points. By comparing the distances $\|P_{k-1}-Q_j\|$, $\|P_k-Q_j\|$, and by the updating search bound $\chi(Q_j)$, we have the following three scenarios.

(1) If $\chi(Q_j)\leq\|P_{k-1}-Q_j\|$, then the final forward step $\lambda_{\text{final}}=\|P_{k-1}-Q_j\|$ and the next CC point Q_{j+1} can be found.
(2) If $\|P_{k-1}-Q_j\|<\chi(Q_j)\leq\|P_k-Q_j\|$, then the final forward step $\lambda_{\text{final}}=\chi(Q_j)=\lambda(P_k)$ and the next CC point Q_{j+1} can be found.
(3) If $\chi(Q_j)>\|P_k-Q_j\|$, then the forward step could be longer, so we update the search bound $\chi(Q_j)=\lambda(P_k)$ and go to Step 2.

Table 2 Point data for a compound surface

Name	Size (mm)	Number of points	Noise	Data
Compound surface	X:100.0 Y:100.0 Z:46.5	16,286	Standard deviation $\sigma=0.15$	

In Table 1, we give an example to illustrate how to find the next CC point Q_{j+1}. The small points are the input data, the big points are the CC points, and the pentacles are the guidance points. The circles represent the self-updating search bounds. Note that $\lambda(P_i)$ is calculated based on the forward normal curvature at the guidance points P_i before the CC points are traced.

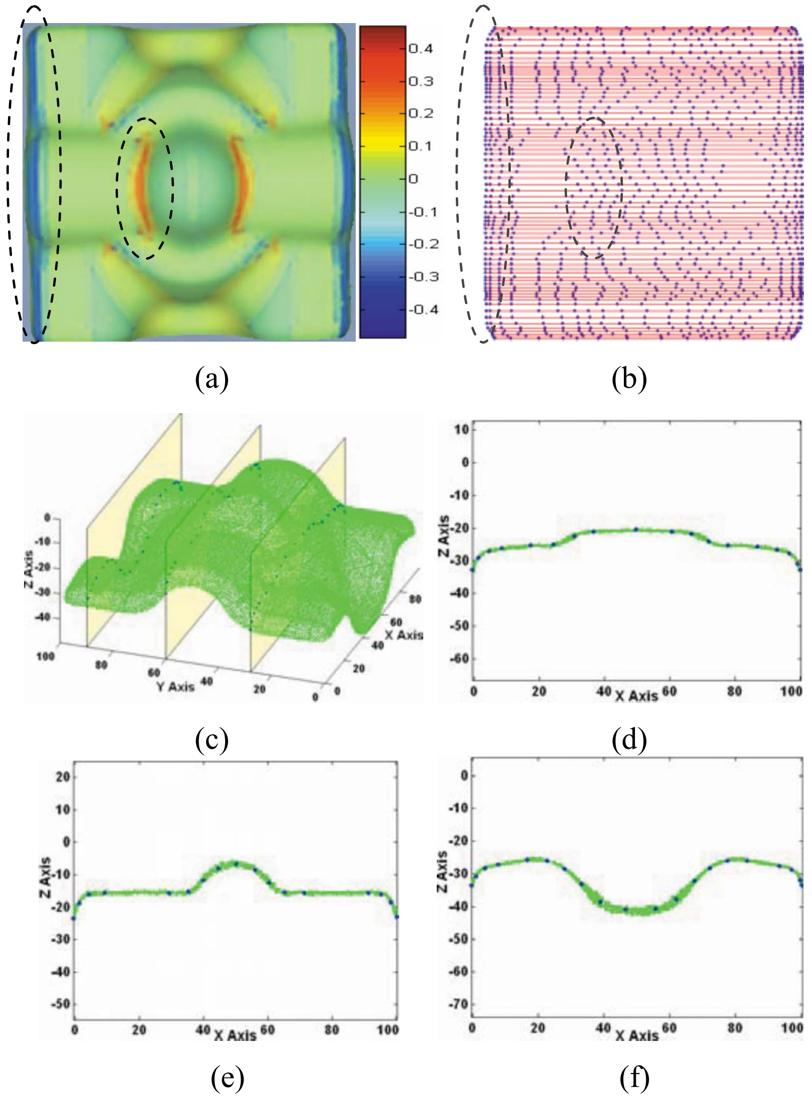
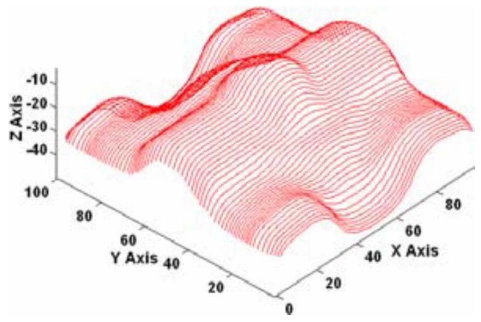
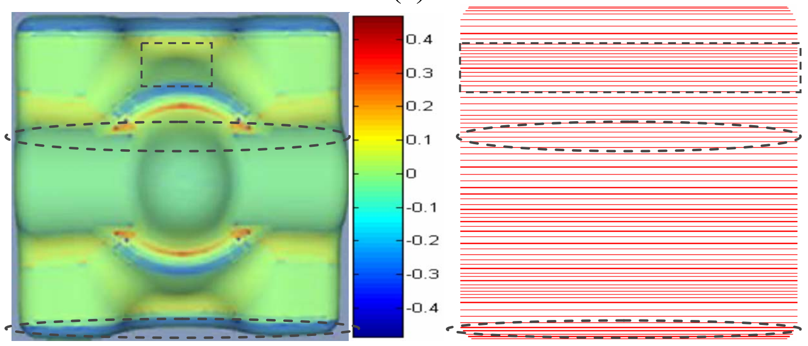


