Approximate Boolean Operations on Large Polyhedral Solids with Partial Mesh Reconstruction

Charlie C.L. Wang, *Member, IEEE*

***Abstract*—We present a new approach to compute the approximate Boolean operations of two freeform polygonal mesh solids efficiently with the help of Layered Depth Images (LDI). After applying the LDI sampling based membership classification, the most challenging part, a trimmed adaptive contouring algorithm, is developed to reconstruct the mesh surface from the LDI samples near the intersected regions and stitch it to the boundary of the retained surfaces. Our method of approximate Boolean operations holds the advantage of numerical robustness as the approach uses volumetric representation. However, unlike other methods based on volumetric representation, we do not damage the facets in non-intersected regions, thus preserving geometric details much better and speeding up the computation as well. We show that the proposed method can successfully compute the Boolean operations of freeform solids with a massive number of polygons in a few seconds.**

***Index Terms*—Boolean operations, freeform solids, robust, approximation, Layered Depth Images.**

I. Introduction

THE conventional methods of Boolean operations for two solids are based on the intersection computation and the surface membership classification. Robustness and efficiency are the major difficulties in implementing Boolean operations on freeform solids with complex geometry, which are usually represented by polygonal meshes. More specifically, when the number of polygons involved is massive, it takes a long time to detect and compute the intersection curves. Furthermore, robustness problems are very common in this kind of approach (e.g., two polygons are tangentially contacted or are only intersected on one of their boundary edges). Simply using approximate arithmetic to implement a conventional Boolean operation algorithm makes the program unstable (e.g., see the results in Fig.18). The inconsistencies arising from numerical error can lead to incorrect topology (such as breaks in the boundary or inconsistent intersection curves on two solids). Although the techniques of exact arithmetic (ref. [5], [24], [25], [34], [35], [50]–[52]) and interval computation (ref. [22], [44], [45], [72]) have been employed in the algorithms of Boolean operations, they are quite expensive in terms of both computing time and memory. Moreover, conventional algorithms can hardly be parallelized to borrow the advanced computational power available on consumer PCs with GPU or multi-core CPU. Another stream of research to solve the robustness problem in Boolean operations is based on volumetric representation (e.g., [26], [46], [54], [59], [77], [78], [80]). Nevertheless, the procedure of surface-volume conversion, processing and volume-surface conversion can easily lose the geometric details and other attributes of the given models unless the geometry of the original models is retained. Retaining the geometry is easier for point-sampled surfaces [2] but much more difficult for mesh surfaces. The purpose of this research is to develop a fast boundary evaluation method which inherits the robustness of volume-based approach but only introduces shape approximation error at intersected regions by retaining facets at non-intersected regions.

Manuscript was submitted to IEEE TVCG on January 9, 2009.
Revised manuscript was prepared on May 8, 2010.

We exploit a new approach to efficiently compute the approximate Boolean operations of two freeform solids with the help of Layered Depth Images (LDI). We assume that the given two freeform solids are represented by non-self-intersected closed triangular mesh surfaces. Firstly, the given triangular meshes are sampled into LDIs. By a clustering algorithm, each triangle on given models has at least one or even more corresponding sample points in the LDIs. As it will be detailed later, the LDIs actually give a semi-implicit representation of the given models; therefore, Boolean operations can be robustly and efficiently performed on them. The resultant samples in the LDI are then employed to give membership classification for triangles on the given models. The model after membership classification is in a LDI/mesh hybrid representation. Lastly, the most challenging part – a trimmed adaptive contouring algorithm is developed to reconstruct the mesh surface in the intersected region from the LDI/mesh hybrid and stitch it to the boundary of the retained mesh surfaces. The trimmed adaptive contouring algorithm is different from a mesh hole filling (e.g., [6]). Since we do not explicitly compute the intersection curves, the missed region to be reconstructed in general has a larger area. The shape of the missed region is however represented by LDI samples. Therefore, the reconstructed surface must capture the shape with geometric details rather than smoothly interpolating the boundaries. Unlike the previous works of polytree (ref. [14], [15]), extended octree (ref. [12], [13]) and adaptive contouring (ref. [46], [48]) which reconstruct the whole mesh surface, we take a similar strategy as [10], [19], [61], [79]

- to retain the surfaces that are far away from the intersected region Ψ,
- to reconstruct the surfaces near Ψ,
- and produce the two-manifold boundary of a solid model by stitching the retained and the reconstructed surfaces.

Sampling is adopted to approximate the boundary of the result of a Boolean operation on triangulated solids and the result of reconstruction is a two-manifold mesh surface. The Hausdorff error with respect to the correct result decreases with the sampling rate.

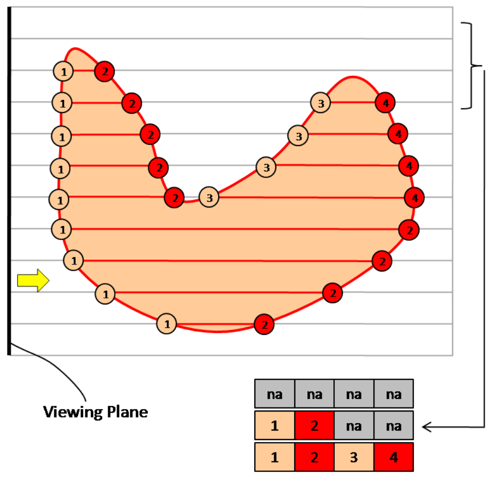

Fig. 1. A 2D illustration of Layered Depth Images (LDI), and the samples sorted by ascending depth values in three pixels are also listed.

We do not reconstruct the intersected region of surfaces at the finest resolution of a LDI. Instead, an octree is employed to make the resolution of the reconstructed mesh adaptive to 1) topology complexity on retained surfaces and 2) geometry complexity of the shape to be reconstructed. Building an octree for partial reconstruction and stitching the reconstructed surface to the boundary of the retained mesh surface is an open problem. We propose a new algorithm in this paper to solve this problem. Another major difference between [10], [61], [79] and ours is that the membership classification is accelerated by graphics hardware in our algorithm.

A. *Previous work*

The previous work about Boolean operations on 3D polyhedral objects can be classified into three broad categories: approaches based on exact arithmetic and interval computation, special algorithms using approximate arithmetic, and volumetric methods. In addition, another relevant stream of research is the techniques in image space.

Exact arithmetic and interval computation

The algorithms for Boolean operations on 3D solid models have been studied for many years (see [55], [63], [64]). When implementing these algorithms (ref. [40]–[42]), the problem of robustness is the major concern. Many of the researches use exact arithmetic as it provides the most promising approach to the numerical robustness problem. The authors in [5], [24] and [25] considered Boolean operations on solids bounded by piecewise linear surfaces, which are recently optimized and integrated in CGAL (ref. [16], [34], [35]). More advanced approaches of Boolean operations by exact arithmetic in the curved domain can be found in [50]–[52]. Some other approaches employ the technique of interval computation. The rounded interval arithmetic was adopted in [44] and [45] to compute Boolean operations on solids with spline surfaces. Fang et al. [22] and Segal [72] conducted the tolerances (which are actually intervals) to keep the information of the algorithm's decision-making history. That means, when a new decision is to be made, the algorithm makes it consistent with all previous ones by checking the tolerances. A more comprehensive review can be found in [68]. However, significant extra computation and memory are required by the approaches based on exact arithmetic and interval computation. It is impractical to apply them on freeform solids with hundreds of thousands of triangles as the examples shown in this paper.

Approximate arithmetic

Since how to effectively deal with degenerated surface-surface intersections is one of the most challenging aspects of exact solid modeling calculations, some approaches were proposed to compute approximate Boolean operations instead of the exact ones. Biermann et al. [8] introduced a method for computing approximate results of Boolean operations applied to freeform solids bounded by multi-resolution subdivision surfaces. Different from the aforementioned algorithms, the cutting and merging steps are applied at the coarse mesh level, and the resultant coarse mesh is then fit to the given detailed mesh surfaces. The robustness problem in intersection computation is solved by the numerical perturbation method applied to the coarse meshes. However, it becomes time-consuming when applying such a perturbation method to the given models with a massive number of polygons. Recently, Smith and Dodgson [73] described a topologically robust algorithm, which uses approximate arithmetic. After carefully defining the relationship between a series of interdependent operations, the consistency in output is ensured and therefore a correct connectivity is guaranteed in the final results. One major defect of the approach is its inability to borrow the power of parallel computing, which is available at consumer level PCs nowadays, since the operations are interdependent. In addition, the implementation of such a complex algorithm as [73] is not easy.

Volumetric methods

An alternative to directly computing on mesh surfaces is to convert them into volumetric representation, and then the results of Boolean operations can be easily and robustly obtained on volumetric data (e.g., the adaptive distance-field method in [26] and the level-set based method in [59]). However, sharp edges and corners of the original surface are removed by the sampling process. As observed by Kobbelt et al. in [54], even if an over-sampling is applied, the associated aliasing error is not absolutely eliminated since the surface normals in the reconstructed model usually do not converge to the normal field of the original model. Therefore, the normal information is encoded during the sampling in [54] and [46] to provide the ability of reconstructing sharp features on resultant surfaces. A variety of variants of [46] have been developed thereafter trying to enhance the results in the aspects of topology preservation (ref. [77], [78], [80]), intersection free [48], and manifold preservation [71]. The authors in [2] employed the octree structure to give inside-outside detection of surfels that resulted in Boolean operations on surfel-bounded solids. The major drawback of all these approaches except [2] is that the surface-volume-surface conversion often damages the geometric details and other attributes of the given models. Nevertheless, our algorithm has overcome the problem.

