Fast Intersection-free Offset Surface Generation from Freeform Models with Triangular Meshes

Shengjun Liu and Charlie C.L. Wang, *Member, IEEE*

***Abstract*—A fast offset surface generation approach is presented in this paper to construct intersection-free offset surfaces, which preserve sharp features, from freeform triangular mesh surfaces. The basic spirit of our algorithm is to sample a narrow-band signed distance-field from the input model on a uniform grid and then employ a contouring algorithm to build the resultant offset mesh surface from the signed distance-field. Four filters are conducted to generate the narrow-band signed distance-field around the offset surface in a very efficient way by alleviating computation redundancies in the regions far from the offset surfaces. The resultant mesh surfaces are generated by a modified dual contouring algorithm which relies on accurate intersections between the grid edges and the isosurfaces. A hybrid method is developed to prevent the expensive bisection search in the configurations that the analytical solutions exist. Our modified intersection-free dual contouring algorithm is based on convex-concave analysis, which is more robust and efficient. The quality and performance of our approach are demonstrated with a number of experimental tests on various examples.**

***Note to Practitioners*—This research is motivated by the problem about how to generate intersection-free offset surfaces from a general freeform model bounded by triangular mesh surfaces. Current commercial 3D/2D Computer-Aided Design and Manufacturing (CAD/CAM) systems cannot support the offsetting operation on a general freeform 3D model. This paper presents a new approach which considers the problem in a different manner. Firstly, the offset surfaces are sampled into an implicit representation – a narrow-band signed distance-field. Secondly, the distance-field is contoured into an intersection-free mesh surface that gives the approximated offset surface. The approach can be integrated into commercial CAD/CAM systems to enrich their offsetting functions which are widely used in various CAD/CAM applications, such as filleting, rounding and hollowing of 3D models, tolerance and clearance analysis for assembly, coordinate measuring machines (CMM), tool path generation for 3D numerically controlled (NC) machining, and robot path planning.**

***Index Terms*—Offset surface generation; freeform surface; signed distance field; filtering; intersection-free**

Shengjun Liu is now with School of Mathematical Science and Computing Technology, Central South University, China. This work was completed when he was with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong.

Charlie C.L. Wang (Corresponding Author) is with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong (Tel: (852) 2609 8052; Fax: (852) 2603 6002; E-mail: cwang@mae.cuhk.edu.hk)

I. Introduction

OFFSET surface generation is an important operation in many CAD/CAM applications. An offset surface of a solid H is the set of points having the same offset distance r from the boundary ∂H of H. When considering the sign of distances, we can have a grown offset H_r^+ with the points on H_r^+ being outside of H and a shrunk offset surface H_r^- that has all its points inside the solid H.

Problem Definition: Given a solid model H with its boundary surface ∂H represented by a triangular mesh, the boundary mesh surface of H_r^+ (or H_r^-) is to be computed.

Although the offsetting operation is mathematically well defined [1], offsetting a solid model exactly has proven to be difficult. In general, for a polygonal mesh, the Minkowski sum of it with a sphere is decomposed into a set of spheres, cylinders, and prisms corresponding to vertices, edges, and faces of the mesh. A constructive solid geometry approach to offset surface computation is based on computing the union of all these elements. However, the computation for the union operations of these solids has been proven difficult because of its computational complexity and numerical instability [1-3]. In recent years, volumetric approaches (e.g., [4, 5]) and point-based algorithm [6] have been proposed to overcome these difficulties. These algorithms first generate volumetric grids and sampling points to approximate the offset model and then employ a distance-field or collision detection technique to calculate the resultant implicit surface or sample points. However, they usually suffer the problems of missing sharp features (e.g., [4, 6]) or long computing time (e.g., [4], [5]).

Our method for offset surface generation can also be classified as a volumetric approach. According to the offset surface's intrinsic property that every point on the offset surface with an offset value r has a minimal distance $|r|$ to the original mesh, we use a distance function to define the offset surface directly on triangles. Specifically, an offset surface of H with distance r can be implicitly defined by the following equation

$$f(p) = dis(p, \partial H) - r = 0, \quad (1)$$

where p is a point on the offset surface of the given model H in R^3, and $dis(\ldots)$ is a function returning the signed distance with '+' representing a point outside H and '–' for a point inside.

The offset value r can be either positive or negative. The signed distance-field is sampled on uniform grids in a narrow-band manner – i.e., accurate distances are only computed on the grid nodes near the offset surface. Efficient algorithm for constructing such a signed distance-field is developed by introducing four filters that can greatly alleviate the computational redundancy. As the mesh surface generated by the dual contouring (DC) algorithm [7] can automatically preserve sharp features, our distance-field is also converted into the final offset surface through a dual contouring step that needs to compute accurate intersections between the grid edges and the offset surface. A hybrid method is developed to prevent the expensive bisection search in the configurations that analytical solutions exist. To get a resultant mesh surface that has no self-intersection, we modify the intersection-free DC algorithm [8] by a convex/concave analysis, which is more robust and efficient.

In short, the major contributions of our approach include:

- To identify the grid cells which are intersected by the offset surface and create a narrow-band signed distance-field quickly, four filters are introduced in section IV. The swept sphere volume hierarchy (SSVH) filter is adopted to filter out most of the triangles on the input model from the distance computation. A bounding box filter, a signed distance filter, and an octree filter are employed to filter out most of the grid nodes which are far from the offset surface. These filters can remove around 97% – 99.9% of the unnecessary distance computation on grid nodes.
- To employ the strategy of DC to generate the resultant meshes for offset surfaces, the intersections between the grid edges and the implicit offset surface must be computed. A straightforward way to compute them is by the bisection search, which however is very slow. We introduce a hybrid method in section V that combines the analytical solution (whenever is applicable) and the bisection search to compute the intersections. To reconstruct sharp features on the resultant offset mesh surfaces, every intersection point must be equipped with a normal vector, which can be determined by the closest point search.
- Although self-intersections of the offset solid have been automatically eliminated due to the distance-field based representation, they may be re-generated on the resultant mesh surfaces during the contouring procedure – see the discussion and the first approach to address this problem in [8]. We present a modified DC method based on convex/concave analysis (in section VI), which is more robust and efficient.

II. Related Work

Offset operations can be considered as a special case of the Minkowski sum [1]. There are several approaches in literature for the evaluation of Minkowski sum on solid models. Varadhan et al. proposed a method in [4] to approximate the Minkowski sum of polyhedral models. This method avoids the union step and its computation is simple and effective for convex objects. For general 3D polyhedrons, a divided-and-conquer approach is employed where it first applies convex decompositions, then computes the pair-wise Minkowski sums between those convex pieces, and finally extracts the boundary from the union of all the pairwise Minkowski sums [9]. However, such decomposition is not easy for complex objects [10] and makes too many components. In order to avoid the last union step among a number of components, a distance-field based method is employed in [4], which however has difficulty to reconstruct sharp features. Recently, Zhang et al. [11] generate the resultant boundary surface from sweeping volume with a dual-contouring like approach to retain shape features. However, self-intersections may happen (as discussed in [8]). Moreover, the volumetric region of resultant solid is computed by a flooding algorithm in [11]. This will miss the inner voids that are very important for the shrunk offsetting operation. Pavic and Kobbelt [5] extract the boundary surfaces by a Marching Cubes (MC) like algorithm that does not generate self-intersection. However, sharp features cannot be automatically reconstructed and MC algorithm usually generates much more triangles than DC. A post-processing step is used to recover sharp features. This step however can lead to self-intersection on the resultant mesh surfaces. A point-based method is presented to robustly compute an approximate and accurate representation of the Minkowski sums boundary in [6]. Its resultant boundary is a point-based representation and will lose those small but important sharp features. These methods run much more slowly as they are not specially designed for offset surface generation.

Offset operations can be applied to curves, surfaces, or entire 3D models. In earlier work [1], the mathematical basis for offsetting of solids was described. The offset techniques for curves and surfaces have been extensively studied by Pham [2] and Maekawa [3]. For 3D solid models, it will be more complicated to generate offsets, which involve not only the geometrical issue of offsetting each individual surface in the model but also the topological issue of reconnecting these offset surfaces into a closed 3D model. Generally, offsets of 3D models are achieved by first offsetting all surfaces of the model and then trimming or extending these offset surfaces to reconstruct a closed 3D model [1, 12, 13, 14]. These earlier approaches first compute a superset of the offset surface by offsetting 1) vertices into spheres, 2) edges into cylinders, and 3) faces into parallel faces. Then, they trim that superset by subdividing its elements at their common intersections and deleting the pieces that are too close to the original solid. This is a very expensive computing process and the trimming at tangential contacted regions is numerically unstable. The computation in our approach is much more stable since we work on a volumetric representation that can handle the tangential contact robustly. Some surface-based approaches for generating offset surfaces simply shift the original vertices in the offsetting direction [15]. This is problematic as self-intersections may occur either locally in areas of high

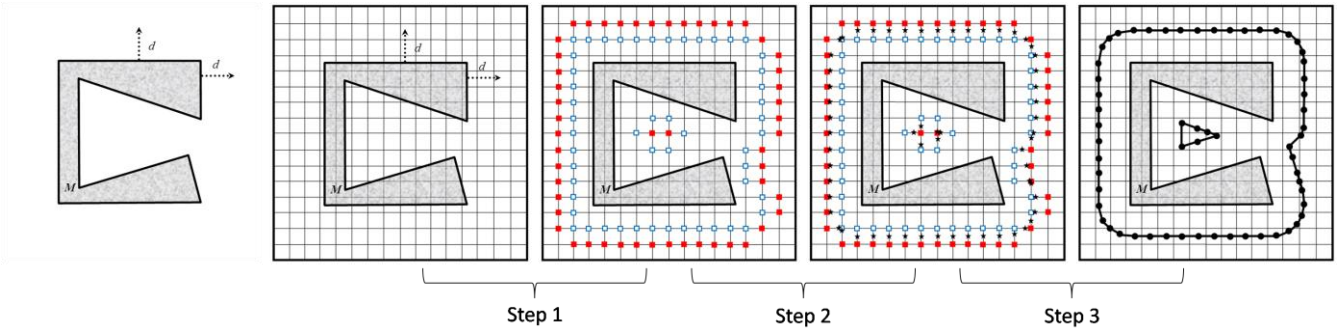


Fig. 1. The overview of our algorithm illustrated in 2D: (leftmost) the given model bounded by oriented two-manifold mesh surfaces, (left) the input model is embedded in a space that is sampled into uniform grids, (middle) classifying the grid nodes inside/outside the offset surface in a narrow-band manner – nodes that are *not* in the narrow-band region of the offset surface do not need to be considered, (right) intersections between the offset surface and the grid edges are computed, and (rightmost) the resultant offset mesh surface is extracted by an intersection-free dual contouring algorithm.

curvature or globally when different parts of the input surface meet during the vertex shifting. On the contrary, self-intersections are eliminated by our method since the volumetric representation and an intersection-free contouring method used in the surface extraction.