Fig. 6 Curvature-adaptive 2D contour generation on the compound surface: (*a*) forward normal curvature display; (*b*) top-view of CC paths (parallel lines) with CC points; (*c*) isoview of input data and three drive planes; (*d*) 2D contour on the plane *Y*=28 mm; (*e*) 2D contour on the plane *Y*=60 mm; and (*f*) 2D contour on the plane *Y*=90 mm

5.3 Determining Tool Path Intervals. With the sampled guidance points on the current drive plane $P_S, P_1, \ldots, P_n, P_E$, the guidance tool path interval $\varsigma(P_i)$ can be assigned to every guidance point P_i, using the following procedures. First of all, the side normal curvature $k^N_{\text{side}}(P_i)$ at the guidance point P_i can be calculated from Eq. (10). Then the side step at point P_i, denoted as $\ell(P_i)$, is determined from Eq. (3) or Eq. (4). With the discussion in Ref. [2], the slope information and the side step $\ell(P_i)$ are combined to determine the guidance tool path interval $\varsigma(P_i)$ at every guidance point P_i. Therefore, the final tool path interval is defined as the minimum of $\varsigma(P_i)$, represented as $\varsigma = \min(\varsigma(P_i))$, $i \in [S, 1, \ldots, n, E]$, and the position of the next drive plane could

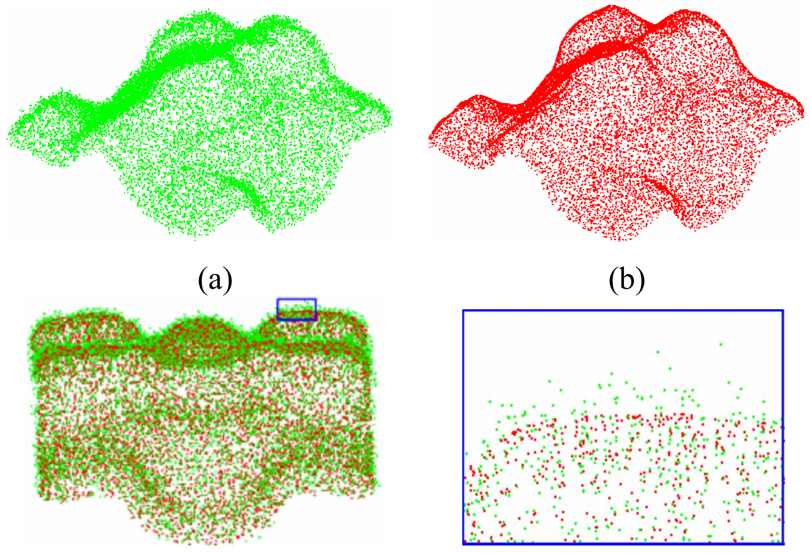


(a)



(b)

Fig. 7 Curvature-adaptive tool path intervals of the compound surface: (*a*) CC paths generated from point data and (b) display of the side normal curvature and adaptive tool path intervals. Dashed rectangles stand for the regions with a high slope in the side direction.