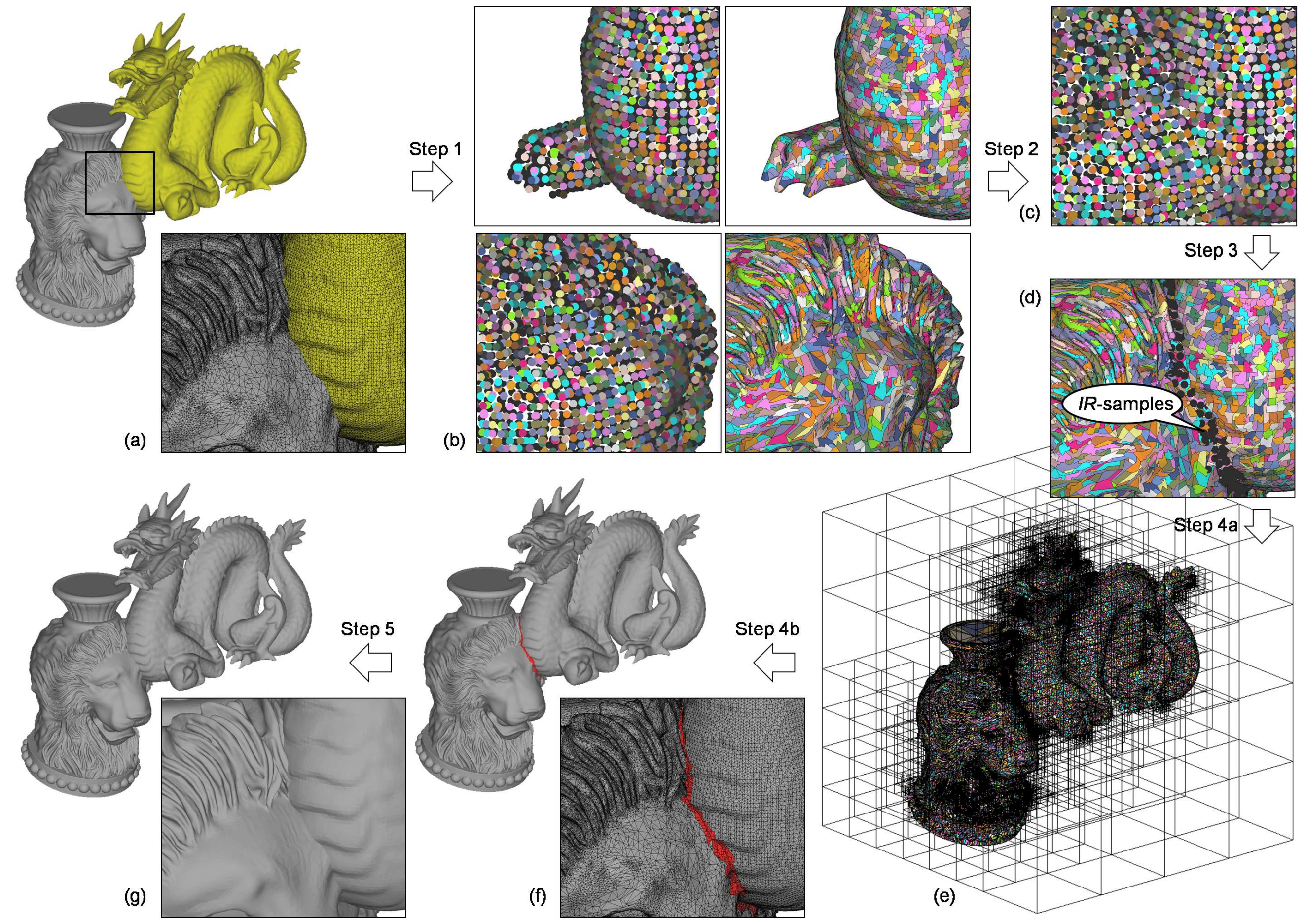

Fig. 2. An overview of our algorithm for fast approximation Boolean operation on large polyhedral solids. The vase-lion model and the dragon model have 400k and 277k triangles respectively — it is impractical to compute the Boolean operations on such models using exact arithmetic based commercial solid modeling systems. Our algorithm consists of five steps: 1) LDI sampling and surface clustering, 2) Boolean operation on the LDI samples, 3) membership classification, 4) trimmed adaptive contouring, and 5) post-processing. Note that, for the purpose of having a clear illustration, the LDI is only sampled at 128×128 in this figure.

Techniques in image space

Using graphics hardware to speed up the evaluation of Boolean operations in image space has a long history (ref. [4], [27], [29], [31]–[33], [49], [66], [67], [74]). The purpose of these algorithms is to provide a quick feedback in rendering rather than boundary evaluation, which indeed is our mission.

There are also many approaches in literature using graphics hardware for interference and collision computations in the image space, where a review can be found in [28] and [75]. Among them, the approaches of [38], [60] and the recent variants in [23] and [76] decompose a given non-self-intersected closed object into Layered Depth Images (LDI) through a specified viewing, where each pixel in a LDI contains a sequence of numbers that specify the depths from the intersections (between a ray passing through the center of a pixel along the viewing direction and the object to be sampled) to the viewing plane and the depths are sorted in ascending order. Figure 1 gives a 2D illustration of the samples stored in a LDI. Sampling a given model with LDI along a single direction $\mathbf{t}$ will miss the surface regions that are nearly perpendicular to $\mathbf{t}$ (e.g., the bottom region of the surface shown in Fig.1). This situation of miss-sampling can be improved by conducting another sampling along the direction perpendicular to $\mathbf{t}$. The sampling can be executed on graphics hardware with the help of stencil-buffer and depth-buffer. For a correctly sampled solid model represented by a LDI, the number of sampled depths on a pixel must be even — this can be guaranteed by stencil-buffer based peeling (see [38]). The inside/outside detection can be conducted very efficiently (see [23] and [76]), so a LDI can actually be considered as a semi-implicit representation. LDI adopted in this paper is indeed an extension of the ray-rep (or dexel) in solid modeling (ref. [7], [21], [37], [43], [56]–[58], [69]). The common defect of these approaches is similar to the volumetric representation based ones – when contouring the computed results back into mesh surfaces, many geometric details are easily destroyed. The approaches using voxels (e.g., [11], [17], [20]) have the same problem as ray-rep has in this aspect. Our algorithm outperforms them on this by retaining the facets from original models in non-intersected regions.

B. Main results

We sample the given models into LDI solids by the graphics hardware accelerated rasterization. Boolean operations are robustly conducted on the LDI solids. After applying LDI sampling based membership classification, a trimmed adaptive contouring algorithm is exploited to reconstruct triangular faces for the intersected region by the samples of a LDI. These result in a robust and fast method of approximate Boolean operations on freeform solids with the help of LDI. Our approach maintains the good property of robustness from the volumetric approaches while keeping the triangles in non-intersected regions, which can preserve geometric details better than other volumetric approaches.

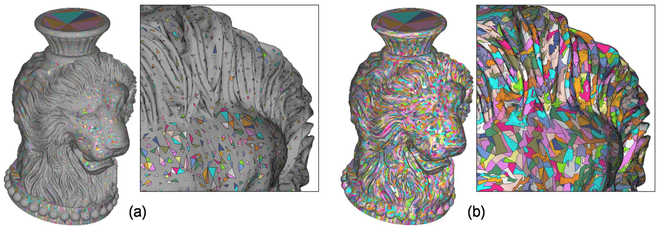

Fig. 3. The surface clustering by sampled triangles: (a) with many miss-sampled triangles — displayed in dark-gray, and (b) with all triangles having region ID specified.

II. Overview

In order to sample a given freeform model M well and also ease the later surface reconstruction, we adopt a structured set of LDIs consisting of x-LDI, y-LDI and z-LDI sampled along $x-$, $y-$ and $z-$axis respectively. The images have the same resolution and are located in a way that their rays intersect at the $w \times w \times w$ nodes of uniform grids. A similar sampling can be found in the approaches of triple ray-rep in [7] and [58], and the sampling of surfels in [62]. With the help of structured LDIs, our algorithm can efficiently compute the partial approximate Boolean operations on large polyhedral solids in five steps.

Step 1: LDI sampling and surface clustering

The first step of our algorithm is to sample the piecewise-linear surfaces of given models (see Fig.2(a)) into structured LDIs. Two given models M_A and M_B are sampled into the LDI solids L_A and L_B. To make the rays of L_A exactly overlap the rays of L_B, the sampling of M_A and M_B must be conducted in the same working envelope — the common bounding box of M_A and M_B. The IDs of triangles on M_A and M_B are transferred to the samples of L_A and L_B with the help of color buffer. We first assign a unique ID to every triangle of M_A and M_B. The number of every ID is then mapped into an RGB-color. Therefore, after rendering all faces by the colors according to their IDs, we can easily identify from which triangle the sample at a fragment is sampled by its RBG-color. As each color component is represented by a number with 8 bits, we can render up to $2^{24} = 16,777,216$ distinguished triangles, which is much more than the required number in practical use. For small triangles that are missed during the sampling, we cluster them into the same region group as that of their neighboring sampled triangles by a flood fill method (see Fig.2(b) and 3). Note that this clustering does not prevent recovering them by the LDI samples. In fact, they are approximated by the polygons constructed in step 4.

Step 2: Boolean operation on LDI samples

The computation of a Boolean operation on two LDI solids L_A and L_B is decoupled into the Boolean operation on the overlapped rays $r_t^A(i,j) \in L_A$ and $r_t^B(i,j) \in L_B$ in parallel. In other words,

$$L_{res} = \forall_{t=\{x,y,z\}} \{(i,j) | r_t^A(i,j) \diamond r_t^B(i,j)\} \tag{1}$$

with "$\diamond$" denoting one of the Boolean operations: "$\cup$", "$\cap$" or "$\setminus$". From the definition of ray-rep, we know that the samples on the rays actually represent a one-dimensional solid where a ray enters into the solid at the *odd*-th samples and leaves at the *even*-th samples. Therefore, the Boolean operations on them can be implemented by moving forward incrementally on sorted samples of corresponding rays.

Step 3: Membership classification

In this step, the samples at the LDI solid L_{res} are employed to determine which triangles on M_A and M_B are to be retained, and remove other triangles. Two given models M_A and M_B have been sampled into LDI solids L_A and L_B. For the new solid L_{res} obtained by Boolean operations, a region $R \in M_A$ (or $\in M_B$) remains if and only if it has the same number of corresponding samples in L_{res} and L_A (or L_B). Putting the triangles of the remaining region of M_A and M_B together, a mesh surface M_P that is part of the resultant solid is obtained. M_P is an open surface. The samples in L_{res} with their corresponding triangles removed are called *IR-samples* (e.g., the black dots in Fig.2(d)), where "IR-" stands for intersected region. The IR-samples are a discrete representation of surfaces near the intersected regions on the Boolean result of M_A and M_B. The IR-samples on L_{res} together with M_P give a Mesh/LDI-hybrid representation of the resultant solid. The surface regions represented by IR-samples are named as *intersected regions*. Figure 2(d) shows such a hybrid representation. Note that for the "$\setminus$" operation, the triangles from M_B must have their normals flipped by reversing the order of their vertices. To ensure M_P does not contain any part not on the real resultant model, the regions next to the boundary of M_P are removed as well.

Step 4: Trimmed adaptive contouring

This is the most difficult step of our algorithm. A trimmed adaptive contouring algorithm is exploited to generate the mesh surface approximating the shape represented by the IR-samples while interpolating the boundary of M_P. We first construct the octree structure according to the geometry, topology and surface stitching criteria (see Fig.2(e)). Then the mesh surface M_I is contoured from the octree and stitched to M_P. An example is shown in Fig.2(f), where the red polygons belong to M_I. Details of the trimmed adaptive contouring algorithm are presented in the following section.

Step 5: Post-processing

The post-processing step conducts the remaining work of triangulating the quadrilateral faces on M_I, eliminating the non-manifold entities on the resultant mesh surface, and filling the local topology information in the data structure of the mesh surface — see the result in Fig.2(g).

