Boolean Operations for Free-form Models Represented in Geometry Images

Yan Fu and Bingfeng Zhou† ‡

State Key Lab of Text Information Processing
Institute of Computer Science & Technology, Peking University, Beijing, China

Abstract

We present a Boolean operation algorithm for free-form solid models represented in geometry images. By taking advantage of the regular data organization of geometry images, our algorithm can perform efficient surface division using boundary-fill algorithm which is previously used for digital image processing. A quadtree subdivision scheme is also applied to the geometry images to accelerate the intersection line calculation. Experimental result shows that the algorithm can generate well-defined closed triangle meshes for Boolean operations. The resulted triangle mesh can also be converted into a geometry image for further processing.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Boolean operation of solid models is a key algorithm for geometry modeling. Many efforts have been made to develop Boolean operation algorithms oriented to different ways of model representations, which include level-set surfaces [KDRA02], multi-resolution meshes [BKZ01] and point clouds [AD03]. Some methods even support Boolean operations on hybrid of implicit models and explicit ones [FGF05]. Though Boolean operations on volumetric representation [FL00] is typically more stable than those on surface-based manner, the result objects have no continuous geometric representation.

Geometry image is a novel boundary representation proposed by Gu et al. [GGH02]. It represents a surface as a two-dimensional matrix, which can be stored in the format of image. The "color" at a pixel of the image represents the geometric information or other properties of the surface. For free-from models, the representation of geometry image avoids the storage of connectivity information since they have been implied in the regular grid of geometry image. Due to the regularity of the data organization of geometry image, many techniques previously developed for digital images can be applied to geometry images [NYC05]. Moreover, the processing of geometry images can be accelerated by hardware [LHSW03].

In this paper, a Boolean operation algorithm for two geometry images is described. We take advantage of the regularity of the data organization of geometry image to increase the efficiency of Boolean operations. The two input geometry images are organized in the structure of quadtree to accelerate the intersection test. Afterwards, the intersection lines are mapped onto the parameter domain, which define connected close boundaries in geometry images. As these boundaries are closed, algorithms such as boundary-fill used in the digital image processing can be employed to classify the inside / outside regions over the surface. When the regions are classified, the result of the Boolean operation can be obtained by triangulating and gluing the retained regions, and the results can be converted into geometry images for further processing.

2. Problem Definition and Algorithm Overview

Definitions and Notations Given a triangle mesh $\mathcal{M}$ as a set of triangles, the geometry image $\mathbf{G}$ of $\mathcal{M}$ can be written in the form of a matrix $\mathbf{G} = (\mathbf{g}_{ij})_{m\times m}$, where

$$\mathbf{g}_{ij} = \mathbf{p}^{-1}(s_i, t_j), \quad s_i = \frac{i-1}{m-1}, t_j = \frac{j-1}{m-1} \tag{1}$$

† Corresponding author. E-mail: cczbf@pku.edu.cn

‡ This work is supported in part by NSFC(No. 60573149) and NSF-Beijing(No. 4072013).

and $\mathbf{p}(\cdot)$ is the conformal parameterization [FH05] from $\mathcal{M}$ to 2D parameter domain $\mathcal{D}$. As the elements of $\mathbf{G}$ correspond to a set of points lining on the orthogonal grids in $\mathcal{D}$, $\mathbf{G}$ corresponds to a 2D mesh $\mathcal{G}$ in $\mathcal{D}$:

$$\mathcal{G} = \{\mathbf{t}_k \mid k = 1,\ldots,2(m-1)(m-1)\}, \qquad (2)$$

where $\mathbf{t}_k$ are triangles that are generated from grids in $\mathcal{D}$ by creating two triangles $\mathbf{t}_{k_1}, \mathbf{t}_{k_2}$ for each grid cell. Let $\mathbf{d}_{i,j} = (s_i, t_j)$ be a point from the grids, then

$$\mathbf{t}_{k_1} = (\mathbf{d}_{i,j}, \mathbf{d}_{i+1,j}, \mathbf{d}_{i+1,j+1}), \mathbf{t}_{k_2} = (\mathbf{d}_{i,j}, \mathbf{d}_{i,j+1}, \mathbf{d}_{i+1,j+1}).$$

