*Computer-Aided Design and Applications*

http://www.cadanda.com

# Direct Digital Design and Manufacturing from Massive Point-Cloud Data

Pinghai Yang, Tim Schmidt and Xiaoping Qian

Illinois Institute of Technology, {yangpin@iit.edu, schmtim1@iit.edu, qian@iit.edu}

**ABSTRACT**

In this paper, we present an approach that can enable product geometric design and its subsequent manufacturing applications directly from massive point-cloud data, without the usual laborious CAD model reconstruction. We term this approach direct digital design and manufacturing (D3M). This approach is based on a moving least-squares (MLS) formulation that defines a continuous surface directly from a set of points. We derive formulae for differential geometric analysis of the MLS surface based on its implicit form. From the resulting closed curvature formulae, we develop a set of error-bounded, adaptive intersecting algorithms that can intersect the MLS surface with lines, planes, polygonal meshes, and NURBS surfaces. These algorithms have been successfully applied in applications ranging from product shape design, CNC machining to rapid prototyping.

**Keywords:** point-sampled geometry, moving least-squares surface, CAD/CAM.
**DOI:** 10.3722/cadaps.2009.685-699

## 1. INTRODUCTION

Rapid advancement of three-dimensional (3D) scanning techniques has fueled the growing use of point-cloud data in product design and manufacturing. It promises to revolutionalize the way digital geometric models are created and used throughout the product development cycle. However, despite the rapid progress in data accuracy, data density and acquisition speed from various 3D scanners, one bottleneck issue remains in various 3D scanning applications. That is, acquired data points still need to undergo lengthy, tedious and error-prone reconstruction of intermediate surface models, such as CAD surface models, with substantial human-intervention and multitude of model conversions. This has severely impeded the realization of the full potentials of 3D scanning such as mass customization and digital inventory.

Recently research has appeared focusing on how to bypass CAD model reconstruction in digital design and manufacturing applications involving massive point-cloud data. For example, point-based curve drawing [3], layered manufacturing [8], [12], [15],[16], and NC machining [4], [5], [11], [13] based on points have recently been proposed. However, these methods are *ad hoc* in nature, e.g. by assuming grid structures of points, or relying on some voting schemes to "thin" the dense point cloud to produce approximate contours. There has not been any systematic study of how points can be used to directly represent continuous surfaces for CAD/CAM uses. The fundamental deficiency with existing attempts is the absence of a suitable mathematical basis for representing surfaces directly with points that can be used consistently in product development cycles.

In this paper, we propose to define the surface from discrete points based on the moving least-squares (MLS) surface definition [10, 45] due to its many desirable properties such as explicit definition in

terms of implicit surfaces, local computing, projection operation, and $C^\infty$ smoothness. The D3M approach presented in this paper exploits these properties of the MLS surface. More specifically, we use the MLS surface as the underlying surface representation for acquired point-sampled geometry in the D3M. This paper summarizes three critical components in D3M.

- Mathematically, closed formulae for differential geometric analysis are derived. Through the implicit definition of an MLS surface, we derive formulae for curvature computing for an MLS surface and planar curves that lie on the MLS surface. This enables the development of subsequent efficient (i.e. curvature-adaptive) and accurate (i.e. error-bounded) surface intersection algorithms.
- Computationally, algorithms for intersecting an MLS surface with lines, planes, polygonal mesh and NURBS surfaces are given. The key to these intersection algorithms is a new algorithm for line/MLS surface intersection, which is based on the projection property of the MLS surface. A curvature-adaptive marching algorithm is developed for plane/MLS surface intersection. The intersection between a triangular mesh and an MLS surface is through a curvature-adaptive marching process that produces a series of intersection points with the separation distance adaptive to the local curvature. The intersection between a NURBS surface and an MLS surface is through adaptive subdivision of NURBS surfaces into polygonal meshes.
- Application-wise, we apply the above algorithms to several point-cloud data based design and manufacturing applications, including Boolean intersection between NURBS surfaces and point-cloud data, slicing in rapid prototyping and path generation in NC machining.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction on the MLS surface. Section 3 presents our results of differential geometric analysis on the MLS surface. Section 4 details a set of algorithms for intersecting an MLS surface with CAD geometry. Section 5 demonstrates the D3M applications of these intersection algorithms on one point-cloud data. An example use of D3M in rapid development of custom headmasks is given in Section 6. This paper concludes in Section 7.

## 2. MOVING LEAST-SQUARES SURFACE

This section gives a brief introduction on the definition of an MLS surface, which forms the basis of our D3M approach.

Levin [9],[10] defined an MLS surface $S$ as the stationary set of a projection operator $\psi_P$, i.e.,

$$S = \left\{ \mathbf{x} \in R^3 \mid \psi_P(\mathbf{x}) = \mathbf{x} \right\} \tag{2.1}$$

Upon such a projection operation, a point on the MLS surface is projected onto itself. Such projection based MLS surfaces are referred to as projection MLS surfaces. Amenta and Kil [1],[2] gave an explicit definition for projection MLS surfaces as the local minima of an energy function $e(\mathbf{y},\mathbf{a})$ ($\mathbf{y}$ is a position vector and $\mathbf{a}$ is a direction vector) along the directions given by a vector field $\mathbf{n}(\mathbf{x})$, as shown in Fig. 1. Based on this definition, they derived a projection procedure for taking a point onto the MLS surface $S$ implied by $\mathbf{n}$ and $e$, which can be summarized and intuitively illustrated in Fig. 1.

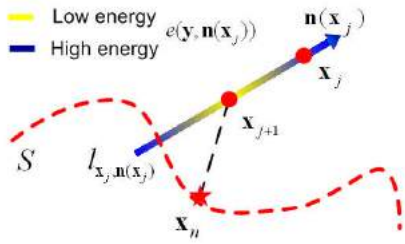



Fig. 1: Illustration of the MLS projection process.

For details of this projection procedure, please refer to [1],[2]. Here we briefly present two key points in this procedure: evaluating the normal direction through a vector field $\mathbf{n}(\mathbf{x})$ 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 is a set of surfels. When the normal information is not readily available as in some applications, we can easily compute this normal information, for example through eigen analysis. 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:

$$\mathbf{n}(\mathbf{x}) = \frac{\sum_{\mathbf{q}_i \in \mathbf{Q}} \mathbf{v}_i \theta(\mathbf{x},\mathbf{q}_i)}{\left\| \sum_{\mathbf{q}_i \in \mathbf{Q}} \mathbf{v}_i \theta(\mathbf{x},\mathbf{q}_i) \right\|} \tag{2.2}$$

where $\theta(\mathbf{x},\mathbf{q}_i) = e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}$ is a Gaussian weighting function, where $h$ is a scale factor that determines the width of the Gaussian kernel.