III. Trimmed Adaptive Contouring on Mesh/LDI-Hybrid Representation

A trimmed adaptive contouring algorithm is developed to generate the mesh surface M_I in intersected regions by samples of L_{res}. Meanwhile, M_I is stitched to the mesh surface M_P, which is obtained after membership classification. The adaptivity is gained by using a hierarchical structure — octree. The algorithm consists of two major steps: 1) octree construction and 2) mesh generation and stitching. As a result, M_I and M_P are tightly stitched together to produce the resultant surface of Boolean operations. Although some small holes (or gaps) are generated on the resultant mesh, they can be easily eliminated by a cleaning process.

A. Octree construction

An octree is constructed to govern the mesh generation of M_I, where the space occupied by each node is defined as a *cell*. However, different from the construction strategy of the octree as shown in [71], [77], [78], we need to consider both the topology and the shape of M_I and the boundary of M_P. The octree is constructed in a top-down manner. Starting from the root cell, the cells are recursively refined into eight sub-cells based on the conditions of 1) the topology simplicity, 2) the geometry approximation, and 3) the surface stitching. For a LDI with $w \times w$ resolution and that has the orthogonal distance r between rays, if we assume that $w = 2^n + 1$ with n being an integer, we can define the root cell's width as $2^n r$ and locate the root cell to the position where its edges overlap with rays in L_{res}. By this, the refined cells can always have their edges overlapped with rays in L_{res} as long as the widths of the cells are $\geq r$. Therefore, the inside/outside status of the eight nodes in a cell can be detected quickly.

Topology simplicity For the face of a cell C, when the diagonal nodes on the face have the same inside/outside status, the topology of the surface inside the cell is ambiguously defined (e.g., the right face in Fig.4(a)). This is called *face ambiguous*. When any pair of diagonally opposite nodes in the cell C has one sign (inside or outside) while the other vertices have a different sign, the topology of the surface inside the cell is also ambiguous (e.g., Fig.4(b)). This is named as *voxel ambiguous*. When a cell has any of the above ambiguity, it needs to be further refined to figure out the real topology inside. Furthermore, as shown in Fig.4(c), when any edge of a cell contains more than one LDI sample (i.e., have multiple intersections between the ray and the real resultant surface), the cell also contains complex topology and needs to be further subdivided.

A cell that contains a part of the surface but all its eight nodes are detected as *inside* is defined as a *fake solid volume*. If a cell is a fake solid volume, its inside topology is complex. For a cell whose width is greater than r, such a case can be detected by checking if any LDI sample is in the cell when the cell's eight nodes are all *inside* the resultant solid. A similar situation occurs when a cell has a *fake solid face* — a face with all its four nodes inside (or outside) but having some 1D solids from the LDI passing through. Figure 5 gives some examples. Fake empty volumes and fake empty faces can be checked in a similar way.

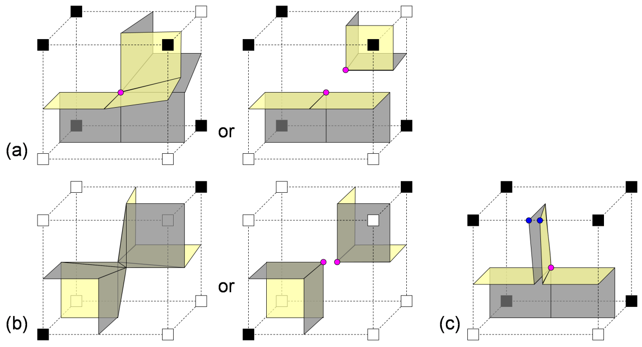

Fig. 4. Examples of cells with complex topology. (a) The topology of the right-face of cell is ambiguously defined by the inside/outside status of nodes. (b) The topology in the cell is ambiguously defined by the nodes' configuration. (c) The cell holding an edge with multiple samples (the blue ones) has complicated topology.

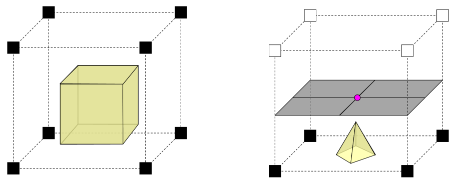

Fig. 5. Examples of a fake solid volume (left) and a fake solid face (right; at the bottom) — we assume that the width of the cell is greater than r and the cavities can be sampled by rays of the LDI.

For a cell C with its width longer than r, if it is in any of the following cases,

- face ambiguous;
- voxel ambiguous;
- having more than one sample on a cell-edge;
- being a fake solid (or empty) volume;
- having a fake solid (or empty) face;

the cell C must be further subdivided into eight sub-cells. To find the right balance between speed and accuracy, the refinement is bounded by the resolution of the LDI.

Geometry approximation The samples of L_{res} inside a cell C can be considered as a set of Hermite data points which specify the geometry in C. The normal vector of a Hermite point in L_{res} is obtained by the triangle from which the point is sampled. In the later mesh generation algorithm, the geometry of a leaf-node cell C containing the reconstructed surface M_I is approximated by a vertex and its adjacent faces. The vertex is located at a *QEF-minimizing-point* $\mathbf{q}_c$, whose position minimizes the *Quadratic Error Function* (QEF), $Q_F(\mathbf{q}_c) = \sum_{\mathbf{h}_i} ((\mathbf{q}_c - \mathbf{h}_i) \cdot \mathbf{n}_{h_i})^2$, defined by all Hermite data points $(\mathbf{h}_i, \mathbf{n}_{h_i})$ in C. To be robust, the position $\mathbf{q}_c$ minimizing $Q_F(\mathbf{q}_c)$ can be computed by the singular value decomposition (SVD). Therefore, to control the geometry approximation error on M_I, we check the error between $\mathbf{q}_c$ and the planes defined by the Hermite samples. If $\exists(\mathbf{h}_i, \mathbf{n}_{h_i})$ that $|(\mathbf{q}_c - \mathbf{h}_i) \cdot \mathbf{n}_{h_i}| > \varepsilon_g$, the cell C needs to be further refined. ε_g is a coefficient used to control the shape approximation error. We choose $\varepsilon_g = 0.1r$ in all our examples.

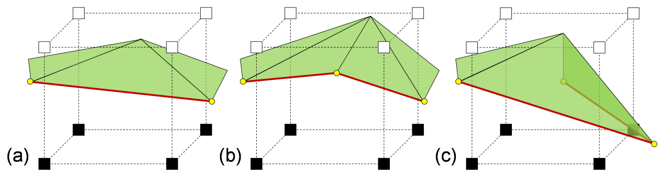

Fig. 6. Illustrations of the only two configurations in a cell C with simple boundary topology: (a) C contains a single boundary edge (in red), and (b) C contains two adjacent boundary edges and their shared vertex (in yellow). (c) Complex topology is presented in C if it contains only two adjacent boundary edges but not their shared vertex – the yellow vertex is outside the cell.

Surface stitching According to the IR-samples defined in the previous section, a cell C can be classified into the following different categories.

- *reconstructed-interior-cell*: if all samples of L_{res} inside C are IR-samples;
- *retained-interior-cell*: if no sample inside C is an IR-sample;
- *boundary-cell*: others that have both IR-samples and other LDI samples.

As we will stitch the vertices generated in the leaf-node cell to the boundary curves of the open surface M_P, the topology of the boundary embedded in a cell C must also be considered when constructing the octree. Every boundary-cell spans the space intersecting M_A (or M_B) that has some triangles removed and some retained. Thus, it must hold some boundary curves of M_P. The surface M_P in the cell C has a boundary curve with simple topology if C contains

- a single boundary edge of M_P (e.g., Fig.6(a)), or
- two adjacent boundary edges and their shared vertex (e.g., Fig.6(b)).

By this rule, all cells containing boundary edges of M_P are classified as *simple-boundary-cells* – cells containing boundary with simple topology, or *complex-boundary-cells* – other cells holding boundary edges of M_P. If the cell C is a *boundary-cell*, the refinement of C only stops when it is a *simple-boundary-cell*.

This criterion is adopted to ensure that the final mesh surface generated from the octree can be tightly stitched to M_P. Details of the stitching are discussed in the subsection of surface stitching below. In practice, to speed up the computation, topology simplicity and geometry approximation error are only detected on those *reconstructed-interior-cells* as the mesh surface M_I is only constructed by the *reconstructed-interior-cells* and *boundary-cells*. The pseudo-code is as shown in **Procedure 1**. Calling **Procedure** *CellRefinement* with the root cell and all boundary edges on M_P can construct the octree T_r for mesh generation and stitching. The refinement of cells can be run in parallel on multi-core CPUs.

Procedure 1 CellRefinement (cell C, edge-list E)

```
Require: $E$ containing the boundary edges of $M_P$
Ensure: the edges in $E$ intersect the domain of $C$
1: if (IsCellRefinementNeeded($C$, $E$)) then
2:     Subdivide cell $C$ into eight sub-cells $C_i\ _{(i=1,\ldots,8)}$;
3:     Subdivide $E$ into eight lists $E_i$ corresponding to $C_i$;
4:     for all $i$ such that $1 \leq i \leq 8$ do
5:         Call CellRefinement($C_i$, $E_i$); {Recursively}
6:     end for
7: end if
```

Procedure 2 IsCellRefinementNeeded (cell C, edge-list E)

```
1: if the width of $C \leq r$ then
2:     return false;
3: else if no LDI sample in $C$ is an IR-sample then
4:     return false;
5: else if $C$ is NOT with a simple topology then
6:     return true;
7: end if
8: Compute the QEF-minimizing-point $\mathbf{q}_c$ by all Hermite
   samples $(\mathbf{h}_i, \mathbf{n}_{h_i})$ in $C$;
9: for all $i$ do
10:    if $|(\mathbf{q}_c - \mathbf{h}_i) \cdot \mathbf{n}_{h_i}| > \varepsilon_g$ then
11:        return true;
12:    end if
13: end for
14: if $C$ is complex-boundary-cell then
15:     return true;
16: end if
17: return false;
```

B. Mesh generation and stitching

After constructing the octree T_r, the mesh surface M_I stitching to the retained mesh surface M_P (obtained by surface membership classification) is generated by contouring the octree. Similar to other octree contouring algorithms in [46], [77], [78], the polygonal faces are only constructed on minimal-edges of T_r that exhibit a sign change on their two endpoints. The *minimal-edge* is an edge on the leaf-node cell that does not properly contain an edge of a neighboring leaf. Mesh generation can be compactly implemented by recursively calling the functions to process the volumes, the faces and the edges of cells. More implementation details of contouring an octree can be found in [70]. Here, we only focus on the difference between our trimmed adaptive contouring algorithm and standard contouring.

First of all, we do not construct faces for all minimal-edges with a change of sign. In conventional contouring, quadrilateral (or triangular) faces are constructed around the minimal edges that exhibit a sign change when the minimal edges are adjacent to four (or three) leaf-node cells. However, we are going to generate the mesh surface M_I, which is a mesh surface only in the space spanned by *boundary-cell* or *reconstructed-interior-cell*.

