Computer-Aided Design 41 (2009) 81–94

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Direct boolean intersection between acquired and designed geometry

Pinghai Yang, Xiaoping Qian *

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

ARTICLE INFO

Article history:
Received 11 April 2008
Accepted 31 December 2008

Keywords:
Shape modeling
Point-sampled geometry
Surface intersection
Boolean operations

ABSTRACT

In this paper, a new shape modeling approach that can enable direct Boolean intersection between acquired and designed geometry without model conversion is presented. At its core is a new method that enables direct intersection and Boolean operations between designed geometry (objects bounded by NURBS and polygonal surfaces) and scanned geometry (objects represented by point cloud data).

We use the moving least-squares (MLS) surface as the underlying surface representation for acquired point-sampled geometry. Based on the MLS surface definition, we derive closed formula for computing curvature of planar curves on the MLS surface. A set of intersection algorithms including line and MLS surface intersection, curvature-adaptive plane and MLS surface intersection, and polygonal mesh and MLS surface intersection are successively developed. Further, an algorithm for NURBS and MLS surface intersection is then developed. It first adaptively subdivides NURBS surfaces into polygonal mesh, and then intersects the mesh with the MLS surface. The intersection points are mapped to the NURBS surface through the Gauss–Newton method.

Based on the above algorithms, a prototype system has been implemented. Through various examples from the system, we demonstrate that direct Boolean intersection between designed geometry and acquired geometry offers a useful and effective means for the shape modeling applications where point-cloud data is involved.

1. Introduction

Rapid advancement of 3D sensing technologies has spurred the growing interest in shape modeling from scanned point-cloud data [1] as evidenced by growing need for reverse engineering of physical artifacts in automotive, aerospace, consumer product and entertainment industries, and increasing practice of patient-specific biomedical implants, customer-specific product design and manufacturing (e.g., apparel and footwear). Shape modeling from scanned geometry is usually based on polygonal models after a set of pre-filtering and post-processing steps that reconstruct polygonal models from noisy point-cloud data. Shape modeling based on non-uniform rational B-spline (NURBS) surfaces, the standard surface representation in CAD systems, reconstructed from scanned geometry requires a further laborious patch layout and fitting process. These multiple steps make it hard to maintain error bound in the complex modeling pipeline. Many of these steps require substantial human intervention, and make it inefficient for many applications.

This paper aims to develop techniques toward the goal of direct shape modeling from acquired and design geometry. Here direct shape modeling refers to a shape modeling approach that enables shape modeling with the native representations of acquired geometry (i.e. discrete points) and designed geometry (i.e. NURBS surfaces and triangle mesh) without representation conversion. Such a shape modeling capability can potentially benefit a host of product development applications such as mass customization and product redesign where designed geometry frequently interacts with the acquired geometry.

An example application is shown in Fig. 1 where a customer-specific headform for chemical masks is developed in 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 geometric modeling system for Boolean operations with the designed template to produce the customer-specific headform. Fig. 1(c) presents the result from our direct Boolean intersection. The direct Boolean intersection operation between the design model and the acquired point cloud has led to substantial time reduction in design and prototyping (details are in Section 7.3). The final computer model is shown in Fig. 1(c) and the resulting fabricated custom-part is shown in Fig. 1(d).

More specifically, we use the moving least-squares (MLS) surface as the underlying surface representation for acquired point-sampled geometry in the shape modeling operations. The key to our direct Boolean intersection is a new algorithm for

* Corresponding author. Tel.: +1 312 567 5855.
E-mail address: qian@iit.edu (X. Qian).

0010-4485/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2008.12.006

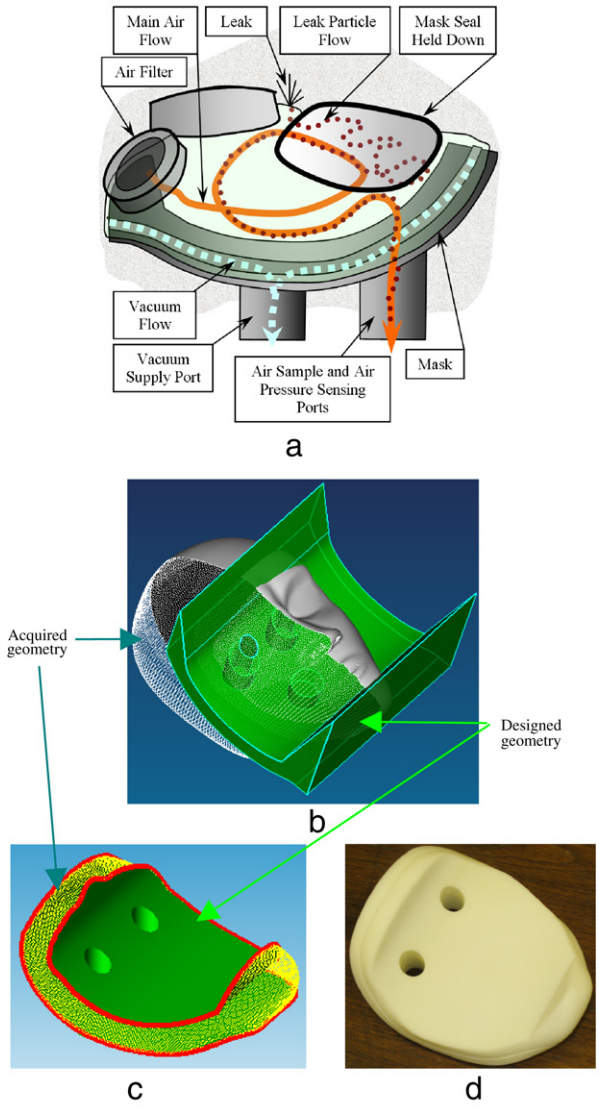


Fig. 1. Designing and manufacturing of a customer-specific headform: (a) Physical mask, (b) Acquired head model and the designed mask template. (c) Direct Boolean intersection result. (d) Rapid prototyped part from the intersection resulting model.

line/MLS surface intersection, which is based on the projection property of the MLS surfaces. The intersection between a NURBS surface and an MLS surface is through a subdivision process, a marching process, and a mapping process. We adaptively subdivide a NURBS surface into a planar triangular mesh that is denser at intersection regions and sparser at non-intersection regions. 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. This intersection process can handle both opening branches and closed internal loops. The resulting intersection points are then mapped back to the NURBS surface and hence reside on both the MLS and NURBS surfaces. Intersection curves generated from these points are used in Boolean operations between objects defined by NURBS and MLS surfaces.

Since the polygonal mesh is an intermediate model used for NURBS/MLS surface intersection, our method thus supports shape modeling from designed geometry in either NURBS or polygonal forms or any other surface representations that can be converted into the polygonal form. Since an MLS surface has the intrinsic ability to handle noisy input, our method supports shape modeling directly from acquired geometry in its native point form without any pre- or post-processing steps.

The contribution of our work includes the components mentioned below.

- Mathematically, closed formula for direct curvature computing in planar curves is derived. Through the implicit definition of an MLS surface, we derive a formula for curvature computing for 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.
- Application-wise, this paper contributes a method that enables direct Boolean intersection between designed and acquired geometry.

The rest of this paper is structured as follows. Section 2 presents related work. Section 3 reviews the MLS surface definition and presents how it enables closed formula for direct curvature computing in planar curves. Section 4 details the projection-based line/MLS surface intersection and the curvature-adaptive plane/MLS surface intersection. Section 5 introduces the triangular mesh/MLS surface intersection. Section 6 presents the NURBS/MLS surface intersection. Section 7 presents the experimental results. Discussion on threshold parameters used in the intersection process is in Section 8. This paper concludes in Section 9.

2. Related work

Shape modeling from point-sampled geometry has recently gained popularity. A number of point-based representations, such as surfel [1,2] and MLS [3–5], have been proposed and proven to be successful in point-based 3D modeling and rendering. Based on these representations, algorithms for surface intersections and Boolean operations on point-sampled geometry are developed: e.g. point-based shape editing in Pointshop3D [6], Adams et al. presented an algorithm to perform interactive Boolean operations on free-form solids bounded by surfels [7]; Pauly et al. presented an MLS-based free-form shape modeling framework for point-sampled geometry [1]. Yang and Qian presented a method for computing surface curvatures in MLS surfaces [8].

