Adaptive Slicing of Moving Least Squares Surfaces: Toward Direct Manufacturing of Point Set Surfaces

Pinghai Yang

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

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

Rapid advancement of 3D sensing techniques has led to dense and accurate point cloud of an object to be readily available. The growing use of such scanned point sets in product design, analysis, and manufacturing necessitates research on direct processing of point set surfaces. In this paper, we present an approach that enables the direct layered manufacturing of point set surfaces. This new approach is based on adaptive slicing of moving least squares (MLS) surfaces. Salient features of this new approach include the following: (1) It bypasses the laborious surface reconstruction and avoids model conversion induced accuracy loss. (2) The resulting layer thickness and layer contours are adaptive to local curvatures, and thus it leads to better surface quality and more efficient fabrication. (3) The curvatures are computed from a set of closed formula based on the MLS surface. The MLS surface naturally smoothes the point cloud and allows upsampling and downsampling, and thus it is robust even for noisy or sparse point sets. Experimental results on both synthetic and scanned point sets are presented.

[DOI: 10.1115/1.2955481]

Keywords: moving least squares surface, point-set surface, curvatures, adaptive slicing, layered manufacturing, direct manufacturing

1 Introduction

Rapid advancement of 3D sensing technologies and rising demands for individualized manufacturing have spurred the growing need for geometric processing of point clouds, a process that converts data output from 3D sensors into suitable geometric models that can be used in downstream product design, analysis, and manufacturing. On the one hand, various 3D sensors of different modalities, such as tactile, optical, X-ray, magnetic, and acoustic sensors, can now readily output dense and accurate point cloud of a physical object. On the other hand, the use of geometric models of existing physical objects is becoming increasingly common in the product development as evidenced by the growing need for reverse engineering of physical artifacts in automotive, aerospace, and consumer product industries, increasing practice of patient-specific biomedical implants, customer-specific product design and manufacturing (e.g., apparel and footwear), and service-condition based product repair.

Current approaches in geometric processing of point cloud for subsequent manufacturing are illustrated in Fig. 1 (in this paper, we focus on additive manufacturing processes, hereinafter layered manufacturing (LM)). The prevalent approach involves surface reconstruction where the discrete point cloud (point set) is converted into certain high order continuous surface representations, e.g., conical surfaces or *B*-spline surface patches. The reconstructed surface model is subsequently discretized into a .stl file for LM. However, the surface reconstruction process, a process that involves segmentation and surface fitting, is laborious and far from being fully automatic. The multitude of model conversions from point sets, to continuous surfaces, to a discrete model (.stl), and to the LM process model (a stackup of 2.5*D* layers) lend itself vulnerable to accuracy loss. An alternative approach is through triangulation of point cloud into a .stl file. This approach also involves the model conversion from .stl to the LM process model. As such, the need for accurate .stl model often leads to the size of triangles to be extremely small, which negates the benefit of triangular representation of surface shapes.

Recently, a new approach that directly processes point cloud for LM emerges. For example, in LM, several direct slicing methods have been reported [1–4]. However, these direct slicing methods employed a projection process that projects the neighboring points within Δd distance away from the slicing plane onto the plane; a trade-off between the projection error and the truncation error is thus introduced. For example, when slicing the rabbit data shown in Fig. 2(*a*), an appropriate plane distance Δd is hard to find due to this tradeoff. When Δd is too large, the projection error may become large (depending on the local normal direction), which is illustrated by the large bandwidth in Fig. 2(*b*); when Δd is small, the truncation error becomes big, which is illustrated by the sparse data and the discretized loss of high frequency (sharp corners) information in Fig. 2(*c*).

In this paper, we present an approach that can generate a LM model directly from the point cloud. This approach is based on adaptive slicing of moving least squares (MLS) surfaces from the point set. It effectively circumvents the tradeoff between the truncation error and the projection error. Instead of using the projection, this algorithm employs an intersection method that generates each 2D contour by directly intersecting a slicing plane with the underlying MLS surface defined by the input point data. In comparison with the current direct slicing methods, our new method tends to give more accurate 2D contours, as shown in Fig. 2(*d*), due to the following reasons: (1) The projection error is avoided by employing an intersection method instead of the projection method. Meanwhile, the accuracy of the intersection is precisely controlled by any given error bound. (2) The truncation error is reduced by upsampling enabled by MLS. While in the projection

[1]Corresponding author.

Contributed by the Computer-Aided Product Development (CAPD) Committee for publication in the JOURNAL OF COMPUTING INFORMATION SCIENCE AND ENGINEERING. Manuscript received May 18, 2007; final manuscript received February 28, 2008; published online August 19, 2008. Assoc. Editor: A. Fischer.

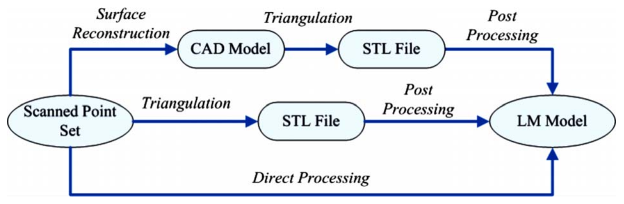


Fig. 1 Approaches for geometric processing of scanned point cloud data for LM

based methods, the density of projected points is limited by the local density of the input point set. (3) The measurement noise is relieved by the smoothing effect of MLS.

Salient features of our adaptive direct slicing method include the following.

- *Direct processing*. The method does not involve any intermediate geometric model, completely bypasses the laborious surface reconstruction, and avoids the potential model conversion induced accuracy loss.
- *Adaptive contour/layer generation*. The contour profile and the thickness of each layer are adaptive to surface curvature; thus, the resulting surface quality and build time are improved. Existing direct slicing methods do not allow such adaptive slicing feature due to the fact that the slice is computed from projection without knowledge of local curvature.
- *Closed formulas for curvature computing*. Unlike other curvature estimating methods [5,6], where subjective parameters and heuristic assumptions are needed to estimate the curvatures, this paper presents a set of closed formulas for computing curvatures from MLS surfaces without any extra data specific parameters.
- *Single underlying point set surface*. The MLS surface provides a single underlying surface representation of the physical object that can be applied in a variety of downstream design, analysis, and manufacturing applications. It circumvents the trade-off between the projection error and the truncation error. It allows upsampling and downsampling. It is robust even for noisy or sparse point sets.

Figure 3 presents a flowchart of our adaptive direct slicing method, in which the differential geometric analysis and MLS are two fundamental components. The differential geometric analysis provides the normal and curvature information for point set surfaces. MLS gives the underlying surface representation of the input scanned point set. The 2D contours/2.5*D* layers are then adaptively generated based on the computed curvatures.

