International Journal of Computer Assisted Radiology and Surgery
DOI 10.1007/s11548-017-1637-y

3D Boolean operations in virtual surgical planning

Jerome Charton · Mathieu Laurentjoye · Youngjun Kim

Received: 12 January 2017 / Accepted: 29 June 2017

Abstract
Purpose: Boolean operations in computer-aided design or computer graphics are a set of operations (e.g. intersection, union, subtraction) between two objects (e.g. a patient model and an implant model) that are important in performing accurate and reproducible virtual surgical planning. This requires accurate and robust techniques that can handle various types of data, such as a surface extracted from volumetric data, synthetic models, and 3D scan data.
Methods: This article compares the performance of the proposed method (Boolean operations by a robust, exact, and simple method between two colliding shells (*BORES*)) and an existing method based on the Visualization Toolkit (*VTK*).
Results: In all tests presented in this article, *BORES* could handle complex configurations as well as report impossible configurations of the input. In contrast, the *VTK* implementations were unstable, do not deal with singular edges and coplanar collisions, and have created several defects.
Conclusions: The proposed method of Boolean operations, *BORES*, is efficient and appropriate for virtual surgical planning. Moreover, it is simple and easy to implement. In future work, we will extend the proposed method to handle non-colliding components.

Keywords Virtual surgery planning · Boolean operations · Computer-aided surgery · Computer-aided design

PACS 1 · 2 · 03 · Surgical/Interventional informatics

PACS 1 · 3 · 05 · PACS-CAD integration

1 Introduction

Virtual surgery has recently become increasingly popular. It consists of the use of acquired patient volume data to simulate surgery by means of a virtual representation. The simulation can be employed for training [1,2,3] or for providing assistance during the actual surgery. Virtual surgery planning aims at controlling the human factor and simplifying the surgery process. This assistance can be simple visual support for guiding surgical instruments [4]. More recently, with advances in 3D printers, virtual surgery planning has evolved and become consolidated. It is presently replacing and becoming an extension of the current mechanical planning, i.e. the creation of surgical splints shaped to fit patient's anatomy. For instance, sagittal split ramus osteotomy (SSRO) of the mandible is a common procedure in jaw deformity correction. This surgery requires an articulator and positioning splints built using the plaster attached to the patient preoperatively, and the shaping of a condylar positioning device (CPD) intraoperatively [5]. The feasibility of the virtualisation of occlusal splints has been presented in [6]. Moreover, [7] presented a full virtual planning that includes the shaping of the CPD. Virtual surgery planning enhances accuracy [8,9].

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 research was supported in part by the KIST institutional program (2E26276, 2E26880).

Jerome Charton[1]
charton@lab327.net

Mathieu Laurentjoye[2,3]
mathieu.laurentjoye@yahoo.fr

Youngjun Kim[1]
junekim@kist.re.kr

[1] Korea Institute of Science and Technology
5, Hwarang-ro 14-gil, Seongbuk-gu, Seoul, 136-791 - Republic of Korea

[2] Surgical school of Bordeaux University
Campus Carreire, 146 rue Léo Saignat, 33076 Bordeaux - France

[3] Polyclinique Bordeaux Nord Aquitaine
15 rue Claude Boucher, 33000 Bordeaux - France

Virtual surgery planning uses various types of data. The main data are the surface extracted from volume acquisition. Starting from it, splints are created by adding synthetic shapes. These shapes may be primitives (e.g. cube, cylinder, sphere), as in [10], procedurally shaped using landmarks, as in [11], or transformations of the extracted surface itself, usually offsetting, as in [12]. Furthermore, some planning techniques, as in [13], use other types of acquisitions such as the 3D laser scan. That may be required in the case of the surfaces of teeth. The surfaces of teeth cannot be obtained from the volume data of the patient if teeth are in occlusion. In that case, teeth surfaces in virtual planning can be replaced by a mesh reconstructed from surface acquisition. Using and handling of such different types of data have to respect various mathematical properties that preserve a consistent definition of the interior and exterior of the objects during the entire processing chain.

The creation of splints is based on CAD (Computer-Aided Design) mechanics, in particular, mesh cutting for simulating osteotomies or cropping pieces, and Boolean operations (i.e. union, intersection and difference) for combining the various objects and eventually generating the expected splints. However, the wide variety of data yields complex definitions of the surfaces. Surface extraction from the volume data or/and Boolean operations may generate singular edges (edges with more than two adjacent faces or with inconsistent orientations). Multiple Boolean operations may result in large coplanar collisions and partial duplication of surfaces. Moreover, surface acquisition, e.g. 3D laser scan, may generate a partial surface of the acquired object.