There are many other methods for offset surface generation with the help of different representations. Volumetric methods were presented based on distance-fields and the fast marching method in [16, 17]. The approximation properties of the fast marching [18] do not allow for high accuracy. Recently, a point-based offsetting approach was introduced in [19], where point samples are first generated on the input surface and are then moved in normal direction. After that, the Minkowski sum volume is rasterized on voxel grids in order to remove self-intersections. In [20], another offsetting approach was presented which aims mainly at visualizing the offset via surface splats. In their cases, the classification on whether a cell intersects the offset surface is based on conservative estimates for both the minimum and maximum distance. While this is sufficient for visualization purposes, it would be non-trivial to extract a proper manifold offset surface from it. On the contrary, our method gives intersection-free mesh surfaces as the results.

A hybrid method combining surface and volume was proposed in [5]. The algorithm is also based on computing the union of a set of primitives (spheres, cylinders and planes). Its computations are stable since the method works on a volumetric representation and the self-intersections can be easily removed by computing the min/max operations applied to distance functions. However, there are still some limitations of this method. The first one is that it generates two offset surfaces simultaneously without identifying the outer or inner ones. The second is the sharp feature extraction problem as aforementioned. Nevertheless, our method defines the offset surface in a narrow-band signed distance-field which results in an identical surface, and the sharp features are intrinsically preserved during the surface extraction. Another hybrid method which combines point and implicit surface was presented in [21]. This method reconstructs the zero-level surface and an offset surface simultaneously from an oriented point set. It first generates the offset point set, and then fits the surfaces with an implicit function combining two point sets. It is stable and robust for different point data, such as noisy points and irregular points. However, using this method, the sharp features cannot be reconstructed, and there are large artifacts on the offset surface where there are no offset points from the sharp features on the original model when the offset distance is large. On the contrary, our method preserves the sharp features and provides highly accurate results even if the offset distance is very large.

III. Overview

Given a solid model bounded by oriented two-manifold meshes, our goal is to obtain its intersection-free offset surface with sharp features preserved. The whole algorithm consists of three steps.

- Firstly, the given model H is embedded into a space Φ bounding the model and its offset surfaces. The space Φ is sampled into a narrow-band signed distance-field on uniform grids, where the narrow-band region is around the offset surface but not the surface of H. Four filters are developed to efficiently evaluate the distance-field (see section IV).
- Secondly, the intersections between the grid edges and the offset surface are computed. We analyze the possible configurations of the intersections, and derive a more efficient method to compute the intersections analytically whenever possible (see section V).
- Lastly, a modified intersection-free dual contouring algorithm is introduced to extract the mesh of offset surfaces (see section VI). Our method is based on convex/ concave analysis, which is more robust and efficient.

Figure 1 gives a 2D illustration of the overall algorithm. Our approach is simple and easy to implement. Moreover, as the narrow-band distance-field is sampled on uniform grids, the minimal distance evaluation and the intersection computation can be easily parallelized on PC with multi-core CPUs.

IV. Filters for Distance-Field Construction

To construct a signed distance-field of a given model H sampled on a uniform grid with $n \times n \times n$ grid nodes, the most

straightforward method is to compute the minimal distance from these n^3 nodes to the surface, ∂H, of H represented by triangular meshes. The minimal distance from a query point q to ∂H can be evaluated by an exhaustive search of all triangles on ∂H. After finding the closest point c_q of q on ∂H, the sign of distance between c_q and q is decided by the angle weighted pseudonormal on c_q [22]. Although the sign of distance can be determined very efficiently, the computation of closest point is time-consuming – especially when such a distance computation is conducted on all n^3 nodes with a large value of n (e.g., n = 513 as shown in our examples in this paper). Four filters are introduced in this section to speed up the evaluation.

A. Swept Sphere Volume Hierarchy (SSVH) Filter

The SSVH filter is applied to the triangles of ∂H to speed up the distance computation between a query point q and the triangles on ∂H. The basic idea is to establish a volume bounding hierarchy (BVH) of the triangles so that the computation on triangles that are far from the query point q can be discarded. The *Swept Sphere Volume Hierarchy* (SSVH) presented in [23] is adopted here as it can be easily modified to compute the minimal distance between a point and a set of polygons. Instead of computing the minimal distance between polygons, we employ a fast algorithm in [24] to compute the distance between the query point and a triangle, where the triangle is defined as

$$t(u,v) = b + ue_0 + ve_1$$

with the parametric domain $D = \{(u,v) : u,v \in [0,1], u+v \le 1\}$. The minimal distance is computed by finding the optimal parameters $(\bar{u},\bar{v}) \in D$ that lead to the closest point to the query point q. Besides the minimal distance from q to ∂H, we also record the closest point c_q, the triangle holding c_q, the parameters $(\bar{u},\bar{v})$ for c_q, and the sign of distance, which will be used in the following computations.

B. Bounding Box Filter

Although the evaluation of function $f(p)$ using the SSVH filter is fast, the construction of a signed distance field is still time consuming since there are n^3 nodes. Running on a PC equipped with Intel Core 2 CPU 6600 2.4GHz with 2GB RAM, the closest point computation using SSVH for a model with 400K triangles (i.e., the vase-lion model shown later in Fig.16) can be completed in 1.047ms on average. When computing the signed distances on 513 × 513 × 513 grid nodes, it takes 513 × 513 × 513 × 1.047ms ≈ 39.26 hr, which is impractical. A wiser solution is to construct a narrow-band distance field around the offset surface of H.

Without loss of generality, the offset surface is located in a narrow-band region of the grid cells.

Definition 1 For any grid node, if its position p satisfies

$$\left|dis(p,\partial H) - r\right| \le l \qquad (2)$$

with l being the width of grid boxes, this grid node is adjacent to the r-offset surface and defined as *valid grid nodes*.

For a grid node not agreeing with Eq.(2), it is called an *invalid grid node*. In the middle figure of Fig.1, only the valid nodes are displayed with small cubes. However, directly checking whether a grid node is valid also involves the distance computation between a point and the given triangular meshes. To avoid redundant evaluation, we introduce the bounding box filter below.

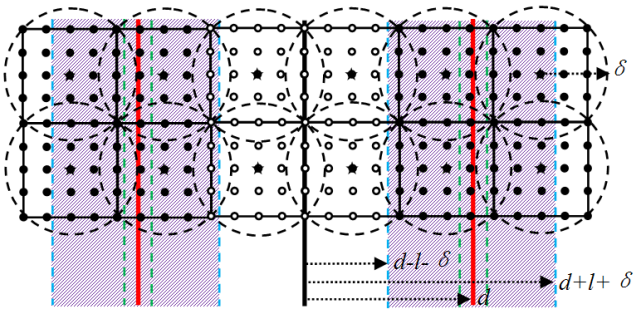


Fig. 2. Illustration of the bounding box filter in 2D with d = | r |. The bold black line represents the surface of the given mode, and the two bold red lines refer to the grown and the shrunk offset surfaces. The grid nodes in a bounding box, the center of circumsphere that does not fall in the purple regions, are invalid grid nodes – illustrated by empty circular dots. The black circular dots are candidates of valid grid nodes.

The grid cells are grouped into cubic bounding boxes with a larger size, where each bounding box consists of $m \times m \times m$ grid boxes (i.e., including $(m+1)^3$ grid nodes). The uniform grids become $\lceil (n-1)/m \rceil \times \lceil (n-1)/m \rceil \times \lceil (n-1)/m \rceil$ bins.

Remark 1 For a sphere with radius δ, if the minimal distance from its center s_c to the boundary surface of H satisfies

$$\left|dis(s_c,\partial H) - r\right| > l + \delta, \qquad (3)$$

all grid nodes in this sphere must be invalid.

The radius of the circumsphere for a bounding box is $\sqrt{3}ml/2$, therefore we can use the above remark to detect whether the grid nodes in a bounding box are invalid with $\delta = \sqrt{3}ml/2$.

Checking remark 1 directly needs to compute the signed distances on all bounding boxes. To speed up the computation, in the bounding box filter, we compute the unsigned distance, $undis(s_c, \partial H)$, between the center of the circumsphere of a bounding box and the surface ∂H. If the circumsphere of a bounding box agrees with

$$\left|undis(s_c,\partial H) - d\right| > l + \delta$$

where d = | r |, its corresponding signed distance must satisfy Eq.(3) – such a bounding box is named as *invalid bounding box*; otherwise, it is called *valid bounding box*.

Remark 2 All grid nodes in an invalid bounding box must be invalid, but the grid nodes in the valid bounding box could be either valid or invalid.

Figure 2 gives an illustration of the bounding box filter. Note that, the grid nodes around both the grown and the shrunk offset surfaces remain after applying this filter.