The rest of the paper is organized as follows. In Sec. 2, we provide a review of relevant work. In Sec. 3, we introduce MLS surfaces as our underlying representation of the point set surface. In Sec. 4, we present two necessary differential geometric analyses: normal estimation for point data and curvature calculation in MLS surfaces. In Sec. 5, we explain the overall procedure of our adaptive direct slicing method, followed by details of two main algorithms: adaptive 2D contour generation and adaptive layer

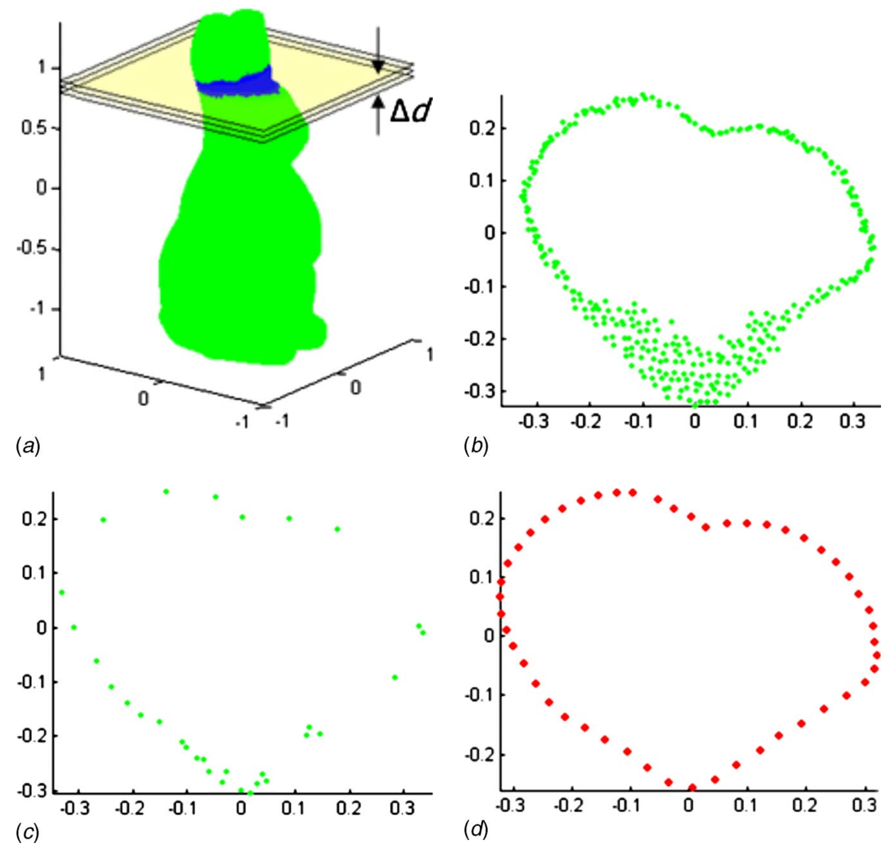


Fig. 2 Comparison of the current projection based and our MLS based slicing approaches. (*a*) Isoview of the rabbit data with slicing planes. (*b*) In the projection based approach, larger Δ*d* leads to more points on the slicing plane and potential projection error. (*c*) In the projection based approach, smaller Δ*d* leads to fewer points projected on the slicing plane and potential truncation error. (*d*) The resulting points of our MLS based approach.

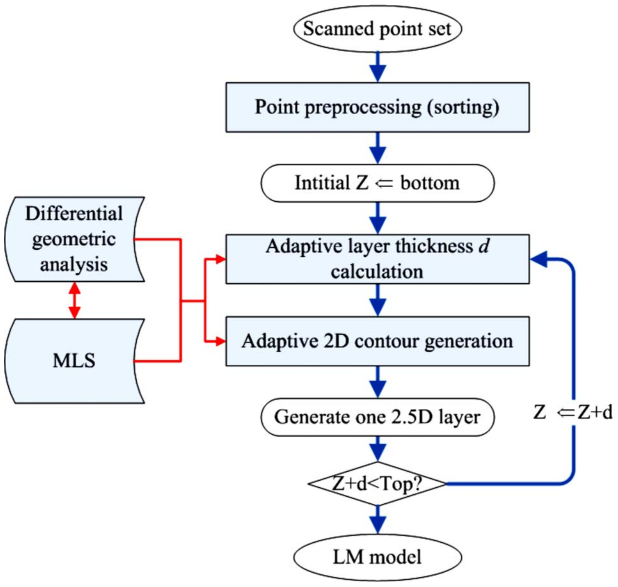

Scanned point set
Point preprocessing (sorting)
Initial Z ⇐ bottom
Differential geometric analysis
MLS
Adaptive layer thickness d calculation
Adaptive 2D contour generation
Generate one 2.5D layer
Z ⇐Z+d
Z+d<Top?
LM model

Fig. 3 Flowchart for our adaptive direct slicing algorithm

thickness calculation. We present experimental examples to illustrate the efficacy of the direct slicing method in Sec. 6 and conclude this paper in Sec. 7.

2 Literature Review

LM is a process in which a part is produced by successively adding material layer by layer. A basic computational task in LM is slicing—the generation of 2.5*D* layers. Various slicing procedures have thus far been proposed and compared in recent surveys [7,8]. Here, we briefly present the works in Refs. [9,10], from which we have adopted adaptive layer generation strategies. In Ref. [9], the accuracy issue of LM in terms of the staircase effect was analyzed. The staircase effect has been relieved by controlling the maximum allowable cusp height using 12 different expressions, depending on geometrical conditions and containment requirements. In Ref. [10], the staircase interaction between the boundaries of the decomposed neighboring volumes has been further studied, which has been eliminated by a feature based volume decomposition algorithm.

Different from the traditional slicing procedures, several direct slicing procedures that avoid both the *B*-spline surface fitting and the triangular mesh construction tasks have been published [1–4]. An image processing based method is proposed in Ref. [1], which collects representative feature points by thinning the projected image. This feature points are then formed into a set of features curves, which are finally used to extract contours for LM. A similar projection based method is presented in Ref. [2], which adaptively determines the slice thickness to control the projection error (not the cusp height). In order to avoid the wideband of 2D point data in the projection process, a new slicing procedure is presented in Ref. [4], which is based on the growing self-organizing map neural network algorithm for piecewise linear reconstruction of curves and surfaces from unorganized thick point data. To relieve the wideband problem in the projection method, a "voting along the surfel" approach is proposed in Ref. [3], which utilizes the quadratic surfel for slicing the point set. However, the trade-off between the projection error and the truncation error remains.

Recently, in computer graphics, a number of point based representations, such as surfel [11,12] or MLS [13–19] have been proposed and proven to be successful in 3D rendering. Moreover, as a smooth surface defined by a projection process [13,14], MLS

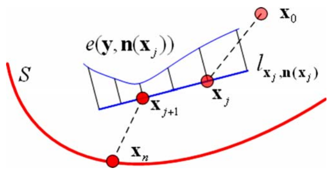

$\mathbf{x}_0$
$e(\mathbf{y},\mathbf{n}(\mathbf{x}_j))$
S
$l_{\mathbf{x}_j,\mathbf{n}(\mathbf{x}_j)}$
$\mathbf{x}_j$
$\mathbf{x}_{j+1}$
$\mathbf{x}_n$

Fig. 4 Illustration of the projection process of a projection MLS

surfaces can also be used for point set denoising, upsampling, downsampling, offsetting, and so on. Based on a more general definition of this projection MLS in Refs. [15,16], a mathematical proof of the convergence of the projection procedure is presented in Refs. [17,18]. Meanwhile, the resulting MLS surface is proven to be isotopic to the original sampled surface.

3 Introduction on MLS Surfaces

This section gives a brief introduction on a MLS surface as described in Refs. [13–18], which forms the basis of our point set surface representation and subsequent direct slicing algorithm in Sec. 5.

Levin [13,14] initially defined the MLS surface S as the stationary set of a projection operator ψ_P, i.e.,

$$S = \{\mathbf{x} \in R^3 | \psi_P(\mathbf{x}) = \mathbf{x}\} \tag{1}$$

Such projection based MLS surfaces are referred to as projection MLS surfaces. Note that this projection is different from the projection based slicing methods where points are simply projected on the z-planes. Amenta and Kil [15,16] gave an explicit definition for projection MLS surfaces as the local minima of an energy function $e(\mathbf{y},\mathbf{a})$ ($\mathbf{a}$ is a direction vector) along the directions given by a vector field $\mathbf{n}(\mathbf{x})$. Based on this definition, they derived a projection procedure for taking a nearby point onto the MLS surface S through the interaction of a normal field $\mathbf{n}(\mathbf{x})$ and the energy function $e(\mathbf{y},\mathbf{a})$, which can be summarized and intuitively illustrated in Fig. 4.