Surface/surface intersection is an important problem in shape modeling. In the area of CAD/CAM, the problem of parametric NURBS surface intersection has been widely investigated. The methods can be categorized in two major types: subdivision based [9,10] and marching based [11,12]. In subdivision-based algorithms, the surfaces are subdivided into a large number of facets and the intersection of surfaces is approximated by the intersection of the facet pairs. In marching algorithms, the basic idea is to trace the points on the intersection curve from a starting point known to be on the intersection curve.

However, to the best of our knowledge, no work on surface intersection and Boolean operations between the designed geometry (NURBS surfaces) and acquired geometry (point-sampled geometry) has been reported in the literature.

3. MLS surface definition and closed curvature formula for planar curves on an MLS surface

This section first gives a brief introduction on the definition of an MLS surface [4,13–15,3], which forms the basis of our line/MLS surface intersection in Section 4.1. Based on our earlier work on curvature computing in MLS surfaces [8], we then give a closed formula for computing the curvature for planar curves on the MLS surface, which forms the basis for curvature-adaptive plane/MLS surface intersection algorithm in Section 4.2.

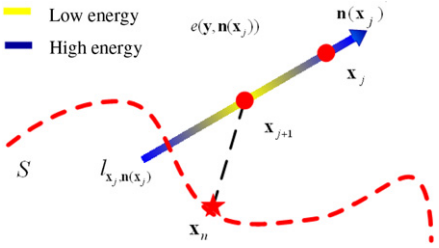


Fig. 2. Illustration of the MLS projection process.

3.1. Projection-based MLS surfaces

Levin [15,3] defined an MLS surface S as the stationary set of a projection operator ψ_P, i.e.,

$$S = \left\{\mathbf{x} \in R^3 \mid \psi_P(\mathbf{x}) = \mathbf{x}\right\}. \tag{1}$$

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 [4,13] 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. 2. 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. 2.

For details of this projection procedure, please refer to [4, 13]. 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:

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

where

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

is a Gaussian weighting function, where h is a scale factor that determines the width of the Gaussian kernel [14].

In the jth 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. 2. Such an energy function $e : R^3 \times R^3 \to R$ can be defined as

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

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

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

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. To improve the efficiency of this scheme, we adopt a standard data sorting algorithm based on the k–d tree structure [16] to identify the neighbors of a given point. This structure also benefits the leaf patch classification in Section 5.1. Throughout the rest of this paper, this projection-based MLS scheme will be used for locally approximating the underlying surface from a set of sample points.

3.2. Computing curvature of planar curves in an MLS surface

We presented a set of analytical equations for direct computing surface curvatures [8] from point-set surfaces based on the implicit definition of an MLS surface. Applying the same method, we present below the analytical curvature formula for planar curves, which is the key in determining the error-bounded curvature-adaptive step length in the plane/MLS surface intersection process.

In this method, the native form of MLS is first converted into an implicit form. It has been proved in [4] that the MLS surface is actually the implicit surface given by the zero-level set of the implicit function

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

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

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

For a more elegant expression of this planar curve, we can transform the plane 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

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

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

Applying a curvature formula given in [17], we have the curvature of this implicit planar curve as

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

where T is the unit tangent vector of the implicit curve $\hat{g}(x, y)$ as

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

and

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

is the gradient of $\hat{g}(x, y)$, $H(\hat{g}(x, y)) = \nabla(\nabla(\hat{g}(x, y)))$ is the Hessian matrix of $\hat{g}(x, y)$. 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

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

The specific expressions for $\nabla(\widehat{g}(x, y))$ and $H(\widehat{g}(x, y))$ are in the Appendix.

4. Intersection of an MLS surface with a line and a plane

The intersection of an MLS surface with a line and a plane forms the basis for our approach in direct Boolean intersection.

4.1. Intersection of an MLS surface with a line

The line/MLS surface intersection has been discussed in the context of ray tracing in [18], where the intersection point is generated as the intersection of a local polynomial and the ray. As such, miss-shooting of the ray with the surface may occur. However, since in this paper we adopt a different definition of the MLS surface [4,13], where the fitting of a local bivariate polynomial is omitted, we must develop a new method for line/MLS surface intersection.

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

$$\|\psi_P(\mathbf{x}) - \mathbf{x}\| = 0. \tag{9}$$

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 Eq. (9) 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 1D problem by substituting $\mathbf{x} = \mathbf{c} + t \cdot \mathbf{n}$ into Eq. (9), where t is the only variable. In this paper, Brent's method is implemented to solve this 1D 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 1D root finding problems [19].

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 ε_0 (e.g., blue circles shown in Fig. 3). The assumption here is that each projected point on the MLS surface is maximally at ε_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. 3). 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. 3, 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. Intersection of an MLS surface with a plane

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.

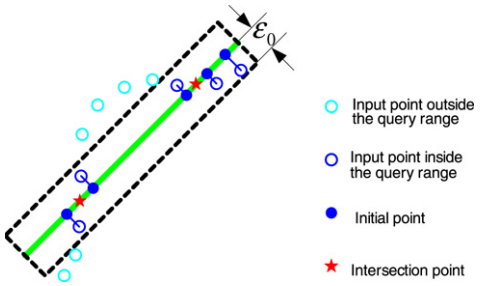


Fig. 3. Strategy for generating different initial points for locating multiple line/MLS surface intersection points: points that are ε_0 distance away from the line are projected onto the line and the projected points are then used as the starting points to find the intersection points between the line and the MLS surface. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

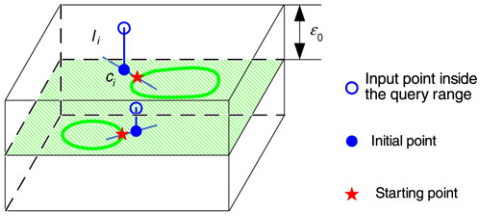


Fig. 4. Strategy for locating the starting points for the marching process in plane/MLS surface intersection: points that are ε_0 distance away from the plane are projected onto the plane and the projected points are then used as the initial points to find the intersection points between the line and the MLS surface. These intersection points become the candidate starting points for the marching process.

4.2.1. Finding starting points for the marching process

Here is the strategy we used to find starting points. First, find all points of the input point data inside a query range, i.e., having a distance to the input plane within a prescribed distance ε_0. Second, project these points onto the plane and use the projected points as initial points. Third, for each of these initial points $\mathbf{c}_i$, apply the MLS projection algorithm introduced in Section 3 and get the corresponding point $\mathbf{c}'_i$ on the MLS surface. Then use the following formula to calculate the direction vector for the intersection line l_i:

$$\mathbf{n}_i = (\mathbf{n}'_i \times \mathbf{n}_H) \times \mathbf{n}_H$$

where $\mathbf{n}'_i$ is the normal of the MLS surface at $\mathbf{c}'_i$ and $\mathbf{n}_H$ is the normal of the plane H. This direction vector $\mathbf{n}_i$ corresponds to the MLS surface point $\mathbf{c}'_i$'s normal projected onto the plane H. Hence, intersection line l_i can be defined by the initial point $\mathbf{c}_i$ and the direction vector $\mathbf{n}_i$. Finally, apply the line/MLS surface intersection algorithm to get the intersection point of the line l_i and the MLS surface S as a candidate starting point. If this candidate point has a distance to all previous intersection curves larger than a given threshold, it will be accepted as a new starting point. With this strategy, multiple starting points can be identified when multiple intersection loops exist as shown in Fig. 4.