When the triangle vertices are mapped back to the 3D space using $\mathbf{p}^{-1}(\cdot)$, the transformed triangles $\{\mathbf{T}_k\}$ form a closed 3D triangle mesh $\mathcal{G}^{-1}$ whose vertices are on the surface of $\mathcal{M}$. We call $\mathcal{G}^{-1}$ *reconstructed mesh* :

$$\mathcal{G}^{-1} = \{\mathbf{T}_k \mid k = 1,\ldots,2(m-1)(m-1)\}, \qquad (3)$$

where

$$\mathbf{T}_{k_1} = \left(\mathbf{g}_{i,j}, \mathbf{g}_{i+1,j}, \mathbf{g}_{i+1,j+1}\right), \mathbf{T}_{k_2} = \left(\mathbf{g}_{i,j}, \mathbf{g}_{i,j+1}, \mathbf{g}_{i+1,j+1}\right).$$

Algorithm Overview For two 3D solid models A and B, their geometry images are denoted as $\mathbf{G}_A$ and $\mathbf{G}_B$ respectively, which correspond to two 2D meshes $\mathcal{G}_A$ and $\mathcal{G}_B$ and two *reconstructed meshes* $\mathcal{G}_A^{-1}$ and $\mathcal{G}_B^{-1}$ respectively.

With the notation given above, our Boolean operation algorithm can be presented as the following steps:

Step 1: Intersection. The algorithm begins with the calculation of intersection lines between $\mathcal{G}_A^{-1}$ and $\mathcal{G}_B^{-1}$. Then the intersection lines are mapped into $\mathcal{G}_A$ and $\mathcal{G}_B$ in 2D domain (Figure 1 (d, g)).

Step 2: Pixel classification. The intersection lines form several closed paths over the parameter domain, these paths divide the pixels of geometry image into subsets (Figure 1 (e, h)). The elements of each individual subset are either inside or outside the other reconstructed mesh. After deciding the inside / outside property of the elements in geometry images, the elements that will be used as the vertices of the resulted triangle mesh can be determined based on the type of the Boolean operation being performed.

Step 3: Triangulation. When pixels of a geometry image to be kept in the result are determined, they can be triangulated using a constrained Delaunay triangulation, where the constrained edges fed into the triangulation algorithm are the intersection lines mapped into the parameter domain (Figure 1 (f, i)). When the triangulations are performed for both geometry images, the set of triangles obtained can be merged to get the resulted triangle mesh(Figure 1 (b, c)).

3. Calculation of Intersection Lines

In our algorithm, the B-reps models are geometry images, but as a matter of fact they represent triangle meshes. Therefore the intersection line calculation is confined to the intersection of triangles. In our algorithm, the reconstructed meshes $\mathcal{G}_A^{-1}$ and $\mathcal{G}_B^{-1}$ are used to perform intersection test

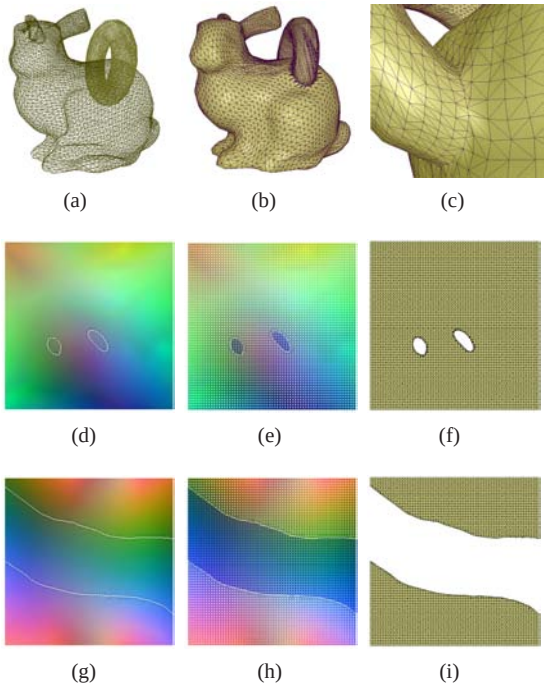


Figure 1: *Procedure of Boolean operations on Geometry Images (See Section 2 for detail).*