For details of this projection procedure, please refer to Refs. [15,16]. Here we just briefly present two key points in this procedure: Evaluating the normal direction through a vector field $n(x)$, and searching for the local minimum of an energy function $e(\mathbf{y},\mathbf{n}(\mathbf{x}))$.

When evaluating the normal vector, we assume that the normal information at each input point data is available. This assumption is naturally true, when the input data are a set of surfels. When the normal information is not readily available as in some applications, we can easily compute this normal information, which we will discuss in the next section. Then we can compute a normal vector for any point with the normals of the nearby sample points, i.e., define a normal vector field as the normalized weighted average of the normals at the sample points. Suppose a normal vector $\mathbf{v}_i$ is assigned to each point $\mathbf{q}_i \in R^3$ of an input point set $\mathbf{Q}$, we have

$$\mathbf{n}(\mathbf{x}) = \frac{\sum_{\mathbf{q}_i\in\mathbf{Q}} \mathbf{v}_i\theta(\mathbf{x},\mathbf{q}_i)}{\|\sum_{\mathbf{q}_i\in\mathbf{Q}} \mathbf{v}_i\theta(\mathbf{x},\mathbf{q}_i)\|} \tag{2}$$

where

$$\theta(\mathbf{x},\mathbf{q}_i) = e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2} \tag{3}$$

is a Gaussian weighting function, in which h is a scale factor that determines the width of the Gaussian kernel, which has been discussed extensively in Refs. [17,18,20]. In these papers, various schemes have been developed in determining h, e.g., defining h as a fraction of the local feature size [17] or varying h based on the local point density [20]. In this paper, for the sake of simplicity, we adopt a constant scale factor h for each example. Applying the scale factors listed in Table 3, the resulting MLS surfaces can properly preserve the small features and avoid potential instabilities due to the quick falloff of the Gaussian weighting function.

In order to search the local minimum of an energy function, we define the energy function $e:R^3\times R^3\rightarrow R$ as

$$e(\mathbf{y},\mathbf{n}(\mathbf{x}_j)) = e(\mathbf{y}) = \sum_{\mathbf{q}_i\in\mathbf{Q}} ((\mathbf{y}-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x}_j))^2\theta(\mathbf{y},\mathbf{q}_i) \tag{4}$$

To facilitate the search of the local minimum, we can substitute $\mathbf{y}=\mathbf{x}_j+t\mathbf{n}(\mathbf{x}_j)$ into Eq. (4) and restate it as a function of variable t as follows

$$e(t) = \sum_{\mathbf{q}_i\in\mathbf{Q}} ((\mathbf{x}_j - t\mathbf{n}(\mathbf{x}_j)-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x}_j))^2\theta(\mathbf{x}_j - t\mathbf{n}(\mathbf{x}_j),\mathbf{q}_i)$$

With a vector field $\mathbf{n}(\mathbf{x})$ and an energy funtion e, we now have an elegant scheme to project a point onto a MLS surface. Throughout the rest of this paper, this projection based MLS scheme will be used for locally approximating an underlying surface from a set of sample points.

4 Differential Geometric Analysis

4.1 Estimating Surface Normal From Points. As we introduced in the previous section, the computation of the normal vector field $\mathbf{n}(\mathbf{x})$ requires a preassigned normal at each point of the input point set $\mathbf{Q}$. When this normal information is missing, a statistical analysis of the neighboring samples can be applied to estimate the normal vectors, e.g., an eigenanalysis of the covariance matrix of the point positions.

Let $\mathbf{c}$ be the weighted centroid of the neighborhood of $\mathbf{q}$, i.e.,

$$\mathbf{c} = \sum_{\mathbf{q}_i\in\mathbf{Q}} \mathbf{q}_i\cdot\theta(\mathbf{q},\mathbf{q}_i)$$

The 3×3 covariance matrix $\mathbf{C}$ for the sample point $\mathbf{q}$ is then given by

$$\mathbf{C} = \sum_{\mathbf{q}_i\in\mathbf{Q}} (\mathbf{q}_i-\mathbf{c})\cdot(\mathbf{q}_i-\mathbf{c})^T\cdot\theta(\mathbf{q},\mathbf{q}_i)$$

Since matrix $\mathbf{C}$ is symmetric and positive semidefinite, all its three eigenvalues λ_0, λ_1, and λ_2 are real valued. Assuming $\lambda_0\leqslant\lambda_1\leqslant\lambda_2$, we can use the eigenvector $\mathbf{v}_0$ of the smallest eigenvalue λ_0 to approximate the surface normal $\mathbf{n}_\mathbf{q}$ at $\mathbf{q}$ [21].

Notice that the normal vectors estimated by this method are unoriented. Although the orientation of a normal vector is not required to define a normal vector field as in Eq. (2), this information is necessary for the adaptive slicing process presented in the next section. A simple scheme can be used to setup a consistent orientation for the normal vectors that point to the "outside" of the object. This algorithm starts with an extremal point, such as the point with the largest x or y coordinate and sets the orientation of its normal as pointing away from the centroid of the point cloud. Then the orientation of the normal vector of the nearest neighbors can be adjusted by guaranteeing the angle between the neighboring normals less than $\pi/2$. Such an adjusting process can finally spread the correct normal orientation to all the sampled points.

4.2 Curvature Calculation in MLS Surfaces. In our direct slicing algorithm, the curvature of a planar curve (or the surface curvature along the intersection of a surface and a plane) plays a key role in determining the step length in 2D contour generation and the layer thickness in layer generation.

To calculate the curvature of planar curves on a MLS surface, we first convert the native form of MLS into an implicit form. It has been proven, in Refs. [15,17] that the MLS surface is actually the implicit surface given by the zero-level set of the implicit function

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

where $\mathbf{n}:R^3\rightarrow R^3$ is the vector field defined by Eq. (2) and $e:R^3\times R^3\rightarrow R$ is the energy function defined by Eq. (4). Let $\mathbf{x}=(x\ y\ z)^T$, then any planar curve on this MLS surface can be defined as the intersection the MLS surface and a plane as follows:

$$\{g(\mathbf{x}) = g(x,y,z)=0\}\cap\{h(\mathbf{x}) = Ax+By+Cz+D=0\}$$

For a more elegant expression of this planar curve, we can transform the problem so that the plane $h(\mathbf{x})=0$ is transformed to the xy-plane $\hat{h}(\mathbf{x})=z=0$. Then the expression for the implicit curve will reduce to

$$\hat{g}(x,y) = \hat{\mathbf{n}}((x\quad y\quad 0)^T)^T\left(\left.\frac{\partial\hat{e}(\mathbf{y},\hat{\mathbf{n}}((x\quad y\quad 0)^T))}{\partial\mathbf{y}}\right|_{(x\quad y\quad 0)^T}\right)=0 \tag{6}$$

which is only a function of variable x and y (since $z=0$ in the xy-plane).

