Bulletin of the JSME Vol.11, No.4, 2017
Journal of Advanced Mechanical Design, Systems, and Manufacturing

Boolean operations between two colliding shells: a robust, exact, and simple method

Jerome CHARTON*, Laehyun KIM* and Youngjun KIM*
*Korea Institute of Science and Technology
5, Hwarang-ro 14-gil, Seongbuk-gu, Seoul, 02792. Republic of Korea
E-mail: junekim@kist.re.kr

Received: 29 November 2016; Accepted: 3 January 2017

Abstract
Boolean operations are classic procedures in computer-aided design, and allow the creation of complex objects by combining simple objects. Although Boolean operations are trivial in implicit surface representations, they are problematic in polygonal meshes. Methods that directly use meshes to compute Boolean operations consistently consider the intersections between two faces without taking into account coplanar collisions. Thus, they either perturb the input meshes when colliding faces are coplanar or simply ignore this kind of collision. Most existing approaches for Boolean operations convert input meshes to volumetric representations such as binary space partitioning (BSP) and voxel grids. The output mesh is obtained by remeshing the resulting volumetric model. We propose a robust, exact, and simple method to manage Boolean operations between colliding shells without conversion and use a pure surface approach. The proposed method consists of three steps: (1) Calculating the intersections of input shells for both non-coplanar and coplanar collisions, (2) Decomposing the whole new mesh into its manifold components, and (3) Preserving only the components related to the requested operation (union or intersection). Subtraction operations are considered by reversing the surface orientation of the subtracted shell using the intersection operation. The output preserves the exact geometry of the input mesh while adding vertices for the remeshed colliding faces. In comparison with existing methods that use the mesh directly, the main advantage of the proposed method is that it processes coplanar collisions without geometrical modification, which avoids creating many small shells when two objects share the same part of the surface. Compared with methods using volumetric representation, the proposed method is faster and does not require input meshes without a boundary. We demonstrated the effectiveness of our method using synthetic models and real-world objects.

Keywords : Polygonal mesh, Triangular mesh, Boolean operations

1. Introduction

Boolean operations, used in constructive solid geometry (CSG) to create complex shapes from simple ones or to fit the interlocking between two objects, are composed of an elementary set of operations such as union, intersection, differences, and symmetric difference. Boolean operations are essential tools in computer-aided design (CAD) and have a long history in research (Requicha and Voelcker, 1985). Boolean operations have been used in mechanical design to model the operations of extrusion and combination for many years. Lately, constructive solid geometry is widely used. Commonly used in 3D animation and video game design, more recently, the expanded use of 3D printers has given rise to multiple applications (e.g., rapid prototyping and virtual surgery planning). In the case of rapid prototyping, the classic way to create a virtual object, with the aim of printing it, is to use a modeller such as Blender and create the object by using a juxtaposition of primitive shapes. However, this approach to construction without operating the union between colliding shells can cause problems during the slicing of the 3D printing process (e.g., local confusion of inside/outside or laminating of the object in coplanar collisions). Virtual surgery planning is a technique that uses surface reconstruction of patient data to simulate bone surgery. This planning is achieved by the 3D printing of virtual guides, which are created using synthetic objects interlocked with the bone surface using the difference operator (Laurentjoye et al., 2014). The

Paper No.16-00645
[DOI: 10.1299/jamdsm.2017jamdsm0041]

acquisition of a "real" object can usually be realized in two main ways. The first, essentially practiced in the medical field, is volumetric acquisition. The reconstructed data resulting from volumetric acquisition is a voxel grid that can be used to extract a mesh representation of the targeted surface. The second is the acquisition of a 3D laser scan. Using a laser line projection, a sampling of the surface of an object is acquired. The mesh of the surface is built by adding polygons based on the sampling. When volumetric acquisition allows a full closed surface of the acquired data to be created, the 3D laser scan can frequently only acquire a partial surface by creating a mesh with boundaries.

However, existing methods based on Boolean operations require input meshes without a boundary. The principle reason is that they use the volumetric properties of the input boundary representations. This paper presents a method for processing Boolean operations between two colliding shells (not only punctually) without constraint of closure or 3-manifoldness on the input meshes.

2. Related work & positioning

Although Boolean operations are trivial in terms of implicit surfaces or other volumetric representations, they are challenging in relation to polygonal meshes. In fact, the polygonal mesh model provides more topological possibilities than the surface of a volume, including especially non-3-manifolds and/or 2-manifolds with boundaries. Thus, the common way to obtain a robust Boolean operation between two polygonal meshes involves a conversion (complete or partial) of these surface representations to volumetric representations (e.g., binary space partitioning (BSP) (Thibault and Naylor, 1987), voxel grid) (Granados et al., 2003) (Chen, 2007) (Bernstein and Fussell, 2009) (Campen and Kobbelt, 2010) (Hachenberger and Kettner, 2016). However, these representations are computationally inefficient, depend on parameters (e.g., the voxel size), or require input meshes without a boundary. The result of these methods is generally a remeshing of the volumetric representation after operation.

Besides, the other approaches use the polygonal mesh data structure directly with a space partitioning tree (e.g., octree, kd-tree, bounding volume hierarchy (BVH)) as meta-structure to reach faces efficiently and define the inside/outside relation by ray tracing and parity counting (Bernstein, 2007) (Chen et al., 2010) (Feito et al., 2013), plane sweeping with a dynamic real tree (R-tree) (Schifko et al., 2010) or even, by using this meta structure to distinguish the inside and outside of each face or constraint component (Pavi et al., 2010) (Mei and Tipper, 2013). These methods are systematically composed of two phases. First, they compute and remesh intersections between input meshes, and second, they classify, in the merged mesh, triangles to remove and to preserve by using the closure property of the input meshes.

This paper presents a method that falls within the second category (using the mesh directly). However, this method uses open and non-necessary 3-manifold shells that are colliding in more than one point. The requirements of the input shells are:

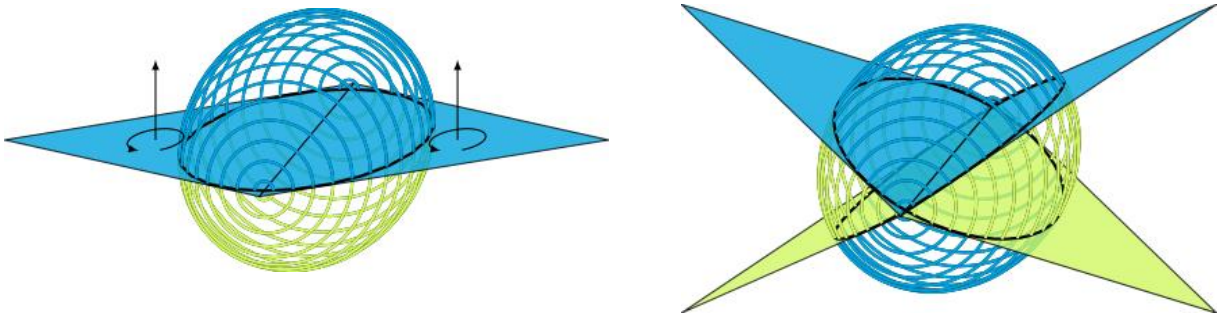
- Each face of both input shells is consistently oriented. The front side is oriented to the outside and the backside to the inside.
- Let e be a non-boundary edge and $\mathscr{B}$ be the maximal ball centred in the middle of e and intersecting all adjacent faces of e by one-half disk. Then all partitions of $\mathscr{B}$, obtained by the intersection with neighbouring faces of e, can be classified as inside or outside of the object by the orientation of splitting faces (Fig. 1(b)).
- The two input shells are colliding with at least one collision larger than a point (e.g. edge, face, and series of edges).

The output of this method is a mesh composed of subsets of the components of the input shells with refinement of the colliding faces. The output mesh preserves the input topological singularities but converts geometrical singularities into topological ones, if there are any.

As for almost all approaches using polygonal mesh directly, the proposed method is composed of two main steps. The first step consists of the creation of a copy of both input meshes in one indexed mesh without geometrical singularity. Intersections between shells are computed and remeshed in this step to convert these geometrical singularities into topological singularities. The second step, using a classification of faces in oriented manifold components (*OMC*s) and non-manifold edges in chains, determines for each component whether it should be preserved, reversed, or removed according to the expected Boolean operation. During this process, the operation can be aborted for non-consistence of the inside and outside. The operation can be aborted in the two steps. In the first step, if a bordering edge is strictly intersecting the inside of a face or a non-bordering edge, the inside/outside cannot be determined. In the second step, if

[DOI: 10.1299/jamdsm.2017jamdsm0041]

a manifold component has a multiple classification, the global operation is inconsistent.



(a) Normal edge. (b) Singular edge with four adjacent faces.

Fig. 1. Examples of edges with consistent inside (green) / outside (blue) definition.

As for almost all approaches using polygonal mesh directly, the proposed method is composed of two main steps. The first step consists of the creation of a copy of both input meshes in one indexed mesh without geometrical singularity. Intersections between shells are computed and remeshed in this step to convert these geometrical singularities into topological singularities. The second step, using a classification of faces in oriented manifold components (*OMC*s) and non-manifold edges in chains, determines for each component whether it should be preserved, reversed, or removed according to the expected Boolean operation. During this process, the operation can be aborted for non-consistence of the inside and outside. The operation can be aborted in the two steps. In the first step, if a bordering edge is strictly intersecting the inside of a face or a non-bordering edge, the inside/outside cannot be determined. In the second step, if a manifold component has a multiple classification, the global operation is inconsistent.