C. Signed Distance Filter

To retain only the bounding boxes around the grown offset

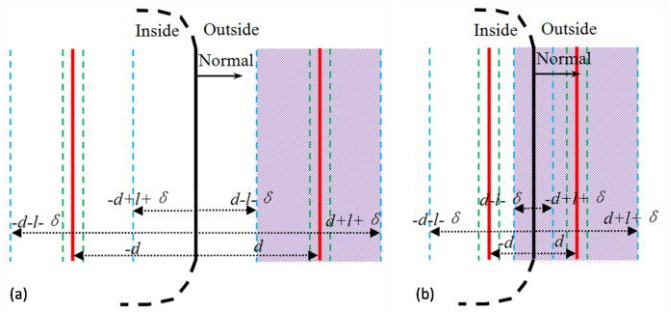


Fig. 3. Two configurations when applying the signed distance filter to grown offsetting – two band regions are (a) separated and (b) overlapped.

surface (or the shrunk surface), we check the signed distance $dis(s_c, \partial H)$ of the valid bounding boxes by a signed distance filter using the remarks below.

Remark 3a For the distance-field for a grown offset surface ($r > 0$), the grid nodes in a bounding box satisfying

$$dis(s_c, \partial H) \in \left[-d-l-\delta, \min(-d+l+\delta, d-l-\delta)\right] \quad (4)$$

are invalid, where $d = |r|$ and s_c is the center of the bounding box's circumsphere.

Remark 3b For the distance-field for a shrunk offset surface ($r < 0$), if a bounding box satisfies

$$dis(s_c, \partial H) \in \left[\max(d-l-\delta, -d+l+\delta), d+l+\delta\right], \quad (5)$$

all grid nodes in it are invalid.

Here, the *invalidity* of a grid node refers to the inability to satisfy Eq.(2) in Definition 2. The min(…) and max(…) functions in remark 3 are caused by the configuration that the two band regions overlap when the offset distance d is very small. Figure 3 shows two configurations when applying the signed distance filter for the grown offsetting.

D. Octree Filter

The bounding box filter and the signed distance filter introduced above can fast recognize many invalid grid nodes. However, when choosing a large bounding box size, many invalid grid nodes survive since they are enclosed by a valid bounding box (see Fig.4(a) for an example). To further exclude the invalid grid nodes from the narrow-band signed distance-field, an octree filter is introduced. Starting from a retained bounding box, we recursively subdivide the bounding box into eight sub-boxes. For each sub-box B, the signed distance dis_B from the center of its circumsphere to ∂H is computed. If the signed distance dis_B agrees with Eq.(3), the recursive subdivision on this sub-box is stopped, and all grid nodes in this sub-box are classified as invalid. Those grid nodes located on the interface of several sub-boxes are marked as invalid if the signed distance from any center of these sub-boxes' circumspheres agrees with remark 3. Figure 4(b) gives the illustration of octree filtering on a retained bounding box.

V. Surface Intersections on Grid Edges

After applying the above four filters, we can efficiently construct a narrow-band signed distance-field around the offset

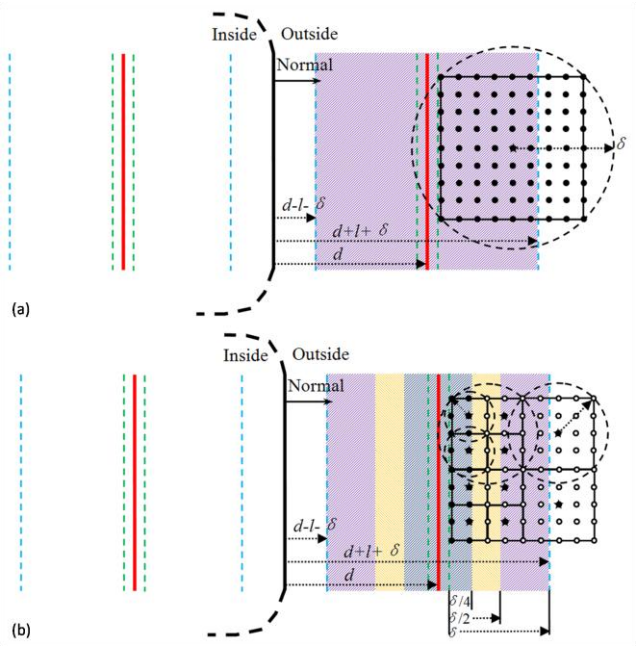


Fig. 4. Using an octree filter to further detect the validity of grid nodes in the retained bounding boxes: (a) the bounding box is valid but the enclosed grid nodes are invalid, and (b) invalid grid nodes are excluded recursively in the octree filter.

surface $f(p) = r$. The signed distances are only evaluated on the valid grid nodes. By the local signed distances, we can detect the grid cells intersecting with the offset surface.

Definition 2 For a grid edge e on the signed distance field with two *valid* grid nodes p_1 and p_2, if p_1 and p_2 satisfy

$$f(p_1) f(p_2) < 0 \quad (6)$$

with $f(p)$ being the function defined in Eq.(1), the grid edge is named as an *intersected edge*.

To extract the mesh surface from the implicitly defined offset surface, we apply a modified dual contouring (DC) algorithm that can automatically reconstruct sharp features and can be parallelized easily. The DC algorithm on uniform grids needs the intersection points between the intersected edge and the implicit surface, as well as the surface normal vectors at these intersection points. The intersection on an intersected edge e is the solution of α for the following equation

$$f((1-\alpha)p_1 + \alpha p_2) = 0, \quad (7)$$

which is nonlinear since the signed distance function $dis(\ldots)$ in $f(\ldots)$ is nonlinear (ref. Eq.(1)). A straightforward way to find the root of Eq.(7) is by the bisection search, which however is slow.

In this section, we introduce a hybrid method which combines the analytical solution (whenever is applicable) with the bisection search to compute the intersections.

A. Feasibility of Analytical Solution

Definition 3 For a query point q, $cp(q)$ denotes the closest point to q on the surface ∂H of the given model H.

The analytical solution is obtained when $cp(q)$ ($\forall q \in e$)

belongs to the same triangle T on ∂H. This configuration happens frequently when the grid edges are short enough.

Remark 4 For an intersected edge e with two ends p_1 and p_2, $cp(p_1)$ and $cp(p_2)$, being in the same triangle T is the necessary condition for $cp((1-\alpha)p_1+\alpha p_2)\in T, \forall \alpha \in [0,1]$.

For an intersected edge which does not satisfy the necessary condition given in remark 4, the bisection search method is used to find the solution of Eq.(7). Note that, to detect whether $cp(p_1)$ and $cp(p_2)$ are in the same triangle, the configurations when any of them is on the boundary of a triangle must be further evaluated. This is because $cp(p_1)$ and $cp(p_2)$ reported by the closest point query may not belong to the same triangle, but could be any of the configurations below.

- $cp(p_1)$ is inside the triangle t_1, and $cp(p_2)$ is on an edge e of triangle t_2 adjacent to t_1 and e is also an edge of t_1.
- $cp(p_1)$ is inside the triangle t_1, and $cp(p_2)$ is on a vertex v of triangle t_2 and t_1 is one of the triangles incident to v.
- $cp(p_1)$ and $cp(p_2)$ are on two edges that are adjacent to a common triangle.
- $cp(p_1)$ is on the edge e of a triangle t_1 and $cp(p_2)$ is on the vertex v of a different triangle t_2, but there is a common triangle incident to e and v.
- $cp(p_1)$ and $cp(p_2)$ are on the vertices of different triangles, but these two vertices are incident to a common triangle.

Symmetric cases of above configurations must also be considered.

When the necessary condition is satisfied, we conduct the analytical solution described below to find the solution of Eq.(7) – α_0. After that, a verification step is performed by checking whether

$$\left|dis((1-\alpha_0)p_1+\alpha_0 p_2), \partial H) - r\right| \le \varepsilon \qquad (8)$$

is satisfied, where $\varepsilon=10^{-5}$ is the error tolerance. If the distance difference is *not* less than ε, the bisection search method is further applied to find the root of Eq.(7). Note that, the terminal criterion of the bisection search is also Eq.(8) and with the same value of ε. Moreover, during the subdivision of bisection search, if the searched line segment satisfies the necessary condition of using an analytical scheme (i.e., Remark 4), we will compute the analytical solution – this can further speed up the computation of intersection points.

After finding the intersection point p, the normal vector at p can be easily assigned to $(p\text{-}cp(p))/\|p\text{-}cp(p)\|$ for grown offsetting and $(cp(p)\text{-}p)/\|p\text{-}cp(p)\|$ for shrunk offsetting.

B. Analytical Solution

For an intersected edge e with two ends p_1 and p_2, if the closest points from all the points on e to ∂H are located in the same triangle T on ∂H and $f(p_1)f(p_2)<0$, we will find a point p on e which has $f(p)=0$.

After analysis, we find that when a query point q is located in different regions around a triangle T, its closest point $cp(q)$ on T should be inside the face, on the edges, or on the vertices respectively. In general, the space around the triangle T is classified into seven regions (see regions 0-6 in Fig. 5(a)), which are separated by the triangle's bounding planes.

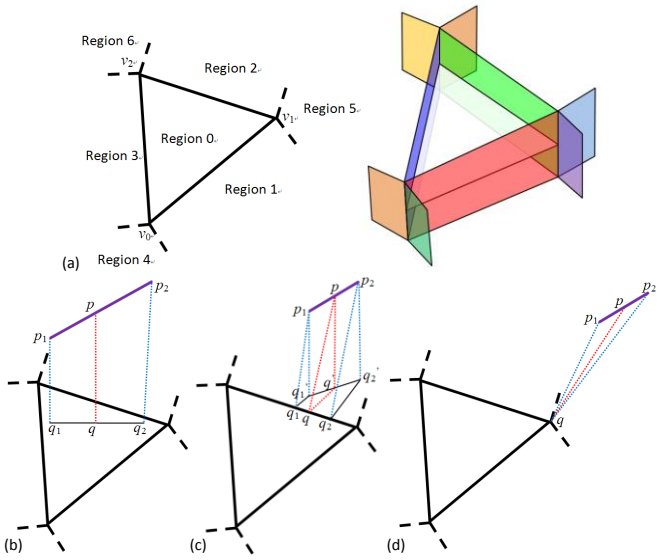


Fig. 5. Illustration of finding the analytical solution of intersection points on the intersected grid edge: (a) space around a triangle T is subdivided into seven regions by the bounding planes (represented by dash lines in the figures), (b) the configuration of a segment in a face region, (c) the segment is fully in an edge region, and (d) at a vertex region.