Applying a curvature formula given in Ref. [22], we have the curvature of this implicit planar curve as

$$k = -\frac{T(\hat{g}(x,y))^T\cdot H(\hat{g}(x,y))\cdot T(\hat{g}(x,y))}{\|\nabla\hat{g}(x,y)\|} \tag{7}$$

where $T(g(x,y))$ is the unit tangent vector of the implicit curve

$$T(\hat{g}(x,y)) = \frac{\left(-\frac{\partial\hat{g}(x,y)}{\partial y}\quad\frac{\partial\hat{g}(x,y)}{\partial x}\right)^T}{\left\|\left(-\frac{\partial\hat{g}(x,y)}{\partial y}\quad\frac{\partial\hat{g}(x,y)}{\partial x}\right)^T\right\|}$$

and

$$\nabla\hat{g}(x,y) = \left(\frac{\partial\hat{g}(x,y)}{\partial x}\quad\frac{\partial\hat{g}(x,y)}{\partial y}\right)^T$$

is the gradient of $g(x,y)$, $H(g(x,y))=\nabla(\nabla(g(x,y)))$ is the Hessian matrix of $g(x,y)$. Notice that

$$T(\hat{g}(x,y)) = \frac{\mathbf{T}\cdot\nabla\hat{g}(x,y)}{\|\nabla\hat{g}(x,y)\|}$$

where $\mathbf{T}$ is a 2×2 matrix defined as

$$\mathbf{T} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Hence, the curvature formula of Eq. (7) can be simplified as

$$k = -\frac{(\mathbf{T}\cdot\nabla\hat{g}(x,y))^T\cdot H(\hat{g}(x,y))\cdot\mathbf{T}\cdot\nabla\hat{g}(x,y)}{\|\nabla\hat{g}(x,y)\|^3} \tag{8}$$

To further expand this formula, we first apply a new notation: $\mathbf{x}=(x\ y\ 0)^T$. Then, taking the derivative of Eq. (4) with respect to $\mathbf{y}$ and setting $\mathbf{y}$ equal to $\mathbf{x}$ give

$$\begin{aligned}
&\left.\frac{\partial\hat{e}(\mathbf{y},\hat{\mathbf{n}}((x\quad y\quad 0)^T))}{\partial\mathbf{y}}\right|_{(x\quad y\quad 0)^T} \\
&= \left.\frac{\partial\hat{e}(\mathbf{y},\hat{\mathbf{n}}(\mathbf{x}))}{\partial\mathbf{y}}\right|_{\mathbf{y}=\mathbf{x}} \\
&= \sum_{\mathbf{q}_i\in\mathbf{Q}} 2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\Bigg(((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))\cdot\hat{\mathbf{n}}(\mathbf{x}) \\
&\quad - \frac{1}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2\cdot(\mathbf{x}-\mathbf{q}_i)\Bigg)
\end{aligned}$$

Substituting it into the transformed implicit function (6) and noticing that $(\mathbf{n}(\mathbf{x}))^T\cdot\mathbf{n}(\mathbf{x})=1$, we have

$$\hat{g}(x,y)=\hat{\mathbf{n}}(\mathbf{x})^T\left(\left.\frac{\partial\hat{e}(\mathbf{y},\hat{\mathbf{n}}(\mathbf{x}))}{\partial\mathbf{y}}\right|_{\mathbf{y}=\mathbf{x}}\right)$$
$$=\sum_{\mathbf{q}_i\in\mathbf{Q}}2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(1-\frac{1}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2\right)\cdot(\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}) \tag{9}$$

From Eq. (9), we can derive the formulas for $\nabla(g(x,y))$ and $H(g(x,y))$. Here we just give the resulting formulas. The gradient of $g(x,y)$ can be expressed as

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

The Hessian of $g(x,y)$ can be expressed as

$$H(\hat{g}(x,y))=\nabla(\nabla(\hat{g}(x,y)))=\sum_{\mathbf{q}_i\in\mathbf{Q}}-\frac{4}{h^2}e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(\frac{2}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))\cdot\left(\frac{1}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2-1\right)\cdot(\mathbf{x}-\mathbf{q}_i)+\left(1-\frac{3}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2\right)\cdot(\hat{\mathbf{n}}(\mathbf{x})+\nabla^T(\hat{\mathbf{n}}(\mathbf{x}))\cdot(\mathbf{x}-\mathbf{q}_i))\right)\cdot(\mathbf{x}-\mathbf{q}_i)^T+2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(\frac{6}{h^4}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2-\frac{2}{h^2}\right)\cdot(\mathbf{x}-\mathbf{q}_i)\cdot(\hat{\mathbf{n}}^T(\mathbf{x})+(\mathbf{x}-\mathbf{q}_i)^T\cdot\nabla(\hat{\mathbf{n}}(\mathbf{x})))+\frac{4}{h^2}e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))\cdot\left(\frac{1}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2-1\right)\cdot\mathbf{I}-\frac{12}{h^2}e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))\cdot(\hat{\mathbf{n}}(\mathbf{x})+\nabla^T(\hat{\mathbf{n}}(\mathbf{x}))\cdot(\mathbf{x}-\mathbf{q}_i))\cdot(\hat{\mathbf{n}}^T(\mathbf{x})+(\mathbf{x}-\mathbf{q}_i)^T\cdot\nabla(\hat{\mathbf{n}}(\mathbf{x})))+2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(1-\frac{3}{h^2}((\mathbf{x}-\mathbf{q}_i)^T\hat{\mathbf{n}}(\mathbf{x}))^2\right)\cdot(\nabla(\hat{\mathbf{n}}(\mathbf{x}))+\nabla^T(\hat{\mathbf{n}}(\mathbf{x}))+\nabla^T(\nabla(\hat{\mathbf{n}}(\mathbf{x})))\cdot(\mathbf{x}-\mathbf{q}_i))$$

where $\mathbf{I}$ is the identity matrix. Notice that the sign of the result curvature k in Eq. (8) has an explicit physical meaning: If k is negative, then the curvature vector $\mathbf{k}$ is opposite to the direction of normal $\mathbf{n}$, or the planar curve is convex at this point; otherwise, $\mathbf{k}$ and $\mathbf{n}$ have the same direction, or the planar curve is concave. It is important to distinguish between the convex and concave property of the planar curve since different formulas of the layer thickness will be applied in these two different cases [9].

5 Direct Slicing of Point Set Surfaces

5.1 Overall Procedure. In our direct slicing scheme for measured data, we assume that the building orientation is already determined. Without loss of generality, we further assume that the building direction is z axis, which means that every slice has a constant z coordinate. Then our direct slicing algorithm can be summarized as using measured data and a prescribed cusp height δ as system input, output successive $2.5D$ layers with layer thickness, and step length of the contour in each slice adaptive to curvature on the underlying MLS surfaces. A stackup of these layers constitute the layered manufactured object. An overall system flowchart for our direct slicing algorithm is shown in Fig. 3. From Fig. 3, we can see that the system flowchart includes three main steps: Point preprocessing, adaptive 2D contour generation, and adaptive layer thickness calculation. The point preprocessing sets up a local neighborhood relation for each input point, which is the base of normal estimation. Since we adopt a standard data sorting algorithm based on the *k-d* tree structure [23] in the first step, we will focus on the last two steps in the remainder of this section.

5.2 Adaptive 2D Contour Generation. Central to the problem of direct slicing of point set surfaces is the generation of 2D contours by intersecting the horizontal slicing plane with the underlying MLS surface defined by the input point data. In this section, we propose a new methodology to generate 2D contours, which contains the following advantages: (1) A marching process is developed to calculate the intersection points, which creates a set of ordered points and form a $2.5D$ layer. (2) The step length between adjacent boundary points on a slice is adaptive to the curvature. It enables the control of the error bound of approximating the MLS surface with polylines from these points, since the error of piecewise linear approximation functions depends linearly on the second derivatives and the spacing of points.

This marching based adaptive 2D contour generation algorithm can be summarized as follows:

Step 1. Given an input point set $\mathbf{Q}$ and an initial line l_0 on the slicing plane H, let $i=0$.

Step 2. Calculate the intersection point $\mathbf{p}_i$ of the MLS surface S and the line l_i.

Step 3. Determine a new line l_{i+1} on the slicing plane H, based on an adaptively computed step length.

Step 4. If $i>2$ and $\|\mathbf{p}_i-\mathbf{p}_0\|<\varepsilon$, stop this process and output $\mathbf{P}=\{\mathbf{p}_0,\mathbf{p}_1,\ldots,\mathbf{p}_i\}$ as the resulting 2D contour. Else let $i=i+1$ and go to Step 2.