In the $j$-th iteration of the overall projection process, we need to search the local minimum $\mathbf{x}_{j+1}$ of an energy function along a line $l_{\mathbf{x}_j,\mathbf{n}(\mathbf{x}_j)}$ given by $\mathbf{x}_j$ and $\mathbf{n}(\mathbf{x}_j)$, as shown in Fig. 1. Such an energy function $e: R^3 \times R^3 \to R$ can be defined as

$$e(\mathbf{y},\mathbf{n}(\mathbf{x}_j)) = e(\mathbf{y}) = \sum_{\mathbf{q}_i \in \mathbf{Q}} \left( (\mathbf{y}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x}_j) \right)^2 \theta(\mathbf{y},\mathbf{q}_i) \tag{2.3}$$

To facilitate the search of the local minimum, we can substitute $\mathbf{y} = \mathbf{x}_j + t \cdot \mathbf{n}(\mathbf{x}_j)$ into Eqn. (2.3) and restate it as a function of variable $t$:

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

With a vector field $\mathbf{n}(\mathbf{x})$ and an energy function $e$, we now have an elegant scheme to project a point onto an MLS surface.

## 3. DIFFERENTIAL GEOMETRIC ANALYSIS OF THE MLS SURFACE

To calculate the principal curvatures of an MLS surface, we first convert the native form of MLS into an implicit form. It has been proved in [1],[2] that the MLS surface is actually the implicit surface given by the zero-level set of the implicit function

$$g(\mathbf{x}) = \mathbf{n}(\mathbf{x})^T \left( \frac{\partial e(\mathbf{y},\mathbf{n}(\mathbf{x}))}{\partial \mathbf{y}} \Big|_{\mathbf{y}=\mathbf{x}} \right) \tag{3.4}$$

where $\mathbf{n}: R^3 \to R^3$ is the vector field defined by Eqn. (2.2) and $e: R^3 \times R^3 \to R$ is the energy function defined by Eqn. (2.3). Applying the curvature formulas for implicit surfaces given in [6], we have the Gaussian and mean curvatures of this implicitly defined MLS surface as

$$\begin{cases} k_{Gaussian} = -\dfrac{Det\begin{pmatrix} H(g(\mathbf{x})) & \nabla^T g(\mathbf{x}) \\ \nabla g(\mathbf{x}) & 0 \end{pmatrix}}{\|\nabla g(\mathbf{x})\|^4} \\ k_{Mean} = -\left( \dfrac{\nabla g(\mathbf{x}) \cdot H(g(\mathbf{x})) \cdot \nabla^T g(\mathbf{x}) - \|\nabla g(\mathbf{x})\|^2 \cdot Trace(H)}{\|\nabla g(\mathbf{x})\|^4} \right) \end{cases} \tag{3.5}$$

where $\nabla g(\mathbf{x}) = \left( \frac{\partial g(\mathbf{x})}{\partial x} \quad \frac{\partial g(\mathbf{x})}{\partial y} \quad \frac{\partial g(\mathbf{x})}{\partial z} \right)^T$ is the gradient of $g(\mathbf{x})$ and

$$H(g(\mathbf{x})) = \nabla(\nabla(g(\mathbf{x}))) = \begin{pmatrix} \frac{\partial g(\mathbf{x})}{\partial x \partial x} & \frac{\partial g(\mathbf{x})}{\partial x \partial y} & \frac{\partial g(\mathbf{x})}{\partial x \partial z} \\ \frac{\partial g(\mathbf{x})}{\partial x \partial y} & \frac{\partial g(\mathbf{x})}{\partial y \partial y} & \frac{\partial g(\mathbf{x})}{\partial y \partial z} \\ \frac{\partial g(\mathbf{x})}{\partial x \partial z} & \frac{\partial g(\mathbf{x})}{\partial y \partial z} & \frac{\partial g(\mathbf{x})}{\partial z \partial z} \end{pmatrix}$$

is the Hessian matrix of $g(\mathbf{x})$. Notice that *Det*(**A**) and *Trace*(**A**) denote the determinant and the trace of the matrix **A** correspondingly.

Note, the principal curvatures can be derived from the Gaussian curvature and mean curvature. To further expand the formula of Eqn. (5), we first take the derivative of Eqn. (2.3) with respect to **y** and setting **y** equal to **x**, which gives

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

Substituting it into Eqn. (3.4), and notice that $(\mathbf{n}(\mathbf{x}))^T \cdot \mathbf{n}(\mathbf{x}) = 1$, we have

$$g(\mathbf{x}) = \mathbf{n}(\mathbf{x})^T \left( \frac{\partial e(\mathbf{y}, \mathbf{n}(\mathbf{x}))}{\partial \mathbf{y}}\Big|_{\mathbf{y}=\mathbf{x}} \right) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2} \cdot \left( 1 - \frac{1}{h^2} \left( (\mathbf{x}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x}) \right)^2 \right) \cdot (\mathbf{x}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x}) \tag{3.6}$$

Suppose that $A = e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}$ and $B = (\mathbf{x}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x})$, then Eqn. (3.6) can be written as

$$g(\mathbf{x}) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2A \cdot \left( B - \frac{1}{h^2} B^3 \right)$$

Hence, the gradient of $g(\mathbf{x})$ can be expressed as

$$\nabla(g(\mathbf{x})) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2 \cdot \left( \left( \frac{\partial A}{\partial \mathbf{x}} \right) \cdot \left( B - \frac{1}{h^2} B^3 \right) + A \cdot \left( 1 - \frac{3}{h^2} B^2 \right) \cdot \left( \frac{\partial B}{\partial \mathbf{x}} \right) \right)$$

Notice that

$$\begin{cases} \frac{\partial A}{\partial \mathbf{x}} = -\frac{2}{h^2} e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2} \cdot (\mathbf{x}-\mathbf{q}_i) \\ \frac{\partial B}{\partial \mathbf{x}} = \mathbf{n}(\mathbf{x}) + \nabla^T(\mathbf{n}(\mathbf{x})) \cdot (\mathbf{x}-\mathbf{q}_i) \end{cases}$$

We finally have

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

Suppose that $\mathbf{C} = (\mathbf{x}-\mathbf{q}_i)$ and $\mathbf{D} = \mathbf{n}(\mathbf{x}) + \nabla^T(\mathbf{n}(\mathbf{x})) \cdot (\mathbf{x}-\mathbf{q}_i)$, then the above equation can be written as

$$\nabla(g(\mathbf{x})) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2A \cdot \left( \frac{2}{h^2} B \cdot \left( \frac{1}{h^2} B^2 - 1 \right) \cdot \mathbf{C} + \left( 1 - \frac{3}{h^2} B^2 \right) \cdot \mathbf{D} \right)$$