(a) (b)

(c) (d)

Fig. 8 Data smoothing from the MLS surface: (*a*) input data with noise ($\sigma = 0.6$); (*b*) points after projecting onto MLS surface; (*c*) overlap of input data and the projected points; and (*d*) zoom-in of the frame in (*c*)

Table 3 Point data for a human face

Name	Size (mm)	Number of points	Noise	Data
Human Face	X:101.9 Y:140.2 Z:43.9	17,938	Standard deviation $\sigma=0.15$	

then be determined. Note that we use guidance points instead of CC points for the interval calculation because the tool path interval calculated only at CC points might miss some smaller tool path intervals calculated at other points on the given drive plane.

6 Implementation and Examples

We implemented the curvature-adaptive NC path generation and the guidance field algorithms in MATLAB 7.0.1. In this section, two examples are presented below to illustrate and validate the proposed approach.

Example 1: Compound surface. Table 2 shows the massive point data (16,286 points) for a compound surface with noise (standard deviation $\sigma=0.15$).

To validate the curvature-adaptive CC point generation algorithm, we show the forward normal curvature (Fig. 6(*a*)) and the adaptive forward steps (Fig. 6(*b*)) based on the massive point data of the compound surface.

In Figs. 6(*a*) and 6(*b*), it can be seen that high curvature areas

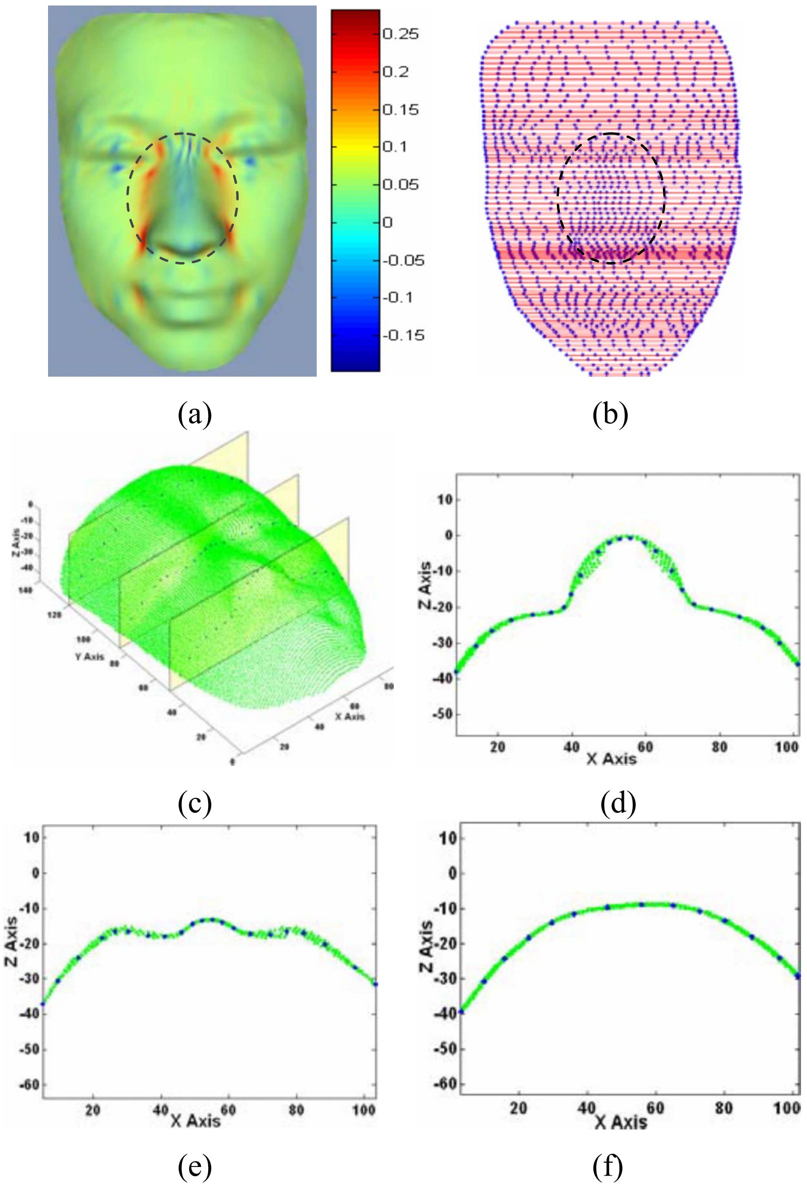