[Mol97]. For a pair of triangles from $\mathcal{G}_A^{-1}$ and $\mathcal{G}_B^{-1}$ respectively, there exist two triangle edges that intersect with the other triangle (See Figure 2 (b, c). It is also possible that the intersection points lie on the edges from the same triangle.). We name the triangle edges that intersect with a triangle from the other reconstructed mesh as *boundary grid line*s and denote the set of *boundary grid lines* over the surface of $\mathcal{G}_A^{-1}$ and $\mathcal{G}_B^{-1}$ as $\mathbf{L}_A$ and $\mathbf{L}_B$ respectively. The end points of the boundary grid lines are called *boundary pixels*, which form the boundary for the boundary-fill algorithm to classify the surface of models. The intersection lines obtained in 3D space are mapped into both $\mathcal{G}_A$ and $\mathcal{G}_B$ and they form one or more closed paths in the parameter domain which are used in the triangulation step to construct the result mesh of Boolean operation.

In our implementation, we apply a quadtree scheme to localize and accelerate the triangle intersection. Since each geometry image represents a 3D model in a regular structure, the construction of quadtree is quite straightforward. For a $m \times m$ geometry image, the depth of the quadtree is at most $\lceil \log_2 m \rceil$.

4. Classification of Pixels in Geometry Image

In this section, we shall use a boundary-fill scheme to decide the inside / outside property of each pixel in the geometry image. The boundary is formed by the *boundary pixels* of all the boundary grid lines. Here, we take $\mathbf{G_A}$ as an example.

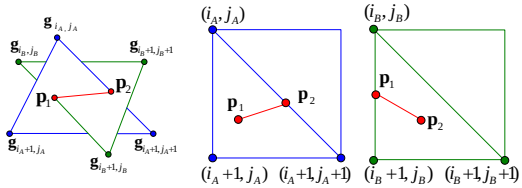


Figure 2: *An example case of intersections of two triangles from $\mathcal{G}_A^{-1}$ and $\mathcal{G}_B^{-1}$.*

Pixels in geometry images are classified into three types: *outside pixels*, *inside pixels* and *intersecting pixels*. Since all the intersection lines are obtained, it is straightforward to find out all *intersecting pixels* first. Other pixels are classified using boundary-fill process outlined in Figure 3. At the beginning of this algorithm, a valid boundary pixel $\mathbf{g}_{i_0,j_0}$ can be randomly picked. Here, we call a boundary pixel *valid* if its property has not been set and the boundary grid line it lies on intersects with only one triangle T_{k_B} in $\mathcal{G}_B^{-1}$. Suppose the normal of triangle T_{k_B} points outward, it is easy to determine whether $\mathbf{g}_{i_0,j_0}$ is inside or outside $\mathcal{G}_B^{-1}$. Then, in procedure FindNextSeed, from the 4-neighboring pixels of $\mathbf{g}_{i_0,j_0}$, we choose a pixel whose property has not been set and itself is not a boundary pixel as the first seed pixel. And the seed is labeled as the same property as that of $\mathbf{g}_{i_0,j_0}$.

After a seed pixel is selected, the geometry image pixels situated in the same connected region with the seed can be found out in BoundaryFill4. Here, the boundary filling process searches the pixels in 4-connected neighborhood. The reason for this is that pixels acting as the boundary for the algorithm are sometimes 8-connected. The calling to the procedure FindNextSeed and BoundaryFill4 is repeated until no seed can be found and thus the pixels of $\mathbf{G}_A$ are classified (Figure 3).