4.2.2. Curvature-adaptive marching for the plane/MLS surface intersection

In this section, we propose a new methodology to march along each intersection curve to get the successive intersection

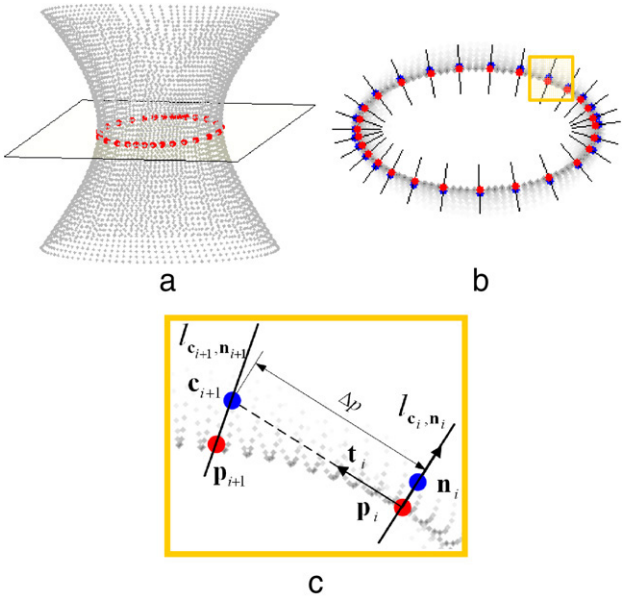


Fig. 5. Illustration of adaptive marching in plane/MLS surface intersection. (a) Iso-view of a point data with resulting 2D contour on a slicing plane. (b) Top-view of the slice at $z = 0.1$. (c) Zoom-in.

points with adaptive step length calculated based on the analytical curvature formula for planar curves derived in Section 3.2. 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 $\mathbf{n}_0$ obtained as in Section 4.2.1. 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 $\mathbf{p}_0$.

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. 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 $\mathbf{n}_i$:

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

where $\mathbf{t}_i$ is the unit vector perpendicular $\mathbf{n}_i$, Δp is the step length. To compute the step length Δp, we first approximate the planar 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:

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

where δ_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 by Eq. (8) 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

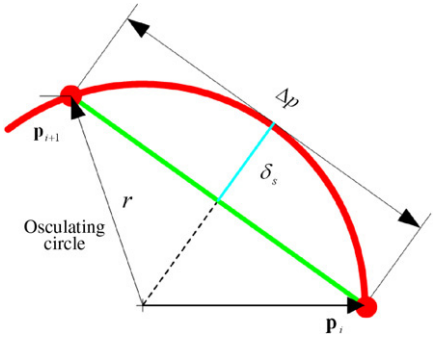


Fig. 6. Computing error-bounded step length Δp based on an osculating circle.

$r_{\max}$ 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 Eq. (10); 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.

Remark. *The above line/MLS and plane/MLS surface intersection algorithms are resolution complete*. That is, our algorithm is guaranteed to find all intersection points/curves if the resolution (point spacing) of the input point data is fine enough.

5. Adaptive intersection of an MLS surface with a triangular mesh

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 has 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. 7. 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.

5.1. 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.

5.2. 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 ε, we find all points that have a distance to the input triangle within ε; (2) candidate starting points outside the triangle are omitted.

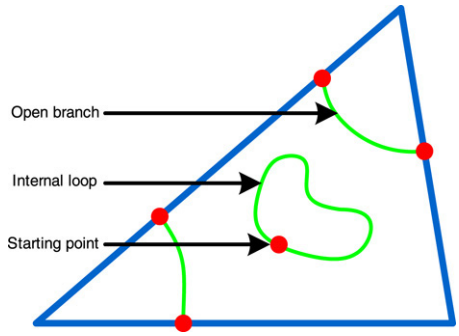


Fig. 7. Types of intersection curves.

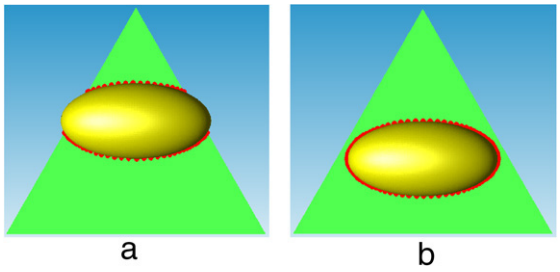


Fig. 8. Curvature-adaptive marching for triangle/MLS surface intersection. (a) Open branches. (b) One internal loop.

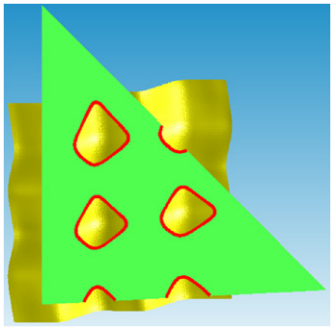


Fig. 9. Intersection between a triangle and an MLS surface resulting in multiple open branches and internal loops.

5.3. 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. 8 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. 9 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.

Repeated use of the triangle/MLS surface intersection would then result in the intersection contours between a triangular mesh and an MLS surface. Examples are shown in Section 7. 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.

6. Intersection and Boolean operations between NURBS and MLS surface models

The intersection and Boolean operations between design geometry (NURBS surfaces) and acquired geometry (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 the combined 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 the potentially intersecting region based on this tree structure.
2. Adaptive intersection of a triangular mesh and an MLS surface.
3. Mapping the intersection points to the NURBS surfaces and represent each intersection curve in either the polyline form or the NURBS curve form.
4. Set membership classification for Boolean operations.

6.1. Adaptive subdivision of NURBS surfaces for accurate and efficient intersection

Since our NURBS/MLS surface intersection is based on plane/MLS surface intersection, we 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. 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. 10 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.

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:

- it deviates from a "best fit" plane within a given tolerance;
- 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.

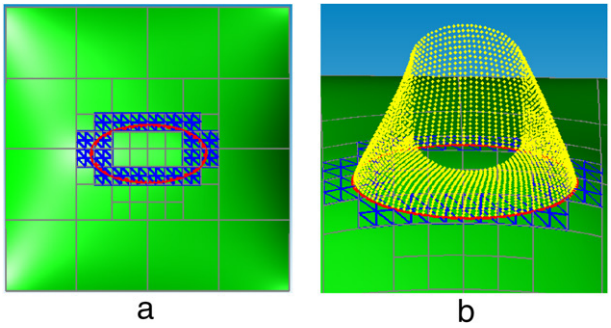


Fig. 10. Adaptive subdivision of a NURBS surface ensuring accurate and efficient NURBS/MLS surface intersection: Patches closer to the intersection curves undergo more times of subdivision. Planar triangles are only generated from the leaf parches that can potentially intersect with the MLS surface. (a) Top-view. (b) Iso-view. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

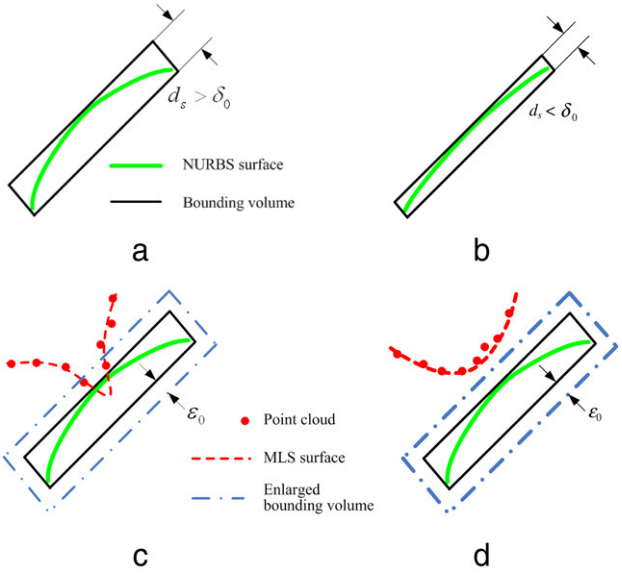


Fig. 11. Two criteria for leaf node identification in quad-tree construction. (a) The patch needs to be further subdivided due to the larger planar distance of the bounding volume according to the 1st condition. (b) The patch is identified as a leaf patch due to the smaller planar distance of the bounding 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.

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 detail. 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. 11), 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 [20].

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. 11(a) and (b)). If d_s is larger than a specified error bound δ_0, the testing patch will be divided into four subpatches; otherwise, this testing patch will be kept as a leaf node of the quad-tree.