Fig. 9 Curvature-adaptive path generation on the human face: (*a*) forward normal curvature display; (*b*) top-view of CC paths (parallel lines) with CC points; (*c*) isoview of input data and three drive planes; (*d*) 2D contour on the plane $Y=50$ mm; (*e*) 2D contour on the plane $Y=85$ mm; and (*f*) 2D contour on the plane $Y=120$ mm

have smaller forward steps. Figures 6(*d*)–6(*f*) further illustrate the curvature-adaptivity and the effect of the guidance field for paths at three different drive planes ($Y=50$ mm, $Y=85$ mm, and $Y=120$ mm) where the distribution of the CC points is adaptive to the local curvature, and the resulting paths are accurate even at a sharp transition.

In our algorithm, we also achieve curvature-adaptive tool path intervals. Furthermore, we use guidance points to calculate the tool path intervals instead of CC points. As shown in Fig. 7, the large side normal curvature corresponds to the small tool path intervals. Note that the dense tool path intervals also happens at the regions with a high slope in the side direction.

We also use the synthetic data with high noise (standard deviation $\sigma=0.6$) to illustrate MLS's noise-handling ability. In Fig.

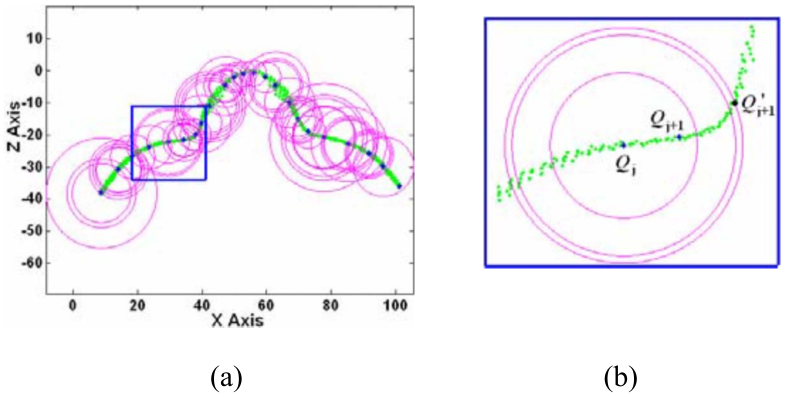


Fig. 10 Self-updating search bound in the guidance field algorithm: (*a*) display of the bound self-updating process and (*b*) zoom-in of the frame in (*a*)