The remarks and propositions below prove how a watertight result can be produced by M_P and M_I.

Remark 1 Faces of M_I are only constructed on the sign-changed minimal-edges whose neighboring cells are either *boundary-cell* or *reconstructed-interior-cell*.

As illustrated in Fig.7(a), if there are four cells around a minimal-edge and one cell among them is a *retained-interior-*

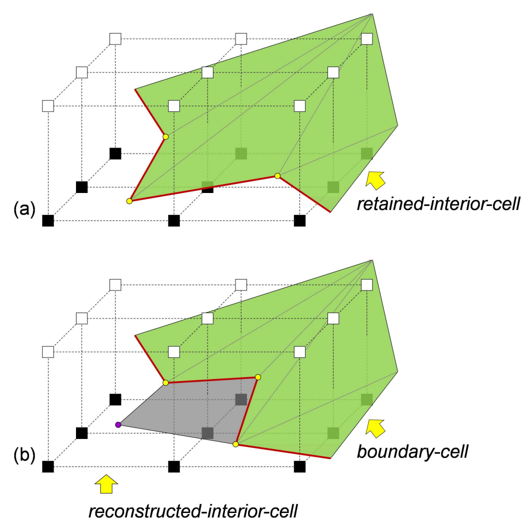

Fig. 7. Illustrations of mesh generation in partial adaptive contouring: (a) no face is constructed as there is a *retained-interior-cell*, and (b) a face (in gray) is constructed to link the vertices in the *boundary-cell* and the *reconstructed-interior-cells*. The vertex in the *reconstructed-interior-cell* is displayed in purple.

cell, no face is generated for the minimal edge. When all four cells are either *boundary-cell* or *reconstructed-interior-cell*, a quadrilateral face is generated during the contouring (e.g., the case shown in Fig.7(b)).

Proposition 1 If all leaf *boundary-cells* in the octree are with simple topology, the mesh M_I generated by the above method has the same boundary topology as that of M_P.

Proof. For the leaf cells of the octree constructed by the aforementioned method, *reconstructed-interior-cells* and *retained-interior-cells* are separated by *boundary-cells* and empty (solid) cells. The polygons are constructed only according to the sign changed minimal-edges, thus the boundary on the reconstructed mesh surface M_I is only from the *boundary-cells*. If a boundary-cell C_b is a simple cell, the topology of M_P's boundary in C_b is the same as that of the boundary formed by the reconstructed faces of M_I. Therefore, M_I and M_P have the same boundary topology. ◇

Vertices of the reconstructed mesh M_I are created in both the *reconstructed-interior-cells* and the *boundary-cells* adjacent to sign changed minimal-edges. The vertices in the *reconstructed-interior-cells* are placed at the position of their *QEF-minimizing-points* (e.g., the purple vertex in Fig.7(b)). A more difficult problem about how to locate vertices in the *boundary-cells* is discussed below.

The vertices of the mesh M_I in those leaf boundary-cells are positioned by stitching to the boundary of the mesh surface M_P.

Remark 2(a) For a simple-boundary-cell C with two adjacent boundary edges, their shared vertex $\mathbf{v}_c$ is adopted as the vertex in this cell.

Remark 2(b) The vertex $\mathbf{v}$ in a simple-boundary-cell C (whose QEF-minimizing-point is $\mathbf{q}_c$) contains only a single boundary edge e, and the position of $\mathbf{v}$ is computed by the closest point to $\mathbf{q}_c$ on e.

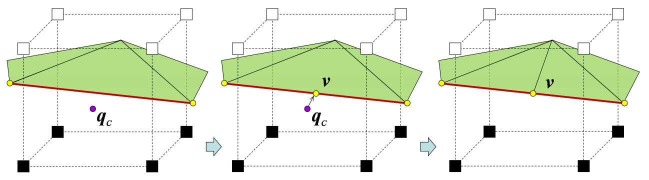

Fig. 8. Illustrations of stitching a vertex in a *simple-boundary-cell* holding only one boundary edge e (in red): (left) the QEF-minimizing-point $\mathbf{q}_c$ is computed, (middle) $\mathbf{q}_c$ is projected to its closest point on e, and (right) a vertex $\mathbf{v}$ is created to split the original face.

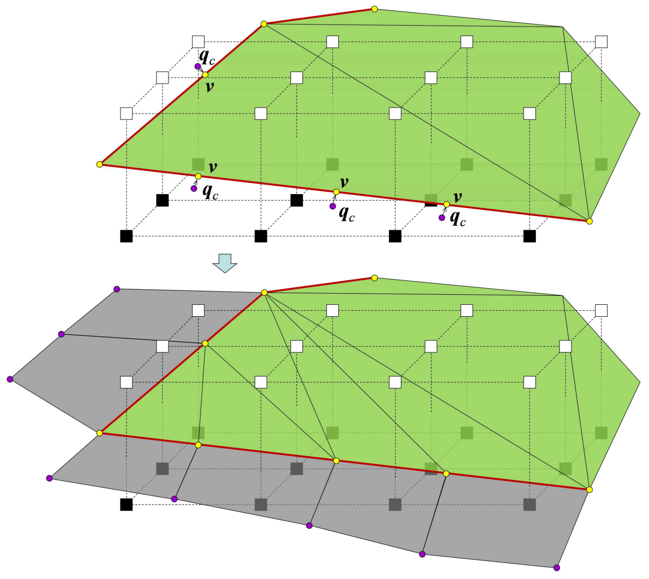

Fig. 9. Illustrations of a more complex example of stitching the reconstructed mesh surface M_I to the boundary edges of the retained mesh surface M_P.

The boundary edge e and its adjacent face must be split by the vertex $\mathbf{v}$ of cell C so that M_I and M_P match in C not only geometrically but also topologically. An illustration is given in Fig.8. A more complex case is that one boundary edge e passes through a few boundary cells (as shown in Fig.9). Therefore, in our implementation, we separate the projection and splitting. After projecting all vertices in the *simple-boundary-cells*, the created vertices are attached to e and sorted by their positions on e. The splitting is conducted thereafter as a face may be split into more than one triangle (see Fig.9).

Proposition 2 If the reconstructed mesh surface M_I and the open mesh M_P have the same boundary topology, M_I and M_P are connected in watertight by locating the boundary vertices on M_I in the way of Remark 2.

The proof of this proposition is straightforward. However, the leaf boundary-cells in the octree are *NOT* intrinsically with simple topology. More *complex-boundary-cells* are generated when the sampling distance of LDI solids is much larger than the size of triangles on given models (e.g., as the case shown in Fig.3). Increasing the sampling rate of LDI decomposition helps reduce the number of *complex-boundary-cells* in the octree. However, having a very high resolution may greatly increase the time and memory requested, which is impractical. Therefore, three filters are developed and consecutively applied to amend M_P before mesh generation.

- *Edge-Split* filter: For a boundary e in a leaf *boundary-cell* C, if no endpoint of e is in C, a new vertex $\mathbf{v}_m$ is inserted to split e at the middle point of e's part inside C. Meanwhile, the adjacent face of e is also split. Note that, the splitting is conducted after all new vertices have been introduced as one triangle may be split into more than two triangles (e.g., Fig.10(b)).
- *Node-Collapse* filter: For a leaf *boundary-cell* C, all boundary vertices of M_P in C are collapsed into one boundary vertex $\mathbf{v}_c$ — the relevant edges are merged and the faces are eliminated as well. See Fig.10(b) and (c) for the illustration. Implementation details can be found in [36].
- *Node-Position* filter: For a *complex-boundary-cell* C, the only boundary vertex $\mathbf{v}_c$ obtained after applying the *Node-Collapse* filter is moved to its *QEF-minimizing-point*.

Figure 10 illustrates the procedure of processing the boundary of M_P by these filters. After applying the *Edge-Split* filter and then the *Node-Collapse* filter on all leaf *boundary-cells* in the octree T_r, each boundary vertex of the remaining mesh surface M_P is in a different leaf *boundary-cell* in T_r. Although these filters do not fully convert the topology in the *complex-boundary-cells* to a simple one, they make the resultant mesh surface M_I stitch to M_P with a very limited number of holes. The number of holes can also be reduced by increasing the resolution of LDI sampling (e.g., Fig.11).

Similar to the cell refinement step, the mesh generation step can also be parallelized and run on multi-core CPUs. However, mesh generation is memory-access intensive rather than compute intensive. Therefore, the speed up by parallelization is not significant for mesh generation.

Since only one vertex is constructed in each cell, the complex topology inside a cell cannot be successfully reconstructed if the sampling rate is insufficient. In other words, the resultant model of our approach would not be homeomorphic to the correct result as ours only approximates the real shape and topology at the intersected regions. This however can be improved by increasing the sampling rate of the LDI.

C. Processing for two-manifold topology

To generate a two-manifold mesh surface as the result of Boolean operations, we need to carefully process the topology entities when reconstructing the mesh surface M_I for stitching. When creating a triangle f_{new} to link three vertices $\mathbf{v}_p$, $\mathbf{v}_q$ and $\mathbf{v}_r$, the existing topological entities around the vertices must be checked. For vertices $\mathbf{v}_p$ and $\mathbf{v}_q$, if there is already an edge $\mathbf{v}_p\mathbf{v}_q$ that has triangles on both its sides, we give up constructing the triangle f_{new}. The construction of f_{new} is also canceled when a similar case occurs on the edge $\mathbf{v}_q\mathbf{v}_r$ or $\mathbf{v}_r\mathbf{v}_p$. As a result, the resultant surface stitching to M_P is either a two-manifold surface or an open mesh surface with a few hanging vertices. The generation of hanging edges is prevented.

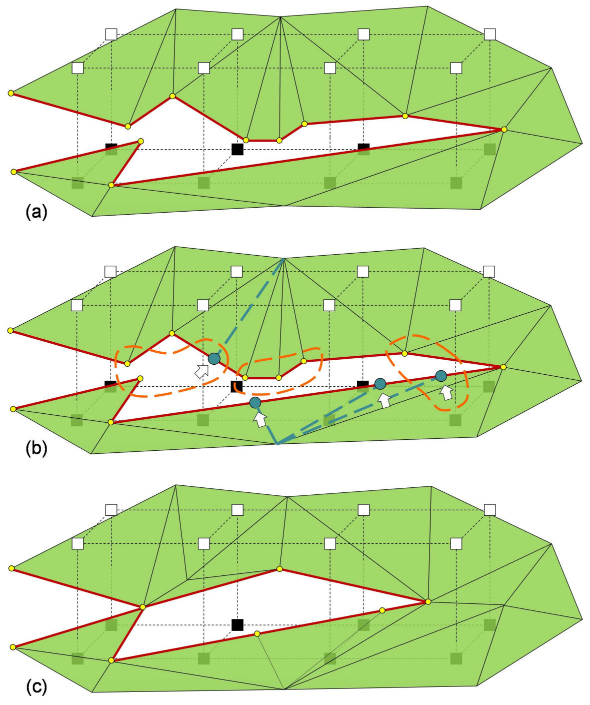

Fig. 10. Illustrations of filtering the boundary of the mesh surface in leaf *boundary-cells*. (a) The mesh surface M_P leads to *complex-boundary-cells*. (b) After applying the *Edge-Split* filter, new vertices and new edges are introduced (in cyan). (c) The result obtained after using the *Node-Collapse* filter, where the vertices in (b) circled by orange dash lines are collapsed into one vertex in the cell – the related triangles and edges are eliminated and merged as well. The hole will be filled by the quadrilateral faces constructed on the sign-changed minimal edges.