3. Creation of the global mesh without geometrical singularity

3.1. Removing duplicated entities and degenerated faces

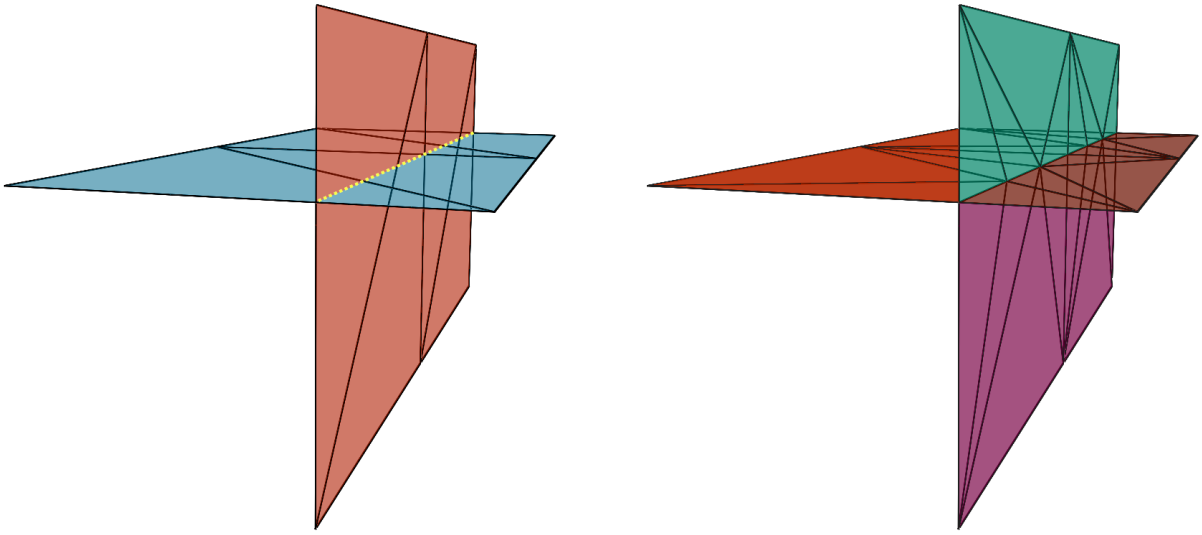
This first step starts by copying all the data of both inputs (A and B) in one indexed mesh termed C_{Init}as adding a tag ω_A or ω_B to faces to preserve the information of their origin. C_{Init} is composed of shells with geometrical singularities which are self-intersections and potentially duplicated vertices and degenerated faces. In terms of implementation this indexed mesh is only composed of a table of vertex coordinates and a table of faces indexing the first table. The aim of this first step of the algorithm is to create a mesh containing geometrical information of inputs A and B without geometrical singularity. The first geometrical singularities that can be directly managed are the duplicated vertices. This is achieved by sorting vertices in lexicographic order to bring the duplicated vertices closer. Using an ε value relative to the numerical precision, duplicate vertices can be merge with an ε Euclidean distance criteria. Note: two duplicated vertices do not necessarily occur side by side. ε has to be used for the Euclidean distance to determine the proximity but also on the x-, y- and z-coordinates independently to define the condition of stopping the search of duplicated vertices. During the sorting and merging processes, the vertex indices of faces should be maintained. This indexation is maintained by using a third table of double indexation between the coordinates of vertices and faces to operate the actual sorting and merging. If a face indexes the same vertex twice, this face is discarded. If two faces index the same set of vertices, only the one issued by A is preserved. If the two faces have the same orientations the preserved face is tagged ω_D and ω_C otherwise. After this initial removing of geometrical duplications, two geometrical singularities remain: degenerated faces and self-intersections. Even if the merging of vertices can solve several degenerated faces due to close neighbours, the degenerated faces can remain because of the vertices almost lying on the opposite base. These degenerated faces can be processed through edge swapping or collapsing (Chong et al., 2007) (Attene, 2010) but only two faces with the same tag ($\omega_A, \omega_B, \omega_C$ or ω_D) can be swapped. After removing duplicated vertices and degenerated faces, C_{Init} becomes $C_{InitCleaned}$.

[DOI: 10.1299/jamdsm.2017jamdsm0041]

3.2. Remeshing intersections

The remeshing of (self-)intersections is separated into two steps. The first step is the computation of the segments of intersections and the second step, using the set of segments of intersections of each face, refines the face to incorporate these segments as edges in the mesh structure.

Computing intersections. For $C_{InitCleaned}$, the remaining geometrical singularities are due to intersections between faces. A fast computation of the intersections is performed by building two sets of faces F_A and F_B, where F_A and F_B contain faces inherited from A and B, respectively, and wherein the bounding box of each face intersects the intersection of the bounding boxes of A and B (known as the common bounding box). Without loss of generality, suppose that $|F_A| < |F_B|$. Using only the faces of F_A, a BVH structure is built (Hapala et al., 2011). The BVH is queried by using the faces of F_B. This performs a fast and small space partitioning to reach all potential collisions between the faces of A and those of B. The intersection between two faces f_A and f_B can be classified in two categories: coplanar and non-coplanar. If f_A and f_B are non-coplanar, the intersection is a segment that is potentially null (Fig. 2(a)). We computed this collision using the method proposed by (Möller, 1997). An initial test consists of checking if all vertices of f_A are on the same side of the plane of f_B. In this case, there is no collision. After this, the two segments of intersection between f_A and the plane of f_B are computed and vice versa. These two segments are aligned and their collision, if any, is the actual collision. For coplanar collisions, this problem is actually a two-dimensional problem and can be performed by testing collisions between all edges of f_A with all edges of f_B (Antonio, 1992). During this procedure, for all edges in the intersection, we create a vertex and keep the relation between this vertex and the edge. Before the creation of a vertex, we check if the edge(s) is (are) not already associated with a vertex at the intersection point. If the intersection point is related to one and only one bordering edge, the process stops and reports the impossibility to operate a Boolean operation with the input data.



(a) The two shells (A in blue and B in red) are colliding. The yellow dotted line draws the intersection line.

(b) After computation and remeshing of intersections, we obtain a new mesh C. C can be decomposed into oriented manifold components (*OMC*s) rendered in different colours.

Fig. 2. Simple example of intersection between two shells.

Refinement of the mesh to integrate the intersections. After the computation of all intersections, we obtain a set of colliding faces F^*, with a set of segments (potentially null) for each face, and a set of edges involved in the intersections E^*. If we are working with non-exact arithmetic, some faces that are actually colliding at the edge can be missing in F^* due to the ε-precision. We correct this by adding all adjacent faces to an edge of E^* missing in F^*. All faces of F^* have to be subdivided to integrate the segment inside the mesh structure. The integration of the segments of intersections in a triangle is a two-dimensional problem. It can be processed by a constraint Delaunay triangulation (Shewchuk, 1997). Nevertheless, the common way to embed a three-dimensional planar polygon in a two-dimensional space consists of the removal of one of its coordinates using the maximal coefficient of the normal vector. This technique is easy and fast; it can generate a degenerated triangle in the two-dimensional space if we are working with non-exact

[DOI: 10.1299/jamdsm.2017jamdsm0041]

arithmetic. Let (e_0, e_1) be the plane of the orthogonal projection, where $\{e_0, e_1\} \subset \{x, y, z\}$ and let e_2 be the complementary. The projection of the face in the two-dimensional space contracts the triangle with different factors in e_0 and e_1. These factors are relative to the angles (α_0 and α_1) of the face in the planes (e_0, e_2) and (e_1, e_2) (Fig. 3). Preservation of the geometry of the face in this embedding requires the coordinates of each vertex $v = (v_x, v_y, v_z)^T$ to be dilated to compute its image $v' = \left(\frac{v_{e_0}}{\cos(\alpha_0)}, \frac{v_{e_1}}{\cos(\alpha_1)}\right)^T$. This transformation can be computed with a minimum of computation by directly computing $cos(\alpha_i)$ by using the coordinates of the normal $\hat{n}$. In fact the angles between the normal vector and the plane (e_0, e_1) in (e_0, e_2) and (e_1, e_2) are termed β_0 and β_1. We have the properties $\hat{n}_{e_i} = cos(\beta_i)$ and $cos(\beta_i) = sin(\alpha_i)$. By that, we have $cos(\alpha_i) = \sqrt{1 - (\hat{n}_{e_i})^2}$. In this process, we do not actually modify the coordinates of the vertices themselves but of a copy of them. The outcome of this refinement is the creation of sub-triangulation generated and suppression of the original face. During the creation of a face f, if another face f' with the same set of vertices already exists then the face tagged ω_B is discarded. Moreover if f' has an opposed orientation to f, the preserved face is tagged ω_C, otherwise it is tagged ω_D. With these operations $C_{InitCleaned}$ becomes C.