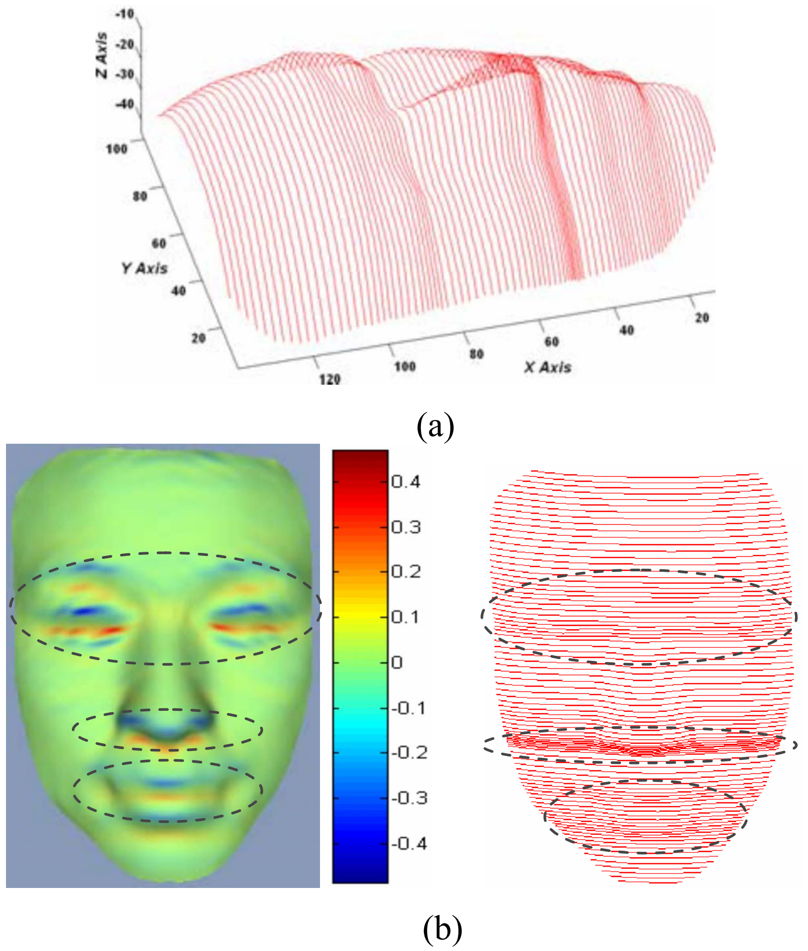


Fig. 11 Curvature-adaptive tool path intervals of the human face: (*a*) CC path generation from point data and (*b*) display of the side normal curvature and adaptive tool path intervals

8(*a*), we can find noisy data. By projecting the input points onto the MLS surface, we can see that the resulting compound surface is much smoother (Fig. 8(*b*)–8(*d*)).

Example 2: Human face. Table 3 shows the massive point data (17,938 points) for a human face with noise (standard deviation $\sigma=0.15$).

The forward normal curvature is displayed in Fig. 9(*a*), and the corresponding curvature-adaptive forward steps are shown in Fig. 9(*b*). Three paths are shown to illustrate the curvature-adaptivity and the guidance field effect (Figs. 9(*c*)–9(*f*)).

In our approach, we use the guidance field algorithm with the self-updating bound to guarantee the machining accuracy. Figure 10 presents a snapshot of how the guidance field works for the drive plane in Fig. 9(*d*). The small points stand for the input point data and the big ones are the adaptive CC points. The circles represent the self-updating search bounds. As in Fig. 10(*b*), Q_j is the current CC point, so based on its forward normal curvature, the next CC point should be point Q'_{j+1}. However, between Q_j and Q'_{j+1}, there exists a point with a larger normal curvature, so the final forward step shrinks and a correct CC point Q_{j+1} is identified. The results in Figs. 6 and 9 both demonstrate that the resulting paths are both curvature-adaptive and accurate due to the use of the guidance field algorithm. Figure 11 presents the resulting curvature-adaptive tool path intervals.

With the above algorithms, we generated the NC codes directly from the input point cloud of the human face for a HASS three-axis milling machine. Note that the cutter location points are determined by the CC points, the CC point normal, and the Z-axis. The part shapes from both the semifinishing and the finishing cut are shown in Fig. 12. The paths were generated from our software with different machining tolerances and cusp heights for the semifinishing and the finishing cut.

7 Discussion

In our algorithm, the accuracy of the computed normal curvature determines the final tool paths. By using the MLS surface as