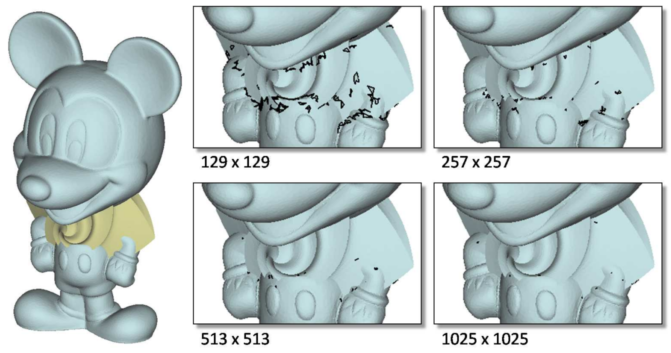

Fig. 11. Scattered small holes are generated on the result of our stitching algorithm. The boundaries of holes are displayed by bold black segments. With the increase in the LDI solids' resolution, the number of holes is reduced quickly.

The following *Node-Open* operator is applied to eliminate the hanging vertices on the boundary.

- *Node-Open* operator: For a boundary vertex $\mathbf{v}_s$ linked with $n(n > 2)$ boundary edges, $((n-2)/2)$ new vertices coincident to $\mathbf{v}_s$ are constructed, and the edges and faces linking to $\mathbf{v}_s$ are separated so that every vertex is linked with only two boundary edges (see Fig.12).

Now every boundary vertex on M_{res} has two boundary edges only. It is easy to walk along the boundary edges to link them into closed boundary loops (i.e., holes). Then, we apply the dynamic programming based triangulation technique in [6] to

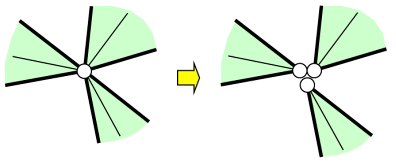

Fig. 12. The hanging vertices on the boundary can be eliminated by the *Node-Open* operator.

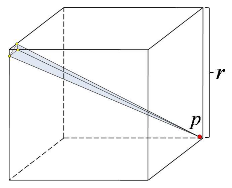

Fig. 13. The configuration with the longest distance from a surface point $\mathbf{p}$ to the samples — the yellow ones on the LDI inside a cell.

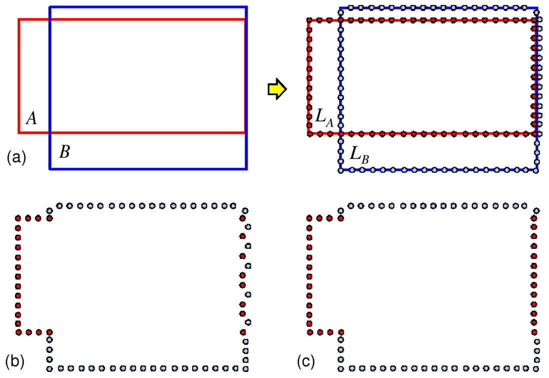

Fig. 14. Illustrations of the tangential contact: (a) two solids A and B and their corresponding LDI solids L_A and L_B, (b) the samples are randomly removed by the intrinsic moving forward algorithm for Boolean operations, and (c) all the samples on one solid are retained at the tangentially contacted region using the strategy of Remark 3.

fill every hole by an optimal set of triangles that minimize the total area of filled triangles. The resultant surface M_{res} is obtained.

IV. Discussion

A. Sampling

We have not analyzed whether L_{res} can provide enough samples for surface membership classification and reconstruction.

Definition The point set S sampled from the model M is defined as a d-covering of M if any point $\mathbf{p}$ on M can find a point $\mathbf{q} \in S$ that $\|\mathbf{p} - \mathbf{q}\| \leq d$.

Lemma A model M sampled into a structured set of LDIs gives a d-covering of M with d bounded by $\sqrt{3}r$, where r is the sampling distance in the LDI.

Proof. The rays from a structured set of LDIs actually form many cubic cells. The samples of LDIs are located at the edges of the cells. After analyzing the possible configurations of surface inside the cells, the configuration with the longest distance from a surface point $\mathbf{p}$ to the samples on the LDI is as shown in Fig.13. The distances from $\mathbf{p}$ to the intersections on the cell edges are $\sqrt{3}r$. Therefore, after sampling a given model M into LDIs with sampling distance r, the obtained point set S gives a d-covering of M with $d \leq \sqrt{3}r$.

$\diamond$

Obviously, $L_{res} = L_A \diamond L_B$ gives the same result as a LDI solid sampled from $M_{res} = M_A \diamond M_B$. By the above Lemma, we can conclude that: L_{res} is a d-covering of $M_{res} = M_A \diamond M_B$. Therefore, L_{res} provides enough samples for the surface membership classification and the subsequent surface reconstruction.

B. Robustness enhancement on tangential contact

The most difficult problem in a boundary evaluation based implementation of Boolean operations is how to handle the tangential contact. Here, it is solved by computing the Boolean operation on LDI solids instead of mesh surfaces — i.e., the tangentially contacted samples are merged (or eliminated) when computing L_{res}. On a ray, the 1D volumes (or gaps) whose thickness are smaller than ϵ are removed from the 1D volumes of the resultant LDI samples. $\epsilon = 10^{-5}$ is chosen in our implementation. This is because the depth values are encoded in single precision floats on graphics hardware, and $\epsilon = 10^{-5}$ is slightly larger than the smallest number that can be presented by single precision float (i.e., with 32 bits).

However, when merging is conducted intrinsically by the moving forward algorithm for Boolean operations, the tangentially contacted samples on L_A and L_B are removed randomly. This may lead the LDI based membership classification to mistakenly remove the contacted triangles. For example, as shown in Fig.14, when computing union of two models A and B, the intrinsic merging may cause some samples from A and some from B to be removed at the tangentially contacted region (see Fig.14(b)). In the worst case, there is no triangle at the contacted region with its corresponding samples all retained after the union operation — thus, no triangle is retained after membership classification. Although the surface can be reconstructed later in the reconstruction step, the unnecessary removal and reconstruction waste much computing time and may introduce unnecessary geometric errors. Such a mistaken removal of triangles can be prevented if the method below is adopted in Boolean operations.

Remark 3 When merging two tangentially contacted samples on two overlapped rays from L_A and L_B, the one with a greater (or smaller) value in its corresponding triangle ID is removed consistently.

C. About self-intersection removal

The algorithm proposed in this paper requires that the surfaces of two given models are closed non-self-intersected mesh surfaces. For objects that are incorrectly presented by mesh surface with small-holes, we fill the holes by the method in [6]. If the models are with big holes, they can be processed by the method in [47] or [9]. Although holes on mesh surfaces can be easily detected, the detection of self-intersections by the mesh representation is difficult. Nevertheless, they can

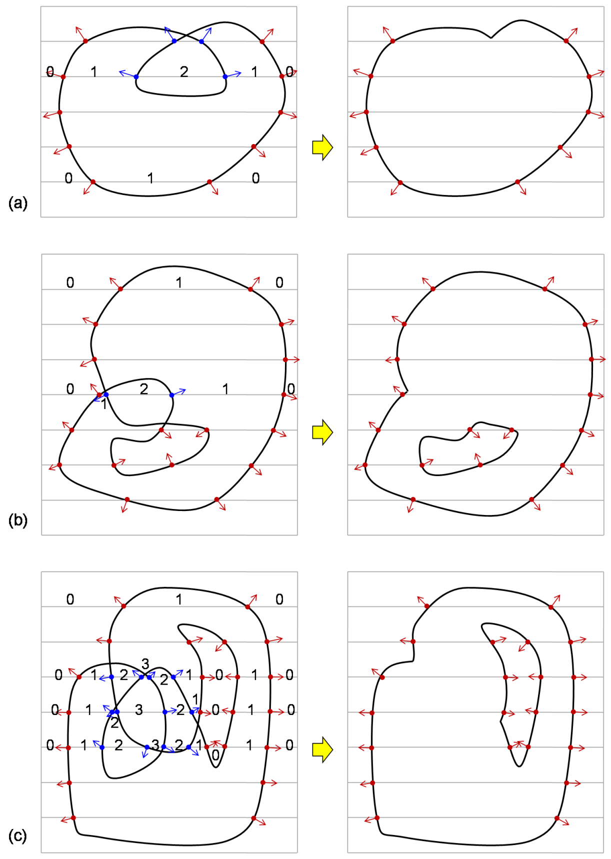

Fig. 15. The illustration of removing (a) a local self-intersection, (b) a global self-intersection and (c) a more complicated self-intersection, where the blue LDI samples are removed.

be detected and removed from the LDI solid by a counting algorithm.

For a ray, we assign each segment a *Normal Index*. Starting from zero, we increase the index by one when meeting a Hermite sample whose normal $\mathbf{n}_p$ is opposite to the ray's direction $\mathbf{n}_r$ (i.e., $\mathbf{n}_p \cdot \mathbf{n}_r < 0$); conversely, the index is decreased by one when meeting a sample that $\mathbf{n}_p \cdot \mathbf{n}_r > 0$. Note that according to the ray-casting based sampling method of LDI, no sample having its normal perpendicular to $\mathbf{n}_r$ can be obtained. After specifying the *Normal Index*, only the samples whose neighboring segments are labeled as $(0, 1)$ or $(1, 0)$ are retained. The formal proof of this algorithm has gone beyond the scope of this paper and can be found in [39]. Figure 15 shows the illustration of the algorithm. The blue ones in Fig.15 are those samples removed in this way to eliminate self-intersections on the input model.

V. Experimental Results

We have implemented the above algorithm in C++ plus OpenGL and tested various examples with a massive number of triangles (see Fig. 1, 16 and 17) on a consumer level PC with Intel Core 2 Quad CPU Q6600 2.4GHz + 4GB RAM and GeForce 8800 GT graphics card. The implementation of LDI sampling is based on the code of OpengGL in [38] which takes advantage of the excellent computational power provided by modern graphics hardware, and the construction of the octree is parallelized using OpenMP.