Hence, the Hessian of $g(\mathbf{x})$ can be expressed as

$$H(g(\mathbf{x})) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2\left(\frac{2}{h^2}B\cdot\left(\frac{1}{h^2}B^2-1\right)\cdot\mathbf{C}+\left(1-\frac{3}{h^2}B^2\right)\cdot\mathbf{D}\right)\cdot\left(\frac{\partial A}{\partial \mathbf{x}}\right)^T + 2A\cdot\left(\frac{6}{h^4}B^2-\frac{2}{h^2}\right)\cdot\mathbf{C}\cdot\left(\frac{\partial B}{\partial \mathbf{x}}\right)^T$$
$$+2A\cdot\frac{2}{h^2}B\cdot\left(\frac{1}{h^2}B^2-1\right)\cdot\left(\frac{\partial \mathbf{C}}{\partial \mathbf{x}}\right)-2A\cdot\frac{6}{h^2}B\cdot\mathbf{D}\cdot\left(\frac{\partial B}{\partial \mathbf{x}}\right)^T+2A\cdot\left(1-\frac{3}{h^2}B^2\right)\cdot\left(\frac{\partial \mathbf{D}}{\partial \mathbf{x}}\right) \quad (3.7)$$

For further expansion of this formula, please refer to [19]. Substituting the above expressions of gradient and Hessian of $g(\mathbf{x})$ into Eqn. (3.5), we can obtain closed formulas for direct computing of surface curvatures for MLS surfaces.

The curvature form for the planar curves can be derived similarly, i.e. by converting the planar curves into an implicit form. The details are available in [17].

## 4. ALGORITHMS FOR INTERSECTING AN MLS SURFACE WITH CAD GEOMETRY

In this section, we present a set of algorithms that can intersect an MLS surface with common analytical geometry such as lines, planes, and solids bounded by polygonal meshes and NURBS surfaces. The basis of these algorithms is two-fold: a) a projection based approach for obtaining intersection points between a line and an MLS surface, and b) a curvature-adaptive marching process that intersects a set of lines on a plane with the MLS surface. We describe these algorithms in the following subsections.

### 4.1 Line/MLS Surface Intersection

Recall the definition of the MLS surface in Eqn. (2.1) that the MLS surface $S$ is the stationary set of a projection operator $\psi_P$. We can easily realize that for any point $\mathbf{x}$ on the MLS surface $S$, we have

$$\|\psi_P(\mathbf{x})-\mathbf{x}\| = 0 \quad (4.8)$$

Then the problem of computing the intersection point $\mathbf{p}$ of a line $l$ with the MLS surface $S$ can be transformed to finding a root of Eqn. (4.8) over the set $\mathbf{x} \in l$. Suppose the line $l$ can be defined by a point $\mathbf{c}$ and a directional vector $\mathbf{n}$, this root finding problem can be further reduced to a one-dimensional problem by substituting $\mathbf{x} = \mathbf{c} + t\cdot\mathbf{n}$ into Eqn. (4.8), where $t$ is the only variable. In this paper, Brent's method is implemented to solve this one-dimensional root finding problem, which combines root bracketing, bisection, and inverse quadratic interpolation to converge from the neighborhood of a zero crossing and is suitable for this kind of one-dimensional root finding problems [14].

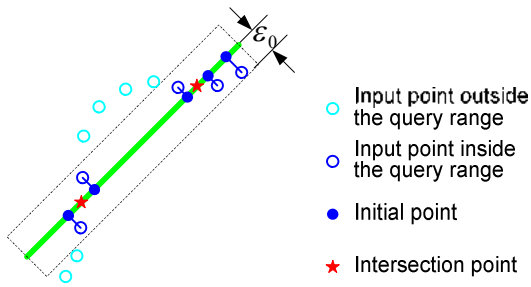



Fig. 2: Strategy for generating different initial points for locating multiple line/MLS surface intersection points: points that are $\varepsilon_0$ distance away from the line are projected onto the line and the projected points are then used as the starting points to find the intersection points between the line and the MLS surface.

When multiple intersection points exist, different initial points are needed to find all intersection points of a line $l$ with the MLS surface $S$. We use the following strategy to generate these initial points:

- Find all points of the input point data inside a query range, i.e., having a distance to the line $l$ within a prescribed distance $\varepsilon_0$ (e.g., blue circles shown in Fig. 2). The assumption here is that each projected point on the MLS surface is maximally at $\varepsilon_0$ distance away from its closest sample in the point cloud.
- Then project all these points onto the line $l$. These projected points will be chosen as initial points (e.g., blue solid circles shown in Fig. 2). Note that it is possible that the Brent's algorithm started at several different initial points may converge to the same point, e.g., in Fig. 2, the left two initial points converged to the left intersection point (represented by a red star) and the right three initial points converged to the right intersection point. In this case, we need further check the resulting intersection points and remove the redundant points.

### 4.2 Curvature-adaptive Plane/MLS Surface Intersection

In this paper, we adopt a marching approach to computing the plane/MLS surface intersection. In this marching approach, the intersection curve(s) is defined in the following way: first find a starting point on the intersection curve and then adaptively march along this curve to get successive intersection points. The line/MLS surface intersection approach described in the previous section is used to determine both the starting points for marching and the intersection points between successive marching lines and the MLS surface. The separation distances between successive lines are adaptive to the curvature in the planar curve on the MLS surface so that the process produces the intersection contour with bounded error. Such curvature-adaptive step length in the marching process circumvents the tradeoff between the intersection accuracy which requires smaller step length and the algorithm efficiency which requires larger step length. This marching algorithm can be summarized as the following steps:

**STEP 1:** Given an input point set $\mathbf{Q}$, an input plane *H*, and an initial line $l_0$ defined by a starting point $\mathbf{p}_0$ and a direction vector $\mathbf{n}_0$. Let $i = 0$;

**STEP 2:** Determine a new line $l_{i+1}$ on the plane *H*, based on a computed step length adaptive to the local curvature on the planar curve on the MLS surface;

**STEP 3:** Calculate the intersection point $\mathbf{p}_{i+1}$ of the MLS surface *S* and the line $l_{i+1}$;

**STEP 4:** Check the stop condition. If true, stop this process and output $\mathbf{P} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{i+1}\}$ as the resulting 2D contour. Else let $i = i+1$ and go back to STEP 2.

In STEP 1, the point $\mathbf{p}_0$ is the starting point computed in the previous section and the initial line $l_0$ is the line used to compute $\mathbf{p}_0$.