Therefore, we need a Boolean operation method that can handle holey and/or non-manifold surfaces (with singular edges) and is capable of reporting possible inconsistences. This paper presents a practical evaluation of a method presented in [14] (called *BORES*). Moreover, this method will be compared with a pending version of another open-source library supporting holey meshes (called *VTK**), which is a Visualization Toolkit (VTK) implementation [15] based on the method of [16]. In this article, we will use the pending version of *VTK* called "vtk-new_boolean" based on *VTK* version 7.1.0 in [15] (called *VTK**), the current version of VTK (7.1.1) that has implanted one of the two modified classes from [15] (i.e. vtkIntersectionPolyDataFilter) as keeping the previous method [17] and the previous released version (7.0.0) that fully uses the previous method without the modification proposed in [15].

2 Related work and positioning

Boolean operations take as input two objects and create a new object, as follows.

- Union: merging of the interiors of the input objects.
- Intersection: extraction of the common interior of the input objects.
- Difference (or subtraction) $A - B$: extraction of the common interior from the object A.

Even though these operations are trivial when volume representations of objects are used, they become complex when boundary representation (B-rep) such as the mesh is used. Thus, a robust way to obtain the Boolean operations between two triangle meshes involves a conversion (complete or partial) of these surface representations to volumetric representations, e.g. *binary space partitioning (BSP)* [18], *voxel gird* [19,20,21,22,23]. However, these conversions have several drawbacks. They are computationally inefficient, cause loss of information, and require surfaces without holes.

Other approaches, using directly the B-rep, construct a space partitioning tree (e.g. octree, kd-tree, bounding volume hierarchy (BVH)) wrapping the triangle set to reach faces efficiently. These approaches usually consist of two steps. The first step aims at computing the intersections between the triangles of the object A with these of the object B to obtain and integrate the colliding segments in the data structure. The second step is the classification of the parts of a surface with the following strategies: preservation, removal, or reversion, according to the expected Boolean operation. In this respect, ray tracing with parity counting [24, 25,26], plane sweeping with a dynamic *real tree (R-tree)* [27], or spatial partitioning [28,16] are often used.

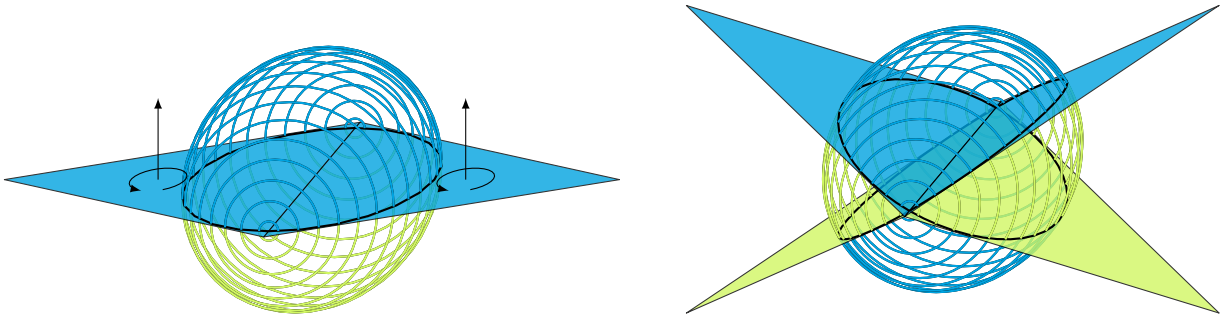
The two methods compared in this paper (*BORES* and *VTK**) use directly the mesh. These methods are distinguished by the way they handle open surfaces. As seen in Section 1, virtual surgery planning uses various types of data that may have holes and singular edges with the following input requirements:

- Each face of both input shells is consistently oriented. The front side is oriented to the outside and the backside to the inside (Fig. 1(a)).
- 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 in a half-disc. Then, all partitions of $\mathscr{B}$ obtained by intersection with neighbouring faces of e can be classified as either the interior or exterior of the object according to the orientation of the splitting faces (Fig. 1(b)).

Both methods consist of the two aforementioned steps. However, these steps are carried out differently in each method. This affects the data structure during the processing and therefore the result.