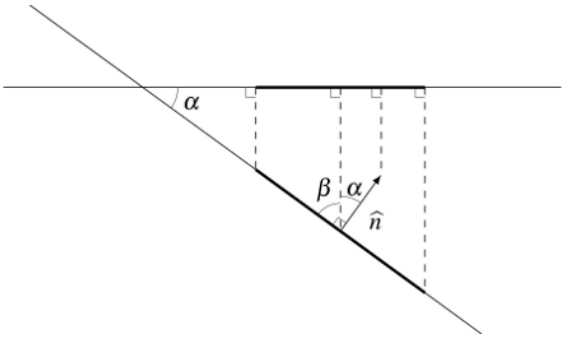


Fig. 3. Orthogonal projection in two-dimensional space.

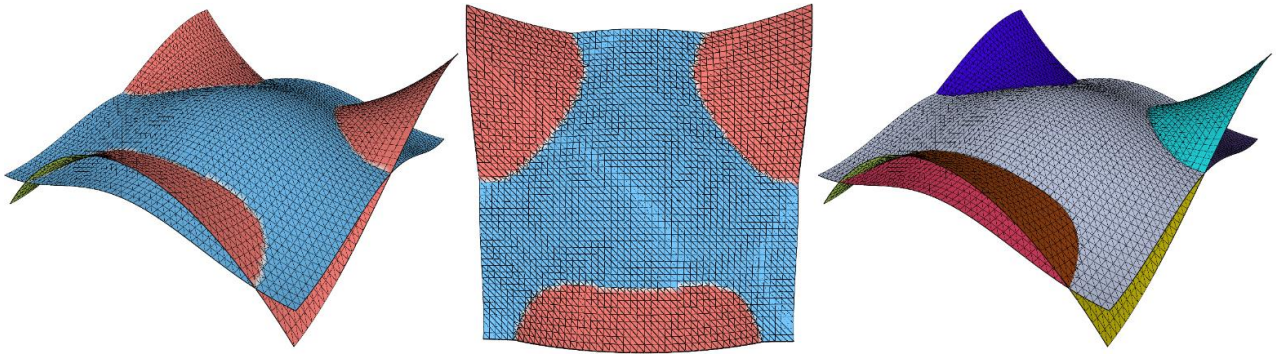
4. Classification

After the previous steps, C does not contain geometrical singularity; rather, it contains several edges with more than two adjacent faces. These non-manifold edges are the result of the instantiations of the non-coplanar intersections between the two input shells. Furthermore, as no bordering edge crosses the inner part of the surface (Section 2), the whole mesh of C can be decomposed into *OMC*s (Fig. 2(b)). An *OMC* is a set of adjacent faces through normal edges. An edge e is considered normal if and only if e has exactly two adjacent faces f_a and f_b and the vertices of e are in opposite order in f_a and f_b. These *OMC*s are inter-connected by subsets (or sequences) of non-manifold edges known as dividing lines. Two edges e_u and e_v sharing a vertex v are related to the same dividing line if for all adjacent faces $f_{(e_u,i)}$ to e_u there exists a succession of adjacent faces to v connected by normal edges ending with a face adjacent to e_v.

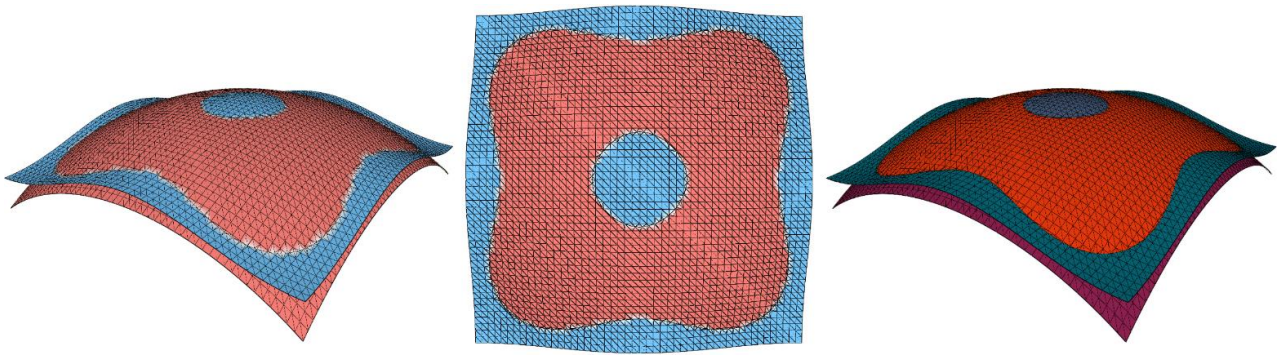
These dividing lines can be closed or opened (Fig. 4). They are closed when all colliding edges are normal and opened either when the two vertices at the ends are the intersections between bordering edges or have more than two non-manifold edges in their neighbourhoods. All edges of a dividing line are adjacent to the same *OMC*s. By that, any edge of the line can represent the connections of the whole dividing line.

The classification of the *OMC*s is achieved by using the geometry of adjacent faces of a random sampled edge e of each dividing line DL_e. The set of adjacent faces to e, termed $F_e = \{f_i\}$, is such that firstly, no face is coplanar to another one and secondly, each face is a separation between the inside and outside of at least one of the input objects. These two properties enable the classification to be simplified as follows. Without loss of generality, we define an arbitrary orientation to e and name this oriented edge $\vec{e}$. Let τ be an orthogonal plane to $\vec{e}$ (Fig. 5(a)). For all $f_i \in F_e$, its orthogonal projection in τ is computed. As f_i and τ are orthogonal, the projection produces a vector $\overrightarrow{t_i}$. Defining an arbitrary $\overrightarrow{t_0} \in \{\overrightarrow{t_i}\}$, faces of F_e are sorted in a cyclic order according to its counter-clockwise oriented angle $\theta_i = \measuredangle(\overrightarrow{t_0}, \overrightarrow{t_i})$ in the plane τ oriented by $\vec{e}$. The cyclic sequence of faces (or *OMC*s) is obtained by the sort termed $F_{\vec{e}}$. All faces $f_{\overrightarrow{t_i}}$ of $F_{\vec{e}}$ are a transition between the inside and the outside of A if $f_{\overrightarrow{t_i}}$ is tagged ω_A, of B if $f_{\overrightarrow{t_i}}$ is tagged ω_B and of A and B if $f_{\overrightarrow{t_i}}$ is tagged ω_C or ω_D.

[DOI: 10.1299/jamdsm.2017jamdsm0041]

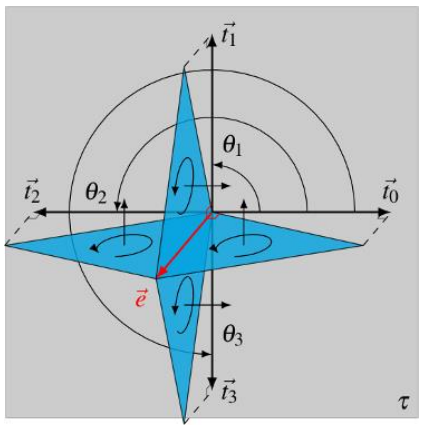


(a) Shells A and B are the discretization of the Trigonometric and Monkey saddle surfaces. All colliding lines are opened and end with collisions between bordering edges.

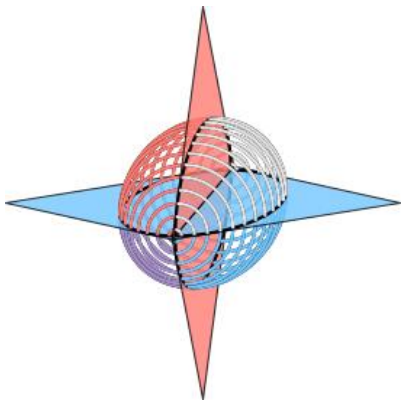


(b) Shells A and B are the discretization of the Trigonometric and Partial sphere surfaces. All colliding lines are closed.

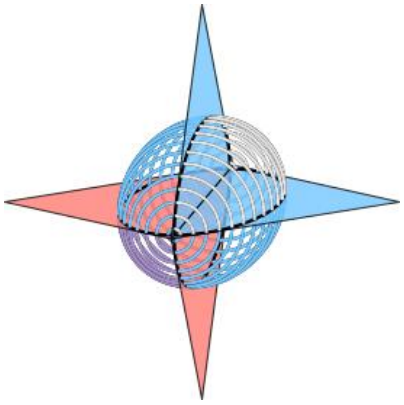
Fig. 4. Two examples of combinations of two meshes with a boundary. On the left and in the middle, the meshes result from the combination of C with two points of view. Components of inputs A and B are rendered in blue and red and added vertices are in white. On the right, the decomposition into *OMC*s is rendered in different colours.



(a) Computation of the θ_i angles.