In Step 1, the initial line l_0 could be defined by any suitable point $\mathbf{c}_0$ and direction vector $\mathbf{n}_0$. For example, we can find a point $\mathbf{q}\in\mathbf{Q}$ with the minimum distance to the slicing plane H, then project $\mathbf{q}$ onto plane H to get the start point $\mathbf{c}_0$. The direction vector $\mathbf{n}_0$ could be the normal estimated at point $\mathbf{c}$ using the algorithm presented in Sec. 4.

In Step 2, computing the intersection point $\mathbf{p}_i$ is the core of the adaptive 2D contour generation algorithm. Recall the definition of the MLS surface in Eq. (1) that the MLS surface S is the stationary set of a projection operator ψ_P, we can easily realize that for any point $\mathbf{x}$ on the MLS surface S, we have

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

Then the problem of computing the intersection point $\mathbf{p}_i$ can be transformed to find a local minimum of $\|\psi_P(\mathbf{x})-\mathbf{x}\|$ over the set $\mathbf{x}\in l_i$. Suppose the line l_i is defined by point $\mathbf{c}_i$ and direction vector $\mathbf{n}_i$, we can further substitute $\mathbf{x}=\mathbf{c}_i+t\cdot\mathbf{n}_i$ into $\|\psi_P(\mathbf{x})-\mathbf{x}\|$ to

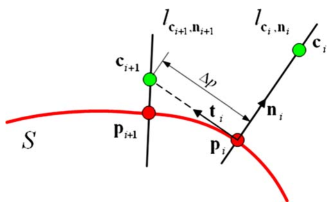


Fig. 5 Illustration of adaptive 2D contour generation

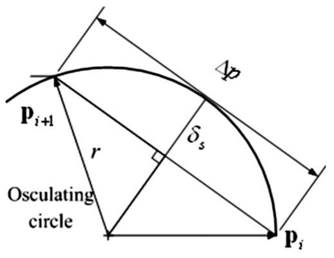


Fig. 6 Calculation of the step length Δp

facilitate the searching process.

In Step 3, to determine a new line l_{i+1}, we first set up a Frenet like frame at point $\mathbf{p}_i$ as shown in Fig. 5, where $\mathbf{n}_i$ denotes the direction vector of l_i. Then we can get a point $\mathbf{c}_{i+1}$ by translating $\mathbf{p}_i$ along the direction perpendicular to $\mathbf{n}_i$ as follows:

$$\mathbf{c}_{i+1} = \mathbf{p}_i + \Delta p \cdot \mathbf{t}_i$$

where $\mathbf{t}_i$ is the unit vector perpendicular to $\mathbf{n}_i$, and Δp is the step length. To compute the step length Δp, we first approximate the horizontal section of the MLS surface S at point $\mathbf{p}_i$ as an osculating circle, as shown in Fig. 6. Then, from Fig. 6, we can derive the following formula to calculate the step length Δp:

$$\Delta p = 2\sqrt{r^2 - (r-\delta_s)^2} = 2\sqrt{2r\delta_s - \delta_s^2} \tag{10}$$

where δ_s is a prescribed approximation error bound, $r=1/|k|$ is the radius of the osculating circle at $\mathbf{p}_i$, and k is the horizontal normal curvature of the MLS surface S at $\mathbf{p}_i$, which is computed by Eq. (8). Additionally, a minimum radius $r_{\min}$ and a maximum radius $r_{\max}$ should be given to limit the permissible radius r to ensure the robustness of the formula in some special cases. For example, setting $r_{\min}=\delta_s$ will avoid the potential negative value inside the square root in Eq. (10); setting a value for $r_{\max}$ will prevent an oversize step length. Finally, by estimating the normal $\mathbf{n}_{i+1}$ at $\mathbf{c}_{i+1}$, we can determine the line l_{i+1} with $\mathbf{c}_{i+1}$ and $\mathbf{n}_{i+1}$, i.e., $l_{i+1} = l_{\mathbf{c}_{i+1},\mathbf{n}_{i+1}}$.

5.3 Adaptive Layer Thickness Calculation. In this paper, we adopt an adaptive layer thickness strategy presented in Ref. [9], where the surface accuracy due to the staircase effect was analyzed, and the accuracy can be controlled by the maximum allowable cusp height. In this strategy, the vertical section of the MLS surface at $\mathbf{p}$ is approximated with the osculating circle, whose radius is given by the reciprocal of the absolute value of the vertical normal curvature at $\mathbf{p}$. This directional curvature can be computed using the method presented in the previous section. Then layer thickness d is determined by eight different cases based on this approximated circle and a prescribed cusp height δ. These eight different cases are a combination of different possible containment requirements and geometrical conditions, i.e., excess/deficient deposition, positive/negative curvature, and $\mathbf{p}$ lying in the upper/lower semicircle of the osculating circle.

For clarity, we choose to introduce the four cases with excess deposition condition as an example. For details of the deficient deposition cases and other deposition requirements, please refer to Ref. [9]. These four cases are illustrated in Fig. 7, where d and δ are as defined before, $\mathbf{n}$ is the surface normal at the point $\mathbf{p}$, θ is the angle between $\mathbf{n}$ and the horizontal plane, ρ is the radius of the osculating circle at $\mathbf{p}$, and is given by the reciprocal of the local vertical normal curvature k at $\mathbf{p}$.

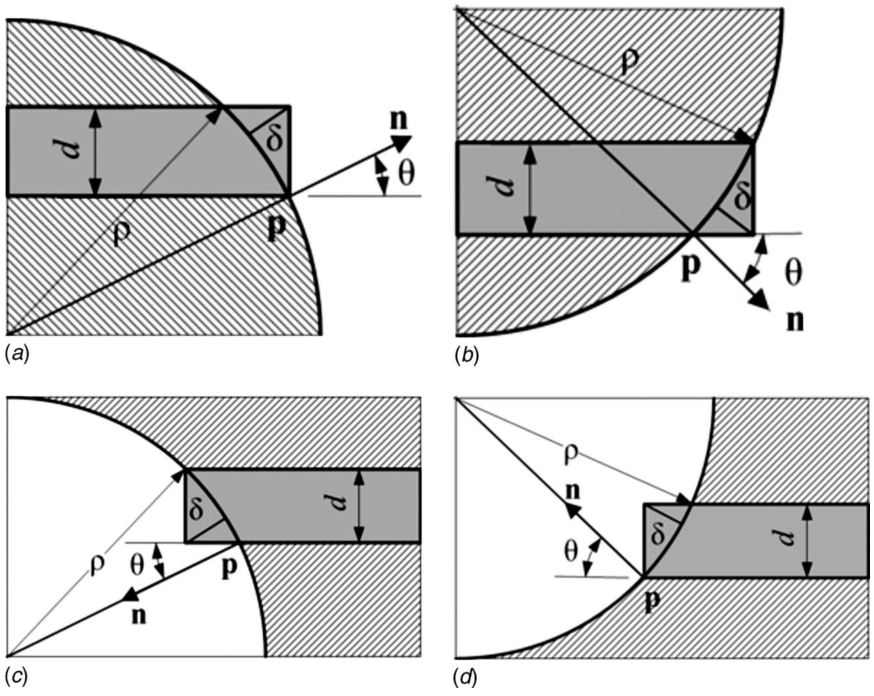
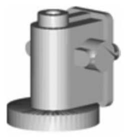


Fig. 7 Four cases in determining the layer thickness d with excess deposition: (*a*) p in the upper semicircle, $k>0$; (*b*) p in the upper semicircle, $k<0$; (*c*) p in the lower semicircle, $k>0$; (*d*) p in the lower semicircle, $k<0$