When the pixels of $\mathbf{G}_B$ are classified in the same way, we are thus ready to proceed to the next step of mesh reconstruction.

5. Triangulation and Mesh Reconstruction

When geometry image pixels are classified and non-surface pixels with respect to the result model determined, we use a constrained Delaunay triangulation library given in [Tri] to triangulate the part of the result mesh surface. To merge the result mesh correctly, the intersection line set must be retained and act as the constrained edges in the triangulation process. After the triangulations are finished, we obtain two triangle meshes. By transforming the vertices of the triangles into 3D space using $\mathbf{p}^{-1}(\cdot)$, we obtain two sets of triangles in 3D space. Suppose they are denoted as $\mathcal{R}_A$ and $\mathcal{R}_B$, then $\mathcal{R} = \mathcal{R}_A \cup \mathcal{R}_B$ is the result triangle mesh, which can be further converted into a geometry image [GGH02].

Procedure:PIXEL-CLASSIFICATION(G)

Input:$\mathcal{G}_A$ and $\mathcal{G}_B$ with *boundary grid lines* information;
Output:$\mathcal{G}_A$ and $\mathcal{G}_B$ with pixels' properties been classified;

Set properties of all *intersecting pixels* ;
$\mathbf{g}_{i_0 j_0} \leftarrow$ a valid boundary pixel;
Set property of $\mathbf{g}_{i_0 j_0}$;
while *FindNextSeed(seed,Prop)=TRUE* **do**
  BoundaryFill4(seed.x, seed.y, Prop);

Procedure: FindNextSeed(seed,SeedProp)

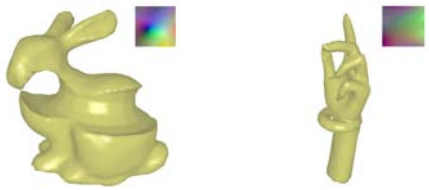
flag $\leftarrow$ FALSE;
foreach *boundary grid line* **l do**
  Get the end points of the grid line:P_l and Q_l;
  if *the property of pixel P_l has been set and P_l is not an intersection point and property of Q_l not set* **then**
    $\mathbf{g}_{i_0 j_0} \leftarrow Q_l$;
    $N_l \leftarrow$number of triangles intersecting with **l**;
    if *N_l is odd* **then**
      property of Q_l is opposite of P_l;
    else
      property of Q_l is the same as P_l;
    if *one neighbor of $\mathbf{g}_{i_0 j_0}$ is not boundary pixel* **then**
      $\mathbf{g}_{i_1 j_1} \leftarrow$the neighbor pixel;
      SeedProp $\leftarrow$property of $\mathbf{g}_{i_1 j_1}$;
      flag $\leftarrow$TRUE ;
      break;
if flag =*FALSE* **then**
  if *select a new valid boundary pixel as seed = TRUE* **then**
    SeedProp $\leftarrow$property of new seed;
    flag =TRUE;
return flag

Figure 3: *Pseudocode of determining properties of geometry image pixels (Section 4).*

6. Experimental Results and Analysis

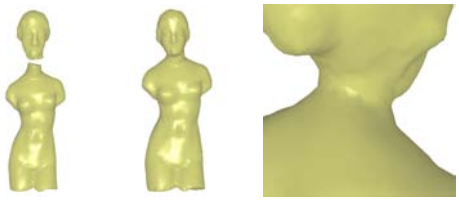
By making use of the regular data organization of geometry images, our algorithm is able to perform efficient surface division for Boolean operations. Figure 4 and 5 show some experimental results of our algorithm, which indicate that our algorithm can be used to sculpt the input models and construct novel ones. The corresponding geometry images of the results are illustrated in the up-right corner of the figures. When applying the algorithm, the resolution of the input geometry image can be freely adjusted for a suitable result.

Table 1 shows the timing comparison between optimized and non-optimized Boolean operations for our experiments.