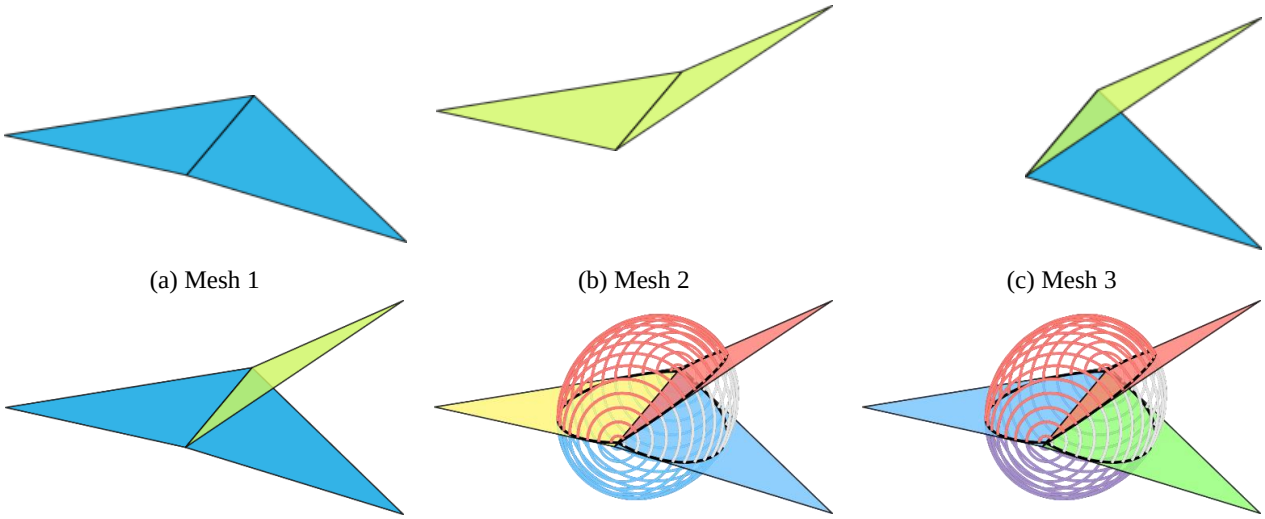
(b) Angle range classification using the faces of (a) and the sequence of tags $(\omega_A, \omega_B, \omega_A, \omega_B)$.



(c) Angle range classification using the faces of (a) and the sequence of tags $(\omega_A, \omega_A, \omega_B, \omega_B)$.

Fig. 5. Simple and classic examples of classification. Legend: (a) blue: outer side. (b)-(c) clear blue: tag ω_A or Δ_A, clear red: tag ω_B or Δ_B, white: $\Delta_{\cup}$, and purple: $\Delta_{\cap}$.

[DOI: 10.1299/jamdsm.2017jamdsm0041]



(a) Mesh 1 (b) Mesh 2 (c) Mesh 3

(d) Faces of C with orientations that can be obtained by merging (a) with (b) or (a) with (c). (yellow-green: inside and blue: outside)

(e) A = (a) and B = (b): example with a face (in yellow) issue of a coplanar collision between two opposed surfaces using the orientations of (d).

(f) A = (a) and B = (c): example with a face (in green) issue of a coplanar collision between two twin surfaces using the orientations of (d).

Fig. 6. Examples of classifications with adjacent coplanar collisions. Legend: (a)-(d) blue: outer side, yellow-green: inner side. (e)-(f) clear blue: tag ω_A or Δ_A, clear red: tag ω_B or Δ_B, yellow: tag ω_C, green: tag ω_D, white: $\Delta_{\cup}$, and purple: $\Delta_{\cap}$.

Classification of the angle ranges. The cyclic sequence $F_{\vec{e}}$ combined with the information of orientations and tags of faces enables the classification of angle ranges between each pair of faces of $F_{\vec{e}}$. Let δ be a pair of Boolean values, where the first element represents the inside/outside of A and the second this of B. δ is initialized with $(-1,-1)$, where -1 means undetermined, 0 outside, and 1 inside. By traversing the sequence $F_{\vec{e}}$, the values of δ changes as follows. The traversing of $f_{\overrightarrow{t_i}}$ with:

- a tag ω_A (respectively ω_B) changes the first (respectively second) element of δ to 1 if the front of $f_{\overrightarrow{t_i}}$ is oriented backward of the sequence $F_{\vec{e}}$ or to 0 in the other sense.
- a tag ω_D has the same effect than the tags ω_A and ω_B but modifies the both values (Fig. 6(e)).
- a tag ω_C, such as the tag ω_D, modifies both elements of δ. With the same effect than a tag ω_A for the first element, but has an opposite effect than the tag ω_B on the second element (Fig. 6(f)).

The first traversal of $F_{\vec{e}}$ initializes the values of δ. A second traversal is required to classify the angle ranges between the pairs of faces of $F_{\vec{e}}$. During the second pass, all changes inside/outside of A or/and B have to operate an actual change of the targeted value(s) of δ. Otherwise, the Boolean operation is not consistent. Moreover, during this second pass, each angle range is classified as: common outside $\Delta_{\cup}$: if $\delta = (0,0)$, inside exclusive to A: $\Delta_{A\oplus}$ if $\delta = (1,0)$, inside exclusive to B: $\Delta_{B\oplus}$ if $\delta = (0,1)$, and common inside $\Delta_{\cap}$: if $\delta = (1,1)$. Applying the angle range classification on the example of Fig. 5(a), with the sequence $(f_{\overrightarrow{t_0}}, f_{\overrightarrow{t_1}}, f_{\overrightarrow{t_2}}, f_{\overrightarrow{t_3}}, [f_{\overrightarrow{t_0}}])$ and:

- $f_{\overrightarrow{t_0}}$ tagged ω_A, $f_{\overrightarrow{t_1}}$ tagged ω_B, $f_{\overrightarrow{t_2}}$ tagged ω_A and $f_{\overrightarrow{t_3}}$ tagged ω_B the result is $(\Delta_{\cup}, \Delta_{B\oplus}, \Delta_{\cap}, \Delta_{A\oplus})$ (Fig. 5(b)).
- $f_{\overrightarrow{t_0}}$ tagged ω_A, $f_{\overrightarrow{t_1}}$ tagged ω_A, $f_{\overrightarrow{t_2}}$ tagged ω_B and $f_{\overrightarrow{t_3}}$ tagged ω_B the result is $(\Delta_{\cup}, \Delta_{A\oplus}, \Delta_{\cap}, \Delta_{A\oplus})$ (Fig. 5(c)).

Classification of the oriented manifold components. Using the angle range classification $\Delta_{A\oplus}$, $\Delta_{B\oplus}$, $\Delta_{\cup}$ and $\Delta_{\cap}$ and the expected Boolean operation union $\cup$, intersection $\cap$, differences $A-B$ and $B-A$ (Note: $A-B = A^{\oplus}$ and $B-A = B^{\oplus}$) and the symmetric difference $\oplus$, the *OMC*s are classified in three classes according to the expected action to apply: $\mathscr{C}_{\text{preserve}}$, $\mathscr{C}_{\text{reverse}}$ and $\mathscr{C}_{\text{remove}}$. $\Delta_{\oplus}$ is defined by $\Delta^{\oplus} = \Delta_{A\oplus} \cup \Delta_{B\oplus}$. A Boolean operation Λ is achieved by keeping only OMCs with the representative faces bordering a Δ_{Λ}. If a face delimits a Δ_{Λ} and has the backside oriented forward Δ_{Λ} then its OMC is classified $\mathscr{C}_{\text{preserve}}$, if the side presented is the front side, its *OMC* component is classified $\mathscr{C}_{\text{reverse}}$ and otherwise $\mathscr{C}_{\text{remove}}$. With two exceptions, for $\Delta_{B\oplus}$ with a face tagged ω_C, the orientation of the face is inverted, and for $\Delta^{\oplus}$ with a face is tagged ω_C, its OMC is systematically classified $\mathscr{C}_{\text{remove}}$. Note: for simplification,

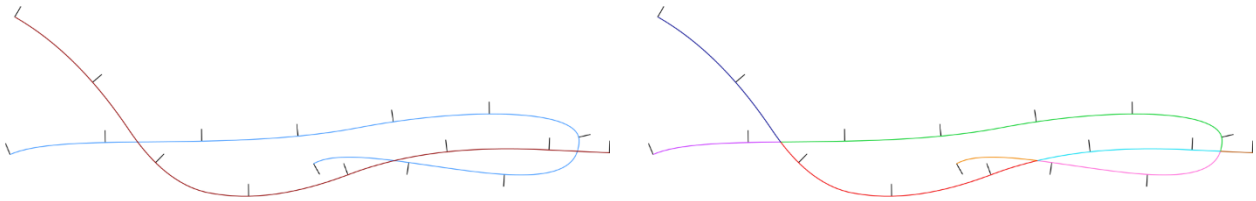
[DOI: 10.1299/jamdsm.2017jamdsm0041]

the difference Boolean operations can be operated by reversing faces of one of the input shells (converting A to $\bar{A}$ or B to $\bar{B}$). The difference Boolean operations are in this case: $A - B = A \cap \bar{B}$ and $B - A = B \cap \bar{A}$. For example, applying this classification to the example in Fig. 5(b) results in the following:

- with the $\cup$ operation, $f_{\overrightarrow{t_0}}$ and $f_{\overrightarrow{t_1}}$ are classified $\mathscr{C}_{\text{preserve}}$ and $f_{\overrightarrow{t_2}}$ and $f_{\overrightarrow{t_3}}$ are classified $\mathscr{C}_{\text{remove}}$.
- with the $\oplus$ operation, $f_{\overrightarrow{t_0}}$ and $f_{\overrightarrow{t_1}}$ are classified $\mathscr{C}_{\text{preserve}}$ and $f_{\overrightarrow{t_2}}$ and $f_{\overrightarrow{t_3}}$ are classified $\mathscr{C}_{\text{reverse}}$.
- with the $\cap$ operation, $f_{\overrightarrow{t_0}}$ and $f_{\overrightarrow{t_1}}$ are classified $\mathscr{C}_{\text{remove}}$ and $f_{\overrightarrow{t_2}}$ and $f_{\overrightarrow{t_3}}$ are classified $\mathscr{C}_{\text{preserve}}$.
- with the $A^{\oplus}$ operation, $f_{\overrightarrow{t_0}}$ is classified $\mathscr{C}_{\text{preserve}}$, $f_{\overrightarrow{t_1}}$ and $f_{\overrightarrow{t_2}}$ are classified $\mathscr{C}_{\text{remove}}$ and $f_{\overrightarrow{t_3}}$ is classified $\mathscr{C}_{\text{preserve}}$.