Definition 4 For a region around the triangle T, if the closest point of any point q in this region is
1) not on the boundary of T – the region is defined as a *face region* of T;
2) on the unique edge e_i (i=1,2,3) of T – the region is called an *edge region* of e_i on T;
3) on the unique vertex v_j (j=1,2,3) of T – this region is named as a *vertex region* of v_j on T.

As illustrated in Fig. 5(a), region 0 is a face region, regions 1-3 are edge regions, and regions 4-6 are vertex regions. Based on this classification, we develop the following method to find the intersection point (i.e., the solution of Eq.(7)) analytically.

Remark 5 For a line segment p_1p_2, if all points on the segment fall in the face region around the triangle T, the solution of $f((1-\alpha)p_1+\alpha p_2)=0$ is

$$\alpha = f(p_1)/(f(p_1)-f(p_2)). \qquad (9)$$

An illustration of remark 5 has been given in Fig.5(b). For the segment whose points all fall in an edge region, the solution of Eq.(7) is not obvious. As shown in Fig.5(c), suppose p is the intersection point on p_1p_2 that we are going to find, we first project p, p_1 and p_2 onto the plane of T to get q', q'_1 and q'_2. Then, q', q'_1 and q'_2 are projected onto the line segment of the edge that defines the edge region – q, q_1 and q_2 are obtained. Because of the two projections, we have the following ratio between the line segments held.

$$\alpha = \frac{pq'\cdot n_T - p_1q'_1\cdot n_T}{p_1q'_1\cdot n_T - p_2q'_2\cdot n_T} = \frac{\|qq'\|-\|q_1q'_1\|}{\|q_2q'_2\|-\|q_1q'_1\|} \qquad (9)$$

By this, we could have

$$pq'\cdot n_T = \alpha(p_1-p_2)\cdot n_T + p_1q'_1\cdot n_T, \qquad (10)$$

$$\|qq'\| = \alpha(\|q_2q_2'\| - \|q_1q_1'\|) + \|q_1q_1'\|. \quad (11)$$

Thus, the value of α can be determined by the function

$$(pq' \cdot n_T)^2 + \|qq'\|^2 = r^2. \quad (12)$$

Remark 6 For a line segment p_1p_2, if all points on the segment fall in an edge region of the triangle T, any α determined by Eq.(12) with $\alpha \in [0,1]$ is a valid solution of Eq.(7).

When all points of a segment fall in a vertex region, the spherical equation is used to obtain the root of Eq.(7) – see the illustration in Fig.5(d).

Remark 7 For a line segment p_1p_2, if all points of it are in the vertex region for a vertex q on T, any α determined by

$$\|q - (1-\alpha)p_1 - \alpha p_2\|^2 = r^2 \quad (13)$$

with $\alpha \in [0,1]$ is a valid solution.

Although the above analytical solution can fast evaluate accurate intersection point on an intersected edge e, the edge in practice usually passes through more than one region. In order to apply the above analytical solution to find the intersection point on e, we need to subdivide the segment of e into several segments where each is in a unique single face, edge or vertex region around T. For a sub-segment with two endpoints s_{p1} and s_{p2}, if it satisfies $f(s_{p1})f(s_{p2})>0$, we simply neglect it. For the sub-segments satisfying $f(s_{p1})f(s_{p2})<0$, the first root of Eq.(7) will be employed as the intersection point in the grid edge.

In our implementation, we compute the sub-segment in a faster way. By knowing the region where the first endpoint of a grid edge e falls in, we can sort the bounding planes of the triangle T in either clockwise or anti-clockwise order and then calculate the intersections between e and the boundary planes progressively. Therefore, the sub-segments are found one by one. Before going to the next sub-segment, we check whether this segment's two endpoints lead to the sign-flip on the function $f(\ldots)$. If they do, Remark 5-7 are employed to find the analytical root; and the rest sub-segments are not considered any more.

VI. Contouring: An Intersection-Free Method

The purpose of the work presented in this section is to investigate a more efficient approach than [8] for generating intersection-free iso-surface from the uniformly sampled grids using dual contouring (DC). Crack-free mesh surface can be generated by DC, and DC is able to reconstruct sharp features when Hermite data (i.e., intersection points equipped with surface normal vectors) are available. However, surfaces produced by dual contouring are rarely intersection-free (see Fig.2 in [8]). The only work existing in literature to address the problem of geometric intersection on the mesh surface generated by DC is [8], which combines the primal and dual contouring. Some necessary conditions have been derived in [8] to try to reduce the number of triangles. However, although the conditions in [8] are sufficient, the detection involves the detection of polygon-edge intersection which is neither as robust nor as efficient as our new approach based on convex/concave detection.

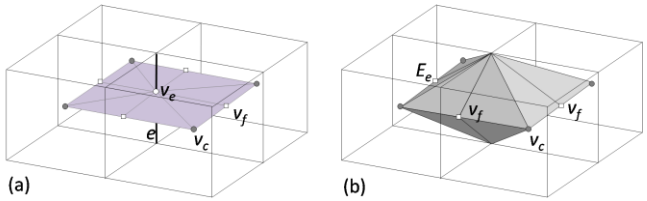


Fig. 6. (a) Triangle fans generated by the hybrid method at a grid edge e with sign change, and (b) the envelope E_e of triangles formed by the union of tetrahedra.

Basically, when applying the dual contouring algorithm to volumetric data sampled on uniform grids, the isosurface can be generated by linking vertices generated in boundary grid cells. A grid cell with its eight grid nodes having inconsistent *inside* (or *outside*) configurations is a boundary grid cell. In each boundary cell c, a vertex v_c on the resultant mesh surface is created and located at the position minimizing the *Quadratic Error Function* (QEF) defined by the Hermite data samples on the grid edges of c. For each cell edge e that contains a sign change – with one end *inside* but the other *outside*, two triangles will be constructed to link four vertices in the cells around it. However, the mesh surface constructed in this way is rarely intersection-free.

A hybrid approach of a primal and dual method is developed in [8]. As illustrated in Fig.6, a triangle fan with eight triangles is generated around a cell edge e with sign change in the hybrid approach, where each triangle connects the edge vertex v_e on e, the face vertex v_f on a cell face f containing e and the cell vertex v_c in a cell c sharing f. The authors of [8] then proved that the hybrid approach generates intersection-free triangles if each cell vertex, face vertex and edge vertex lies interior to the corresponding cell, face or edge. The envelope E_e at a grid edge e is defined as the union of eight tetrahedra, each of which is formed by the edge e, the face vertex v_f and the cell vertex v_c (see Fig.6). It has proven that the envelopes E_e of different edges e are disjoint, and the triangles generated by the hybrid method around e are contained in separate tetrahedron of the envelope E_e. Therefore, the triangulated surface is guaranteed to be intersection-free. To reduce the number of triangles generated by the hybrid method, three rules are defined in [8] for contouring an octree grid. The rules are based on the detection of polygon-edge intersections. However, when contouring uniform grids, a simpler and more robustness method based on convex/concave analysis can be developed as follows, which is one of the major contributions in this paper.

First of all, as long as all cell vertices are located in their corresponding cell boxes, the intersection between the cell face f_{pq} of two neighboring cells c_p and c_q (containing cell vertices v_p and v_q) and the line segment v_pv_q is always in the face f_{pq} when contouring uniform grids. Thus, the face vertices in the hybrid method of [8] are not needed. By removing face

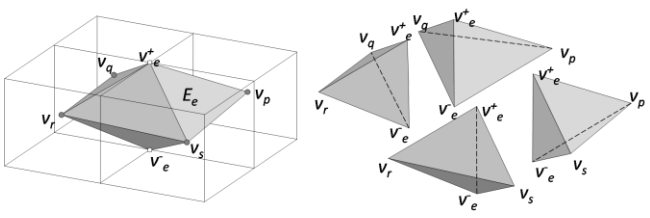


Fig. 7. Four tetrahedra form the intersection-free envelope E_e around the edge e.

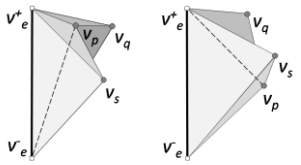


Fig. 8. Two cases that v_p becomes a concave envelope vertex: (left) the edge v_p-v^-_e is concave and (right) the edge v_p-v^+_e is concave.

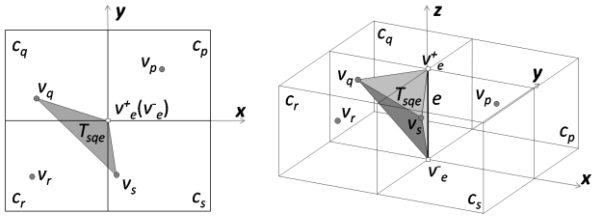


Fig. 9. The tetrahedron formed by two cell vertices in diagonal cells and the edge e can only intersect three of the four cells around e.

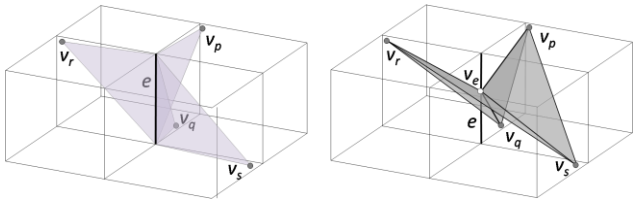


Fig. 10. An example with two concave envelope vertices: (left) location of vertices around edge e and (right) the triangulation by inserting an edge vertex v_e.

vertices, the envelope E_e around e is then formed by four tetrahedra.