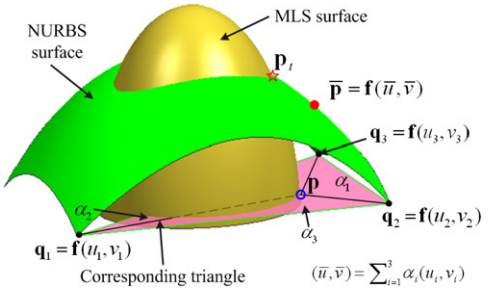


Fig. 12. Procedure for refining an intersection point between a NURBS surface and an MLS surface. Let point p (blue circle) be an intersection point between the MLS surface and the triangular mesh generated from the NUBS surface. Based on its barycentric coordinates, the point is mapped onto the NURBS surface as the point $\bar{p}$ (red solid circle), then it is refined into point p_t (red pentacle) with the Gauss–Newton method. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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 ε_0 on each side to improve the robustness of the algorithm. A 2D example of such classification is shown in Fig. 11(c) and (d). Fig. 11(c) also illustrates potential false classification without the threshold ε_0 due to a relatively low sampling density of the input point data.

6.2. 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.

6.3. Intersection point mapping and intersection curve generation

The accuracy of intersection points from the above algorithm is bounded by the user-specified triangulation error δ_0. However, the intersection points between the triangular mesh and the MLS surface may not lie exactly on the corresponding NURBS surface, due to the approximation error between the triangular mesh and the NURBS surface, as shown in Fig. 12. We introduce below a set of steps to obtain accurate intersection points that lie on both MLS and NURBS surfaces. Due to the accurate mapping to the NURBS surface, this makes the overall intersection process robust, i.e. the accuracy does not depend on the user-specified triangulation error δ_0.

STEP 1: Directly mapping the intersection points onto the input NURBS surface. To do this, we adopt the following method based on barycentric coordinates: For each intersection point **p**, we can find the triangle that contains **p**. We know the (u, v) parameter values for each vertex in the NURBS surface subdivision process. Let barycentric coordinates of **p** with respect to a particular triangle T are α_1, α_2 and α_3, the projected parameter of **p** can be obtained as,

$$(\bar{u}, \bar{v}) = \sum_{i=1}^{3} \alpha_i (u_i, v_i)$$

where T's vertices have parameter values (u_i, v_i). Then the point $\mathbf{p}$ can be mapped onto the underlying NURBS surfaces by computing the surface point for the parameter value $(\bar{u}, \bar{v})$.

STEP 2: Adjusting the projected points. Having finished the projection procedure, the projected points are located on the input NURBS surfaces but, in general, not on the MLS surface. Denote a mapped point by $\bar{\mathbf{p}} = \mathbf{f}(\bar{u}, \bar{v})$, where

$$\mathbf{f}(u, v) = (\mathbf{x}(u, v), \mathbf{y}(u, v), \mathbf{z}(u, v))$$

presents the corresponding NURBS surface. We evaluate the deviation between $\bar{\mathbf{p}} = \mathbf{f}(\bar{u}, \bar{v})$ and the MLS surface by $\|\psi_P(\mathbf{x}) - \mathbf{x}\|$. If this deviation exceeds the given tolerance δ_s, a refinement must be performed by solving the non-linear system

$$\begin{cases} g(\mathbf{x}) = 0 \\ \mathbf{x} = \mathbf{f}(u, v) \end{cases} \tag{11}$$

where $g(\mathbf{x})$ is the implicit function of the MLS surface, which is given in Eq. (5). Note, the initial parameter values $(\bar{u}, \bar{v})$ is already a good approximation of the parameter values (u_t, v_t) of the true intersection point $\mathbf{p}_t$ and the Jacobian matrix of g over (u, v) can be analytically obtained by

$$\mathbf{J}(g(\mathbf{f}(u, v))) = \nabla^{\mathrm{T}} g(\mathbf{f}(u, v)) \cdot \nabla \mathbf{f}(u, v),$$

where ∇g is available in [8] and $\nabla \mathbf{f}$ is the standard NURBS surface derivative given in [21]. This non-linear system can be efficiently solved using the Gauss–Newton optimization method [22], where the refined parameter values (u_t, v_t) can be iteratively calculated by

$$\left(u_t^{(k+1)}, v_t^{(k+1)}\right)^{\mathrm{T}} = \left(\mathbf{J}^{(k)\mathrm{T}} \cdot \mathbf{J}^{(k)}\right)^{-1} \cdot \mathbf{J}^{(k)\mathrm{T}} \cdot \left(\mathbf{J}^{(k)} \cdot \left(u_t^{(k)}, v_t^{(k)}\right)^{\mathrm{T}} - g\right),$$

where k is the iteration number.

STEP 3: Refining the intersection points and approximating intersection curves. After adjusting, the resulting points will serve as the point array for constructing intersection curves by fitting (interpolation or approximation) technique.

Obviously, the deviations between the approximated curves and the theoretical curves at those projected points are within a given tolerance. However, the deviation in the corresponding intervals still needs further examination by an intersection point refinement process.

For convenience, we consider that the maximum deviation in every interval is located at the middle point. For example, in interval between points (u_0, v_0) and (u_1, v_1) on the parametrical domain, the point with the maximum deviation is $(\bar{u}_{1/2}, \bar{v}_{1/2}) = ((u_0 + u_1)/2, (v_0 + v_1)/2)$. Similarly, we can evaluate the deviation between $\bar{\mathbf{p}}_{1/2} = \mathbf{f}(\bar{u}_{1/2}, \bar{v}_{1/2})$ and the MLS surface by Eq. (9). If this deviation exceeds the given tolerance, a new intersection point should be added, which is generated by solving Eq. (11) with an initial point of $\bar{\mathbf{p}}_{1/2} = \mathbf{f}(\bar{u}_{1/2}, \bar{v}_{1/2})$.

The final intersection points are obtained by repeating this process for all the intervals, which will then be used to fit the satisfied intersection curves.

Discussion: In the above three steps, the first step is necessary to provide a good initial guess to obtain the intersection points on the NURBS surface.

The second step overcomes the triangular approximation of the NURBS surface by finding accurate intersection points through directly solving the non-linear equation. Therefore, when the triangular mesh is not fine enough, e.g. due to the selection of a large triangulation error bound δ_0, the intersection inaccuracy due to the triangular approximation is eliminated in STEP 2.

The third step ensures accurate distribution of intersection points even when the marching step length is too large due to the selection of a large marching error bound δ_s or the discrepancy between planar curves on the triangular plane and the NURBS surface. That is, the deviation in every interval can be reduced in STEP 3.

Therefore, through the above three steps, the overall algorithm for NURBS/MLS surface intersection is less sensitive to the selection of the two error bounds, δ_0 and δ_s than otherwise.

6.4. Boolean operations between the NURBS and MLS surface models

With the above method for intersecting NURBS surfaces with an MLS surface via a triangular mesh, we can easily extend it to Boolean operations between design geometry bounded by NURBS and polygonal surfaces and acquired geometry bounded by an MLS surface.

The results of Boolean operations are defined by the original surfaces and the resulting boundaries. For example, a trimmed MLS surface is composed of the original MLS surface and a set of boundary points, which are analogous to the boundary "p-curves" in NURBS surface trimming. Note, the linear interpolation of intersection points approximates the true intersection curve with bounded error since the distribution of intersection points are adaptive to local curvature. The original MLS surface is now described only by points bounded by the boundary curve and the points outside the boundary curve but within the Gaussian compact support region of the boundary points.

During the rendering process, the surface points outside the trimming boundary is removed, which is achieved by set membership classification. The classification method for classifying a subset of a point set against NURBS and polygonal surfaces is commonly available, for example, in [23]. The classification of subsets of NURBS and polygonal models against an MLS surface is done by sampling points in the NURBS patches or polygons and classifying them against the MLS surface. The method for the point/MLS surface classification is available in [1].