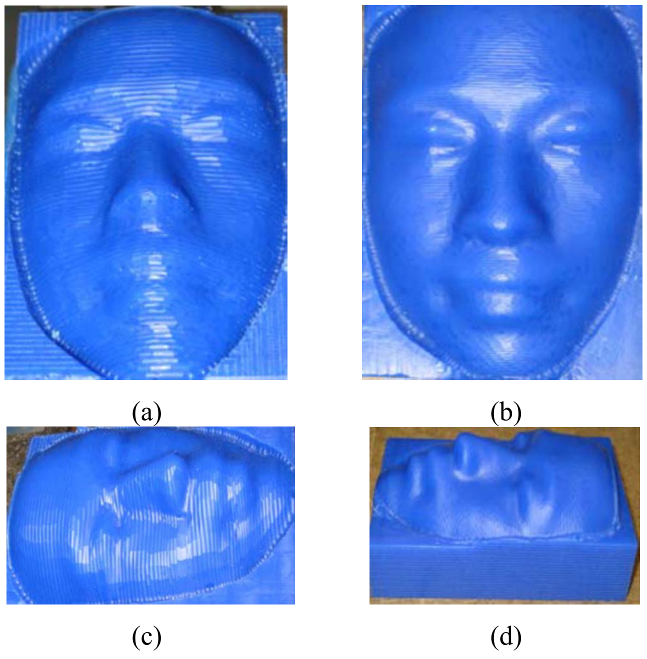


Fig. 12 Actual NC machining results: (*a*) semifinishing machining result: the machining tolerance is 0.2 mm and the cusp height is 0.3 mm; (*b*) finishing machining result: the tolerance and cusp height are 0.1 mm; (*c*) right-view of the semifinishing machining result; and (*d*) left-view of the finishing machining result

Table 4 Results of *k* and *k* errors

	k^N_{forward} mean error	k^N_{forward} std.	k^N_{side} mean error	k^N_{side} std.
Constant $h=0.5$	−0.0114816	0.0199886	−0.00796966	0.0248328
Constant $h=5.0$	0.0117802	0.0600113	0.0146924	0.0640671
Adaptive h	−0.00711186	0.0145441	−0.00394626	0.0121565

the underlying surface representation, the Gaussian kernel h serves as the most critical factor in determining the normal curvature. Therefore, in this section, we discuss the relationships between h and normal curvature as well as the final CC tool paths.

The appropriate h value for accurate curvature estimation obviously depends on the local density and local curvature information. Therefore, only a single constant h may not be suitable for global regions. Both the large and small h could lead to accuracy loss in surface representation and curvature computation as discussed below.

- A smaller h could cause instabilities due to the quick fall-off of the Gaussian weighting function, which would then result in a larger magnitude of the calculated curvature in comparison with the corresponding true curvature.
- A larger h could cause excessive smoothing in regions with small features, which would then result in a smaller magnitude of the calculated curvature in comparison to the corresponding true curvature.

This problem was discussed in detail in Ref. [26]. Based on this discussion, we developed a new experimental formula of h, shown as Eq. (11), to improve the accuracy of MLS in surface representation and curvature calculation

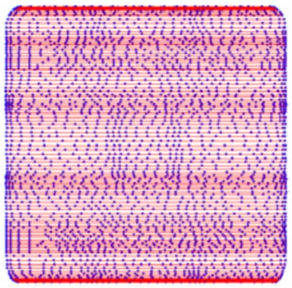
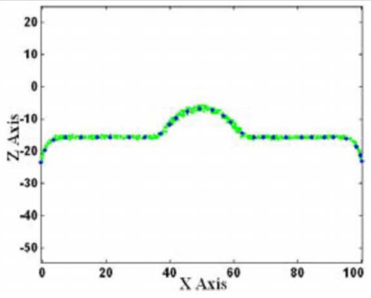
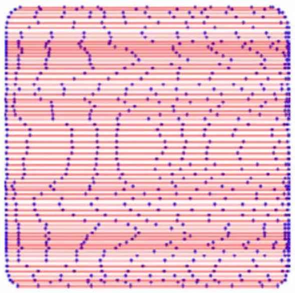
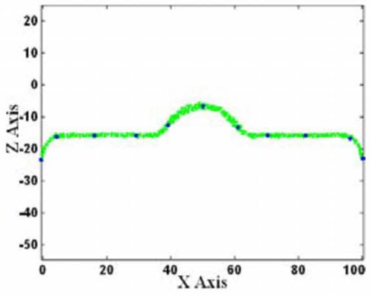
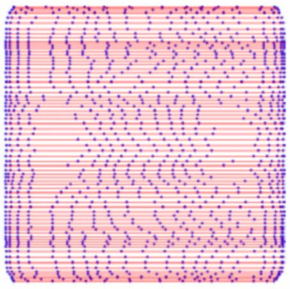
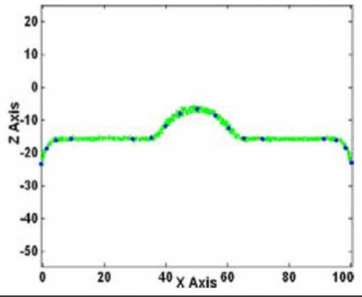
$$h=\frac{\max\|\mathbf{x}-\mathbf{q}_i\|}{\delta} \tag{11}$$