In order to compare the proposed algorithm with the state-of-the-art, we also implemented two other programs for Boolean operations. One calls the API functions provided by the commercial software package ACIS R15 [1], and the other employs the Boolean operation functions on 3D selective Nef Complexes in the newest version of CGAL library [16]. The comparisons of computing time are listed in Table I. It can be found that our method works well on the freeform models with a massive number of triangles (e.g., the models in Fig.2, 16 and 17), which cannot be computed by ACIS and CGAL. Moreover, it is surprising that, although the exact arithmetic is conducted, ACIS fails in the two tangential contact examples from the jewelry industry while ours works well in them (see Fig.16(e) and (f)). CGAL can work out the example (though very slowly) with two rings in Fig.16(f) but fails in the example of ring and bars in Fig.16(e). We also test the models on the commercial software Rhinoceros [65] that claims to offer robust and very fast Boolean operations. It fails in several example models tested here (see Table I). In the failed examples, sometimes the program stops the computation after several seconds and does not modify the input models. In some examples, incorrect results are generated. Figure 18 shows the examples with incorrect output from Rhinoceros. For those examples where correct models can be generated by Rhinoceros, the speed of Rhinoceros is much slower than ours (see also the statistics shown in Table I).

Successfully computing the examples shown in Fig.2, 16 and 17 in a few seconds has verified the efficiency and the robustness of our algorithm presented in this paper. Another interesting phenomenon is that, when increasing the resolution of LDI sampling, most of the additional time is spent on the sampling part. Specifically, when the resolution is increased to two or four times, the time cost of our approach rises much more slowly than that of the increase in resolution.

As our proposed algorithm computes the approximate Boolean operations, the surface errors are measured using the publicly available Metro tool [18] by comparing with the exact Boolean operation's result obtained from CGAL. From Table II, it is not difficult to conclude that our method generates accurate models, and the accuracy converges while increasing the sampling rate.

Why do we not simply reconstruct the resultant mesh surface from L_{res}? This is because the full reconstruction from L_{res} often damages the geometric details in non-intersected regions and also takes a longer computational time. To prove this, some tests are conducted to fully reconstruct mesh surfaces from L_{res} (i.e., let $M_P = \phi$ and reconstruct M_I from all samples in L_{res}). In these tests, we choose the moderate resolution — 513×513 LDI solids. The statistics of both the computing time and the shape error are listed in Table III, which verify our analysis above.

Limitations The major limitation of our algorithm is that self-intersection may happen on the resultant mesh because the mesh generation method is akin to dual-contouring. More details can be found in [48], where the authors have given an

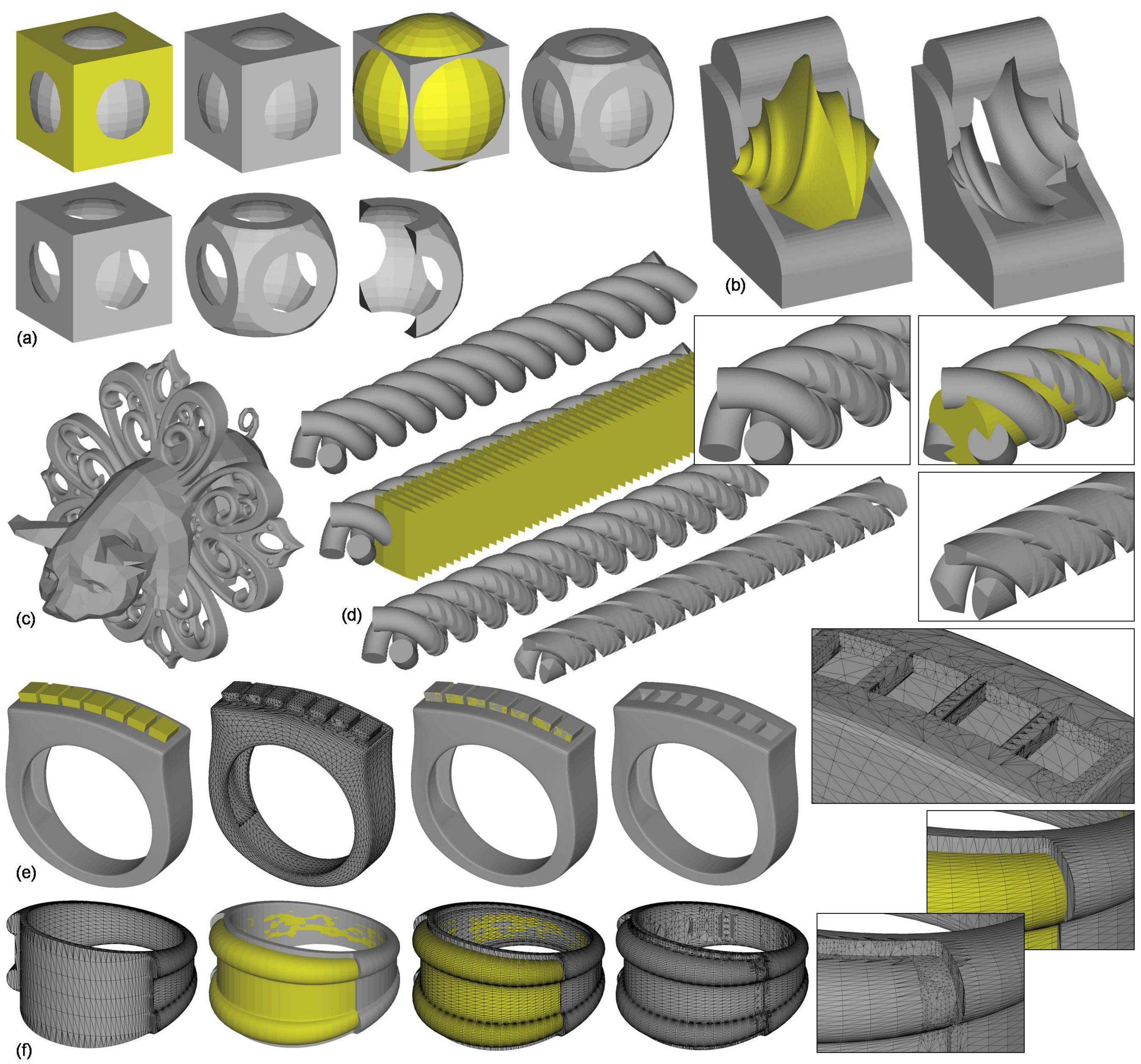

Fig. 16. Examples of our fast approximate Boolean operation algorithm on various models: (a) ((Cube ∪ Sphere) ∩ Sphere2) and ((Cube \ Sphere) ∩ Sphere2), (b) (Chair \ Octa-Flower), (c) (Pig ∪ Filigree), (d) (Helix \ Multi-slices ∩ Cylinder), (e) (Ring ∪ Bars \ Bars) – the "\" operation needs to handle tangential contact, and (f) the union of tangentially contacted Ring-A and Ring-B.

TABLE III
STATISTICS OF FULL RECONSTRUCTION FROM L_{res} AT THE MODERATE RESOLUTION — 513 × 513

Example	$E_{mean}(\%)$*	Computing Time
Chair − Octa-Flower	9.30×10^{-4}	2.90 (sec.)
Helix − Slices	2.21×10^{-3}	4.29 (sec.)
(Helix − Slices) ∩ Cylinder	3.19×10^{-3}	4.15 (sec.)
Ring-A ∪ Ring-B	1.67×10^{-4}	3.59 (sec.)

* The errors are reported in percentage with reference to the diagonal lengths of the models' bounding boxes.

improved contouring algorithm. We will further study it to see how their technique can be integrated into our algorithm. In our current implementation, we simply employ the above self-intersection elimination method to process input models. The second limitation of our algorithm is that complex topology inside the finest resolution of a cell is collapsed, which may miss features whose sizes are smaller than that of the finest cell. However, this can be avoided if the sampling rate of the Layered Depth Images is assigned to bound the d-covering (see Lemma 1). An alternative is to use the method presented in [3] to predict the topology inside the finest cell and then adjust the method to generate polygons.

VI. CONCLUSION

In this paper, we have presented an approximate Boolean operation algorithm using Layered Depth Images (LDI) for freeform solids, which are bounded by mesh surfaces with a massive number of triangles. The major parts of our algorithm are the sampling based membership classification using LDI, and the trimmed adaptive contouring to reconstruct the mesh

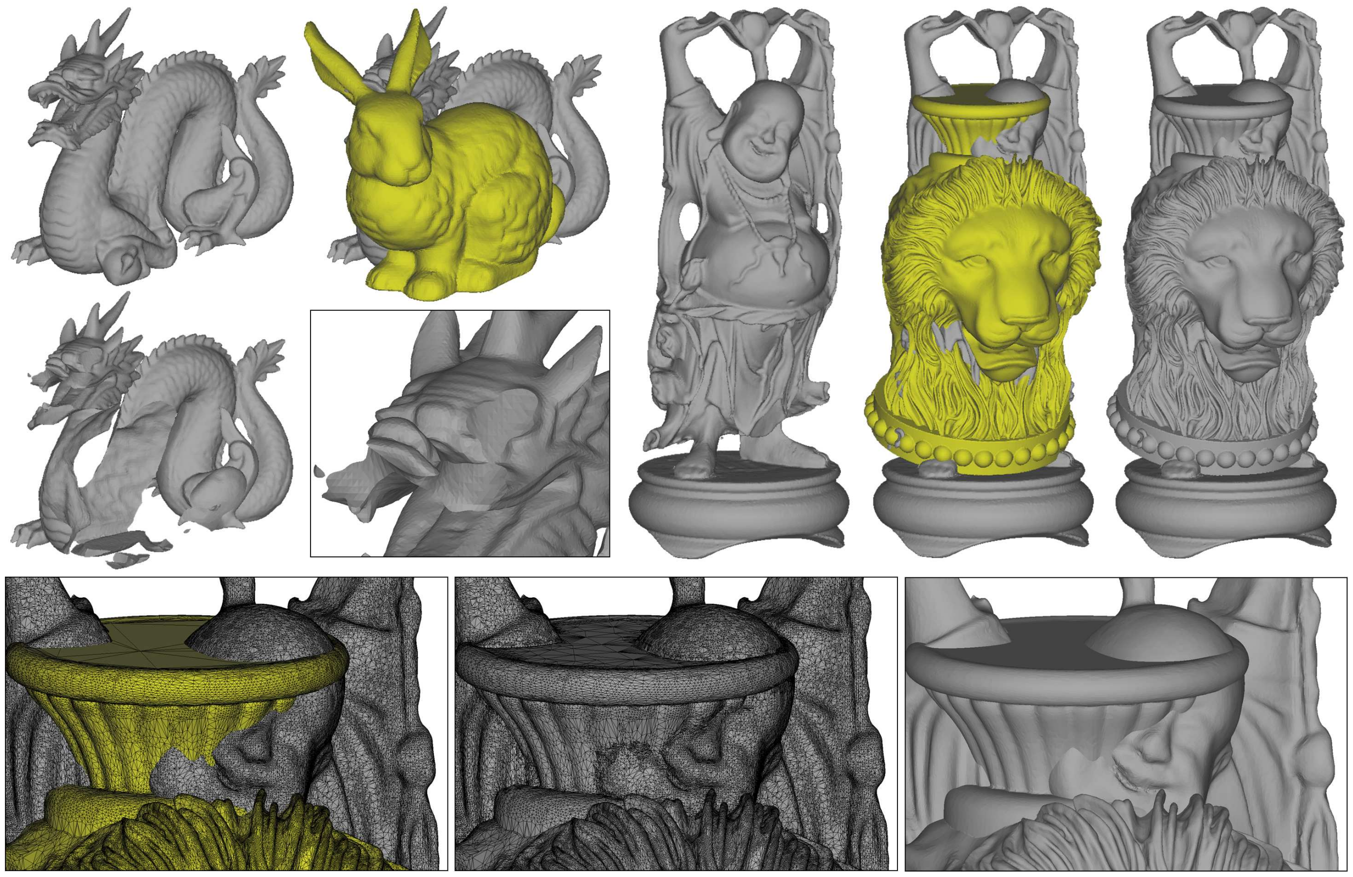
Fig. 17. Boolean operation examples on freeform solids with a massive number of faces: (Dragon $\setminus$ Bunny) and (Buddha $\cup$ Vase-Lion) — triangles at the non-intersected regions are NOT modified.