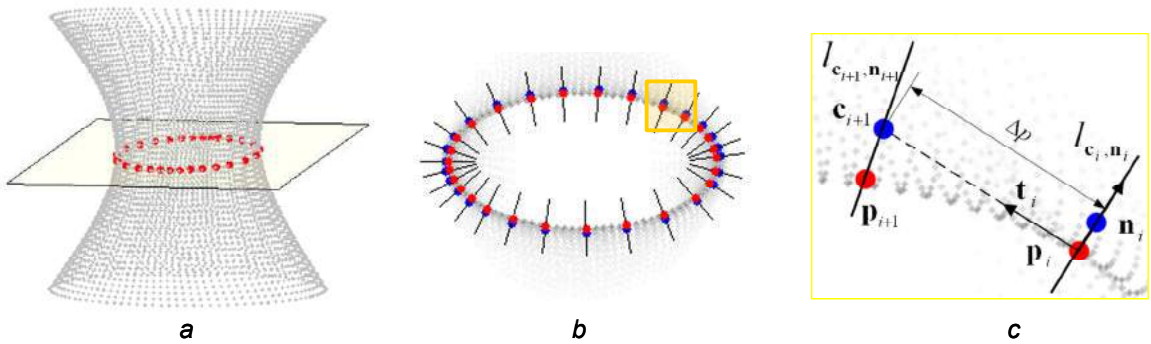



Fig. 3: Illustration of adaptive marching in plane/MLS surface intersection. (a) Iso-view of a point data with resulting 2D contour on a slicing plane. (b) Top view of the slice at $z = 1.0$. (c) Zoom-in.

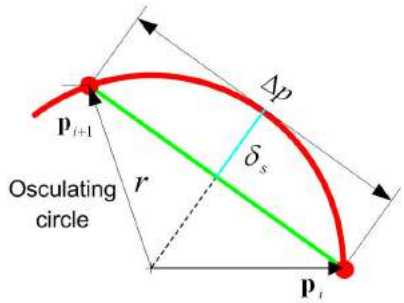



Fig. 4: Computing error-bounded step length $\Delta p$ based on an osculating circle.

In STEP 2, to determine a new line $l_{i+1}$, we first set up a Frenet frame at point $\mathbf{p}_i$ as shown in Fig. 3, 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 $\mathbf{n}_i$: $\mathbf{c}_{i+1} = \mathbf{p}_i + \Delta p \cdot \mathbf{t}_i$,where $\mathbf{t}_i$ is the unit vector perpendicular $\mathbf{n}_i$, $\Delta p$ is the step length. To compute the step length $\Delta p$, we first approximate the planar section of the MLS surface $S$ at point $\mathbf{p}_i$ as an osculating circle, as shown in Fig. 4. Then, from Fig. 4, we can derive the following formula to calculate the step length $\Delta p$:

$$\Delta p = 2 \cdot \sqrt{r^2 - (r - \delta_s)^2} = 2 \cdot \sqrt{2 \cdot r \cdot \delta_s - \delta_s^2} \tag{4.9}$$

where $\delta_s$ is a prescribed approximation error bound for the intersection curve, $r = 1/|k|$ is the radius of the osculating circle at $\mathbf{p}_i$ and $k$ is the curvature computed at $\mathbf{p}_i$ of the planar curve that lies on both the plane $H$, and the MLS surface $S$. Additionally, a minimum radius $r_{\min}$ and a maximum radius $r_{\max}$ can be given to limit the permissible radius $r$ to ensure the robustness of the formula in some special cases. For example, setting $r_{\min} = \delta_s$ would avoid the potential negative value inside the square root in Eqn. (4.9); setting a value for $r_{\max}$ could prevent an over-sized step length since overly un-even distribution of intersection points may cripple many curve interpolation and approximation algorithms when a smooth intersection curve is desired. 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., $l_{i+1} = l_{\mathbf{c}_{i+1}, \mathbf{n}_{i+1}}$.

In STEP 3, the intersection point $\mathbf{p}_i$ is generated by applying the line/MLS surface intersection algorithm.

### 4.3 Plane/MLS Surface Intersection

With the above plane/MLS surface intersection algorithm, we can extend it to the intersection between a triangular mesh and an MLS surface with some minor changes. It involves two main steps:

a) Intersect each individual triangle with the MLS surface defined by the input point cloud to get all the intersection curve segments in a discrete form (polyline);
b) Sort and link the discrete curve segments to construct polylines defining the intersection curve.

In the following sections, we will focus on the first step, where a triangle can be treated as a bounded plane. Due to the existence of the three boundary edges, the intersection curves of a triangle and an MLS surface can be categorized into two main types: (1) *internal loops* and (2) *open branches*, as shown in Fig. 5. However, there is only one type of intersection curves for a plane and an MLS surface, which is corresponding to the *internal loops* for triangle-MLS surface intersection.

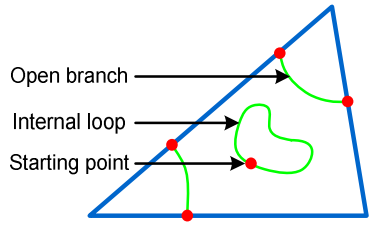
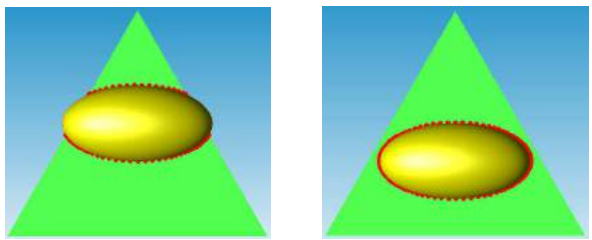



Fig. 5: Types of intersection curves.

*4.3.1 Finding Starting Points for Open Branches*
For an open branch, a starting point is an intersection point between the triangle edges with the MLS surface, which can be obtained by the line/MLS surface intersection algorithm. Notice any starting points outside the range of the edges are omitted.

*4.3.2 Finding Starting Points for Internal Loops*
For an internal loop, we can inherit the strategy of finding starting points in plane/MLS surface intersection algorithm. However, notice, 1) instead of finding all points of the input point data that have a distance to the input plane within a prescribed distance $\varepsilon$, we find all points that have a distance to the input triangle within $\varepsilon$; 2) candidate starting points outside the triangle are omitted.

*4.3.3 Curvature-adaptive Intersection*
The curvature-adaptive marching algorithm for plane/MLS surface intersection can be directly applied to the triangle/MLS surface intersection once the starting points are identified. Fig. 6 shows examples of curvature-adaptive triangle/MLS surface intersection, where the triangle is drawn in green, the point cloud in yellow, and the red points represent the output contours of the adaptive marching algorithm. We can see that the distribution of the points is curvature-adaptive. Fig. 7 shows a more complicated example of the intersection between a triangle and an MLS surface, which results in three open branches and three internal loops.



(a) (b)

Fig. 6: Curvature-adaptive triangle/MLS surface intersection. (a) Open branches. (b) Closed loop.

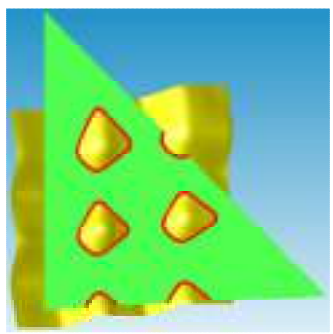


Fig. 7: Intersection between a triangle and an MLS surface resulting in multiple open branches and internal loops.

Repeated use of the triangle/MLS surface intersection would then result in the intersection contours between a triangular mesh and an MLS surface. Note, the connectivity of triangle edges in the mesh is recorded to avoid the duplicate intersection between edges from adjacent triangles and the MLS surface.

