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

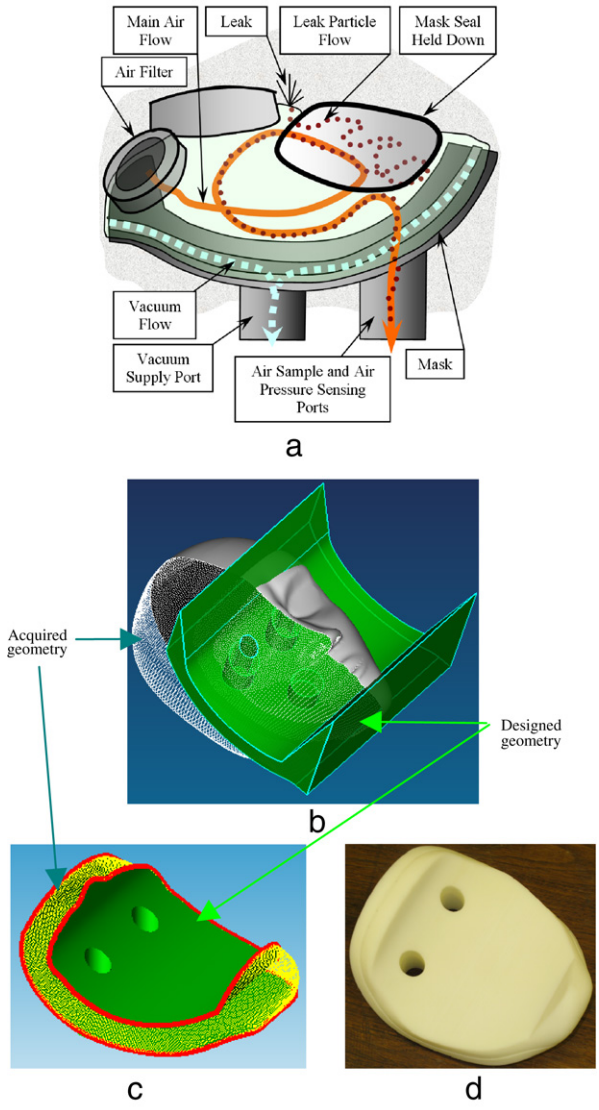

Fig. 1. Designing and manufacturing of a customer-specific headform: (a) Physical mask, (b) Acquired head model and the designed mask template. (c) Direct Boolean intersection result. (d) Rapid prototyped part from the intersection resulting model.

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.

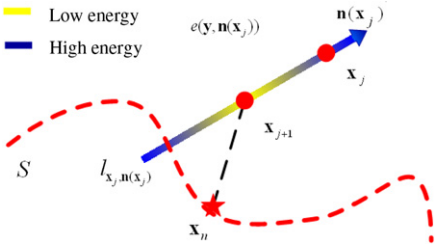

Fig. 2. Illustration of the MLS projection process.

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