(a) (b)

Figure 4: *Results of Boolean operations.*



(a) (b) (c)

Figure 5: *Union of two input models in (a), (c) shows the detail of intersection in result (b).*

We also implemented a Boolean operation algorithm that works directly on the 2D parameterizations of both meshes. The timing of the algorithm for the same model sets is listed in the right column. It is shown in Table 1 that the geometry image representation is superior to triangle meshes in the timing, especially for those complex models.

Table 1: *Timing for Boolean operations on geometry images(G.I.) optimized with quadtree, without quadtree, and direct implementation on 2D parameterizations.*

Results	Size of G.I.	G.I. Without quadtree	G.I. With quadtree	Mesh
Fig 1 (b)	64×64	59.2 s	0.140 s	1.547 s
Fig 4 (a)	64×64	108.5 s	0.328 s	1.563 s
Fig 4 (b)	128×128	1932.2 s	0.906 s	4.453 s
Fig 5	64×64	58.3 s	0.078 s	5.234 s

To obtain a mesh of high quality for further processing, the result mesh can be optimized by some mesh optimization techniques. In our experiment, we collapse tiny edges under the constraints that the collapse will not cause topology error and the visual effect is not affected. The optimized meshes near intersection lines are illustrated in Figure 1 (c) and Figure 5 (c).

7. Conclusion

We have presented a method to perform Boolean operation on free-form solid models represented by geometry images. The algorithm is simple and efficient. The regular data organization of geometry image facilitates the construction of hierarchical quadtrees, thus the time consumed in triangle-triangle intersection is reduced. The effective performance of pixel property classification also owes to the regular data organization of geometry images. Moreover, the Boolean operation can be performed independent of resolutions of geometry images, which offers flexibilities for various application requirements.

As most of the Boolean operations on explicit representations, the robustness of our algorithm mainly relies on the intersection computations, which depends greatly on the quality of geometry images. If the triangles in the reconstructed mesh of geometry image are well-shaped, the algorithm will be more robust. Therefore, how to generate a good geometry image for general meshes is worth to be investigated further. Also we are expecting to achieve performance improvement of Boolean operation on geometry images by utilizing graphics hardware.

References

[AD03] ADAMS B., DUTRÉ P.: Interactive boolean operations on surfel-bounded solids. In *ACM SIGGRAPH 2003* (2003), ACM Press, pp. 651–656.

[BKZ01] BIERMANN H., KRISTJANSSON D., ZORIN D.: Approximate boolean operations on free-form solids. In *ACM SIGGRAPH 2001 Papers* (New York, NY, USA, 2001), ACM Press, pp. 185–194.

[FGF05] FOUGEROLLE Y. D., GRIBOK A., FOUFOU S.: Boolean operations with implicit and parametric representation of primitives using r-functions. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (2005), 529–539.

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in multiresolution for geometric modelling*, Dodgson N. A., Floater M. S., Sabin M. A., (Eds.). Springer Verlag, 2005, pp. 157–186.

[FL00] FANG S., LIAO D.: Fast csg voxelization by frame buffer pixel mapping. In *Proceedings of the 2000 IEEE symposium on Volume visualization* (2000), ACM Press, pp. 43–48.

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. In *ACM SIGGRAPH 2002* (2002), ACM Press, pp. 355–361.

[KDRA02] KEN M., DAVID E. B., ROSS T. W., ALAN H. B.: Level set surface editing operators. 330–338.

[LHSW03] LOSASSO F., HOPPE H., SCHAEFER S., WARREN J.: Smooth geometry images. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2003), pp. 138–145.

[Mol97] MOLLER T.: A fast triangle-triangle intersection test. *J. Graph. Tools 2*, 2 (1997), 25–30.

[NYC05] NGUYEN M. X., YUAN X., CHEN B.: Geometry completion and detail generation by texture synthesis. *The Visual Computer 21*, 9–10 (2005), 669–678.

[Tri] http://www.cs.cmu.edu/ quake/triangle.html.