7. Implementation and examples

We have implemented the algorithms presented in this paper with Visual C++ 6.0. We used ACIS as the solid modeling kernel to model designed geometry and used both synthetic point cloud and acquired real point-cloud data as acquired geometry. We present below a few examples highlighting the direct Boolean intersection results based on our implementation.

7.1. Mesh/MLS surface intersection (Example 1)

Fig. 13 presents an example of intersection between a polygonal mesh and a point-set surface. This example demonstrates the ability of our algorithms in directly handling the intersection between polygonal and MLS surfaces. The time efficiency is shown in Table 2.

7.2. NURBS/MLS surface intersection (Examples 2 and 3)

The second example is on NURBS/MLS surface intersection and is depicted in Fig. 14. Fig. 14(a) shows the original NURBS model of a human head, composed of 54 NURBS patches in blue; Fig. 14(c) and (d) show the result of a cut-and-paste operation: the ear has been extracted from the Max Planck model and pasted onto the NURBS model. This example demonstrates that our direct Boolean intersection can handle intersection between multiple NURBS surfaces and an MLS surface.

Fig. 15 shows the third example on Boolean operations between a NURBS model (the airfoil) and a point-cloud data (the turbine blade). In this example, a new turbine blade is designed based on a point cloud scanned from an old turbine blade (Fig. 15(a)) and a newly designed airfoil (Fig. 15(b)), which illustrates how a new design problem is efficiently and effectively addressed by reusing

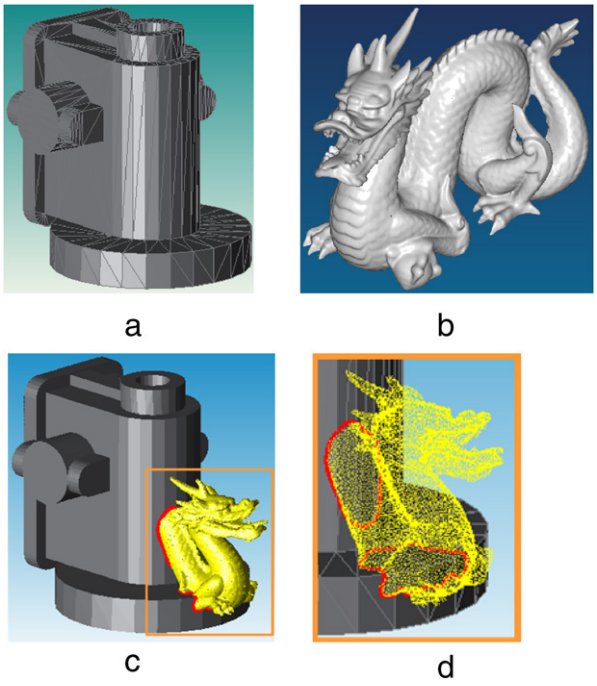


Fig. 13. Example 1: Intersection between polygonal and point-set surfaces. (a) Triangular mesh of a reducer. (b) Point-set surface of a dragon. (c) Left-front-view of the intersection result. (d) Zoom-in of the result in (c).

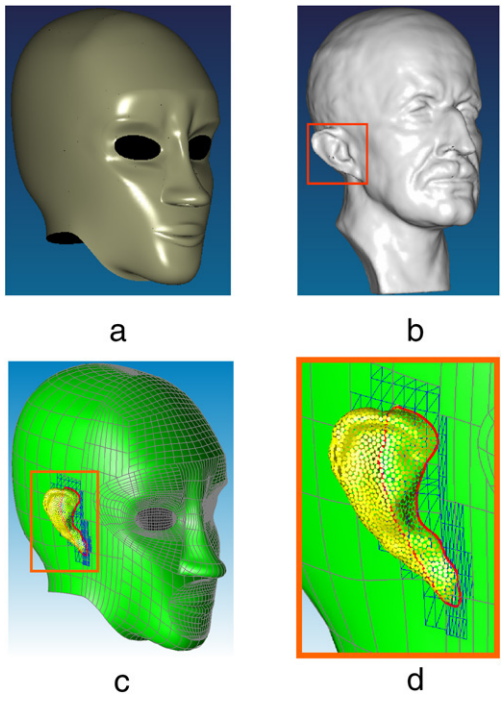


Fig. 14. Example 2: Intersection of a NURBS "head" and the point-set "Max Planck bust's ear". (a) Original NURBS surfaces of a head. (b) Original point-set surface of Max Planck bust. (c) Intersection result. (c) Zoom-in to the intersection region.

the previous design knowledge (the platform of the old turbine blade) and bypassing the tedious model reconstruction process (with our hybrid Boolean operation algorithm).

7.3. Custom headmask development (Example 4)

Let us recall the modeling and manufacturing of a customer-specific headform (shown in Fig. 1), which involves the proposed

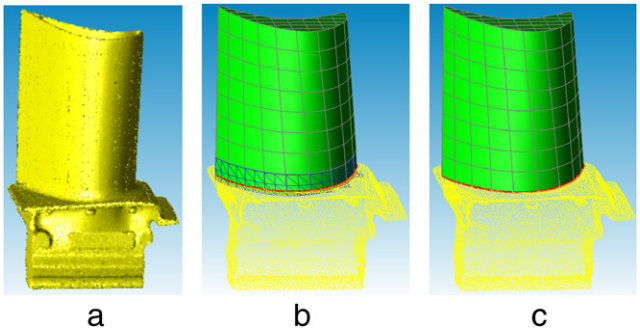


Fig. 15. Example 3: Boolean operation of a NURBS "airfoil" and a point-set "platform". (a) Iso-view of a whole scanned turbine blade. (b) Iso-view of the platform of the scanned turbine blade and a newly designed airfoil. (c) Iso-view of the Boolean result.

direct modeling approach. Row 1 of Table 1 represents two acquired geometry and their Boolean operation with the designed model. The designed model has undergone parametric modification from Model A (same as in Fig. 1) to Model B in a CAD system.

In current 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 proposed direct modeling approach based on point cloud, the above tedious model conversion processes are avoided and the overall modeling time is dramatically reduced to less than 20 s.

In the 4th example in this paper, the resulting hybrid model (shown in row 2 of Table 1) is directly converted into a sliced model for the layer-wise rapid prototyping in a Fused Deposition Modeling machine, as shown in row 3 of Table 1. The final custom-built parts from our direct shape modeling are shown in row 4 of Table 1.

Table 2 summarizes the performance of our algorithms on various examples where the last three columns give the computational times for NURBS surface subdivision T_{sub}, triangular mesh/MLS surface intersection T_{int} and intersection point mapping T_{map} correspondingly, which is based on a computer with dual Pentium IV 2.8 GHz processors and 1 GB RAM. From Table 2, there is a significant increase of the intersection time T_{int} and the mapping time T_{map} for the last two examples, which is due to a significant larger number of input points (see Table 2) and intersection points (around 1000 points, which are twice as many as that of other examples).

The above examples demonstrate that direct Boolean intersection can be a potent tool for bypassing manual surface reconstruction and improving the efficiency for the shape modeling applications where point-cloud data is involved. Even though the current implementation bypasses the significant human interaction time required in CAD surface reconstruction, direct Boolean intersection still cannot take place in real-time yet. Further work will improve its time efficiency by developing, e.g. multi-core processor-based algorithms.

8. Discussion

In this section, we discuss the effect of various parameters involved in the modeling process and recommend a set of procedure for selecting these parameters. We illustrate the sensitivity of these parameters with an example.

8.1. Gaussian factor h

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

Table 1
Custom headform development.