Table 1 Summary of test data

Part	Size (inch)	Number of points	σ (inch)	Source	Picture
Can	2.0×2.0×2.0	5000	0.01	Synthetic data with random noise	
Wine Glass	1.5×1.5×2.0	35109	0.008	Synthetic data with random noise	
Vase	4.5×4.7×8.2	68097	0.008	Scanned data from www.cyberware.com	
Rabbit	1.2×1.4×2.8	67038	0.012	Scanned data from www.cyberware.com	
Reducer	3.5×4.0×4.3	45326	0.002	Synthetic data with random noise	

With these denotations on hand, we can easily derive a formula of d for each of the four cases illustrated in Fig. 7. Finally, we can get four formulas corresponding to the four cases shown in Fig. 7:

(a) $d = -\rho \sin\theta + \sqrt{\rho^2 \sin^2\theta + 2\rho\delta + \delta^2}$

(b) $d = -\rho \sin\theta + \sqrt{\rho^2 \sin^2\theta + 2\rho\delta - \delta^2}$

(c) $d = +\rho \sin\theta - \sqrt{\rho^2 \sin^2\theta - 2\rho\delta - \delta^2}$

(d) $d = +\rho \sin\theta - \sqrt{\rho^2 \sin^2\theta - 2\rho\delta + \delta^2}$

With these layer thickness formulas for one point $\mathbf{p}$ on a 2D horizontal slice, we can find the layer thickness for the whole 2D slice by solving a nonlinear optimization problem stated as follows:

(1) Find $\mathbf{x}$
(2) Minimize $d(\mathbf{x})$
(3) Subject to $z(\mathbf{x})=\text{const},\ g(\mathbf{x})=0$

where $z(\mathbf{x})$ equals to the z coordinate of $\mathbf{x}$, and $g(\mathbf{x})$ is defined in Eq. (5).

6 Implementation and Examples

In this section, we introduce the MATLAB implementation of our direct slicing algorithm and then present the implementation results on a variety of synthetic and real data. Five test cases are summarized in Table 1 and further analyzed and illustrated later in this section.

Following the procedures in Fig. 3, our algorithm is implemented in MATLAB. Here are several key points during the implementation.

(1) *Projection and intersection*. The function *fminbnd* from MATLAB's optimization toolbox is applied to compute the minimal value of the energy function $e(\mathbf{y},\mathbf{n}(\mathbf{x}))$ when projecting a point onto a MLS surface in the projection based MLS scheme in Sec. 3 and the minimal value of $\|\psi_P(\mathbf{x}) - \mathbf{x}\|$ when calculating the intersection point of a line and a MLS surface in adaptive 2D contour generation in Sec. 5. This function finds a local minimum of one variable function using a golden section search and parabolic interpolation.
(2) *Step length*. The most important part in defining the step length is the calculation of the curve curvature, which can be calculated from the curvature formulas developed in the previous section. Meanwhile, it also can be calculated from an alternative numerical method, where a circle is approximated with several (at least three or five in this paper) neighbor points (distance smaller than the step length Δp) by a least-squares fitting. Such a circle is a good approximate of the osculating circle.
(3) *Layer thickness*. The calculation of layer thickness involves a nonlinear optimization problem in finding a minimal layer thickness for a 2D slice. During our implementations, we exhaustively determine the layer thickness for all sliced points and then find the minimal value as the resulting layer thickness.

6.1 Curvature Calculation in MLS Surfaces. To validate the curvature formulas, we use a synthetic example from a known

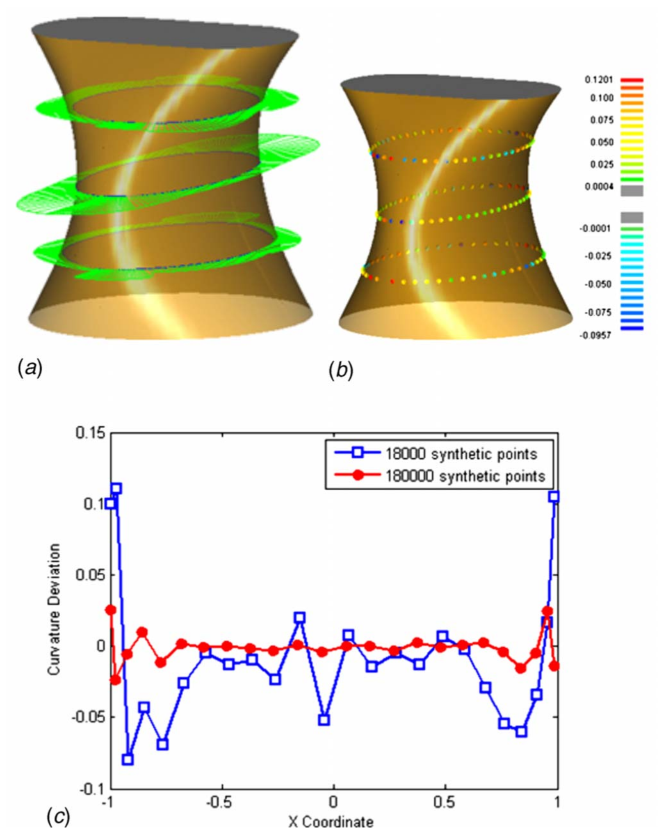


Fig. 8 Validation of the curvature formula for MLS surfaces. (*a*) Needle plot of curvatures of three planar curves. (*b*) Color map of the curvature deviation. (*c*) Comparison of the curvature deviation with different sampling densities of the input point set.

B-spline surface described as the can surface in Table 1. In this example, we select three planar curves determined by the intersection of the original can surface (i.e., the nominal *B*-spline surface) and three horizontal planes at $z=0.5$, $z=1$, and $z=1.5$, whose curvature distribution is shown in Fig. 8(*a*).

Applying the curvature formulas derived in the previous section, we can calculate curvature distribution of three planar curves determined by the intersection of the corresponding MLS surface and the same three horizontal planes. The deviation between these two groups of curvature distributions is illustrated with a color map shown in Fig. 8(*b*). Figure 8(*b*) indicates that larger curvature deviation between the curvatures computed from the nominal *B*-spline surface and the curvatures computed from the closed formula developed in this paper occurs where the original surface has larger curvatures. From the small curvature deviation, we can conclude that (1) the MLS surface approximates the original surface with a small error and (2) the curvature formulas for planar curves are correct. Moreover, when increasing the sampling density of the input point set, the resulting curvature deviation tends to decrease. Using half of the middle curve ($z=1$) as an example, we showed this tendency in Fig. 8(*c*).

Applying a similar validation for the alternative numerical method for computing curve curvature through circle fitting shows that the result of the alternative method is not as accurate as that of our analytical method. This accuracy difference is more distinct in high curvature region, which further demonstrated the advantage of our analytical method in terms of accuracy.

Adaptive 2D Contour Generation. To validate the adaptive 2D contour generation algorithm, three slices at different heights $z=1.2$, $z=1.5$, and $z=1.8$ are shown in Fig. 9(*a*). For better illustration of the sliced data, we further give the top views of these three slices, as shown in Figs. 9(*b*)–9(*d*). In these figures, the yellow planes represent the slicing planes, the red dot points represent the output 2D contours, and the green cross points represent the input part points with intensity fading away when they are farther away from the slicing plane. From these figures, we can see that the distribution of the points is curvature adaptive.

Noisy or Sparse Point Sets. To validate the robustness of MLS in handling noisy or sparse point sets, we generate six synthetic data from the nominal can surface by sampling different number of points (i.e., 2500 and 5000) and by adding random noise with different standard deviations (i.e., 0.01, 0.02, and 0.03). Among them, three data sets that contain 5000 sampling points are further rendered to illustrate the influence of the added noises, as shown in Fig. 10. Using the same parameters and slicing these synthetic data at the horizontal plane of $z=1.2$, we get six sliced data represented by red dot points shown in Table 2. Comparing these sliced data, we can see that there is no significant change in the sliced data when input points become sparse and noisy, which demonstrates the robustness of our algorithm in slicing noisy or sparse point sets.