Definition 5 An envelope E_e around an edge e that does not have volume overlap with other edge envelopes is formed by four tetrahedra: v_p-v_q-v^+_e-v^-_e, v_q-v_r-v^+_e-v^-_e, v_r-v_s-v^+_e-v^-_e and v_s-v_p-v^+_e-v^-_e, where v_p, v_q, v_r and v_s are cell vertices in cells c_p, c_q, c_r and c_s around e, and v^+_e and v^-_e are two vertices at the end of e.

Figure 7 gives an illustration of the four tetrahedra that form the intersection-free envelope E_e around the edge e. Note that the indices p, q, s and r are circularly used here.

Definition 6 For the cell vertex v_c forming the intersection-free envelope E_e around an edge e, if any of the edges v_c-v^+_e or v_c-v^-_e is concave, it is defined as a *concave envelope vertex* of E_e; otherwise, it is named as a *convex envelope vertex*.

Remark 8 The edge v_p-v^+_e is concave if v_p is below the plane defined by the oriented triangle v_s-v_q-v^+_e, and the edge v_p-v^-_e is concave if v_p is below the plane defined by the oriented triangle v_q-v_s-v^-_e.

Figure 8 gives an illustration of Remark 8.

Proposition 1 For two cell vertices v_s and v_q in two non-neighboring cells c_s and c_q, the tetrahedron T_{sqe} formed by the cell vertice v_s and v_q and the edge e can only intersect three of the four cells around e.

Proof can be found in [25].

Since all points in c_p will be above the oriented planes v_s-v_q-v^+_e and v_q-v_s-v^-_e when the tetrahedron T_{sqe} intersects c_r (see Fig.9), by Remark 8 and Proposition 1, we can conclude the following remark.

Remark 9 A cell vertex v_p in the cell c_p will *NOT* be a concave envelope edge if the tetrahedron T_{sqe} formed by its previous cell vertex v_s, its latter cell vertex v_q and the edge e does not intersect the volume of cell c_p.

Proposition 2 For four cell vertices v_p, v_q, v_r and v_s around an edge e, if no concave envelope vertex is found, all the four triangles: v_p-v_q-v_r, v_p-v_r-v_s, v_p-v_q-v_s and v_q-v_r-v_s are enclosed by E_e.

Proposition 3 For four cell vertices v_p, v_q, v_r and v_s around an edge e, if only v_p is a *concave envelope vertex*, triangles v_p-v_q-v_r and v_p-v_r-v_s will be enclosed by the envelope E_e, whereas some part of the triangles v_p-v_q-v_s and v_q-v_r-v_s will *NOT* be enclosed by E_e.

Proofs can be found in [25].

Remark 10 For four cell vertices v_p, v_q, v_r and v_s around an edge e, only one or two vertices can be *concave envelope vertices*, and these two vertices must be in the adjacent cells.

If v_p is a *concave envelope vertex*, the tetrahedron T_{sqe} formed by v_s, v_q and the edge e can only intersect c_p, c_s and c_q but not the cell c_r (by Proposition 1). Therefore, v_r cannot be a concave envelope vertex (by Remark 9). Also, if v_q is a *concave envelope vertex*, v_s cannot be a concave envelope vertex.

Remark 11 If two vertices are *concave envelope vertices*, none of the triangles v_p-v_q-v_r, v_p-v_r-v_s, v_p-v_q-v_s and v_q-v_r-v_s is enclosed by E_e.

If v_p is a concave envelope vertex (or v_r is a concave envelope vertex), the edge v_s-v_q is outside E_e; and the edge v_p-v_r is outside E_e when v_q (or v_s) is outside E_e. Figure 10 shows an example of that. If two vertices are concave envelope vertices, we have to further split them into four triangles by adding an edge vertex v_e on e to ensure that E_e encloses all the triangles.

Robust and Efficient Implementation: Our convex/concave analysis based approach is more robust and efficient comparing with the intersection-free approach presented in [8], which is based on the edge-polygon intersection tests. In general, 4 to 6 edge-triangle intersection tests are needed for constructing two triangles for an intersected cell edge.

TABLE I
COMPUTATIONAL STATISTICS (IN SECOND)

Model	Fig.	*F#*	Offset	T_s	T_d	T_c	T_{tol}
Carve-cube	11	7.1k	-0.04	0.06	13.2	15.7	28.96
			0.05		13.8	17.8	31.66
Anchor-plate	12	9k	-0.041	0.06	6.4	13	16.96
			0.013		7.2	13.9	21.16
			0.2		16.15	18.13	34.33
Octa-flower	13	15.8k	-0.05	0.15	4.95	13.45	18.55
			0.1		12.2	16.2	28.55
Dragon	14	49.4k	-0.02	0.45	8.16	13.65	22.26
			0.03		13.8	15.37	29.62
Vase-lion	15	400k	-0.02	4.2	15.8	15.5	35.5
			0.04		17	17.2	38.4
Rabbit	21	134k	-0.03	1.3	5.6	13.5	20.4

F#: number of triangles for the given model
T_s: time for construction SSVH
T_d: time for generating the narrow-band signed distance field
T_c: time for surface contouring
T_{tol}: the total time for offset surface generation

TABLE II
OFFSETTING TIME ON GRIDS WITH DIFFERENT RESOLUTION (IN SECOND)

Res.	65×65×65	129×129×129	257×257×257	513×513×513
F#: 40k	0.53	0.98	3.34	21.6
F#: 400k	4.6	5.3	8.4	28.7

F#: number of triangles on the chair model.

TABLE III
COMPUTING TIME FOR DIFFERENT OFFSET VALUES (IN SECOND)

Rocker-arm	*r*	0.01	0.02	0.04	0.08	0.16	0.32
	T_{tol}	19.8	20.7	22.4	26	31.7	50.7
Terracotta -warrior	*r*	-0.02	-0.01	0.01	0.02	0.04	0.08
	T_{tol}	18.6	19.2	19.7	21.2	23.1	26.6
Filigree	*r*	-0.005	0.005	0.01	0.02	0.04	0.08
	T_{tol}	26.8	29.5	30.6	33.5	37.4	49.2

r: offset value
T_{tol}: the total time for offset surface generation
*The results can be found in Figs.17-19.

According to the fastest algorithm of edge-triangle intersection test in literature (ref. [26]), each edge-triangle intersection test needs 3 cross-products and 3 (or 4) dot-products. Therefore, totally 18 cross-products and 24 dot-products are required in the worst case. In our convex/concave analysis, we only need to detect whether the four vertices are concave envelope vertices. Specifically, detecting whether a vertex v_p is below the oriented triangle v_s-v_q-v^+_e can be efficiently done by checking if the scalar triple product below satisfies

$$(v_p - v^+_e) \cdot ((v_s - v^+_e) \times (v_q - v^+_e)) < 0.$$

In the worst case, 8 cross-products and 8 dot-products are needed. When computing scalar triple product by using determination, the computation can be further reduced. Moreover, if v_p is found to be a concave envelope vertex, the detection of v_r can be neglected (by Remark 11). The detection on v_q and v_s can be simplified in the same way. Practical tests and comparisons with [8] can be found in the following section (section VII-B).

In the aspect of robustness, the detection based on edge-polygon intersection test is suffered from the extreme cases (e.g., intersection occurs on the edge or vertex of the triangle).

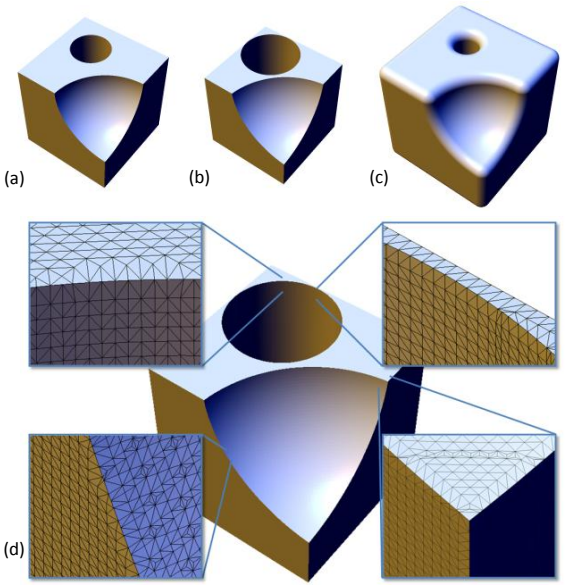


Fig. 11. Offset surfaces generated by our approach on a carved-cube model: (a) the given model, (b) the shrunk offset surface with distance -0.04, (c) the grown offset surface with distance 0.05, and (d) the zoom-in view of the shrunk offset surface – it is easy to find that sharp features are well reconstructed on the offset surface.

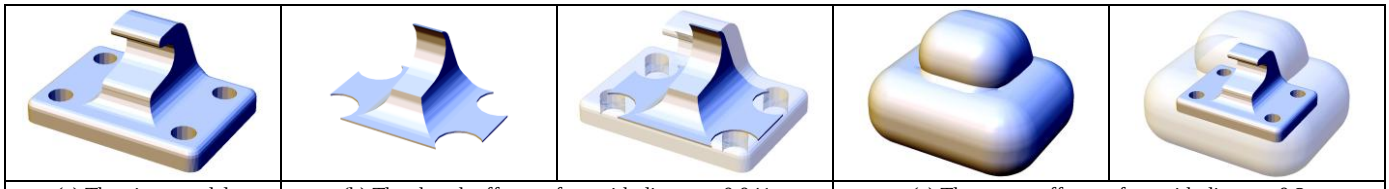
To ensure not missing any possibility of self-collision, some nearly intersected (but not really intersected) cases may be classified as intersection, which leads to more triangles on the resultant mesh surfaces by the unnecessary triangulation. Our detection that is based on convex/concave analysis does not have such a problem.