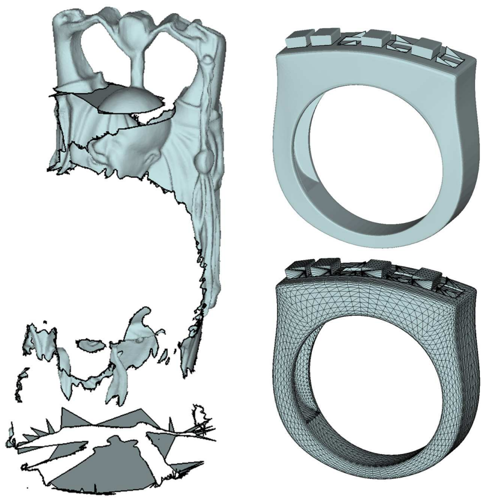
Fig. 18. Incorrect results are generated by Rhinoceros [65].

surface in the intersected regions from the LDI. The advantage in numerical robustness of other approaches using volumetric representations is inherited in our algorithm; however, unlike other volumetric representation based methods, we do not damage the facets in non-intersected regions, thus preserving geometric details and also speeding up the computation. The efficiency and robustness of our algorithm have been verified by various example models with complex geometry and topology.

Acknowledgment

This work was supported by the Hong Kong RGC/CERG Grant (CUHK/417508) and the CUHK Direct Research Grant (CUHK/2050400). Mr. Yunbo Zhang and Mr. Pu Huang from CUHK helped to implement the ACIS and CGAL based Boolean operation program. We deeply thank the valuable comments from the anonymous reviewers which help to improve this paper a lot. The author would also like to thank Dr. Yu Wang (HKUST), Dr. Yong Chen (USC) and the AIM@SHAPE Shape Repository for sharing some models used in this paper, and Ms. Siu Ping Mok for proof reading the manuscript.

References

[1] 3D ACIS Modeling, http://www.spatial.com, 2008.
[2] B. Adams and P. Dutré, “Interactive boolean operations on surfel-bounded solids,” *ACM Trans. on Graphics*, vol.22, no.3, pp.651-656, 2003.
[3] C. Andújar, P. Brunet, A. Chica, I. Navazo, J. Rossignac, and A. Vinacua, “Optimizing the topological and combinatorial complexity of isosurfaces,” *Computer-Aided Design*, vol.37, no.8, pp.847-857, 2005.
[4] R. Banerjee, V. Goel, and A. Mukherjee, “Efficient parallel evaluation of CSG tree using fixed number of processors,” *Proc. the second ACM Symposium on Solid Modeling and Applications*, pp.137-146, 1993.
[5] R. Banerjee and J. Rossignac, “Topologically exact evaluation of polyhedral defined in CSG with loose primitives,” *Computer Graphics Forum*, vol.15, no.4, pp.205-217, 1996.

TABLE I
COMPUTATIONAL STATISTICS IN TIME (SECOND)

Example	Figure	Face Num.		ACIS	CGAL	Rhinoceros	Our Approach with Diff. Res.†		
		First	Second				257×257	513×513	1025×1025
Vase-Lion ∪ Dragon	2	400k	277k	n/a*	n/a	∼71	4.35 (0.967)	5.26 (1.75)	8.33 (4.04)
Chair − Octa-Flower	16(b)	464	15.8k	1.72	7.82	∼2	0.624 (0.250)	1.51 (0.577)	4.68 (1.59)
Pig ∪ Filigree	16(c)	1,208	260k	n/a	n/a	fail (∼7)	2.23 (0.733)	3.00 (1.31)	5.63 (2.95)
(Helix − Slices) ∩ Cyn.	16(d)	74k	920	19.9	77.5	∼47	1.36 (0.593)	2.56 (0.811)	5.16 (1.48)
			180	39.9	56.9	∼58	1.34 (0.515)	2.45 (0.826)	5.82 (2.03)
Ring ∪ Bars − Bars	16(e)	10.4k	4,172	0.56	n/a	fail (∼1)	0.405 (0.203)	0.811 (0.390)	2.23 (1.01)
			4,172	n/a	n/a		0.499 (0.234)	1.17 (0.405)	3.12 (1.19)
Ring-A ∪ Ring-B	16(f)	12.3k	14.3k	n/a	35.0	fail (∼43)	0.608 (0.234)	1.37 (0.531)	4.09 (1.48)
Dragon − Bunny	17	277k	69.6k	n/a	n/a	∼25	2.47 (0.593)	3.37 (1.11)	6.26 (2.71)
Buddha ∪ Vase-Lion	17	871k	400k	n/a	n/a	fail (∼597)	9.00 (1.68)	10.1 (2.59)	15.2 (6.61)

* n/a denotes that the API function reports fail during Boolean operations;
† In bracket is the time used for sampling and reading back from the graphics hardware.

TABLE II
SHAPE ERROR REPORTED BY THE METRO TOOL [18]

Example	Res.: 257×257		Res.: 513×513		Res.: 1025×1025	
	$E_{mean}(\%)$	$E_{max}(\%)$	$E_{mean}(\%)$	$E_{max}(\%)$	$E_{mean}(\%)$	$E_{max}(\%)$
Chair − Octa-Flower	1.03×10^{-3}	1.43	7.66×10^{-4}	0.783	2.89×10^{-4}	0.377
Helix − Slices	3.14×10^{-3}	0.842	1.08×10^{-3}	0.406	4.12×10^{-4}	0.187
(Helix − Slices) ∩ Cylinder	5.67×10^{-3}	0.795	1.88×10^{-3}	0.414	9.95×10^{-4}	0.180
Ring-A ∪ Ring-B	8.22×10^{-4}	0.0496	5.07×10^{-4}	0.966	2.31×10^{-4}	0.0365

* The errors are reported in percentage with reference to the diagonal lengths of the models' bounding boxes.

[6] G. Barequet and M. Sharir, "Filling gaps in the boundary of a polyhedron," *Computer Aided Geometric Design*, vol.12, pp.207-229, 1995.

[7] M.O. Benouamer and D. Michelucci, "Bridging the gap between CSG and Brep via a triple ray representation," *Proc. the Fourth ACM Symposium on Solid Modeling and Applications*, pp.68-79, 1997.

[8] H. Biermann, D. Kristjansson, and D. Zorin, "Approximate Boolean operations on free-form solids," *Proc. ACM SIGGRAPH 2001*, pp.185-194, 2001.

[9] S. Bischoff, D. Pavic, and L. Kobbelt, "Automatic restoration of polygon models," *ACM Trans. on Graphics*, vol.24, no.4, pp.1332-1352, 2005.

[10] S. Bischoff and L. Kobbelt, "Structure preserving CAD model repair", *Computer Graphics Forum* (Eurographics 2005 proceedings), vo.24, no.3, pp.527-536, 2005.

[11] R. Böonning and H. Müuller, "Interactive sculpturing and visualization of unbounded voxel volumes," *Proc. of the seventh ACM symposium on Solid modeling and applications*, pp.212 - 219, 2002.

[12] P. Brunet and I Navazo, "Geometric modelling using exact octree representation of polyhedral objects," *Proc. of Eurographics 85*, pp.159-169, Nice, September 9-13, 1985.

[13] P. Brunet and I Navazo, "Solid representation and operation using extended Octrees," *ACM Trans. on Graphics*, vol.9, no.2, pp.170-197.

[14] I. Carlbom, I. Chakravarty, and D.A. Vanderschel, "A heirarchical data structure for representing the spatial decomposition of 3D objects," *IEEE Computer Graphics and Application*, vol.5, no.4, pp.24-31, 1985.

[15] I. Carlbom, "An algorithm for geometric set operations using cellular subdivision techniques," *IEEE Computer Graphics and Application*, vol.7, no.5, pp.44-55, 1987.

[16] CGAL, http://www.cgal.org, 2008.

[17] H. Chen and S. Fang, "A volumetic approach to interactive CSG modeling and rendering," *Proc. the fifth ACM Symposium on Solid Modeling and Applications*, pp.318-319, 1999.

[18] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," *Computer Graphics Forum*, vol.17, no.2, pp.167-174, 1998.

[19] J. Du, B. Fix, J. Glimm, X. Jia, X. Li, Y. Li, and L. Wu, "A simple package for front tracking ", *Journal of Computational Physics*, vol.213, no.2, pp.613-628, April 2006.

[20] E. Eisemann and X. Décoret, "Single-pass GPU solid voxelization for real-time applications," *Proc. of Graphics Interface 2008*, pp.73-80, 2008.

[21] J.L. Ellis, G. Kedem, T.C. Lyerly, D.G. Thielman, R.J. Marisa, J.P. Menon, and H.B. Voelcker, "The ray casting engine and ray representatives," *Proc. ACM Symposium on Solid Modeling and Applications 1991*, pp.255-267, 1991.

[22] S. Fang, B.D. Brüderlin, and X. Zhu, "Robustness in solid modelling: a tolerance-based intuitionistic approach," *Computer-Aided Design*, vol.25, no.9, pp.567-576, 1993.

[23] F. Faure, S. Barbier, J. Allard, and F. Falipou, "Image-based collision detection and response between arbitrary volumetric objects," *Proc. Eurographics/ACM Siggraph Symposium on Computer Animation*, 2008.