where $\mathbf{x}$ is the point whose Gaussian kernel h needs to be evaluated. $\mathbf{q}_i$ represents the nearest neighboring point of $\mathbf{x}$ and the number of neighbors is denoted as N, so $i\in[1,N]$. $\max\|\mathbf{x}-\mathbf{q}_i\|$ stands for the δ-fold multiplication of the Gaussian kernel h. We use $N=90$ and $\delta=4$ for all the examples in this paper. To validate our adaptive h strategy, we adopt the same synthetic data in Example 1 (compound surface) to evaluate the curvature errors between the computed curvature values and the corresponding true values, which are tabulated in Table 4.

From Table 4, we can see that the normal curvature relies on the selection of h. And our adaptive h gives the best performance in defining the normal curvature, and it produces the best final CC tool paths. Table 5 gives the graphical comparison of CC tool paths with the constant h and adaptive h separately.

From Table 5, we can see that the selection of h has a significant effect on the final CC tool paths: a smaller h causes denser forward steps and smaller tool path intervals due to the quick fall-off of the Gaussian weighting function, which then results in a larger magnitude of the calculated curvature in comparison to the corresponding true curvature. In this case, though the machining accuracy is guaranteed, the adaptivity of tool paths is almost lost. On the other hand, a larger h results in excessive smoothing in regions with small features, and produces a smaller magnitude of the calculated curvature when compared with the corresponding true curvature. Finally our adaptive h produces the most appropriate Gaussian weighing function at different local regions, resulting in the most accurate normal curvature. Therefore, our adaptive h produces the desirable tool paths that not only guarantees the machining accuracy, but also improves the machining efficiency through the curvature-adaptivity.

Table 5 Final CC tool paths with different *h*

	CC tool paths	Y=60mm intersection
Constant *h* (*h*=0.5)		
Constant *h* (*h*=5.0)		
Adaptive *h*		

8 Conclusions and Future Work

This paper presents a novel approach for directly generating curvature-adaptive tool paths with bounded error from massive point data without CAD model reconstruction. At its core is the use of a moving least-squares surface as the underlying surface representation for the massive point data. We derived a closed-form formula for computing the normal curvature in the MLS surface. We developed a new guidance field algorithm with a self-updating search bound that can effectively overcome the excessive locality of the curvature effect. Experimental results on both the simulated paths and actual machining demonstrate that the combination of the curvature-adaptive path generation and the guidance field algorithm produces high-quality NC paths from a variety of noisy point cloud data.

Our further work will include the quantification of the accuracy and efficiency saving of our approach over the current CAD model reconstruction based approach in NC path generation.

Acknowledgment

The financial support from National Science Foundation Grant Nos. 0529165 and 0800912 and Air Force Office of Scientific Research Grant No. FA9550-07-1-0241 are gratefully acknowledged.