VII. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed approach has been implemented in Visual C++ plus OpenMP. OpenMP is employed to parallelize the computation on the PC with multi-core CPUs, which is quite common on consumer-level PCs.

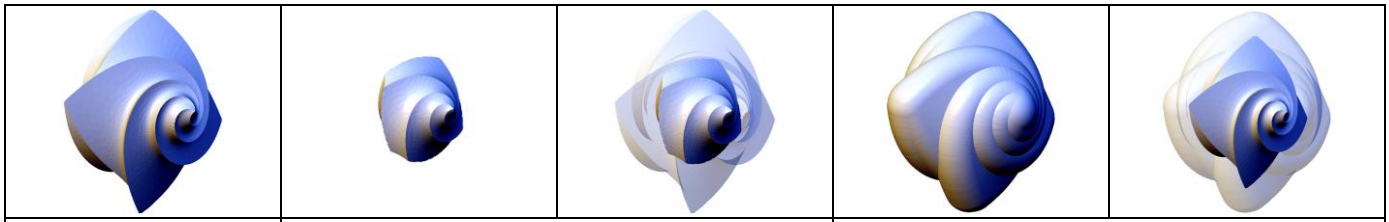
A. Experimental Tests

Our implementation has been tested on a variety of models to generate grown or shrunk offset surfaces in different offset values (as shown in Figs.11-19). The computational statistics in terms of time are shown in Table I. The input mesh models are with different number of triangles varying from 7.1k to 400k. Offset surfaces for these models can be generated in less than 40 seconds. The statistics in this paper are all tested on a PC with two Intel Xeon Quad E5440 CPUs at 2.83GHz + 8GB RAM. All the offset distances are described as relative values to the diagonal lengths of bounding boxes for the given models. Table II gives the statistics on computing time for generating the offset surfaces (r = 0.02) on a chair model (see Fig.17) with different numbers of triangles. From the statistical data, it is easy to find that our approach scales quite well with the increase of grid resolution and the number of triangles. The increase of computing time is approximately linear to the increase of resolution, which outperforms other volumetric approaches on uniform or adaptive grids. The computing time increases less than linear ratio compared with the increment of



(a) The given model (b) The shrunk offset surface with distance -0.041 (c) The grown offset surface with distance 0.2

Fig. 12. The anchor-plate model and its offset surfaces.



(a) The given model (b) The shrunk offset surface with distance -0.05 (c) The grown offset surface with distance 0.1

Fig. 13. The octa-flower model and its offset surfaces.

(a) The given model (b) The shrunk offset surface with distance -0.02 (c) The grown offset surface with distance 0.03

Fig. 14. The offset surfaces for the dragon model.

(a) The given model (b) The shrunk offset surface with distance -0.02 (c) The grown offset surface with distance 0.04

Fig. 15. The offset surfaces for the vase-lion model.

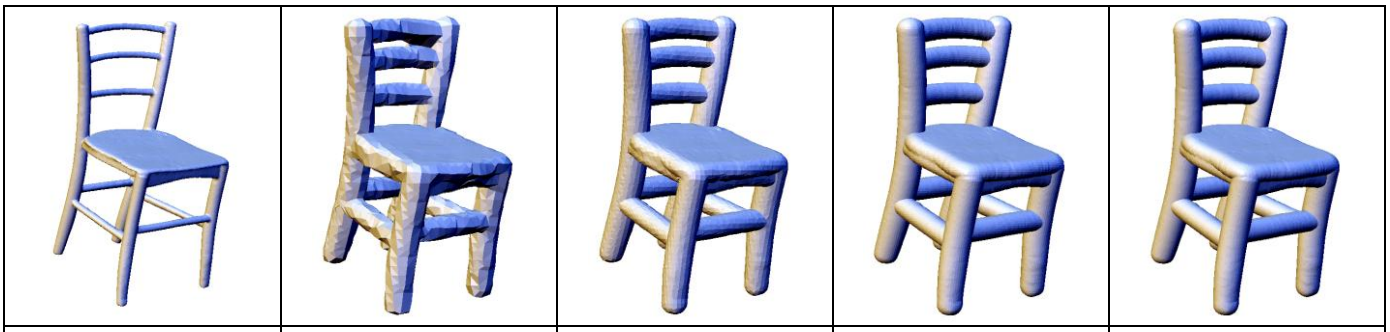
TABLE IV
NUMBER OF TIMES OF DISTANCE COMPUTATION BY APPLYING DIFFERENT FILTERS

Offset Dist.	BB				BB + SD				BB + OT	BB + SD + OT
	BB_8	BB_{16}	BB_{32}	BB_{64}	BB_8	BB_{16}	BB_{32}	BB_{64}	BB_{128}	BB_{128}
-0.04	5,767k	2,982k	2,023k	1,583k	3,670k	1,737k	1,110k	556k	585k (159k)	120k (41k)
-0.02	5,243k	2,785k	1,790k	1,322k	4,194k	2,228k	1,307k	819k	620k (152k)	240k (72k)
-0.01	5,243k	2,654k	1,638k	1,196k	4,981k	2,294k	1,397k	943k	625k (129k)	325k (89k)
0.01	5,505k	2,785k	1,818k	1,269k	5,505k	2,785k	1,818k	1,269k	642k (136k)	476k (122k)
0.02	6,029k	3,113k	2,073k	1,506k	6,029k	3,113	2,073k	1,506k	775k (169k)	534k (139k)
0.04	6,291k	3,899k	2,699k	2,017k	6,291k	3,899k	2,699k	2,018k	750k (203k)	630k (170k)
0.08	10,224k	5,833k	4,227k	2,668k	10,224k	5,833k	4,104k	2,668k	865k (240k)	858k (234k)
0.16	16,777k	11,239k	7,258k	4,248k	16,777k	11,239k	7,258k	4,248k	1,410k (389k)	1,410k (389k)
0.32	40,370k	26,116k	15,667k	8,847k	40,370k	26,116k	15,667k	8,847k	2,846k (797k)	2,846k (797k)

Values in the bracket are the number of times of distance computation for testing valid and invalid bounding boxes during filtering.

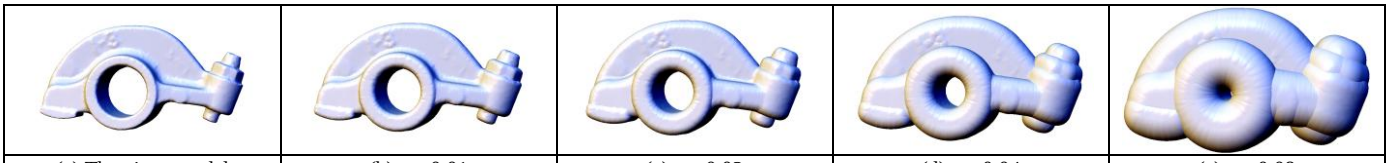
triangle numbers. Another study is the computing time of our approach on generations with different offset values (see Table III). Again, the increment of computing time is less than linear ratio compared with the increment of offset values. In short, our approach scales well on models with different triangles, different offset values, and different grid resolutions.

To study the performance of the filters proposed in this paper, we test the reduction in the number of closest point



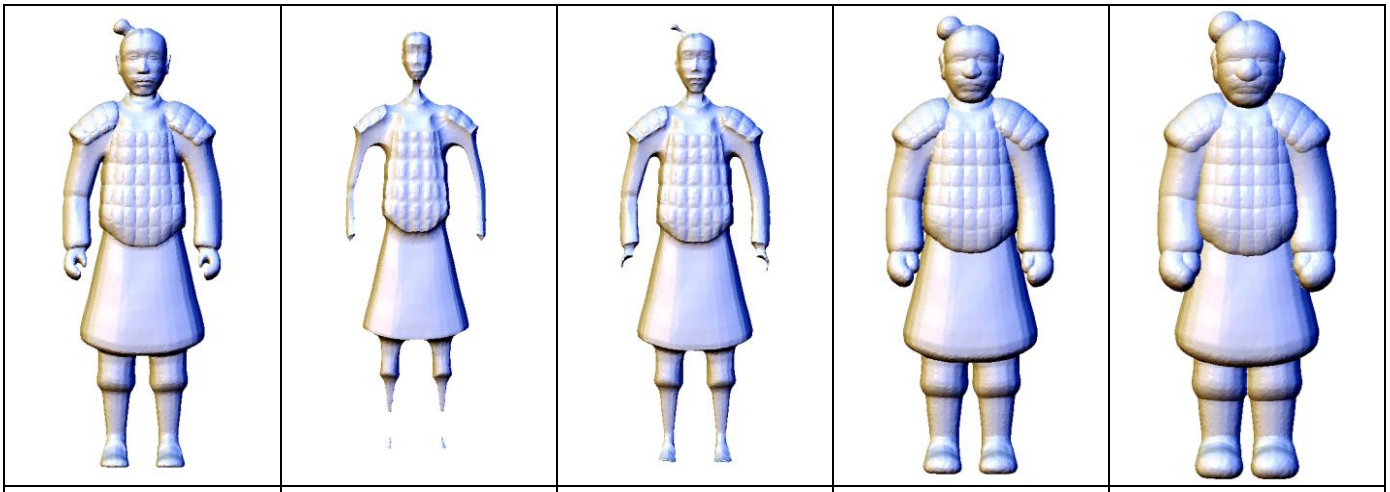
(a) The given model | (b) 64×64×64 | (c) 128×128×128 | (b) 256×256×256 | (b) 512×512×512

Fig. 16. The chair model with 40k triangles and its offset surfaces generated with different grid resolutions.