[24] S. Fortune and C.J. van Wyk, "Efficient exact arithmetic for computational geometry," *Proc. 9th ACM Symposium on Computational Geometry*, pp.163-172, 1993.

[25] S. Fortune, "Polyhedral modelling with exact arithmetic," *Proc. 3rd ACM Symposium on Solid Modeling*, pp.225-234, 1995.

[26] S.F. Frisken, R.N. Perry, A.P. Rockwood, and T.R. Jones, "Adaptively sampled distance fields: a general representation of shape for computer graphics," *Proc. ACM SIGGRAPH 2000*, pp.249-254, 2000.

[27] M. Goodrich, "An improved ray shooting method for constructive solid geometry models via tree contraction," *International Journal of Computational Geometry and Applications*, vol.8, no.1, pp.1-24, 1998.

[28] N.K. Govindaraju, M.C. Lin, D. Manocha, "Fast and reliable collision culling using graphics hardware," *IEEE Trans. Visualization and Computer Graphics*, vol.12, no.2, pp.143-154, 2006.

[29] S. Guha, S. Krishnan, K. Munagala, and S. Venkatasubramanian, "Application of the two-sided depth test to CSG rendering," *Proc. 2003 symposium on Interactive 3D graphics*, pp.177-180, 2003.

[30] A. Guéziec, G. Taubin, F. Lazarus, and B. Horn, "Cutting and stitching: converting set of polygons to manifold surfaces," *IEEE Trans. Visualization and Computer Graphics*, vol.7, no.2, pp.136-151.

[31] J. Hable and J. Rossignac, "CST: Constructive Solid Trimming for Rendering BReps and CSG," *IEEE Trans. Visualization and Computer Graphics*, vol.13, no.5, pp.1004-1014, 2007.

[32] J. Hable and J. Rossignac, "Constructive solid trimming," *In ACM SIGGRAPH 2006 Sketches*, 2006.

[33] J. Hable and J. Rossignac, "Blister: Gpu-based rendering of boolean combinations of free-form triangulated shapes," *ACM Trans. Graphics*, vol.24, no.3, pp.1024-1031, 2005.

[34] P. Hachenberger and L. Kettner, "Boolean operations on 3D selective nef complexes: optimized implementation and experiments," *Proc. ACM*

Symposium on Solid and Physical Modeling (SPM 2005), pp.163-174, 2005.

[35] P. Hachenberger, L. Kettner, and K. Mehlhorn, "Boolean operations on 3D selective Nef complexes: Data structure, algorithms, optimized implementation and experiments," *Computational Geometry: Theory and Applications,* vol.38, no.1-2, pp.64-99, 2007.

[36] B. Hamann, "A data reduction scheme for triangulated surfaces," *Computer Aided Geometric Design,* vol.11, pp.197-214, 1994.

[37] E.E. Hartquist, J.P. Menon, K. Suresh, H.B. Voelcker, J. Zagajac, "A computing strategy for applications involving offsets, sweeps, and Minkowski operations," *Computer-Aided Design*, vol.31, no.3, pp.175-183, 1999.

[38] B. Heidelberger, M. Teschner, and M. Gross, "Real-time volumetric intersections of deforming objects," *Proc. Vision, Modeling, and Visualization 2003,* pp.461-468, Munich, Germany, November 19-21, 2003.

[39] J.A. Heisserman, Generative Geometric Design and Boundary Solid Grammars, Ch5, PhD Dissertation, Carnegie Mellon University, 1991.

[40] C. Hoffmann, *Geometric and Solid Modeling: An Introduction*, Morgan Kauffmann, 1989.

[41] C. Hoffmann, J. Hopcroft, and M. Karasik, "Robust set operations on polyhedral solids," *IEEE Computer Graphics and Applications*, vol.9, no.6, pp.50-59, 1989.

[42] C. Hoffmann, "Robustness in geometric computations," *ASME Journal of Computing and Information Science in Eng.*, vol.1, pp.143-156, 2001.

[43] T. Van Hook, "Real-time shaded NC milling display," *ACM SIGGRAPH Computer Graphics*, vol.20, no.4, pp.15-20, Aug. 1986.

[44] C.-Y. Hu, N.M. Patrikalakis, and X. Ye, "Robust interval solid modelling - Part I: representation," *Computer-Aided Design,* vol.28, no.10, pp.807-817, 1996.

[45] C.-Y. Hu, N.M. Patrikalakis, and X. Ye, "Robust interval solid modelling - Part II: boundary evaluation," *Computer-Aided Design*, vol.28, no.10, pp.819-830, 1996.

[46] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," *ACM Trans. on Graphics*, vol.21, no.3, pp.339-346, 2002.

[47] T. Ju, "Robust repair of polygonal models," *ACM Trans. on Graphics*, vol.23, no.3, pp.888-895, 2004.

[48] T. Ju and T. Udeshi, "Intersection-free contouring on an octree grid," *Proc. Pacific Graphics*, 2006.

[49] M. Kelley, K. Gould, B. Pease, S. Winner, and A. Yen, "Hardware accelerated rendering of CSG and transparency," *Proc. SIGGRAPH 1994*, pp.177-184, 1994.

[50] J. Keyser, S. Krishnan, and D. Manocha, "Efficient and accurate B-rep generation of low degree sculptured solids using exact arithmetic: I - Representations," *Computer Aided Geometric Design,* vol.16, no.9, pp.841-859, 1999.

[51] J. Keyser, S. Krishnan, and D. Manocha, "Efficient and accurate B-rep generation of low degree sculptured solids using exact arithmetic: II - Computation," *Computer Aided Geometric Design,* vol.16, no.9, pp.861-882, 1999.

[52] J. Keyser, T. Culver, M. Foskey, S. Krishnan, and D. Manocha, "ESOLID: A system for exact boundary evaluation," *Computer Aided Design,* vol. 36, no. 2, pp. 175-193, 2004.

[53] H.S. Kim, H.K. Choi, and K.H. Lee, "Feature detection of triangular meshes based on tensor voting theory", *Computer-Aided Design,* vol.41, pp.47-58, 2009.

[54] L.P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, "Feature sensitive surface extraction from volume data," *Proc. ACM SIGGRAPH 2001,* pp.57-66, 2001.

[55] M. Mäntylä, "Boolean operations of 2-manifolds through vertex neighborhood classification," *ACM Trans. on Graphics*, vol.5, no.1, pp.1-29, 1986.

[56] J. Menon, R.J. Marisa, and J. Zagajac, "More Powerful Solid Modeling Through Ray Representations," *IEEE Computer Graphics and Applications,* vol.14, no.3, pp.22-35, May 1994.

[57] J.P. Menon and H.B. Voelcker, "On the completeness and conversion of ray representations of arbitrary solids," *Proc. ACM Symposium on Solid Modeling and Applications 1995,* pp.175-286, 1995.

[58] H. Muller, T. Surmann, M. Stautner, F. Albersmann, K. Weinert, "Online sculpting and visualization of multi-dexel volumes," *Proc. the eighth ACM symposium on Solid Modeling and Applications,* pp.258-261, 2003.

[59] K. Museth, D.E. Breen, R.T. Whitaker, A.H. Barr, "Level set surface editing operators," *ACM Trans. on Graphics,* vol.21, no.3, pp.330-338, July 2002.

[60] F.S. Nooruddin and G. Turk, "Interior/exterior classification of polygonal models," *Proc. IEEE Visualization 2000*, pp.415-422, 2000.

[61] D. Pavic, M. Campen, and L. Kobbelt, "Hybrid Booleans," *Computer Graphics Forum,* to appear, 2010.

[62] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: surface elements as rendering primitives," *Proc. SIGGRAPH 2000*, pp.335-342, 2000.

[63] A.A.G. Requicha and H.B. Voelcker, "Solid modeling: A historical summary and contemporary assessment," *IEEE Computer Graphics and Applications*, vol.2, no.2, pp.9-24, March 1982.

[64] A.A.G. Requicha and H.B. Voelcker, "Boolean operations in solid modelling: Boundary evaluation and merging algorithms," *Proc. IEEE*, vol.73, no.1, pp.30-44, January 1985.

[65] Rhinoceros, ver 4.0 Evaluation, http://www.rhino3d.com, 2009.

[66] F. Romeiro, L. Velho, and L.H. de Figueiredo, "Scalable GPU rendering of CSG models," *Computers and Graphics,* vol.32 no.5, pp.526-539, October, 2008.

[67] J. Rossignac and J. Wu, "Correct Shading of Regularized CSG solids using a Depth-lnterval Buffer," *Advances in Computer Graphics Hardware V,* pp.117-138, Sprinter-Verlag, Berlin, 1990.

[68] J.R. Rossignac, "Solid and physical modeling," *Technical Report*, 2007.

[69] P. Santos, R. de Toledo, and M. Gattass, "Solid height-map sets: modeling and visualization," Proc. ACM symposium on Solid and Physical Modeling 2008, pp.359-365, 2008.

[70] S. Schaefer and J. Warren, "Dual contouring: the secret sauce," *Rice University Technical Report,* 2002.

[71] S. Schaefer, T. Ju, and J. Warren, "Manifold dual contouring," *IEEE Trans. Visualization and Computer Graphics,* vol.13, no.3, pp.610-619, 2007.

[72] M. Segal, "Using tolerance to guarantee valid polyhedral modeling results," *ACM SIGGRAPH Computer Graphics*, vol.24, no.4, pp.105-114, 1990.

[73] J.M. Smith and N.A. Dodgson, "A topologically robust algorithm for Boolean operations on polyhedral shapes using approximate arithmetic," *Computer-Aided Design*, vol.39, pp.149-163, 2007.

[74] N. Stewart, G. Leach, and S. John, "An improved z-buffer CSG rendering algorithm," *Proc. ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pp.25-30, Lisbon, Portugal, 1998.

[75] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino, "Collision detection for deformable objects," *Computer Graphics Forum*, vol.24, no.1, pp.61-81, 2005.

[76] M. Trapp and J. Döllner, "Real-time volumetric tests using Layered Depth Images," *Proc. of Eurographics 2008,* pp.235-238, 2008.

[77] G. Varadhan, S. Krishnan, Y.J. Kim, and D. Manocha, "Feature-sensitive subdivision and isosurface reconstruction," *Proc. IEEE Visualization 2003,* pp.99-106, 2003.

[78] G. Varadhan, S. Krishnan, T.V.N. Sriram, and D. Manocha, "Topology preserving surface extraction using adaptive subdivision," *Proc. 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pp.235-244, 2004.

[79] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Deforming meshes that split and merge", *ACM Transactions on Graphics*, vol.28, no.3, Article 76, 10 pages, August 2009.

[80] N. Zhang, W. Hong, and A. Kaufman, "Dual contouring with topology-preserving simplification using enhanced cell representation," *Proc. IEEE Visualization 2004,* pp.505-512, 2004.

PLACE PHOTO HERE

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.