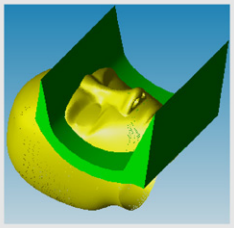
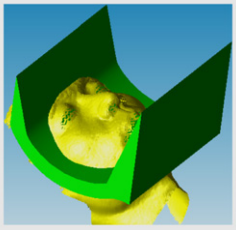
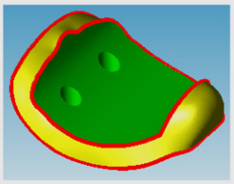
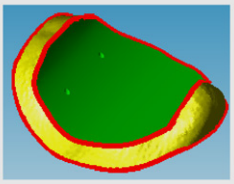
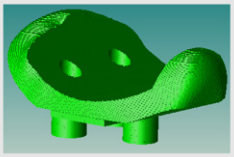
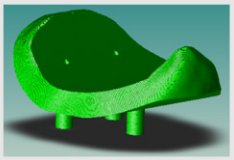
	Headform A	Headform B
1. Original designed and acquired geometries		
2. Boolean operation result		
3. Sliced rapid prototyping model		
4. Rapid prototyped part		

Table 2
Algorithm performances on various examples.

Surface A		Point-set surface B		Times (s)		
Name	Type	Name	Number of points	NURBS subdivision	Intersection/Boolean	Mapping
Ex 1	Triangular mesh	Dragon	39,829	N/A	6.302	N/A
Ex 2	NURBS	Ear	2,935	0.469	2.734	2.123
Ex 3	NURBS	Platform	16,152	0.376	4.913	0.598
Ex 4A	NURBS	Head A	79,284	3.586	15.293	11.541
Ex 4B	NURBS	Head B	157,444	3.612	16.415	12.957

accuracy of the MLS surface. In the above experiments, for the sake of simplicity, we have chosen a constant $h = \eta_{ave}$ for each example, where η_{ave} denotes average local sample spacing, i.e. average distance between sample points and their nearest neighbors.

However such a constant h may not be optimal for non-uniformly sampled surfaces. One scheme to solve this problem is setting the Gaussian factor h such that it is adapted to the local sampling density and the local curvature. The method of computing local sampling density can be found in [1]; while the analytical curvature formulas for MLS surfaces are provided by [8].

8.2. Bounding parallelepiped offset ε_0

Bounding offset ε_0 is another important parameter, which is used in line/MLS surface intersection, plane/MLS surface intersection and adaptive subdivision of NURBS surfaces.

When applying a too large ε_0, the algorithm efficiency will be sacrificed; when applying a too small ε_0, an incomplete intersection may occur and hence may lead to false intersection operation. If the input point set is a *uniform ε-sampling* point set, which means that the distance from any point on the MLS surface to its closest sample point is less than a given value ε, we can set $\varepsilon_0 = \varepsilon$. If not, we can set ε_0 equal to the average point distance.

8.3. Parameter for adaptive triangulation: δ_0

Triangulation error bound δ_0 defines an upper error bound of d_s, which represents the smallest distance between all pairs of parallel planar faces of the bounding parallelepiped of a leaf B-spline patch. Since this bounding parallelepiped contains both the leaf B-spline patch S_b and the resulting triangle pair $\hat{S}_b$, we have

$$d\left(S_b - \hat{S}_b\right) \leq d_s \leq \delta_0$$

where $d\left(S_b - \hat{S}_b\right)$ denotes the Hausdorff distance between the sets of S_b and $\hat{S}_b$.

Since this relationship holds for any leaf B-spline patch and corresponding triangle pair, we claim that the maximum deviation between the final triangle mesh and the underlying NURBS surface is bounded by δ_0. One way to specify δ_0 is to set it as a percentage of the average point distance of the input point cloud.

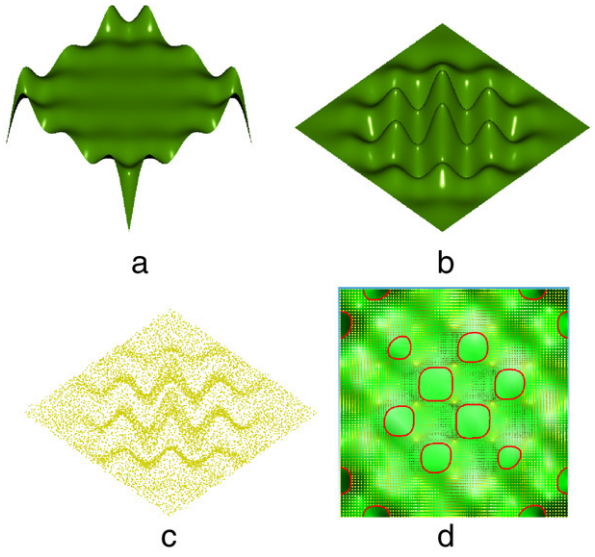


Fig. 16. Intersection of NURBS surface A and Point-set surface C. (a) NURBS surface A. (b) Nominal NURBS surface B. (c) Point-set surface C generated from surface B by sampling 100×100 points and by adding random noise with a standard deviation of 0.1. (d) Intersection result.

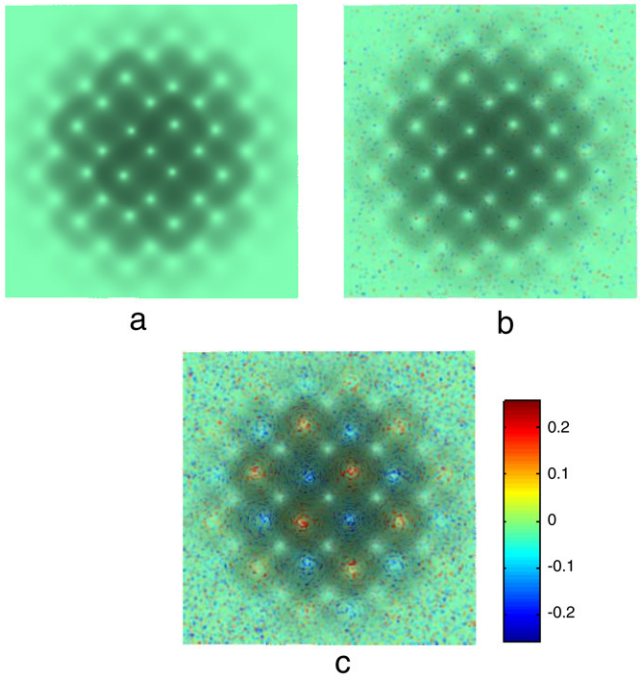


Fig. 17. Error distributions between nominal NURBS surface B and MLS surfaces with different Gaussian factor h. (a) $h = 0.03$. (b) $h = 0.3$. (c) $h = 3.0$.

8.4. Parameters for curvature-adaptive marching algorithm: δ_s, $r_{\min}$ and $r_{\max}$

In our curvature-adaptive marching algorithm for plane/MLS surface intersection, the result is a set of piecewise linear curves, which are linear approximations of the true intersection curves. To control the maximum approximation error, we adopt an error bound δ_s, which in conjunction with an osculating radius r is used to calculate an error-bounded step length Δp by Eq. (10). From Eq. (10), we find that the maximum approximation error is limited by δ_s under the assumption that the intersection curve can be locally represented by the osculating circle. One way to specify δ_s is to set it as a percentage of the average point distance of the input point cloud.

From Eq. (10), we also see that an extremely large or small osculating radius r may result in an extremely large or small step length. If a step length is too big the intersection approximations may be wrong due to overly un-even distribution of intersection points; if it is too small, the efficiency of the algorithm decreases.

Hence, we adopt a minimum osculating radius $r_{\min}$ and a maximum osculating radius $r_{\max}$ to limit the osculating radius r and avoid creating overly small or too large step lengths. In this paper, we choose:

$$\begin{cases} r_{\min} = \delta_s \\ r_{\max} = c_2 \cdot \delta_s \end{cases}$$

where c_2 is a constant value and equals 10.

8.5. An example for parameter selection

In the previous subsections, we have discussed the selection of several parameters for our direct surface intersection algorithm. Now, we will illustrate how these parameters will influence the intersection result of NURBS surface A (18.8 in. × 18.8 in. × 10.0 in.) and point-set surface C (18.8 in. × 18.8 in. × 10.4 in.), as shown in Fig. 16. Note, the point-set surface C is generated from nominal NURBS surface B (Fig. 16(b)) by sampling 100 × 100 points and by adding random noise with a standard deviation of 0.1, as shown in Fig. 16(c).