For better illustrating the difference in the quality of the generated slices, we employ the maximum error between the nominal can surface and the sliced data as a quantitative description of the data quality, as shown in Fig. 11. In Fig. 11, we find that the maximum error is smaller than the given standard deviation for each of the six synthetic data, which reveals the denoising ability of our algorithm.

Adaptive Layer Thickness Calculation. To validate the adaptive layer thickness strategy, we utilize an example of a synthetic wine glass data described in Table 1. In this example, the wine

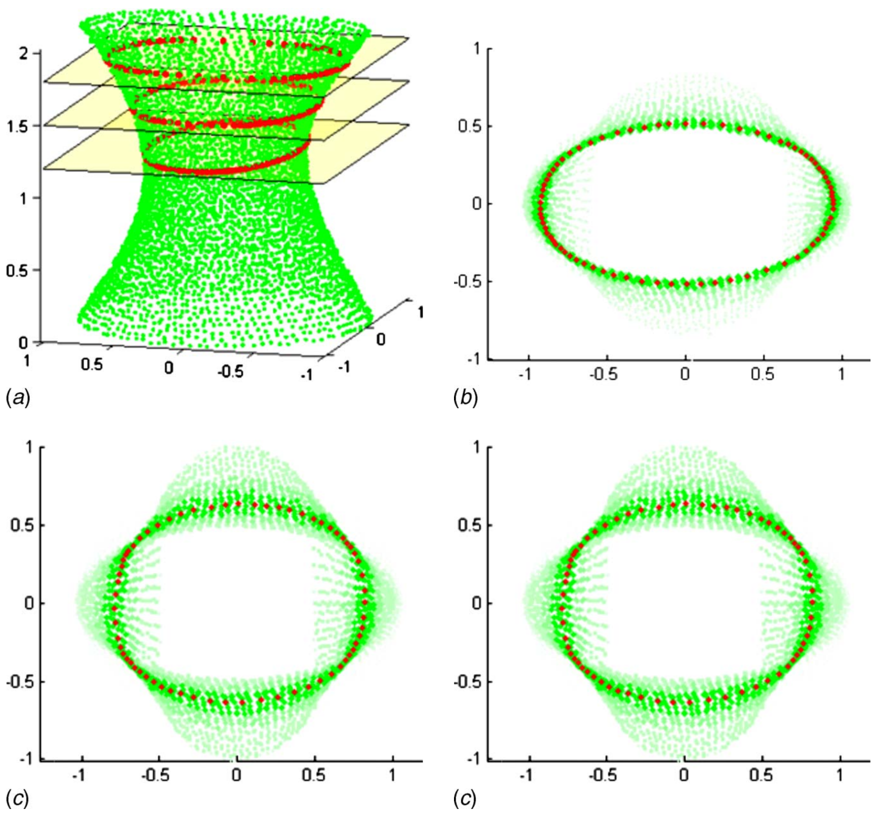


Fig. 9 Slicing of the can data with three horizontal planes. (*a*) Isoview of the can data with resulting 2D contours on three slicing planes. (*b*) Top view of the slice at $z=1.2$. (*c*) Top view of the slice at $z=1.5$. (*d*) Top view of the slice at $z=1.8$.

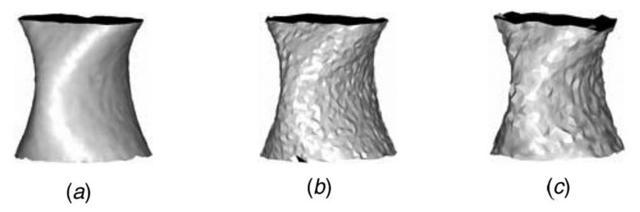
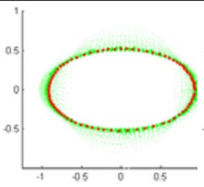
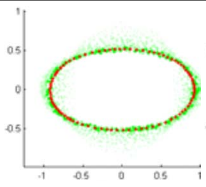
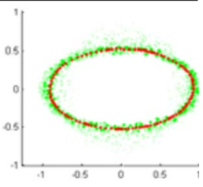
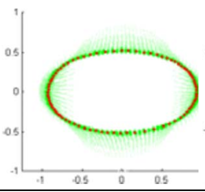
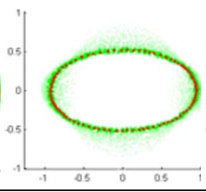
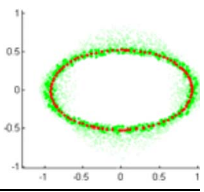


Fig. 10 Rendered models of synthetic can data with different standard deviations of noise: (*a*) $\sigma=0.01$, (*b*) $\sigma=0.02$, and (*c*) $\sigma=0.03$

glass data are sliced into 87 layers with a prescribed cusp height δ as listed in Table 3. Figure 12(*a*) illustrates the input synthetic data with green points and the slicing planes with black lines, where a clear variation in the layer thickness is observed. Moreover, to clearly reveal the relationship between the layer thickness d and the two parameters (the slope angle θ and the curvature k), three curves that represent d, θ, and k as a function of z coordinate are shown in Fig. 12(*b*). Notice that the nominal wine glass surface is generated by revolving a given profile curve about the z axis and all the vertical section curves on the surface are identical. Hence, we just use this profile curve to calculate the angle θ and

Table 2 Slicing of the can data with different noise levels and sample densities

Number of points	Standard Deviation		
	0.01	0.02	0.03
2500			
5000			

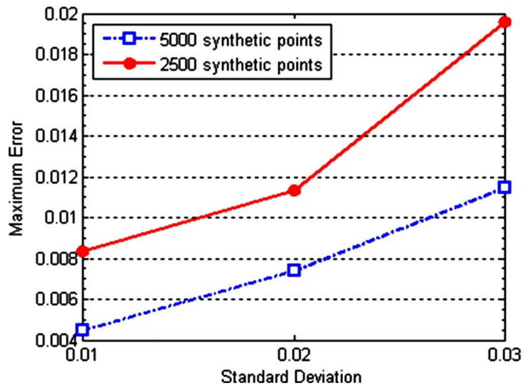


Fig. 11 Illustration of the maximum error between the nominal can surface and the sliced data with different noise levels and sample densities

the curvature k. From Fig. 12(b), we can observe that when either the angle θ or the curvature k becomes larger, the layer thickness d is smaller.

Direct Slicing Algorithm. Using an excess deposition and specifying the cusp height δ as listed in Table 3, we obtained our implementation results illustrated in Table 3, where the yellow planes with black edges represent the slicing planes. These examples demonstrate that our algorithm can handle various point sets, from simple synthetic data to complicated real measured data.

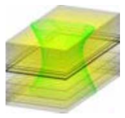
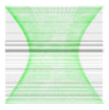
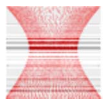
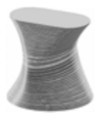
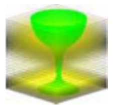
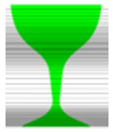
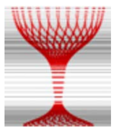
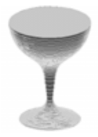
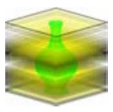
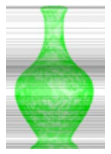
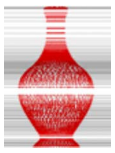
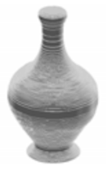
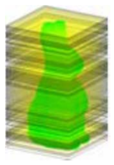
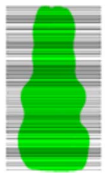
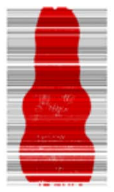
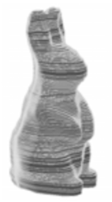
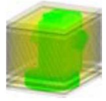
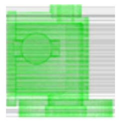
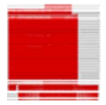
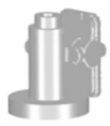
7 Conclusions

In this paper, we present a MLS based approach for directly slicing a point set into the LM model. In comparison with traditional slicing procedures, our method avoids the troublesome surface reconstruction and avoids the potential model conversion induced accuracy loss. When compared with existing direct slicing procedures, our method circumvents the trade-off between the projection error and the truncation error. Further, the resulting contour profile does not depend on the density of the input point set as in prevalent projection based slicing methods due to the use of upsampling from MLS surfaces.