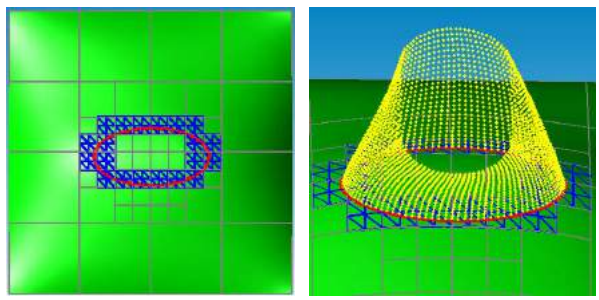
**4.4 NURBS/MLS Surface Intersection**

The intersection and Boolean operations between design geometry (NURBS surfaces) and acquired point-sampled geometry (approximated by an MLS surface) is achieved by first adaptively subdividing the designed geometry (e.g., NURBS surfaces) into a set of planar triangles and then applying the above triangular mesh /MLS surface intersection algorithm, which ensures the generality of our intersection algorithm for shape modeling from both design and acquired geometry. These are the steps of our algorithm.

1. Adaptive subdivision of NURBS surfaces:
   a) Generate an adaptive quad-tree structure for the input NURBS surfaces;
   b) Create a triangular mesh for potentially intersecting regions based on this tree structure.
2. Adaptive intersection of a triangular mesh and an MLS surface.

*4.4.1 Adaptive Subdivision of NURBS Surfaces for Accurate and Efficient Intersection*

Since our NURBS/MLS surface intersection is based on plane/MLS surface intersection, we first subdivide the NURBS surfaces into a set of patches and then divide those patches that can potentially intersect with the MLS surface into planar triangles. Two governing factors that affect the subdivision process are accuracy and efficiency. In order to assure the accuracy of the intersection, the patch subdivision continues until the subdivided triangle mesh represents the underlying NURBS accurately within a bounded error. In order to improve the efficiency of NURBS/MLS surface intersection, we adaptively subdivide the NURBS patches. That is, the NURBS patches are only subdivided when they can potentially intersect with the MLS surface and triangles are only generated from those patches that can potentially intersect with the MLS surface. Fig. 8 shows a triangle mesh (blue) generated from a NURBS surface (green) by the adaptive subdivision algorithm. The resulting mesh encompasses the intersection contour between the NURBS surface and a point-set surface (yellow). Gray curves represent the boundary curves of all leaf surface patches of the quad-tree constructed on the input NURBS surface. It can be seen that patches closer to the intersection contour are smaller and have gone though more times of subdivision. Planar triangles are only generated from the leaf patches that can potentially intersect with the MLS surface.



(a) (b)

Fig. 8: Adaptive subdivision of a NURBS surface ensuring accurate and efficient NURBS/MLS surface intersection: patches closer to the intersection curves undergo more times of subdivision. Planar triangles are only generated from the leaf patches that can potentially intersect with the MLS surface. (a) Top-view. (b) Iso-view.

We now detail the adaptive subdivision process. We first construct a quad-tree based on the adaptive NURBS surface subdivision algorithm. We start with one NURBS surface patch as the root node of the quad-tree. This node is recursively split into four children in the parametrical domain until at least one of the following two conditions is satisfied: 1) it deviates from a “best fit” plane within a given tolerance; 2) it has no intersection with the input point-set surface. Meanwhile, all the leaf nodes of the

quad-tree are classified into two types, i.e., *non-intersection* patches and *intersection candidate* patches according to the testing result of the second criterion, where *non-intersection* denotes a patch has no intersection with the input point-set surface and *intersection candidate* denotes a patch may intersect with the input point-set surface. Through this classification, the amount of actual intersection operation is reduced and it makes our algorithm more efficient in terms of both time and memory space.

After the construction of the quad-tree structure, the final triangle mesh can be easily generated by dividing each of the *intersection candidate* patches into two triangles, where the two triangles share two diagonal points of the patch.

Now we will explain the two criteria of quad-tree construction in details. For clarity, ideas presented here are illustrated in 2D, but they are easily extendable to 3D. Both conditions need to utilize a bounding volume of the input NURBS surface. In this paper, instead of an axis aligned bounding box for the input NURBS surface, we use a tight parallelepiped (parallelogram for 2D cases) as the bounding volume (as shown in Fig. 9), because the axis aligned bounding box generally overestimate the enclosed patches, thus leading to unnecessary subdivisions and intersection tests. Such a parallelepiped is constructed with the help of intervals of the partial derivatives and the mean value theorem of differential calculus [7].

The first condition is used to control the maximum deviation between the final triangle mesh and the underlying NURBS surface. To simplify the computing of such deviation, we turn to control the smallest distance $d_s$ between all pairs of parallel planar faces of the bounding parallelepiped of the testing patch (parallel edges of the bounding parallelogram for 2D cases as shown in Fig. 9(a) and Fig. 9(b). If $d_s$ is larger than a specified error bound $\delta_0$, the testing patch will be divided into four sub-patches; otherwise, this testing patch will be kept as a leaf node of the quad-tree.

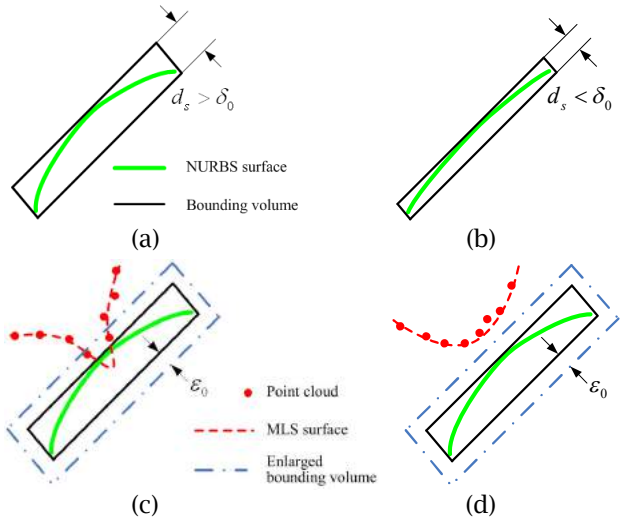



Fig. 9: Two criteria for leaf node identification in quad-tree construction. (a) The patch needs to be further subdivided due to the larger planar distance of the bounding volume according to the 1st condition. (b) The patch is identified as a leaf patch due to the smaller planar distance of the bounding volume according to the 1st condition. (c) The patch needs to be further subdivided according to the 2nd condition. (d) A patch is identified as a leaf patch according to the 2nd condition.

The second condition is used to classify the leaf patches of the quad-tree structure. Here is an easy way for leaf patch classification: first set-up a bounding parallelepiped for the testing patch, then check if there is at least one point of the point-set surface inside an enlarged bounding parallelepiped.