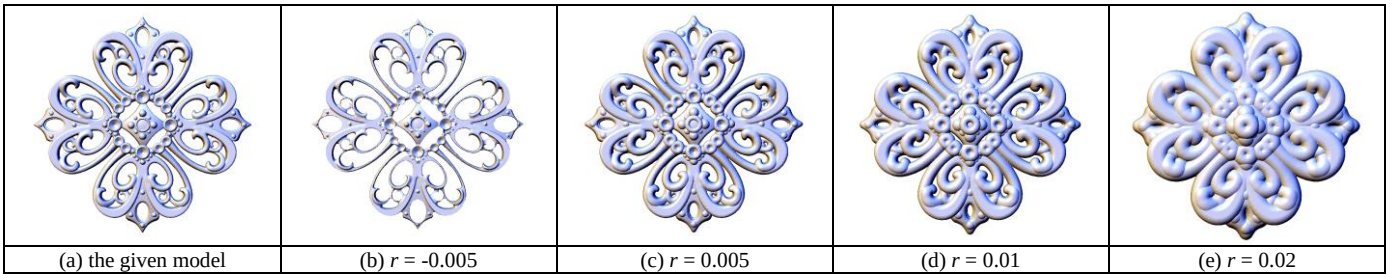
(a) The given model | (b) r = 0.01 | (c) r = 0.02 | (d) r = 0.04 | (e) r = 0.08

Fig. 17. The rocker-arm model with 20k triangles and its offset surfaces using different r.



(a) the given model | (b) r = -0.02 | (c) r = -0.01 | (d) r = 0.01 | (e) r = 0.02

Fig. 18. The terracotta warrior model with 200k triangles and its offset surfaces using different r.



(a) the given model | (b) r = -0.005 | (c) r = 0.005 | (d) r = 0.01 | (e) r = 0.02

Fig. 19. The filigree model with 260k triangles and its offset surfaces using different r.

query conducted in the construction of narrow-band signed distance field. The terracotta-warrior model with 60k triangles and the uniform grid with resolution 513×513×513 are employed in these tests. Four classes of cases are tested here: 1) bounding box filter BB_m with m denoting the number of grid cells in the bounding box in one direction, 2) bounding box filter plus the signed distance filter SD, 3) bounding box filter plus the octree filter OT, and 4) BB_m + SD + OT. Table IV lists the statistical data. It is not difficult to find that by applying BB_m + SD + OT together, around 97% – 99.9% of the unnecessary distance computations on grid nodes can be removed.

The efficiency of our hybrid model for intersection-point computation on intersected edges is also studied. We count the number of intersected edges that can have the intersection point been found on the terracotta-warrior model with different

TABLE V
STATISTICS OF HYBRID METHOD FOR INTERSECTION POINT ON EDGES

Tests on the Terracotta-warrior Model with Different Number of Triangles						
F#	*IntE#*	Hybrid				Bisect.
		$S_d = 0$	$S_d = 1$	$S_d = 2$	$S_d \le 6$	$S_d \le 6$
60k	85032	71.5%	15.4%	7%	99.5%	6.4%
200k	85032	51.3%	20.6%	13.3%	98.8%	6.4%
600k	85032	34.7%	20.3%	17.7%	97.6%	6.4%
Tested on Terracotta-warrior Model with 60k Trgl. with Diff. Off. Dist.						
Offset Dist.	*IntE#*	Hybrid				Bisect.
		$S_d = 0$	$S_d = 1$	$S_d = 2$	$S_d \le 6$	$S_d \le 6$
-0.04	16646	64.8%	15.2%	9.5%	99.2%	6.9%
-0.02	34952	52%	19.8%	13.4%	98.8%	6.6%
-0.01	49208	44.3%	22.7%	15.8%	98.7%	6.9%
0.01	77356	59.1%	21.5%	10.7%	99.4%	6.6%
0.02	85032	71.5%	15.4%	7%	99.5%	6.4%
0.04	101458	82.1%	9.5%	4.5%	99.7%	6.6%
0.08	139356	89.6%	5.5%	2.5%	99.8%	6.5%
0.16	229504	94.8%	2.7%	1.2%	99.9%	6.6%
0.32	463838	97.6%	1.2%	0.6%	99.9%	6.4%
Tested on Terracotta-warrior Model with 60k Trgl. with Diff. Grid Res.						
Grid Res.	*IntE#*	Hybrid				Bisect.
		$S_d = 0$	$S_d = 1$	$S_d = 2$	$S_d \le 6$	$S_d \le 6$
64	1268	6.2%	15.5%	27%	97.7%	0.8%
128	5520	20.4%	25.5%	22.2%	93.4%	1.5%
256	21030	47.6%	24.7%	14.8%	99.3%	3%
512	85032	72.1%	14.9%	7%	99.5%	6.4%

*The percentages shown here describe the number of intersected edges that have found the intersection points after S_d times of subdivision.

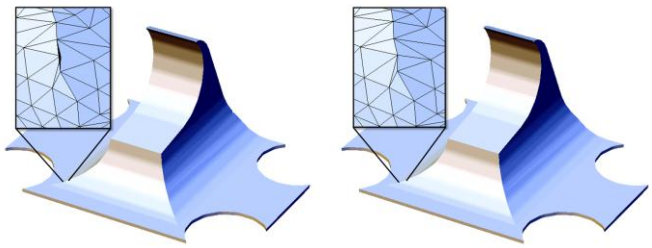


Fig. 20. Self-intersection on triangles can be successfully removed on our modified intersection-free dual contouring algorithm. The surface is an offset of the anchor-plate model given in Fig.12.

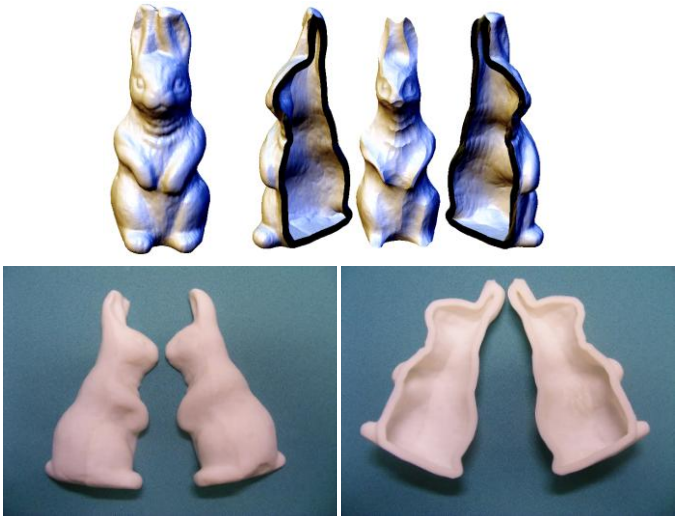


Fig. 21. Hollowing a rabbit model: (top) the rabbit and its hollowed model, and (bottom) the fabricated model by a *Fused Deposition Modeling* (FDM) RP machine.

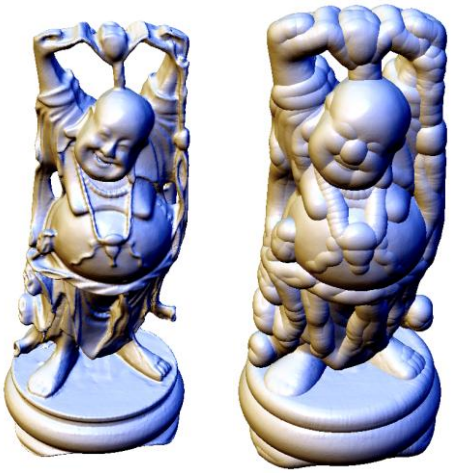


Fig. 22. Offset surface generating on a Buddha model with 1M triangles: (left) the given model and (right) the offset surface with offset value equaling to 2% of the diagonal length of the bounding box.

resolutions – with 60k, 200k and 600k triangles. The similar test is also conducted on the same terracotta-warrior model (with 60k triangles) but with different offset values and on grids with different resolutions. All statistical data are listed in Table V. It is easy to conclude that the analytical solution can be found on more than 90% of the intersected edges after six times of subdivision; whereas, the bisection search based method can find intersection points on less than 7% of the intersected edges after six times of subdivisions. In short, our hybrid method greatly reduces the time of intersection computation.

The resultant models of offsetting are required to be self-intersection-free in general. Figure 20 gives an example of some successfully removing self-intersected triangles on the offset surface of the anchor-plate model which has been previously shown in Fig.12(b).

Our offsetting approach is volumetric which is immune to self-intersection, and the offset surface is defined by a signed distance function (Eq.(1)) so that it guarantees the uniform wall thickness – which is important for generating the hollowed model in rapid prototyping. A rabbit fabricated from the hollowed model generated by our method is shown in Fig.21. It is easy to find that the wall thickness is uniform and the sharp features are preserved on the inner surface of the hollowed model.

B. Discussions

To verify the performance of our algorithm, we test the Buddha model with 1M triangles used in [5] on our implementation with the same offset value (2% of the diagonal length of the bounding box) and the same resolution (512^3). Moreover, to fairly conduct the comparison, we turn off the parallel speedup in our implementation – in other words, only one core of the CPU is employed. When using one core (i.e., no speed optimization using parallel computing), the offset surface can be generated in 180.5 seconds where we spend 9 seconds on SSVH construction (T_s), take 114.5 seconds in generating the signed distance fields in narrow-based (T_d) and

TABLE VI
COMPARISONS FOR SELF-INTERSECTION DETECTION METHODS

Models	Offset	Method in [8]		Our method	
		Trgl. #	Time (s)	Trgl #	Time (s)
Buddha(1M)	0.02	974k	0.094	964k	0.046
Filigree(500k)	0.02	1,031k	0.093	1,021k	0.046
Chair (400k)	0.02	1,041k	0.11	1,041k	0.062
Terracotta warrior (200k)	0.02	567k	0.062	565k	0.031
	-0.01	327k	0.031	325k	0.015

TABLE VII
STATISTICS FOR SHAPE APPROXIMATION ERRORS

Models	Figure	Face #	Offset	Avg. Err.	Max. Err.
Carve-cube	11	7.1k	-0.04	0.000029	0.0009
			0.05	0.000038	0.00074
Anchor-plate	12	9k	-0.041	0.00017	0.033
			0.013	0.00072	0.016
			0.2	0.000024	0.0017
Octa-flower	13	15.8k	-0.05	0.00006	0.02
			0.1	0.000063	0.0027
Dragon	14	49.4k	-0.02	0.000094	0.094
			0.03	0.0008	0.061
Vase-lion	15	400k	-0.02	0.00078	0.07
			0.04	0.0004	0.027
Rabbit	21	134k	-0.03	0.000096	0.053

use 57 seconds for surface contouring (T_c). The computing time reported in [5] is more than 3,000 seconds. We consider this as a significant speed up (with more than 20 times). When eight cores on our test-bed are all used, the time for offsetting the surface can be further reduced into 51.1 seconds with T_s = 9 sec., T_d = 22.8 sec. and T_c = 19.3 sec. – more than 50 times speed up. Figure 22 shows our offsetting result.