In regard to with singular edges, if the initial meshes contain singular edges, these edges can create a dividing line without neighbouring *OMC* of both input meshes. For this case, all the *OMC*s connected to this dividing line are considered to be the same one. Note: if no Δ_A is found around a dividing line and the tags of all adjacent faces are not exclusively A or B, then all adjacent *OMC*s are classified $\mathscr{C}_{\text{remove}}$. For example, the $\cap$ operation in Fig. 5(b) and the $A^{\oplus}$ operation in Fig. 5(c) are empty.

The classification of the *OMC*s is operated locally at the dividing lines. However, this classification in the global view can be inconsistent. That results in *OMC*s classified in different categories at different dividing lines. In that case the Boolean *operation* itself is not consistent. For example, Fig. 7 shows, by a 2D representation of the surfaces, an example in which all angle ranges and *OMC*s can be classified locally. However, the red *OMC* has different classifications at the dividing lines (e.g., for the $\cup$ operation the red component would be classified $\mathscr{C}_{\text{remove}}$ at the node on the right and $\mathscr{C}_{\text{preserve}}$ at the node on the left).



(a) Input shells A (blue) and B (red). (b) After remeshing in C decomposition in *OMC*s.

Fig. 7. Example of inconsistent Boolean operation.

The Boolean operation is performed by applying the classification strategy. The *OMC*s classified $\mathscr{C}_{\text{preserve}}$ are preserved, those classified $\mathscr{C}_{\text{reverse}}$ are reversed and those classified $\mathscr{C}_{\text{remove}}$ are removed. However, if an *OMC* is not connected to a DL, it would not be classified in $\mathscr{C}_{\text{preserve}}$, $\mathscr{C}_{\text{reverse}}$ or $\mathscr{C}_{\text{remove}}$. That can occur when the models A and B represent the same surface. Note: the case with an empty A or B is excluded by the input requirements. In this case the *OMC* can have only two tags ω_C or ω_D. If the OMC is tagged ω_C (the surface of B = the surface of A), the OMC is classified $\mathscr{C}_{\text{preserve}}$ for the $A^{\oplus}$ operation ($A - A = A$), $\mathscr{C}_{\text{reverse}}$ for the $B^{\oplus}$ operation ($A - A = A$) and $\mathscr{C}_{\text{remove}}$ otherwise ($A \cup A = A \oplus A = \mathbb{R}^3$ and $A \cap A = \emptyset$). If the OMC is tagged ω_D (the surface of B = the surface of A), the OMC is classified C_{preserve} for the $\cup$ and $\cap$ operations ($A \cup A = A \cap A = A$) and $\mathscr{C}_{\text{remove}}$ otherwise ($A - A = A \oplus A = \emptyset$).

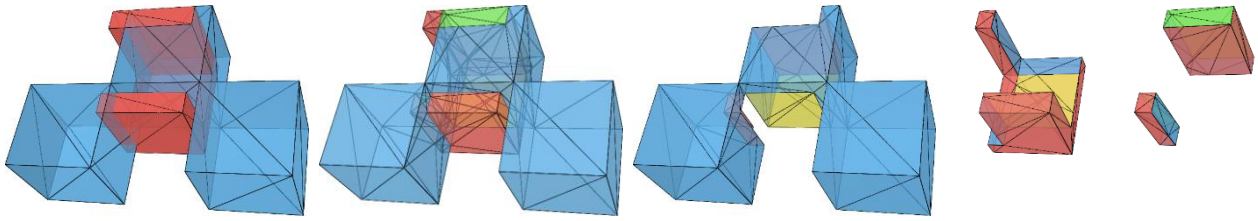
5. Experimental results & discussion

The presented algorithm has been implemented in C++ using double precision and tested with several synthetic and acquired meshes with an i7-2620M 2.70GHz CPU with 8GB of memory. Computing time was checked in mono-threaded mode. Some synthetic meshes have been created to improve critical cases, i.e., coplanar collisions, singular edges existing in the input meshes, and singular edges created during the process. Other synthetic meshes have been tested to improve the algorithm for classic use in CAD. Its robustness has been assessed on acquired meshes by testing it on 3D laser scans and volumetric reconstructions with classic models and a real-use case. In the following, the tests presented have been selected to present a representative sample of these tests. The first test is a simple synthetic test that involves both types of coplanar collisions and two configuration of singular edges. One involved into the collision and another not. The

[DOI: 10.1299/jamdsm.2017jamdsm0041]

second test uses the classic model of the *Bunny* from the Stanford repository[1], the *Scanned sphere* model (a 3D scan acquisition of a sphere) and an ico-sphere (a synthetic sphere twin of the acquired one). The third test is an application of Boolean operations in the surgical domain. It uses the different parts of a patient skull (*Maxilla* and *Mandible*) and a neutral splint used to obtain the negative of teeth that is used during the surgery to define different positions (Laurentjoye et al., 2014). Finally, the fifth test is an experiment in the CAD application of Boolean operations. By using a model composed of several shells with holes (*Tron light cycle*[2]), this model is separated into multiple groups without collision. These groups are combined by $\cup$ operations. When a Boolean operation is not consistent, shells are modified by a minimum to fit with the requirements of the algorithm.

Simple test: *Three cubes* and *U*. For this test, the models A and B are *Three cubes* and *U*, respectively (Fig. 8(a)). The model Three cubes is composed of three cubes touching at the edges, creating singular edges and the model U wraps the central cube of the model A creating multiple and various coplanar collisions (coplanar collisions with opposed orientations are in yellow Fig. 8(d) and coplanar collisions with the same orientation are in green see Fig. 8(e)). This test shows that non-manifold edges are managed as well if they are in the colliding line (such as the edge between the cube on the left and that in the middle) or not (such as the edge between the cube on the right and that in the middle).



(a) *Three cubes* (A in blue) and *U* (B in red). (b) After computation and remeshing of intersections, C is obtained (c) $A - B = A^{\oplus}$. (d) $B - A = B^{\oplus}$. (e) $A \cap B$.

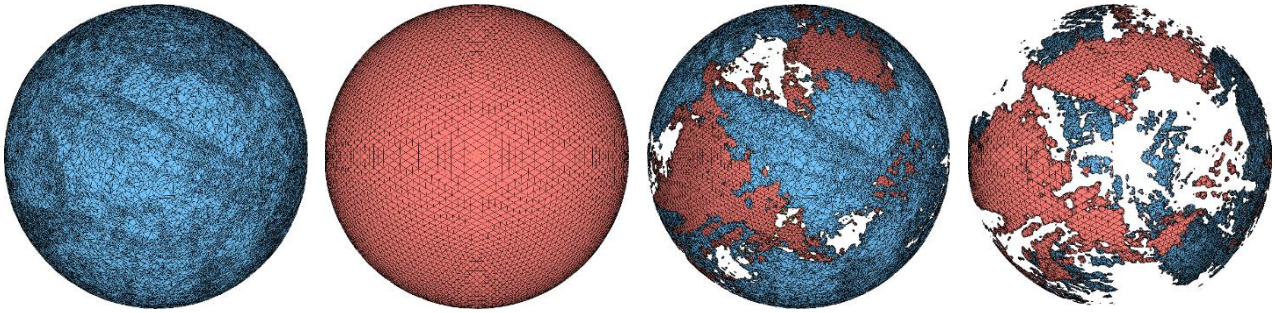
Fig. 8. Boolean operations between *Three cubes* (model A in blue) and *U* (model B in red). Legend: clear blue: tag ω_A, clear red: tag ω_B, yellow: tag ω_C, and green: tag ω_D.