If yes, this patch is classified as an *intersection candidate* patch; otherwise it is classified as a *non-intersection* patch. Note, the bounding volume is enlarged by a threshold value $\varepsilon_0$ on each side to improve the robustness of the algorithm. A 2D example of such classification is shown in Fig. 9(c) and Fig. 9(d). Fig. 9(c) also illustrates potential false classification without the threshold $\varepsilon_0$ due to a relatively low sampling density of the input point data.

*4.4.2 Adaptive Intersection Between the Triangular Mesh and the MLS Surface*
With the above obtained triangular mesh, the algorithm for mesh and MLS surface intersection described in Section 5 is then applied to obtain the intersection contours between the triangular mesh and the MLS surface. When needed, the intersection points can be mapped precisely onto the parametric domain. For details, please refer to [18].

## 5. D3M APPLICATIONS

We present below the applications of the above differential geometric analysis and MLS surface based intersection algorithms in digital design and manufacturing from one set of point-cloud data.

### 5.1 Surface Curvature

Fig. 10 presents the example of applying the curvature formula derived in Section 3 on an input data cloud (consisting of 30, 246 points). The resulting principal curvatures are displayed in the Figure. Curvatures for planar curves can also be computed, which are used in the adaptive slicing and NC path generation in the following subsections.

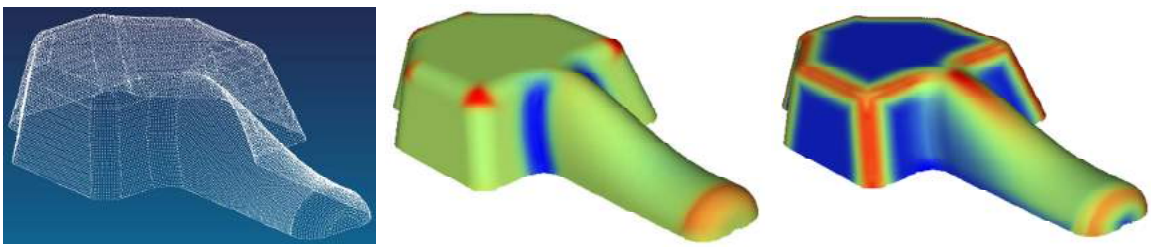



Fig. 10: Principal curvatures computed analytically from the MLS surface approximation of the input data points: (a) 30,246 data points, (b) maximum principal cuvatures; (c) minimal principal curvatures.

### 5.2 Boolean Intersection with NURBS Surfaces

The above point-cloud data can now be directly intersected with objects bounded by NURBS surfaces. Fig. 11 presents the result from the Boolean intersection between the input data set and a flower-like object represented by NURBS surfaces. In Fig. 11(d) and Fig. 11(e), the red curve represents the intersection curve. The grey triangles are those away from the intersection curve and thus are discarded much earlier then the blue triangles.

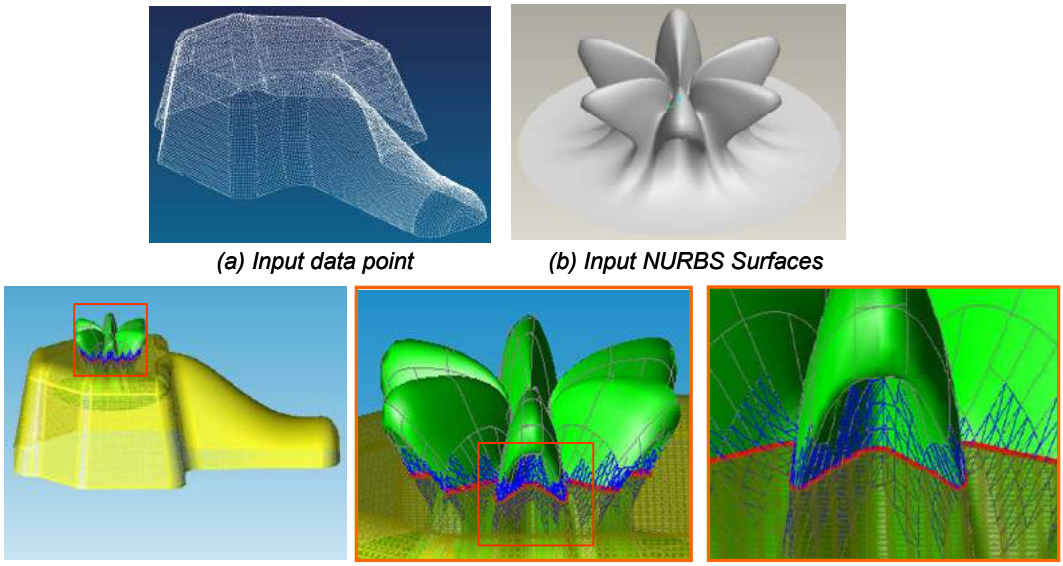
### 5.3 Slicing in Rapid Prototyping

The plane/MLS surface intersection algorithm can be applied for slicing for rapid prototyping [17]. Fig. 12 shows the adaptive slicing result from the same point-cloud data as shown in Fig. 10 and Fig. 11. By computing the curvature of the planar curve for each slice, we can obtain curvature-adaptive intersection points for each slice, as shown in Fig. 12(c). That is, in corner regions, intersection points are denser. By computing the normal curvature along the build direction, we can adaptively determine layer thickness to control the stair-case effect and thus the geometric accuracy. The resulting layer thickness is adaptive to local geometric variation, as shown in Fig. 12(d). This adaptivity benefits both build accuracy and build efficiency.

### 5.4. NC path generation

The curvature-adaptive plane/MLS surface intersection algorithm can be directly applied to NC path generation. Although, research has been done in the past to generate NC paths directly from data

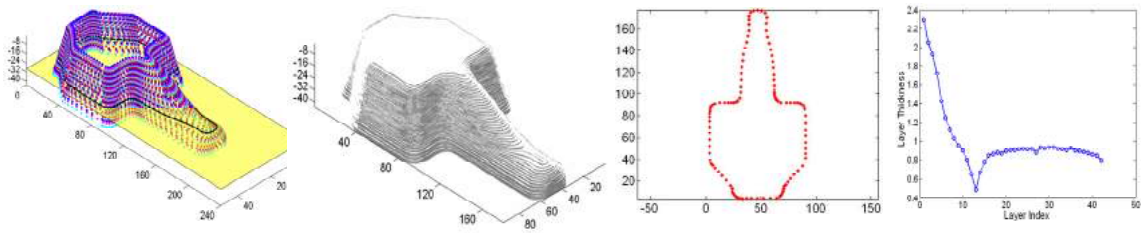
points, this is the first reported approach that can generate curvature-adaptive NC paths from discrete data points. This ensures the balance between the machining accuracy and efficiency since short step intervals usually mean accurate machining results and long machining time. Fig. 13 shows the curvature-adaptive NC paths generated directly from the point-cloud data. That is, in Y direction (feed forward direction), the forward steps are adaptive according to the normal curvature in Y direction. In X direction (side direction), the side steps are adaptive to the normal curvature in X direction. Physical prototypes machined based on this approach has been demonstrated in [20].