Since we employ the MLS as our underlying surface representation, our algorithm inherits various desirable properties of MLS surfaces. For example, due to the smoothing effect of MLS, the slicing procedure is robust against measurement noise.

Further, we present closed formulas for computing the curvature of planar curves based on the implicit definition of the MLS. As a result, the use of the curvature based adaptive 2D contour and layer thickness generation algorithms increases the surface quality of the resulting LM model and allows the control of the slicing error when approximating the MLS surface with discrete points.

Table 3 Examples of direct slicing

Name	*Cusp height (inch)*	*Gaussian factor h (inch)*	*Number of layers*	*Input point set*		*Output LM model*	
				Iso-view	*Front-view*	*Front-view of slices*	*Rendered Iso-view*
Can	0.02	0.12	61				
Wine Glass	0.01	0.02	87				
Vase	0.04	0.02	106				
Rabbit	0.02	0.06	130				
Reducer	0.03	0.035	98				

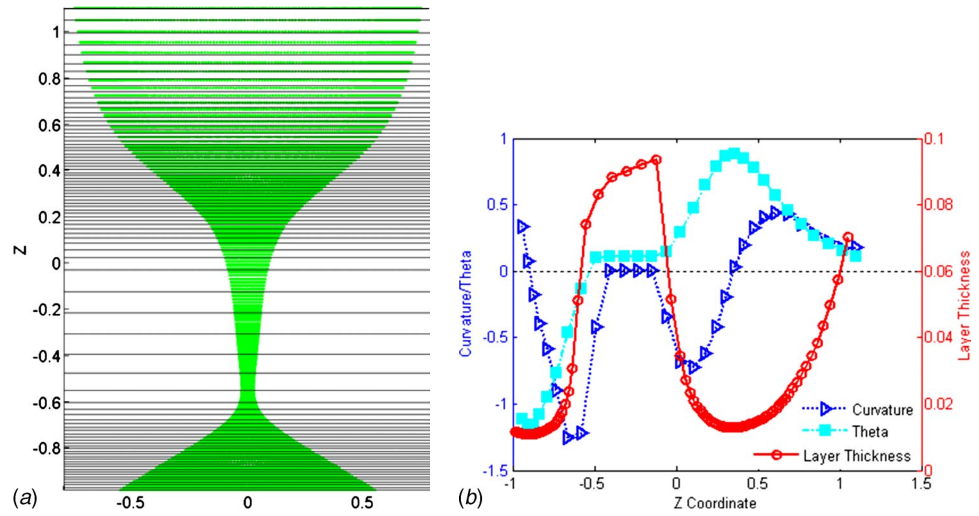


Fig. 12 Illustration of adaptive layer generation. (*a*) Front view of the wine glass data with slicing planes in black. (*b*) Comparison of *d*, *θ*, and *k* as a function of *z* coordinate.

Acknowledgment

This work was supported in part by the National Science Foundation Grant No. 0529165 and Air Force Office of Scientific Research Grant No. FA9550-07-1-0241. P.Y. is also supported by the 2007 Fieldhouse Fellowship.

References

[1] Liu, G. H., Wong, Y. S., Zhang, Y. F., and Loh, H. T., 2003, "Error Based Segmentation of Cloud Data for Direct Rapid Prototyping," Comput.-Aided Des., **35**(7), pp. 633–645.

[2] Wu, Y. F., Wong, Y. S., Loh, H. T., and Zhang, Y. F., 2004, "Modelling Cloud Data Using an Adaptive Slicing Approach," Comput.-Aided Des., **36**(3), pp. 231–240.

[3] Shin, H., Park, S., and Park, E., 2004, "Direct Slicing of a Point Set Model for Rapid Prototyping," *Proceedings of CAD'04*, Pattaya, Thailand, May.

[4] Saravana, K. G., Kalra, P. K., and Dhande, S. G., 2004, "Direct Layered Manufacturing of Point Sampled Geometry," Int. J. Manufacturing Technology & Management, **6**(6), pp. 534–549.

[5] Mitra, N. J., Nguyen, A., and Guibas, L. J., 2004, "Estimating Surface Normals in Noisy Point Cloud Data," Int. J. Comput. Geom. Appl., **14**(4–5), pp. 261–276.

[6] Surazhsky, T., Magid, E., Soldea, O., Elber, G., and Rivlin, E., 2003, "A Comparison of Gaussian and Mean Curvatures Estimation Methods on Triangular Meshes," *2003 IEEE International Conference on Robotics & Automation (ICRA2003)*, pp. 1021–1026.

[7] Kulkarni, P., Marsan, A., and Dutta, D., 2000, "A Review of Process Planning Techniques in Layered Manufacturing," Rapid Prototyping J., **6**(1), pp. 18–35.

[8] Pandey, P. M., Reddy, N. V., and Dhande, S. G., 2003, "Slicing Procedures in Layered Manufacturing: A Review," Rapid Prototyping J., **9**(5), pp. 274–288.

[9] Kulkarni, P., and Dutta, D., 1996, "An Accurate Slicing Procedure for Layered Manufacturing," CAD, **28**(9), pp. 683–697.

[10] Qian, X., and Dutta, D., 2001, "Feature Based Fabrication in Layered Manufacturing," ASME J. Mech. Des., **123**(3), pp. 337–345.

[11] Pauly, M., Keiser, R., Kobbelt, L. P., and Gross, M., 2003, "Shape Modelling with Point-Sampled Geometry," ACM Trans. Graphics, **22**(3), pp. 641–650.

[12] Kobbelt, L., and Botsch, M., 2004, "A Survey of Point-based Techniques in Computer Graphics," Comput. Graphics, **28**(6), pp. 801–814.

[13] Levin, D., 1998, "The Approximation Power of Moving Least-Squares," Math. Comput., **67**, pp. 1517–1531.

[14] Levin, D., 2003, "Mesh-Independent Surface Interpolation," *Geometric Modelling for Scientific Visualization*, G. Brunnett, B. Hamann, H. Muller, and L. Linsen, eds., 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 Workshop on Point-based Graphics*, pp. 139–147.

[17] 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.

[18] Dey, T. K., and Sun, J., 2005, "Adaptive MLS Surfaces for Reconstruction with Guarantees," *Proceeding of the Eurographics Symposium on Geometry Processing*, pp. 43–52.

[19] Katz, S., Tal, A., and Basri, R., 2007, "Direct Visibility of Point Sets," ACM Trans. Graphics, **26**(3), 24:1–24:11.

[20] Pauly, M., 2003, "Point Primitives for Interactive Modeling and Processing of 3D Geometry," Ph.D. thesis, Computer Science Department, ETH Zurich, Zurich, Switzerland.

[21] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., 1992, "Surface Reconstruction From Unorganized Points," Comput. Graphics, **26**(2), pp. 71–78.

[22] Goldman, R., 2005, "Curvature Formulas for Implicit Curves and Surfaces," Comput. Aided Geom. Des., **22**(7), pp. 632–658.

[23] de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O., 1997, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Berlin.