3D laser scan: acquired sphere and synthetic sphere. The comparison of an acquired mesh with its virtual model is a common operation and can be visualized by operating the difference operations between the two models. The model acquired minus the synthetic one gives an over-estimation and in the other sense, an under-estimation. With this aim, the first test on 3D laser scan data proposes to use as model A the sphere reconstructed from a 3D laser scan acquisition of a 3D printed synthetic *ico-sphere* of radius 20 mm and built by six subdivisions, named the *Scanned sphere* model (Fig. 9(a)) and as model B its initial virtual model, named the *ico-sphere* model (Fig. 9(b)). The acquired model has been re-centred to be realigned with the virtual model. This test is challenging for Boolean operations because the 3D laser scan acquisition creates a mesh with widespread vertices that creates irregular triangulations. Moreover, the high accuracy of the 3D laser scan brings the surfaces of models A and B in close proximity (the distances between A and B are in max +0.013/-0.014mm, in mean 0.001mm -0.002mm/+0.003mm and the standard deviation is 0.003mm). Because of that, during the processing, model C is decomposed by several dividing lines that are the base of the classification of the presented algorithm. The result of difference operators is given in Fig. 9(c) and 9(d). The model *Scanned sphere* contains 98,304 faces and 49,154 vertices and the model *ico-sphere* contains 20,480 faces and 10,242 vertices. The mean time to process the different operations is 24.386s for 22,582 colliding faces. Remark that, for all operations, the time variation is around 0.5s. However, to measure the impact of variation of the density of faces, in terms of processing time, on the different phases of the proposed algorithm, a comparison with different levels of subdivisions has been realized. The numerical results of this test are listed in Table 5. Observations: as expected, the cost in time of building the BVH, the computation of the intersections, and extraction of *OMC*s are linearly proportional to the number of input faces. However, the Boolean operation itself (classification and modifications) is also proportional to the number of input faces. In fact, the time required for classification is negligible compared to the modification time to remove unwanted faces and vertices. That confirms the efficiency of the classification of the *OMC*s by angles of one edge per dividing line.

[1] http://graphics.stanford.edu/data/3Dscanrep/

[2] http://tf3dm.com

[DOI: 10.1299/jamdsm.2017jamdsm0041]



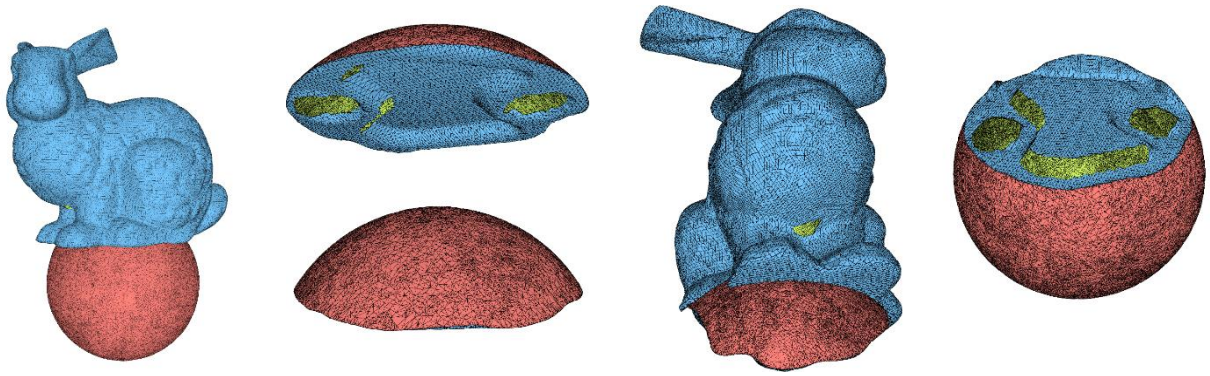
(a) *Scanned sphere*. (b) *ico-sphere*. (c) $A - B = A^{\oplus}$. (d) $B - A = B^{\oplus}$.

Fig. 9. Difference operators between the Scanned sphere (model A in blue) and the ico-sphere of six subdivisions (model B in red).

Table 1. Experimental results between the *Scanned* sphere model and the *ico-sphere* with different subdivisions and the *Bunny*.

#subd.	$\|F\|$	$\Delta_T(M)$	$\Delta_T(BVH)$	$\Delta_T(I)$	$\#CF$	$\Delta_T(R)$	$\Delta_T(OMC)$	$\#DL$	$\Delta_T(OB)$	$\Delta_T(G)$
4	1280	1.099	3.884	4.286	5815	1.483	2.490	279	0.446	13.692
5	5120	1.127	3.979	6.094	16163	3.092	4.541	615	1.172	20.011
6	20480	1.395	4.058	7.625	22582	3.565	5.969	634	1.846	24.465
7	81920	2.281	4.534	15.139	30103	5.460	8.306	609	2.931	38.658
8	32768	5.386	4.733	27.664	44713	8.167	14.133	612	6.266	66.355
Model	$\|F\|$	$\Delta_T(M)$	$\Delta_T(BVH)$	$\Delta_T(I)$	$\#CF$	$\Delta_T(R)$	$\Delta_T(OMC)$	$\#DL$	$\Delta_T(OB)$	$\Delta_T(G)$
Bunny	69451	1.817	0.808	0.344	1080	0.993	2.701	1	1.917	8.586

Legend: #subd.: number of subdivisions of the *ico-sphere* model, $|F|$: number of faces of the *ico-sphere* model, $\Delta_T(M)$: computing time in seconds of the creation of $C_{InitCleaned}$, $\Delta_T(BVH)$: computing time in seconds of the creation of the BVH, $\Delta_T(I)$: computing time in seconds of the intersections, $\#CF$: number of colliding faces, $\Delta_T(R)$: computing time in seconds of the remeshing of the intersections, $\Delta_T(OMC)$: computing time in seconds of the creation of *OMC*s and dividing lines, $\#DL$: number of dividing lines created, $\Delta_T(OB)$: computing time in seconds of the Boolean operation (classification and modifications), $\Delta_T(G)$: global computing time in seconds.



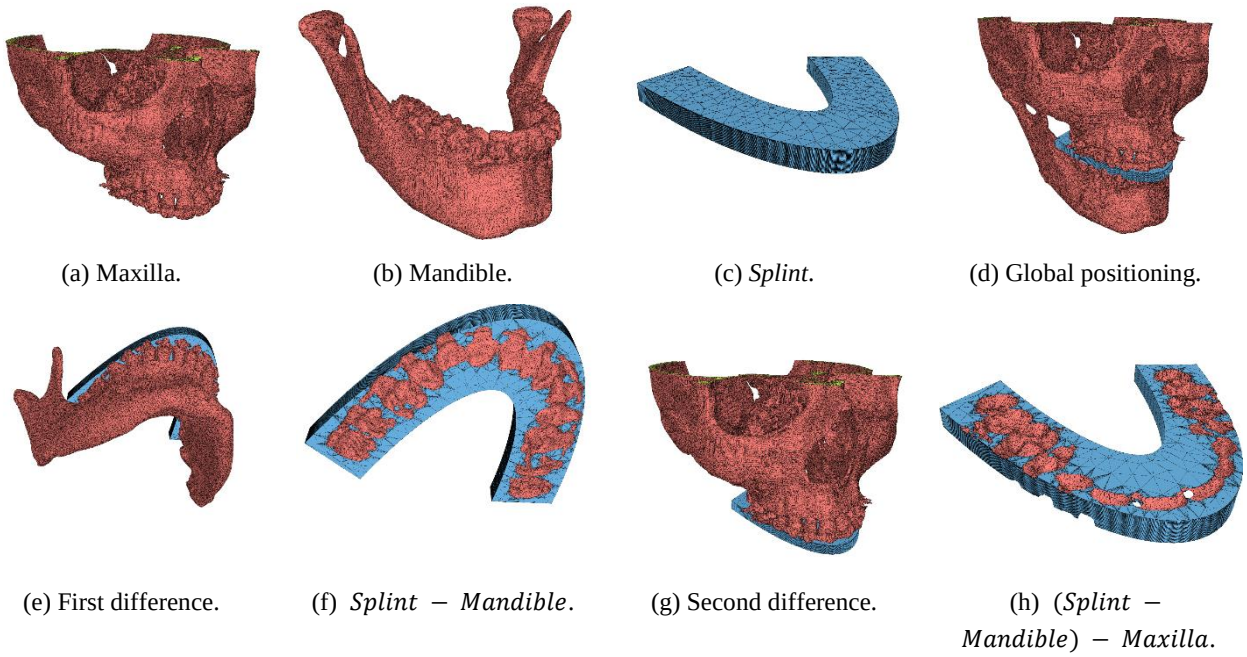
(a) $A \cup B$. (b) $A \cap B$. (c) $A - B = A^{\oplus}$. (d) $B - A = B^{\oplus}$.

Fig. 10. Boolean operations between the *Bunny* (model A in blue) and the *Scanned sphere* (model B in red). The yellow-green colour represents the backside of faces.

3D laser scan: sphere and bunny. The tests with the spheres presented above aim to compare the result of a 3D laser scan acquisition and its virtual representation. For that, both models have to be closed. Experimentation with the proposed algorithm on two different 3D laser scan acquisitions with potential holes was carried out for this test by using

[DOI: 10.1299/jamdsm.2017jamdsm0041]

the *Scanned sphere* of the previous test and the classic acquired model of Standard, the *Bunny*. The *Bunny* model is a hollow statue acquired by surface scan; thus, the reconstructed mesh contains holes. It contains exactly five holes of which four are under the feet and one under the head (Fig. 10)). The *Bunny* has been scaled and moved to place it on the *Scanned sphere* such that the four holes at the bottom are inside the sphere and the fifth under the head is outside. The processing times are presented in Table 5 and the results are shown in Fig. 10. The proposed algorithm produces the expected results as the holes of the *Bunny* model are preserved without modification of the meshes except in the colliding faces. The quality of the resulting mesh is similar with those realized by (Feito et al., 2013) or (Schifko et al., 2010) but these methods require closed shells as input.



(a) Maxilla. (b) Mandible. (c) *Splint*. (d) Global positioning.