*(a) Input data point* *(b) Input NURBS Surfaces*

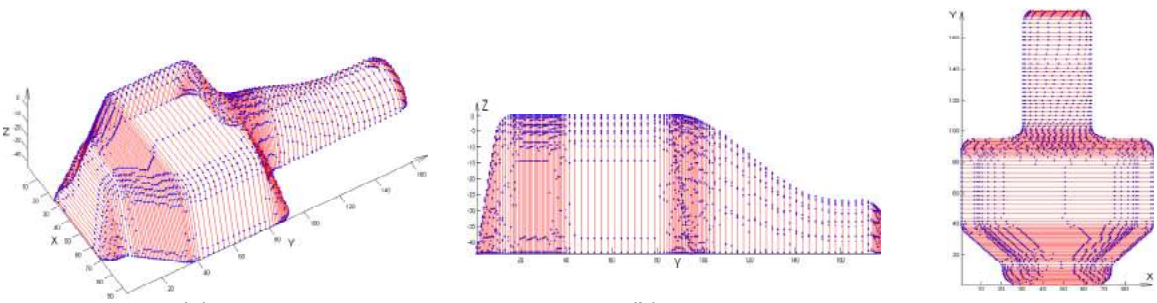
*(c) Results of adaptive intersection* *(d) Zoon-in view* *(e) Zoon-in view*

Fig. 11: Boolean intersection between point-cloud data and NURBS surfaces.



*(a) Sliced model* *(b) Shaded slices* *(c) Slice profile* *(d) Layer thickness*

Fig. 12: Adaptive slicing from point-cloud data: (a) sliced model, (b) shaded slices, (c) profile of the highlighted slice in (a), (d) the adaptive distribution of layer thickness.



*(a)* *(b)* *(c)*

Fig. 13: Curvature-adaptive NC path generated directly from point-cloud data.

## 6. D3M IN HEADMASK DEVELOPMENT

The section above presents the examples demonstrating basic capabilities of D3M from massive point-cloud data. We now present the application of D3M on a specific custom product development: developing a custom headmask which involves both designed geometry and custom face geometry obtained as point-cloud data.

This example application is shown in Fig. 14 where a customer-specific headform for chemical masks is to be developed for leak testing. In this example, the base template part is created in a CAD system. The mask surface shape comes from customer-specific faces. Existing approach would involve a lengthy point cloud cleaning process prior to the polygonal model reconstruction, and a laborious and error-prone NURBS surface reconstruction before the reconstructed head model is imported into a CAD system for Boolean operations with the designed template to produce the customer-specific headform. The D3M approach allows direct Boolean intersection between the design model and the acquired point cloud and it has led to substantial time reduction in design and prototyping. We now examine the process efficiency and the resulting model accuracy.

### 6.1. Process Efficiency

Fig. **15** presents the development of the headmask for two scanned geometry. The designed model has undergone parametric modification from Head A to Head B in the CAD system. In the existing

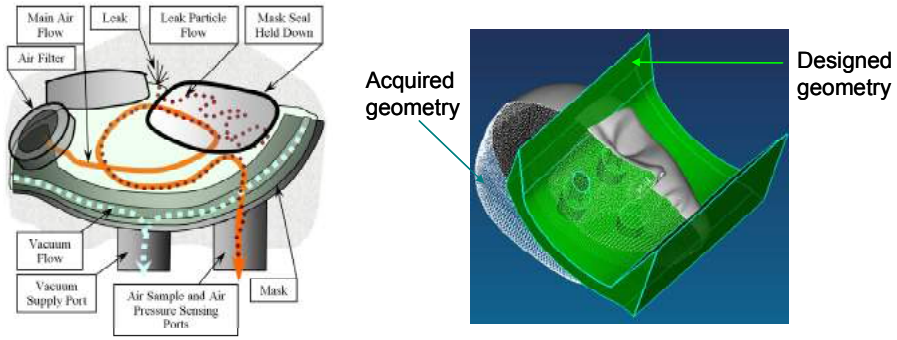



Fig. 14: Designing and manufacturing of a customer-specific headform: (a) Physical mask, (b) Acquired head model and the designed mask template.

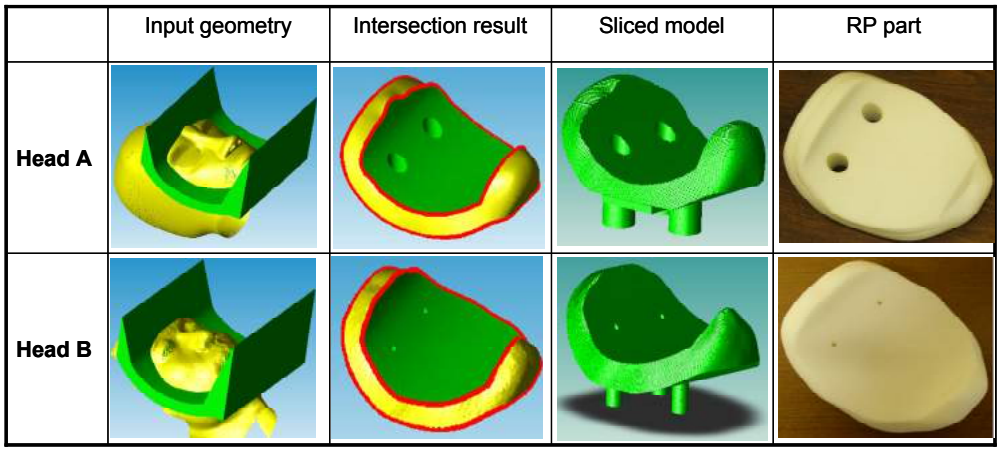



Fig. 15: D3M enabling mass customization: Row A and Row B are the Boolean intersection results, sliced models and resulting RP parts of the same D3M process on different scanned head models.

| *Task Step* | *Description* | *Task Type* | *Required Time* | *Iterations* |
|---|---|---|---|---|
| 1 | Data Orientation | Manual | 5 min. | 1 |
| 2 | Create Design Geometry as NURBS Surfaces | Manual | 24 min. 34.64 sec. | 1 |
| 3 | NURBS Subdivision | Automatic | 2.851 sec. | 1 |
| 4 | Calculate Intersection of NURBS Surfaces and Point Cloud | Automatic | 16.203 sec. | 1 |
| 5 | Set Membership Classification for Boolean Operations | Manual* | 11.315 sec. | 1 |
| 6 | System Calculates Contours and Generates SLC Files | Automatic | 16.415 sec. | 1 |
| 7 | Create Manufacturing File from SLC | Automatic | 27 sec. | 1 |
| Total Time | | | 30 min. 48 sec. | |

* Currently manual and its automation is being implemented.