3 Methods

As other methods using directly the mesh data structure, *BORES* and *VTK** consist of two main steps. The first step



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

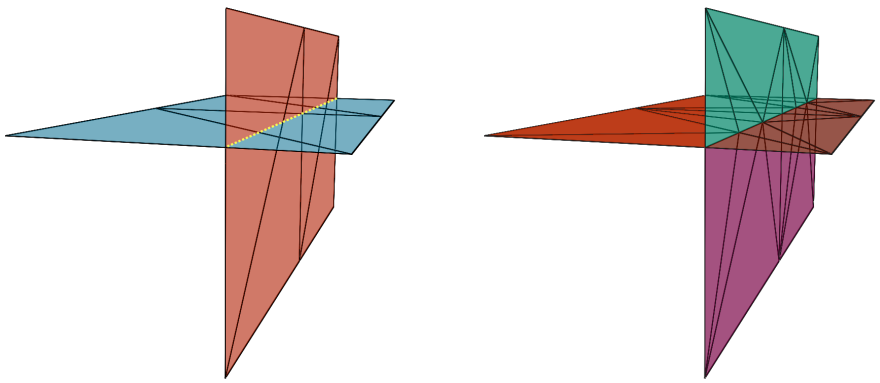
Fig. 1 Examples of edges with consistent definition of the interior (green) / exterior (blue) (figure from [14]).

creates a merging mesh of the input data by removing all geometrical duplication, and the second step classifies each component of the mesh in order to perform the expected operation.

Merging of input data Merging of inputs, for *BORES*, starts by embedding all triangles composing the models A and B, with tags ω_A and ω_B, respectively, in the same data structure, while preserving their origin (Fig. 2(a)). The merged mesh contains (self-)intersections between triangles from A and B. An intersection between two triangles can be classified in two categories: coplanar and non-coplanar. If the collision is coplanar, the collision segments are the edges of the polygon bounding the common area of the two colliding triangles [29]. In the case of a non-coplanar collision, the collision is a simple segment, possibly null [30]. Using these colliding segments, all colliding faces are remeshed by a constrained Delaunay triangulation [31], and the segments become edges linking faces from both input models. We note that for coplanar collisions, remeshing of the common area should be the same for both triangles (Fig. 2(b)). After remeshing, duplicate faces may remain, due to coplanar collisions. In that case, faces tagged ω_B are removed, and those tagged ω_A are preserved; however, their tag changes to ω_C if the removed face had the opposite orientation, and ω_D otherwise (Fig. 3). The mesh resulting from merging, called C, does not contain duplicate data, but rather several singular edges forming chains connecting parts of surfaces from the models A and B. For *VTK**, this merging is similar to that of *BORES*. However, it does not take in account coplanar collisions. The result of merging in *VTK** is a combined mesh of the two inputs with several singular edges forming loops (as in *BORES*) without non-coplanar collisions, but possibly with coplanar collisions, duplicate faces, and duplicate vertices.

We note that at the end of this step, if a bordering edge collided with the inner part of a surface of the second mesh, then an open chain (using the terminology of [15]) of singular edges would be created. In that case, *BORES* considers the Boolean operation inconsistent and reports this anomaly, whereas *VTK** continues processing.

Component classification The classification step will determine which patches should be preserved, reversed, or re-



(a) (b)

Fig. 2 Simple example of intersection between two shells (figure from [14]). (a) Two shells (A in blue and B in red) are colliding. The yellow dotted line is the intersection line. (b) After computation and remeshing of intersections, we obtain C. C can be decomposed in patches rendered in different colours.

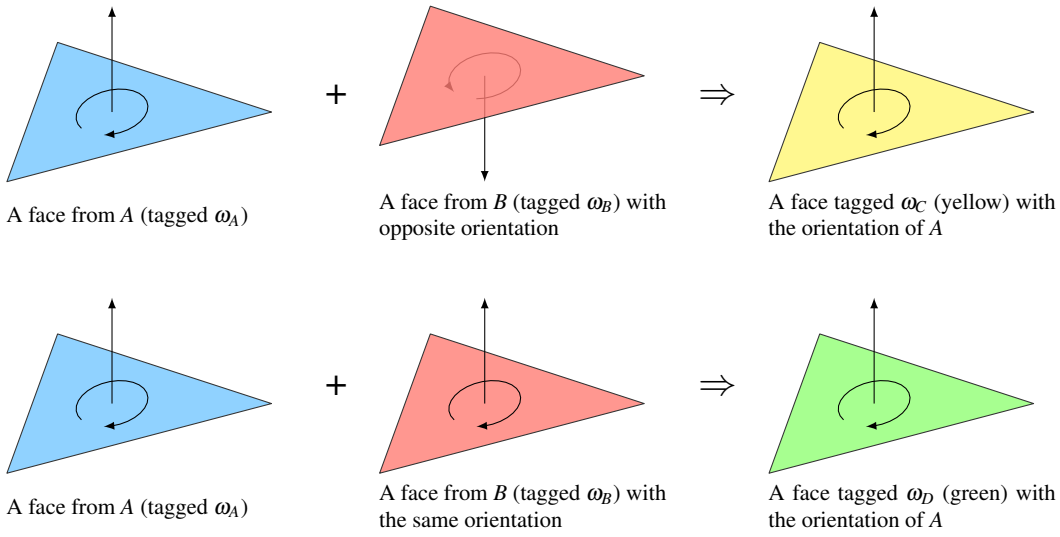


Fig. 3 Applying tags ω_C and ω_D on duplicate faces. For duplicate faces, the face tagged ω_B is removed and that tagged ω_A changes to ω_C or ω_D.