(e) First difference. (f) $Splint - Mandible$. (g) Second difference. (h) $(Splint - Mandible) - Maxilla$.

Fig. 11. Tests of Boolean operations in the context of surgical application. (The yellow-green colour represents the backside of faces.)

Maxillofacial surgery: Maxilla, Mandible, and Splint. This third test under real conditions of use presents the application of the proposed method to the medical context. Boolean operations are an essential component in the recent application known as virtual surgery planning. For this experimentation, the building of an occlusal splint has been chosen. Using the *Maxilla* part (Max) (Fig. 11(a)) and the *Mandible* part ($Mand$) (Fig. 11(b)) of the patient, a negative of the teeth is inserted in a neutral *Splint* (S) (Fig. 11(c)). The negative of the teeth is made by the differences: $(S - Mand) - Max$. The result of the first difference is given in Fig. 11(f) and that of the second one in the Fig. 11(h). The results presented in Table 5 show, as expected, that optimization of the BVH constrained in the common bounding box and containing only the smaller set of faces from A or from B allows a fast construction of the spatial partition tree and a fast computation of intersections as reaching only pairs composed of A and B potentially colliding. The global time of the operation $S' - Max$ is similar with that of the operations between the *Scanned sphere* and the *ico-sphere* with eight subdivisions. However, the number of faces used in both cases is not comparable. For the sphere, the time consumption is mainly by the phases of computation of intersections as the common bounding box is wide (both objects are sharing the same space) when for the operations in the medical application, the time consumption is essentially used by the construction of the *OMC*s and the construction of $C_{InitCleaned}$. This example shows the advantage to use a BVH containing only one of the two sets of faces intersecting the common bounding box against an octree containing both sets. For the operation $S - Mand$, the computation time of the BVH is two times less than for the $S^* - Mand$. The main difference between those two operations is that the S^* contains the negative of faces of the Maxilla inserted in the Splint. This addition is comparable to the construction of the BVH with both sets of faces. Note: the symmetric operations show similar observation ($S - Max$ and $S' - Max$). Moreover, the BVH data structure is on average lighter and faster than the octree (dos Santos et al., 2014).

[DOI: 10.1299/jamdsm.2017jamdsm0041]

Finally, the order of the Boolean operations is arbitrary $((S - Mand) - Max$ or $(S - Max) - Mand)$. Therefore, two orders were run to control the quality of the result of the proposed method. That gives $S'' = (S - Mand) - Max$ and $S^{**} = (S - Max) - Mand$. Theoretically, S'' and S^{**} represent the same discrete surface and $S'' \oplus S^{**} = \emptyset$. The result (Table 5) shows that the meshes of S'' and S^{**} are numerically different, but the intersection is empty. The expected result is reached.

Table 2. Experimentation in virtual surgery planning application.

Operation	$\Delta_T(M)$	$\Delta_T(BVH)$	$\Delta_T(I)$	$\#CF$	$\Delta_T(R)$	$\Delta_T(OMC)$	$\#DL$	$\Delta_T(OB)$	$\Delta_T(G)$
$S' = S - Mand$	5.325	0.684	1.694	3536	2.735	6.332	12	1.090	17.865
$S'' = S' - Max$	22.334	3.141	3.173	3637	11.485	27.145	34	3.580	70.864
$S^* = S - Max$	20.679	2.270	3.146	3235	10.894	26.150	24	3.156	66.300
$S^{**} = S^* - Mand$	5.045	1.680	2.018	3950	3.061	7.403	22	1.504	20.717
$S'' \oplus S^{**}$	2.609	0.183	0.045	39	0.393	1.204	0	1.201	5.670

Models information:

S: *Splint* (5,129 vertices, 10,216 faces and 0 holes).

$Mand$: *Mandible* (237,606 vertices, 475,412 faces and 0 holes).

Max: *Maxilla* (1,021,179 vertices, 2,042,810 faces and 35 holes).

S': result of $S - Mand$ (20,893 vertices and 41,786 faces).

S^*: result of $S - Max$ (18,963 vertices and 37,934 faces).

S'': result of $S' - Max$ (34,969 vertices and 69,990 faces).

S^{**}: result of $S^* - Mand$ (34,974 vertices and 70,000 faces).

$S'' \oplus S^{**}$: 0 vertex and 0 face.

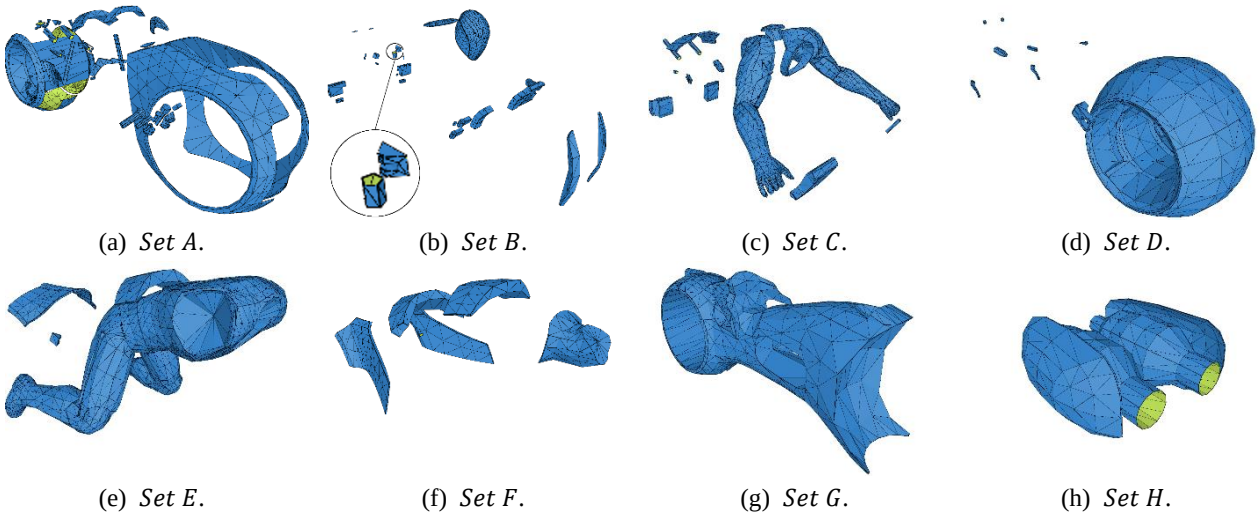
$S'' \cup S^{**}$: 34,976 vertices and 70,003 faces.

Legend: $\Delta_T(M)$: computing time in seconds of the creation of $C_{InitCleaned}$, $\Delta_T(BVH)$: computing time in seconds of the creation of the BVH, $\Delta_T(I)$: computing time in seconds of the intersections, $\#CF$: number of colliding faces, $\Delta_T(R)$: computing time in seconds of the remeshing of the intersections, $\Delta_T(OMC)$: computing time in seconds of the creation of *OMC*s and dividing lines, $\#DL$: number of dividing lines created, $\Delta_T(OB)$: computing time in seconds of the Boolean operation (classification and modifications), $\Delta_T(G)$: global computing time in seconds.

CAD: Tron light cycle. This last test is a CAD application with the particularity that we start from an existing mesh model that was initially designed for rendering and is not an object. This model is the *Tron light cycle*. It is composed of 11,061 vertices, 20,964 faces, and 141 shells, and it contains 117 holes. The main idea of this experience is to use the singularity of the proposed method to process non-manifold edges and shells with boundary edges to perform the construction of the represented object with a minimum of modifications. For that the error detection of the proposed method is used to guide the user to solve the errors of inconsistency. In the first step, the whole model is decomposed into height sets of shells without inner collision in each set. These sets are named $Set\ A, Set\ B, \ldots,$ and $Set\ H$. The fusion of these sets is achieved by the $\cup$ operator with a preservation of shells that are not colliding. For all errors of consistence of the Boolean operation, faces have been added to solve the problem targeted by the algorithm. During the fixing process, 84 small holes have been filled and two pieces of surface have been offset to solve non-manifold edges with odd adjacent faces. After these modifications, the model is composed of 12,154 vertices, 23,580 faces, and contains 30 holes. Figures 12(a)-(h) show the $Set\ A$ to H after these modifications and Fig. 13 shows the difference steps between each $\cup$ operation. After removing small shells, the resulting mesh is composed of 17,298 vertices, 35,040 faces, and two shells. The main shell and the engine part (Fig. 12(h)). There also remain six small holes which are opened to the outside of the main shell.

This test is meaningful because the input model contains several coplanar and nearly coplanar collisions (Fig. 13(h)), holes, and edge contacts that are challenging for existing Boolean operation algorithms to preserve a consistent topology after operations. When we performed this test with the proposed algorithm, we did not encounter any problem except the initial problems of inconsistency of the Boolean operations due to the geometry of the initial mesh. However, the series of operations has created 56 non-manifold edges due to edge collisions. This configuration combines at once the problems of shells with holes and non-manifoldness of edges that can be separately solved by existing methods.