The self-intersection detection approach proposed in section VI is also compared with the method presented in [8]. The testing results agree with our analysis – around two times of speed up can be achieved. Furthermore, as ours is more robust from numerical errors, less triangles are generated. The computational statistics on several examples can be found in Table VI.

The precision analysis on the resultant model is given in Table VII, where the average distance errors and the maximal distance errors with reference to the offset value have been listed. Generally, the average distance errors are small but the maximal distance errors are relative big in some examples. This is because that the surface contouring is limited by the highest resolution of distance and may generate some triangle edges that have high shape approximation error. This considered as the first limitation of our approach.

The second major limitation of our approach is a common problem of all approaches when using a fixed resolution to sample the problem domain. The features whose size is less than the sampling distance may be missed during the reconstruction step. This may result in an offset mesh surface which is not topologically homeomorphic to the actual offset surface. Although the surface reconstruction for guaranteeing the topology is important, the homeomorphic requirement can be released in some applications (e.g., rapid prototyping, CNC manufacturability analysis), where our approach can be employed. An adaptive sampling based approach like [27] can be considered to preserve the topology. This is one of our possible future works.

Another limitation of our current implementation is that some sharp features whose size is less than the sampling distance between neighboring grid nodes may be harmed during the intersection-free reconstruction. The shape of such features however can be recovered by the original dual contouring [7]. One of our near future works is to develop an incremental algorithm to recover the shape of these small sharp features while preventing self-intersection.

VIII. CONCLUSION

In this paper, we present a fast offset surface generation approach to construct intersection-free offset surfaces from a solid bounded by triangular mesh surfaces. Sharp features are preserved on the resultant models. The basic spirit of our algorithm is to sample a signed distance field in a narrow-band from the input model on a uniform grid and then employ a contouring algorithm to build the resultant offset mesh surface from the signed distance field. Four filters are developed to generate the distance-field around the offset surface in a very efficient way by neglecting the redundancies in the regions far from the offset surfaces. In our approach, the resultant mesh surfaces are generated by a modified dual contouring which relies on accurate intersections between the grid edges and the isosurfaces. A hybrid method is developed to prevent the expensive bisection search in the configurations where the analytical solutions exist. Our modified intersection-free dual contouring is based on convex/concave analysis, which is more robust and efficient. The quality and performance of our approach have been demonstrated by a number of tests on various models and the comparisons with the state-of-the-arts.

ACKNOWLEDGMENT

The authors would like to thank the support of HKSAR Research Grants Council GRF Grant (CUHK/417508), CUHK Direct Research Grant (CUHK/2050400), and Shun Hing Institute of Advanced Engineering (SHIAE) Research Grant (CUHK/8115022). The first author is also partially supported by the Natural Science Foundation of China (NSFC - Grant No.: 60970097) and the Open Project Program of the State Key Lab of CAD&CG (Grant No.: A0805) at Zhejiang University. Some models shown in this paper are obtained from the Aim@Shape Shape Repository.

REFERENCES

[1] J.R. Rossignac, and A.A.G. Requicha, "Offsetting operations in solid modeling", *Computer Aided Geometric Design*, vol.3, no.2, pp.129-148, 1986.
[2] B. Pham, "Offset curves and surfaces: a brief survey", *Computer-Aided Design*, vol.24, no.4, pp.223-229, 1992.
[3] T. Maekawa, "An overview of offset curves and surfaces", *Computer-Aided Design*, vol.31, no.3, pp.165-173, 1999.
[4] G. Varadhan, and D. Manocha, "Accurate Minkowski sum approximation of polyhedral models", *Graphical Models*, vol.68, no.4, pp.343–355, 2006.

[5] D. Pavic, and L. Kobbelt, "High-resolution volumetric computation of offset surfaces with feature preservation", *Computer Graphics Forum*, vol.27, no.2, pp.165–174, 2008.
[6] J. M. Lien, "Covering Minkowski Sum Boundary using Points with Applications," *Computer Aided Geometric Design*, vol. 25, no. 8, pp.652–666, 2008.
[7] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data", *Prof. of SIGGRAPH 2002*, pp.339-346, 2002.
[8] T. Ju, and T. Udeshi, "Intersection-free contouring on an octree grid", *Proc. of Pacific Graphics 2006*, IEEE Computer Society.
[9] P. Hachenberger, "Exact Minkowski sums of polyhedral and exact and efficient decomposition of polyhedral in convex pieces", *Proc. 15th Annual European Symposium on Algorithms (ESA)*, pp. 669-680, 2007.
[10] J.M. Lien, and N.M. Amato, "Approximate convex decomposition of polyhedra", *Proc. of the ACM Symposium on Solid and Physical Modeling 2007*, pp.121-131, 2007.
[11] X.Y. Zhang, Y.J. Kim, and D. Manocha, "Reliable sweeps", *Proc. of the ACM Symposium on Solid and Physical Modeling 2009*, pp.373-378, 2009.
[12] R.T. Farouki, "Exact offset procedures for simple solids", *Computer Aided Geometric Design*, vol.2, no.4, pp.257-279.
[13] M. Bala, and T.C. Chang, "Automatic cutter selection and optimal cutter path generation for prismatic parts", *International Journal of Production Research*, vol.29, no.11, pp.2163-2176, 1991.
[14] M. Forsyth, "Shelling and offsetting bodies", *Proc. of the ACM Symposium on Solid Modeling and Applications*, pp.373-381, 1995.
[15] X. Qu, and B. Stucker, "A 3D surface offset method for STL-format models", *Rapid Prototyping Journal*, vol.9, no.3, pp.133–141, 2003.
[16] D.E. Breen, and S. Mauch, "Generating shaded offset surfaces with distance, closest-point and color volumes", *Proceedings of the International Workshop on Volume Graphics*, pp. 307–320, 1999.
[17] D.E. Breen, S. Mauch, and R.T. Whitaker, "3D scan conversion of CSG models into distance volumes", *Proceedings of the 1998 IEEE symposium on Volume visualization*, ACM Press, pp.7–14, 1998.
[18] J.A. Sethian, *Level Set Methods and Fast Marching Methods*, Second Ed., Cambridge University Press, Cambridge, 1999.
[19] Y. Chen, H. Wang, D.W. Rosen, and J. Rossignac, "A point-based offsetting method of polygonal meshes", *Technical Report*, 2005.
[20] J. Huang, Y. Li, R. Crawfis, S.C. Lu, and S.Y. Liou, "A complete distance field representation", *Proceedings of the conference on Visualization'01*, pp. 247–254, 2001.
[21] S.J. Liu and C.C.L. Wang, "Duplex fitting of zero-level and offset surfaces", *Computer-Aided Design*, vol.41, no.4, pp.268-281, 2009.
[22] J.A. Baerentzen, and H. Aanaes, "Signed distance computation using the angle weighted pseudonormal", *IEEE Transactions on Visualization and Computer Graphics*, vol.11, no.3, pp.243-253, 2005.
[23] E. Larsen, S. Gottschalk, M.C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes", *Proc. of International Conference on Robotics and Automation*, pp.3719-3726, April 2000.
[24] D. Eberly, "Distance between point and triangle in 3D", *Geometric tools*, LLC, http://www.geometrictools.com, 2008.
[25] C.C.L. Wang, "Intersection-free dual contouring on uniform grids: an approach based on convex/concave analysis", Technical Report, 2009, http://www2.mae.cuhk.edu.hk/~cwang/pubs/TRIntersectionFreeDC.pdf.
[26] N. Chirkov, "Fast 3D line segment–triangle intersection test", *Journal of Graphics, GPU, & Game Tools*, vol.10, no.3, pp.13-18, 2005.
[27] G. Varadhan, S. Krishnan, T.V.N. Sriram, and D. Manocha, "Topology preserving surface extraction using adaptive subdivision", *Proceedings of Eurographics Symposium on Geometry Processing*, pp.235-244, 2004.

Shengjun Liu is currently a lecturer at the school of mathematical sciences and computing technology, Central South University, China. He received his B.S. degree in Computational Mathematics and Applied Software in 2000 and M.Sc. degree in Applied Mathematics in 2002 from Central South University, D.Eng. degree in Computer Science and Technology in 2007 from Zhejiang University. His research interests include geometric modeling in CAD/CAM, reverse engineering, and computer graphics.

Charlie C.L. Wang is currently an Associate Professor at the Department of Mechanical and Automation Engineering, the Chinese University of Hong Kong, where he began his academic career in 2003. He gained his B.Eng. (1998) in Mechatronics Engineering from Huazhong University of Science and Technology, M.Phil. (2000) and Ph.D. (2002) in Mechanical Engineering from the Hong Kong University of Science and Technology. He is a member of IEEE and ASME, and an executive committee member of Technical Committee on Computer-Aided Product and Process Development (CAPPD) of ASME. Dr. Wang has received a few awards including the ASME CIE Young Engineer Award (2009), the CUHK Young Researcher Award (2009), the CUHK Vice-Chancellor's Exemplary Teaching Award (2008), and the Best Paper Awards of ASME CIE Conferences (in 2008 and 2001). His current research interests include geometric modeling in computer-aided design and manufacturing, biomedical engineering and computer graphics, as well as computational physics in virtual reality.