References

[1] Choi, Y. K., Banerjee, A., and Lee, J. W., 2007, "Tool Path Generation for Free Form Surfaces Using Bezier Curves/Surfaces," Comput. Ind. Eng., **52**(4), pp. 486–501.
[2] Peng, Y. H., and Yin, Z. W., 2008, "The Algorithms for Trimmed Surfaces Construction and Tool Path Generation in Reverse Engineering," Comput. Ind. Eng., **54**(3), pp. 624–633.
[3] Yin, Z., 2004, "Adaptive Tool Path Generation From Measured Data," Proc. Inst. Mech. Eng., Part B, **218**(1), pp. 103–111.
[4] Lin, A. C., and Liu, H. T., 1998, "Automatic Generation of NC Cutter Path From Massive Data Points," CAD, **30**(1), pp. 77–90.
[5] Feng, H. Y., and Teng, Z. J., 2005, "Iso-Plannar Piecewise Linear NC Tool Path Generation From Discrete Measured Data Points," CAD, **37**(1), pp. 55–64.
[6] Chui, K. L., and Lee, T. C., 2002, "Direct Tool-Path Generation From Massive Point Input," Proc. Inst. Mech. Eng., Part B, **216**(2), pp. 199–206.
[7] Park, S. C., and Chung, Y. C., 2003, "Tool-Path Generation From Measured Data," CAD, **35**(5), pp. 467–475.
[8] Kim, S. J., and Yang, M. Y., 2005, "Triangular Mesh Offset for Generalized Cutter," CAD, **37**(10), pp. 999–1014.
[9] Choi, B. K., and Jerard, R. B., 1998, *Sculptured Surface Machining: Theory and Applications*, Kluwer, Dordrecht.
[10] Choi, Y. K., and Banerjee, A., 2007, "Tool Path Generation and Tolerance Analysis for Free-Form Surfaces," Int. J. Mach. Tools Manuf., **47**(3–4), pp. 689–696.
[11] Ding, S., Mannan, M. A., Poo, A. N., Yang, D., and Han, Z., 2003, "Adaptive Isoplanar Tool Path Generation for Machining of Free-Form Surfaces," CAD, **35**(2), pp. 141–153.
[12] Lee, S. H., Kim, H. C., Hur, S. M., and Yang, D. Y., 2002, "STL File Generation From Measured Point Data by Segmentation and Delaunay Triangulation," CAD, **34**(10), pp. 691–704.
[13] Levin, D., 1998, "The Approximation Power of Moving Least-Squares," Math. Comput., **67**(224), pp. 1517–1531.
[14] Levin, D., 2003, "Mesh-Independent Surface Interpolation," *Geometric Modeling for Scientific Visualization*, Springer-Verlag, Berlin, pp. 37–49.
[15] Amenta, N., and Kil, Y. J., 2004, "Defining Point-Set Surfaces," ACM Trans.

Graphics, **23**(3), pp. 264–270.
[16] Amenta, N., and Kil, Y.-J., 2004, “The Domain of a Point Set Surface,” Eurographics Symposium on Point-Based Graphics, pp. 139–147.
[17] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., and Levin, D., 2001, “Point Set Surfaces,” *Proceedings of the Conference on Visualization ‘01, SESSION: Session P1: Point-Based Rendering and Modeling*, pp. 21–28.
[18] Dey, T. K., Goswami, S., and Sun, J., 2005, “Extremal Surface Based Projections Converge and Reconstruct With Isotopy,” Technical Report No. OSU-CISRC-4–05–TR25.
[19] Dey, T. K., and Sun, J., 2005, “Adaptive MLS Surfaces for Reconstruction With Guarantees,” Eurographics Symposium on Geometry Processing, pp. 43–52.
[20] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T., 2003, “Computing and Rendering Point Set Surfaces,” IEEE Trans. Vis. Comput. Graph., **9**(1), pp. 3–15.
[21] Scheidegger, C. E., Fleishman, S., and Silva, C. T., 2005, “Triangulating Point Set Surfaces With Bounded Error,” Eurographics Symposium on Geometry Processing, pp. 63–72.
[22] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., 1992, “Surface Reconstruction From Unorganized Points,” Comput. Graph., **26**(2), pp. 71–78.
[23] Pauly, M., 2003, “Point Primitives for Interactive Modeling and Processing of 3D Geometry,” Ph.D. thesis, Computer Science Department, ETH Zurich.
[24] Goldman, R., 2005, “Curvature Formulas for Implicit Curves and Surfaces,” Comput. Aided Geom. Des., **22**(7), pp. 632–658.
[25] Yang, P., and Qian, X. P., 2008, “Adaptive Slicing of Moving Least Squares Surfaces: Toward Direct Manufacturing of Point Set Surfaces,” ASME J. Comput. Inf. Sci. Eng., **8**(3), p. 031003.
[26] Yang, P. H., and Qian, X. P., “Direct Computing of Surface Curvatures for Point-Set Surfaces,” *Proceedings of 2007 IEEE/Eurographics Symposium on Point-Based Graphics (PBG)*, Prague, Czech Republic, Sept. 2007.