Tab. 1: Time for D3M on the head form development.

NURBS based modeling approach, it takes weeks of engineering times and due to practical logistical constraints about one month for a senior engineer to create a NURBS surface model from the acquired points. By applying the D3M approach, the above tedious model conversion processes are avoided and the overall modeling time, involving the point-cloud data, is dramatically reduced to less than 1 minute. Tab. 1 lists the specific time for each step involved in this process. As we can see, the initial creation of design geometry in a CAD system takes up majority of the product development time. However, the surface reconstruction from point-cloud data is bypassed.

### 6.2. Resulting Model Accuracy

The Gaussian parameter is the most important parameter in defining an MLS surface. Like parameters in other surface fitting methods (e.g., the number of control points in NURBS surface approximation), the Gaussian parameter affects smoothness and accuracy of the MLS surface. Fig. 16, depicting half of the scanned face corresponding to Column 3 in
Fig. **15** shows the error distribution of each data point against the MLS surface under different Gaussian parameters. As one would expect, at large Gaussian parameter h, there is large smoothing effect and the resulting surface is smoother but with substantial and systematic bias. When h is too small, the resulting surface interpolates the data. In the middle, the error approaches random and has the effect of smoothing the data noise. Hence a good choice of h can be determined by examining the distribution of data error.

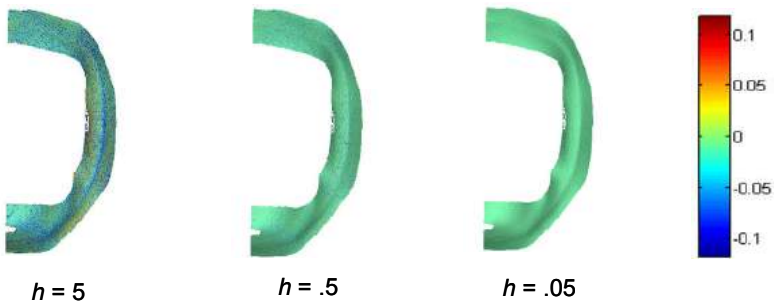



Fig. 16: Influence of Gaussian width on the MLS surface accuracy: larger h leading to biased surface (h = 5), smaller h leading to data interpolation and good choice of h smoothing out the noise (h = 0.05) and yielding near-random error distribution (h = 0.5).

## 7. CONCLUSIONS

In this paper, a direct digital design manufacturing (D3M) approach has been introduced for product development involving scanned point-cloud data. Due to the use of the moving least-squares (MLS)

surface as the underlying surface representation for acquired point-sampled geometry, it affords us many desirable properties, including projection-based line/MLS surface intersection, closed formula for computing curvature for planar curves, which enables curvature-adaptive intersection between MLS surface and CAD geometry. Examples demonstrate D3M offers an effective and efficient means for rapid development of custom products where scanned geometry is directly used in product development without CAD model reconstruction.

## 8. ACKNOWLEDGMENTS

This work was supported by the U.S. National Science Foundation Award #0529165 and Award #0800912, and Air Force Office of Scientific Research Award #FA9550-07-1-0241. We also acknowledge the help from Dongdong Zhang for generating NC paths for this paper.

## 9. REFERENCES

[1] Amenta, N.; Kil, Y. J.: Defining point-set surfaces, ACM Trans. Graph., 23(3), 2004, 264-270.
[2] Amenta, N., Kil, Y. J.: The domain of a point set surface, In: Eurographics Workshop on Point-based Graphics, 2004, 139-147.
[3] Azariadis, P. N.; Sapidis, N. S.: Drawing curves on a cloud of points for point-based modeling, Computer-Aided Design, 37, 2005, 109 – 122.
[4] Cripps, R. J.: Algorithms to support point-based CADCAM, International Journal of Machine Tools & Manufacture, 43, 2003, 425 – 432.
[5] Feng, H. Y.; Teng, Z.: Iso-planar piecewise linear NC tool path generation from discrete measured data points, Computer-Aided Design, 37(1), 2005, 55-64.
[6] Goldman, R.: Curvature formulas for implicit curves and surfaces, Computer Aided Geometric Design, 22(7), 2005, 632-658.
[7] Huber, E.; Barth, W.: Surface-to-surface Intersection with Complete and Guaranteed Results, In Developments in Reliable Computing, T. Csendes (ed.), Kluwer, 1999, 185-198.
[8] Kumar, S.; Kalra, G.; Dhande, S. G.: Direct Layered Manufacturing of Point Sampled Geometry, Int. J. Manufacturing Technology & Management, 6(6), 2004, 534-549.
[9] Levin, D.: Mesh-independent surface interpolation, In: Brunnett G, Hamann B, Muller H, Linsen L, editors, Geometric modeling for scientific visualization, Springer-Verlag; 2003, 37-49.
[10] Levin, D.: The approximation power of moving least-squares, Mathematics of Computation, 67, 1998, 1517-1531.
[11] Lin, A.; Liu, H. T.: Automatic generation of NC cutter path from massive data points, Computer-Aided Design, 30, 1998, 77 – 90.
[12] Liu, G. H.; Wong, Y. S.; Zhang, Y. F.; Loh, H. T.: Error Based Segmentation of Cloud Data for Direct Rapid Prototyping, Computer-Aided Design, 35(7), 2003, 633-645.
[13] Park, S.; Chung, Y. C.: Tool path generation from measured data, Computer-Aided Design, 35, 2003, 467 – 475.
[14] Press, W.; Flannery, B.; Teukolsky, S.; Vetterling, W.: Numerical recipes in C. 2nd ed. Cambridge University Press, 1992.
[15] Shin, H.; Park, S.; Park, E.: Direct Slicing of a Point Set Model for Rapid Prototyping, Computer-Aided Design and Applications, 1(1-4), 2004, 109-115.
[16] Wu, Y. F.; Wong, Y. S.; Loh, H. T.; Zhang, Y. F.: Modeling Cloud Data Using an Adaptive Slicing Approach, Computer-Aided Design, 36(3), 2004, 231-240.
[17] Yang, P.; Qian, X.: Adaptive Slicing of Moving Least Squares Surfaces: Toward Direct Manufacturing from Point Cloud Data, ASME Transactions Journal of Computing and Information Science in Engineering, 8(3), Sep 2008.
[18] Yang, P.; Qian, X.: Direct Boolean Intersection between Acquired and Designed Geometry, Computer-Aided Design, accepted.
[19] Yang, P.; Qian, X.: Direct computing of surface curvatures for point-set surfaces, Proceedings of 2007 IEEE/Eurographics Symposium on Point-based Graphics (PBG), Prague, Czech Republic, Sep 2007.
[20] Zhang, D.; Yang, P.; Qian, X.: Adaptive NC Path Generation from Massive Point Data with Bounded Error, ASME Transactions Journal of Manufacturing Science and Engineering, Vol. 131, 011001-113, Feb 2009.