Based on the above recommended procedure for setting the parameters, we set $\delta_s = 0.003$ and h, δ_0 and ε_0 equal to the average local sample spacing $\eta_{\text{ave}} = 0.3$. Fig. 16(d) illustrates the intersection result which correctly includes 16 intersection curves (red): eight open branches and eight inner loops.

By adopting different values of h, we obtain a set of different MLS surfaces defined from the same point cloud. The deviations between the sampled points and these MLS surfaces are illustrated in Fig. 17. Fig. 17(a) shows that when h is too small (=0.03), the resulting MLS surface tends to interpolate the input noise data and yields to unpleasantly rough features. Fig. 17(c) shows that a large h (=3.0) may cause excessive smoothing in regions with small features and there is bias in the resulting MLS surface.

Keeping h and δ_s unchanged and adopting different values of ε_0 and δ_0, we get a set of different intersection results, as shown in Fig. 18. Fig. 18(a) shows that a mis-classification of the leaf NURBS patches and an incomplete intersection may occur, due to an extremely small ε_0 (=0.0).

Fig. 18(b), (c) and (d) illustrate three different cases of intersection results by choosing different δ_0's, which are further compared in Table 3: (1) large δ_0 (=1.2) results in a large difference between the original NURBS surface and the generated triangular mesh, which leads to missing intersection curves (such as the open branches shown in Fig. 18(c)), as shown in Fig. 18(b); (2) a smaller δ_0 (=0.6) guarantees the correctness of the intersection topology, however, the accuracy of the intersection curves is poor. However, it is improved by the intersection point mapping process, also shown in Fig. 20; (3) an even smaller δ_0 (=0.03) guarantees both the correctness and accuracy of the intersection curves.

Then keeping h, δ_0 and ε_0 unchanged and adopting different values of δ_s, we obtain a set of different intersection results, as shown in Fig. 19. Fig. 19(a) shows that when δ_s is too large (= 0.03), the output points of the marching algorithm may be too sparse (Fig. 19(a) and (c)), which means a low accuracy of the resulting intersection contours.

We finish this section with an example of mapping a set of intersection points generated with the parameters given in Fig. 19(b). In Fig. 20(a), the blue curves represent the initial u–v pairs, which are generated by STEP 1 (direct mapping the intersection points onto the NURBS surface A) in NURBS/MLS

Table 3
The effect of δ_0 on intersection results.

Parameter δ_0		1.2	0.6	0.03
Topology	Number of intersection curves	8	16	16
	Correctness	×	√	√
Geometry	Average[a] deviation to MLS surface	0.1637	0.0746/1.0752e−5[b]	2.3924e−4
	Accuracy	×	×/√[b]	√
Times (s)		3.685	12.912	23.565

[a] Evaluated after direct mapping the intersection points onto the NURBS surface A.
[b] Results before/after refinement respective.

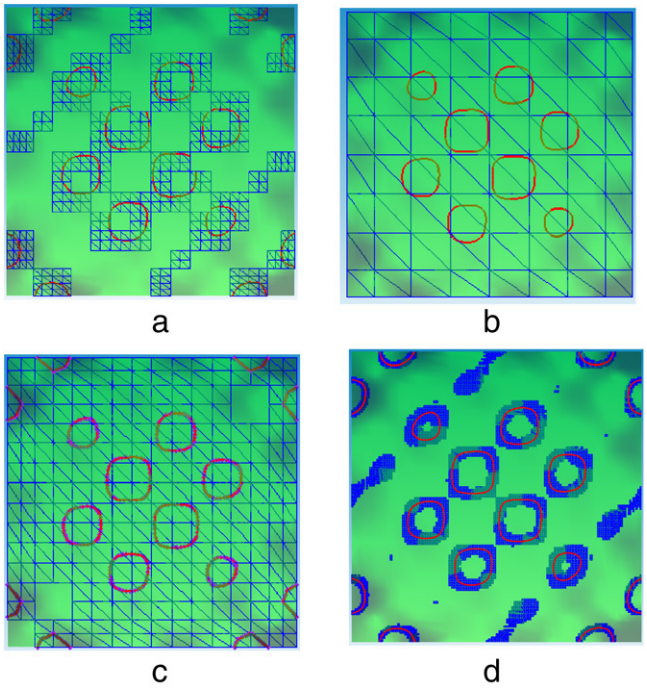


Fig. 18. Intersection results with different δ_0 and ε_0. (a) $\delta_0 = 0.3, \varepsilon_0 = 0.0$ (broken inner intersection curves). (b) $\delta_0 = 1.2, \varepsilon_0 = 0.3$ (missing boundary intersection curves). (c) $\delta_0 = 0.6, \varepsilon_0 = 0.3$ (correct topology and poor accuracy). (d) $\delta_0 = 0.03, \varepsilon_0 = 0.3$ (correct topology and good accuracy).

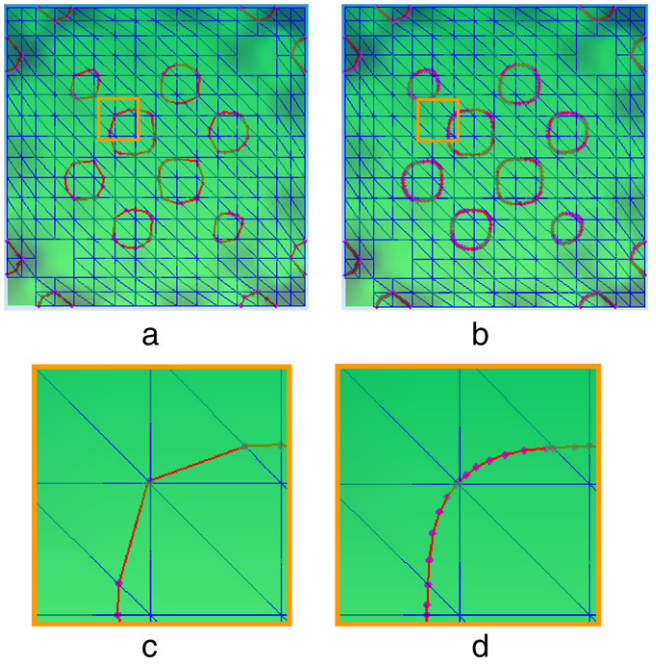


Fig. 19. Intersection results with different δ_s ($\delta_0 = 0.6, \varepsilon_0 = 0.3$). (a) $\delta_s = 0.03$. (b) $\delta_s = 0.003$. (c) Zoom-in of (a). (d) Zoom-in of (b).

surface intersection; the red curves represent the refined u–v pairs based on STEP 2; the background color map represent the values of Eq. (11) in the parametrical domain (the white area indicate that the absolute function value is larger than 6.0).

Before the mapping, the average absolute value of Eq. (11) is 0.1315 and the average distance to the MLS surface is 0.0746 (evaluated at the initial u–v pairs); after the refinement, the average absolute value of Eq. (11) is 6.4536e−8 and the average distance to the MLS surface is 1.0752e−5 (evaluated at the refined u–v pairs). From these results we find that: (1) the mapping process significantly reduces the intersection error; (2) the initial intersection error (0.0746) is much smaller than the triangulation error bound δ_0 (0.6).

In this example, it takes 6.829 s to refine a total number of 629 intersection points with a termination tolerance of 1e−7.

9. Conclusions

In this paper, a direct Boolean intersection approach based on designed geometry and acquired geometry has been presented. It combines the convenience and flexibility of existing CAD systems with the representation flexibility and fine shape details of acquired point-sampled geometry. At its core is a new approach that enables direct intersection and Boolean operations between objects bounded by NURBS and polygonal surfaces and objects bounded by point-cloud data without model conversion.

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 plane/MLS surface intersection. The adaptive subdivision of NURBS surfaces into a triangular mesh simplifies the NURBS/MLS surface intersection problem to a set of plane (triangle)/MLS surface intersection problem, which leads to an efficient and accurate solution of this problem. The resulting intersection points are further refined by a Gauss–Newton method, which makes the selection of triangulation error bound δ_0 and the marching error bound δ_s less critical.