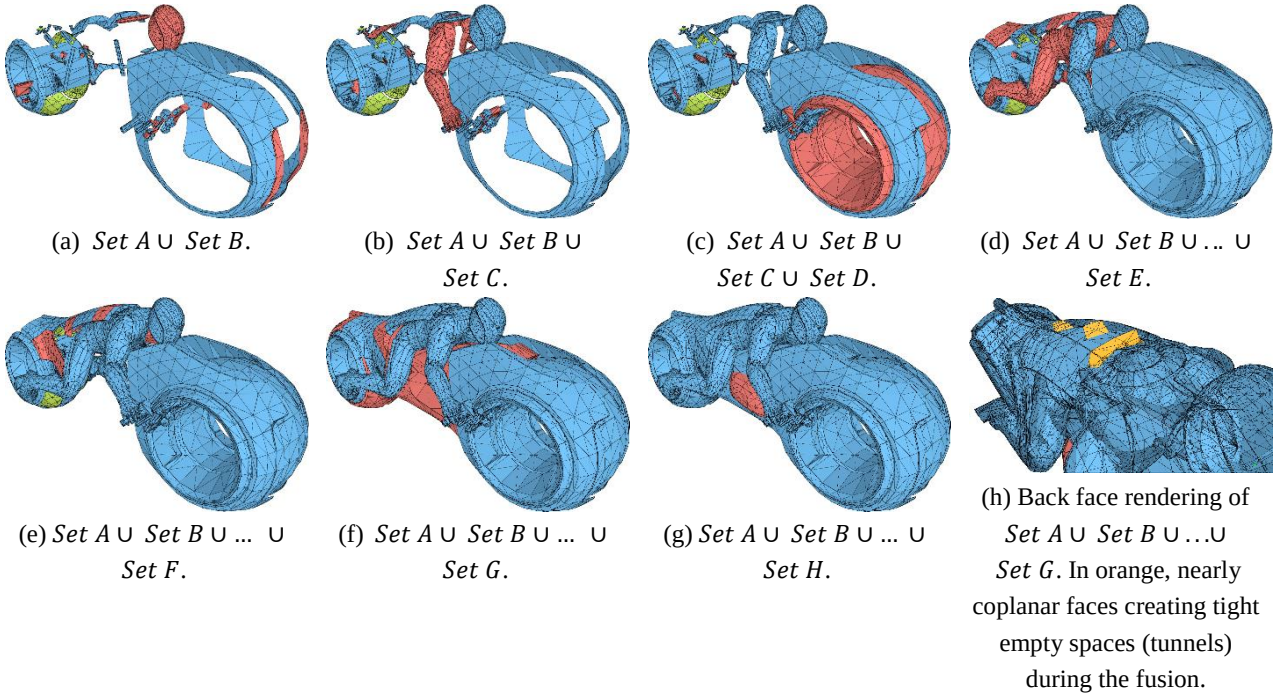
[DOI: 10.1299/jamdsm.2017jamdsm0041]



(a) $Set\ A$. (b) $Set\ B$. (c) $Set\ C$. (d) $Set\ D$.

(e) $Set\ E$. (f) $Set\ F$. (g) $Set\ G$. (h) $Set\ H$.

Fig. 12. Sets of shells of the *Tron light cycle* model without collision after minimal modifications to allow Boolean operations. (The yellow-green colour represents the backside of faces.)



(a) $Set\ A \cup Set\ B$. (b) $Set\ A \cup Set\ B \cup Set\ C$. (c) $Set\ A \cup Set\ B \cup Set\ C \cup Set\ D$. (d) $Set\ A \cup Set\ B \cup \ldots \cup Set\ E$.

(e) $Set\ A \cup Set\ B \cup \ldots \cup Set\ F$. (f) $Set\ A \cup Set\ B \cup \ldots \cup Set\ G$. (g) $Set\ A \cup Set\ B \cup \ldots \cup Set\ H$.

(h) Back face rendering of $Set\ A \cup Set\ B \cup \ldots \cup Set\ G$. In orange, nearly coplanar faces creating tight empty spaces (tunnels) during the fusion.

Fig. 13. Merging of the sets of non-colliding shells of the Tron light cycle model. (The yellow-green colour represents the backside of faces.)

6. Conclusion

This paper presents a method for Boolean operations (union, intersection, differences, and symmetric differences) between two colliding shells. This method is based on two main steps: merging of inputs and classification of the *OMC*s using one edge of each dividing line. This method, unlike existing methods, has demonstrated its capabilities to handle topological problems such as holes and non-manifold edges and large meshes (e.g., over two million faces for the Maxilla model). These capabilities have been demonstrated on specialized synthetic tests, medical data, 3D laser scan acquisitions, and CAD applications. The maxillofacial surgery test firstly shows the efficiency of the BVH data structure built with a minimal subset of faces from one of both of the input shells (A and B) to reach the potential colliding pairs of faces between A and B ; secondly, it shows the quality of its results in terms of experimentally reproducing a theoretical result $(((S - D) - X) \oplus ((S - X) - D) = \emptyset)$. As future work, we plan to extend this method to compute floating shells (bodies) as well as colliding shells. This method is expected to preserve the flexibility and robustness.

[DOI: 10.1299/jamdsm.2017jamdsm0041]

Acknowledgement

This work was supported by Korea Research Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (2015H1D3A1065744). This work was supported in part by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (2017-0-00955).

References

Antonio, F., Graphics gems III. Chapter Faster Line Segment Intersection. Academic Press Professional, Inc., San Diego, CA, USA, (1992), pp.199–202.

Attene, M., A lightweight approach to repairing digitized polygon meshes. The Visual Computer, Vol.26, No.11 (2010), pp.1393–1406.

Bernstein, G. Cork. https://github.com/gilbo/cork (2007).

Bernstein, G. and Fussell D., Fast, exact, linear Booleans. In Proceedings of the Symposium on Geometry Processing (2009), pp.1269–1278, Aire-la-Ville, Switzerland, Switzerland.

Blender, Online Community. Blender - a 3D modelling and rendering package. Blender Foundation, Blender Institute, Amsterdam, (2016).

Campen, M and Kobbelt, L., Exact and robust (self-)intersections for polygonal meshes. Comput. Graph. Forum, Vol.29, No.2 (2010), pp.397–406.

Chen, M., Chen, X.Y., Tang, K. and Yuen, M.M.F., Efficient Boolean operation on manifold mesh surfaces. Computer-Aided Design and Applications, Vol.7, No.3 (2010), pp.405–415.

Chen, Y., Robust and accurate Boolean operations on polygonal models. In ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol.2 (2007), pp.357–369.

Chong, C.S., Kumar A.S. and Lee H.P., Automatic mesh-healing technique for model repair and finite element model generation. Finite Elements in Analysis and Design, Vol.43, No.15 (2009), pp.1109 – 1119.

dos Santos, A.L., Teichrieb, V. and Lindoso, J., Review and comparative study of ray traversal algorithms on a modern gpu architecture. Proceedings: International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS Association, Vol.22 (2014), pp.203–212.

Feito, F.R., Ogayar, C.J., Segura, R.J. and Rivero, M.L., Fast and accurate evaluation of regularized Boolean operations on triangulated solids. Computer-Aided Design, Vol.45, No.3 (2013), pp.705 – 716.

Granados, M., Hachenberger, P., Hert, S., Kettner, L., Mehlhorn, K. and Seel, M., Boolean Operations on 3D Selective Nef Complexes: Data Structure, Algorithms, and Implementation, European Sympos. Algorithms, Springer Berlin Heidelberg, Vol.11, September 16-19 (2003), pp.654–666.

Hachenberger, P. and Kettner, L., 3D Boolean operations on nef polyhedra. In CGAL User and Reference Manual. CGAL Editorial Board, 4.8 edition, (2016).

Hapala, M., Davidovic, T., Wald, I., Havran, V. and Slusallek, P., Efficient stack-less bvh traversal for ray tracing. In 27th Spring Conference on Computer Graphics (2011).

Laurentjoye, M., Charton, J., Desbarats, P. and Montaudon, M., Mandibular surgery planning and 3d printed splint design. In International Journal of Computer Assisted Radiology and Surgery, Vol.9 (Suppl. 1) (2014), S253–54.

Mei, G. and Tipper, J.C., Simple and robust Boolean operations for triangulated surfaces. CoRR (2013), abs/1308.4434.

Möller, T., A fast triangle-triangle intersection test. Journal of Graphics Tools, Vol.2 (1997), pp.25–30.

Pavi, D., Campen, M. and Kobbelt, L., Hybrid booleans. Computer Graphics Forum Vol.29, No.1 (2010), pp.75–87.

Requicha, A.A.G. and Voelcker, H.B., Boolean operations in solid modeling: Boundary evaluation and merging algorithms. Proceedings of the IEEE, Vol.73, No.1 (1985), pp. 30–44.

Schifko, M., Jüttler, B. and Kornberger, B., Industrial application of exact Boolean operations for meshes. In Proceedings of the 26th Spring Conference on Computer Graphics, SCCG, ACM, (2010) pages 165–172, New York, NY, USA.

Shewchuk, J.R., Delaunay Refinement Mesh Generation. PhD thesis (1997), School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania. Available as Technical Report CMU-CS-97-137.

Thibault, W.C. and Naylor, B.F., Set operations on polyhedra using binary space partitioning trees. In ACM SIGGRAPH computer graphics, ACM, Vol.21 (1987), pp.153–162.

[DOI: 10.1299/jamdsm.2017jamdsm0041]