Based on the above algorithms, a prototype system has been implemented. Through various examples from the system, we demonstrate that designed geometry and acquired geometry can be directly integrated for shape modeling without pre-filtering or post-processing on the point-cloud data. In these examples, direct Boolean intersection between designed geometry and acquired geometry proves to be a useful and effective means for point-cloud-data-based shape modeling applications.

In the future, we plan to improve the time efficiency of the proposed modeling approach so that direct Boolean intersection can take place in real-time.

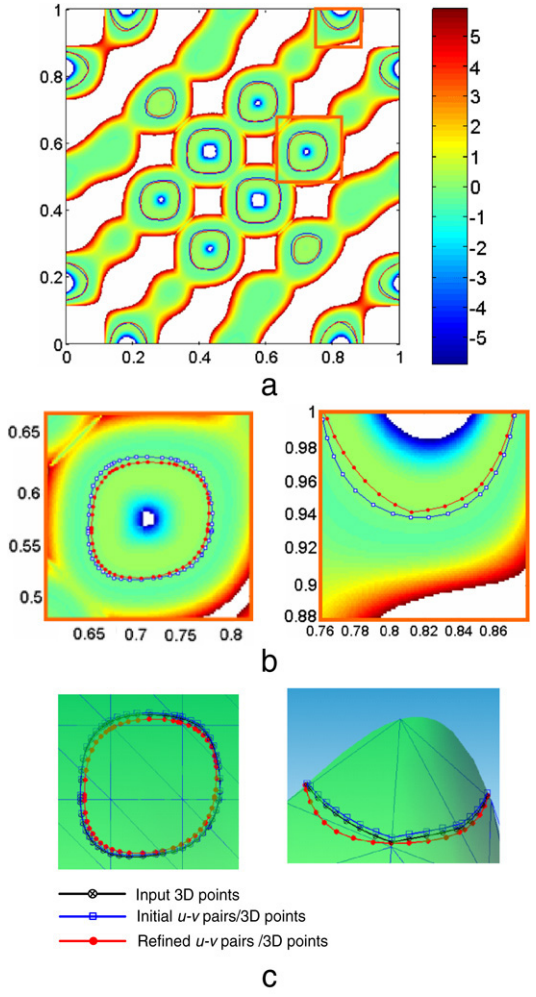


Fig. 20. An example for intersection point mapping with the NURBS surface A and the Point-set surface C shown in Fig. 16. (a) Initial u–v pairs (blue curves) and refined u–v pairs (red curves) with a background color map representing the values of Eq. (11) in parametrical domain. (b) Zoom-in of (a). (c) 3D points corresponding to u–v pairs in (b) and the input 3D intersection points between the triangular mesh and the MLS surface. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Acknowledgements

This work was supported by the US National Science Foundation Award #0529165 and Award #0800912, and Air Force Office of Scientific Research Award #FA9550-07-1-0241.

Appendix. Curvature form expansion

To further expand the gradient and Hessian in the curvature formula in Eq. (8), we first apply a new notation of $\mathbf{x} = \begin{pmatrix} x & y & 0 \end{pmatrix}^{\mathrm{T}}$. Then, taking the derivative of Eq. (4) with respect to $\mathbf{y}$ and setting $\mathbf{y}$ equal to $\mathbf{x}$, gives

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

Substituting into the transformed implicit function of Eq. (6), and notice that $(\hat{\mathbf{n}}(\mathbf{x}))^{\mathrm{T}} \cdot \hat{\mathbf{n}}(\mathbf{x}) = 1$, we have

$$\hat{g}\,(x,y) = \hat{\mathbf{n}}(\mathbf{x})^{\mathrm{T}} \left(\frac{\partial\, \hat{e}\,(\mathbf{y},\, \hat{\mathbf{n}}(\mathbf{x}))}{\partial \mathbf{y}}\Big| \mathbf{y}=\mathbf{x} \right) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2\mathrm{e}^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2} \times \left(1 - \frac{1}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^{\mathrm{T}} \hat{\mathbf{n}}(\mathbf{x})\right)^2\right) \cdot (\mathbf{x}-\mathbf{q}_i)^{\mathrm{T}} \hat{\mathbf{n}}(\mathbf{x}). \quad (12)$$

From Eq. (12), we can derive the formulas for $\nabla(\hat{g}\,(x,y))$ and $H(\hat{g}\,(x,y))$. The gradient of $\hat{g}\,(x,y)$ can be expressed as

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

The Hessian of $\hat{g}\,(x,y)$ can be expressed as

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

where $\mathbf{I}$ is the identity matrix.

References

[1] Pauly M, Keriser R, Kobbelt L, Gross M. Shape modeling with point-sampled geometry. ACM Transactions on Graphics 2003;22(3):641–50.
[2] Kobbelt L, Botsch M. A survey of point-based techniques in computer graphics. Computers & Graphics 2004;28(6):801–14.
[3] Levin D. The approximation power of moving least-squares. Mathematics of Computation 1998;67:1517–31.
[4] Amenta N, Kil YJ. Defining point-set surfaces. ACM Transactions on Graphics 2004;23(3):264–70.
[5] Alexa M, Behr J, Cohen-Or D, Fleishman S, Levin D, Silva CT. Computing and rendering point set surfaces. IEEE TVCG 2003;9(1):3–15.
[6] Zwicker M, Pauly M, Knoll M, Gross M. Pointshop3d: An interactive system for point-based surface editing. ACM Transactions on Graphics 2002;21(3):322–9.

[7] Adams B, Dutre P. Interactive Boolean operations on surfel-bounded solids. ACM Transactions on Graphics 2003;22(3):651–6.
[8] Yang P, Qian X. Direct computing of surface curvatures for point-set surfaces. In: Proceedings of the IEEE/eurographics symposium on point-based graphics(PBG). 2007.
[9] Houghton EG, Emnett RF, Factor JD, Sabharwal CL. Implementation of a divide-and-conquer method for intersection of parametric surfaces. CAGD 1985;2: 173–83.
[10] Jean BA, Hamann B. An efficient surface–surface intersection algorithm using adaptive surface triangulations and space partitioning trees. Mathematical Engineering in Industry 1998;7(1):25–40.
[11] Abdel-Malek K, Yeh HJ. Determining intersection curves between surfaces of two solids. Computer-Aided Design 1996;28(6–7):539–49.
[12] Barnhill RE, Kersey SN. A marching method for parametric surface/surface intersection. CAGD 1990;7:257–80.
[13] Amenta N, Kil YJ. The domain of a point set surface. In: Eurographics workshop on point-based graphics. 2004. p. 139–47.
[14] Dey TK, Sun J. Adaptive MLS surfaces for reconstruction with guarantees. In: Proceedings of eurographics symposium on geometry processing. 2005. p. 43–52.
[15] Levin D. Mesh-independent surface interpolation. In: Brunnett G, Hamann B, Muller H, Linsen L, editors. Geometric modelling for scientific visualization. Springer-Verlag; 2003. p. 37–49.
[16] de Berg M, van Kreveld M, Overmars M, Schwarzkopf O. Computational geometry: Algorithms and applications. Berlin: Springer-Verlag; 1997.
[17] Goldman R. Curvature formulas for implicit curves and surfaces. Computer Aided Geometric Design 2005;22((7):632–58.
[18] Adamson A, Alexa M. Ray tracing point set surface. In: Shape modeling international, vol. 299. 2003. p. 272–82.
[19] Press W, Flannery B, Teukolsky S, Vetterling W. Numerical recipes in C. 2nd ed. Cambridge University Press; 1992.
[20] Huber E, Barth W. Surface-to-surface intersection with complete and guaranteed results. In: Csendes T, editor. Developments in reliable computing. Kluwer; 1999. p. 185–98.
[21] Piegl L, Tiller W. The NURBS Book. 2nd ed. Springer-Verlag; 1997.
[22] Coleman TF, Li Y. An interior trust region approach for nonlinear minimization subject to bounds. Technical report. Ithaca (NY,USA) 1993.
[23] Tilove RB. Set membership classification: A unified approach to geometric intersection problems. IEEE Transactions on Computers 1980;C-29(10): 874–83.