moved. Therefore, both methods define different delimiters among the patches. *BORES* creates chains of singular edges and *VTK** loops of singular edges. A chain is defined by a set of connected singular edges. Specifically, let v be an inner vertex of a chain c, then v is adjacent to exactly two singular edges, and these edges are links of c. If a vertex is adjacent to one or more than two singular edge(s), it is called extremity. A loop is an oriented sequence of singular edges that may be classified as hard closed, soft closed, or open. The orientation of a loop is defined by the orientations of its related faces (clockwise or counter-clockwise). A hard closed loop is composed of only one chain without extremities. A soft closed loop is composed of several chains shaping an actual loop. Each vertex of a soft closed loop is connected to exactly two edges of this loop. However, it may be contained in several other loops. An open loop is a non-cyclic sequence of oriented singular edges.

In *BORES*, a random edge is chosen to represent an entire chain, and around this edge interval angles are classified as common outside, common inside, inside exclusive of A, and inside exclusive of B, using the tags and orientations of the adjacent faces. This classification is used to determine the expected inside according to the requested Boolean operation. Finally, only patches surrounding the expected inside are preserved and re-oriented, if necessary, towards the outside. We note that a singular edge can be adjacent to more or less than four faces. If the angular classification is not consistent (e.g. in case of odd adjacent faces and inconsistent definition of interior/exterior of the inputs), the Boolean operation is not consistent and the problem is reported.

In *VTK**, all singular edges are considered adjacent to four faces. Therefore, a loop may classify two groups of two adjacent patches as clockwise and counter-clockwise. For each group, both patches are from different inputs. One of them is a component of the intersection surface, and the other is a component of the union surface. Using this decomposition, the bounding boxes of each patch, and face orientations, the expected Boolean operation is performed by preserving the largest patch of the two groups, with a special treatment for the open loops, which are processed last.

Theoretical comparison *BORES* and *VTK** adopt similar approaches; however, they differ in the definition of contexts and tolerances.

In the context of *BORES*, inputs may contain singular edges and have coplanar collisions. Regarding open surfaces, *BORES* can determine whether Boolean operations are consistent, for example, when a bordering edge of A collides with the inside of a triangle of B. If the creation of a surface by Boolean operations is impossible, *BORES* detects and reports the problem. A problem in the operation is reported:

- When a bordering edge is colliding with any object that is not another bordering edge,
- When the classification of angular intervals cannot be performed, and
- When a patch is classified in different categories at different edges.

This detection of anomalies ensures that a problem preventing the Boolean operation will be identified and reported.

In the context of *VTK**, inputs can neither contain singular edges nor have coplanar collisions. *VTK** considers that all Boolean operations having collision between the two inputs have solutions and applies a special treatment for open loops.

BORES appears more versatile with respect to the input requirement, whereas *VTK** is more versatile with respect the feasibility of the Boolean operation itself.

Preliminary experimentation When applied to synthetic data, namely the *three cubes* with the U, the two methods yield the results shown in Figs. 4 and 5. This initial test involves coplanar collisions of both orientations, with and without singular edges. *BORES* performs the operations without creating error in the output meshes and yields the expected result, whereas *VTK** creates multiple empty volumes (coplanar faces of opposite orientations) and performs some partially wrong classifications of triangles, creating holes in the union operation and unexpected faces in the intersection operation.

4 Clinical cases

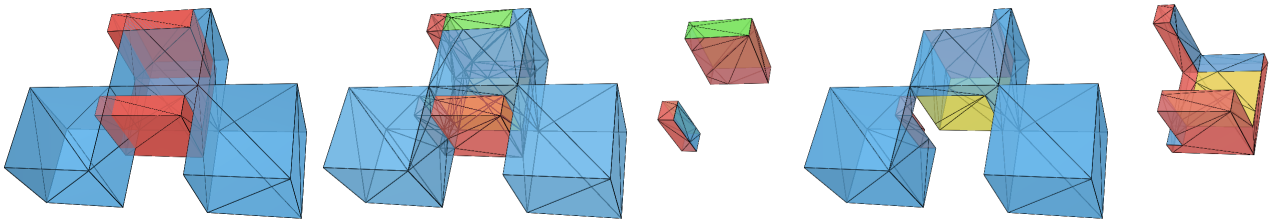
In that follows, the two methods will be tested on real data used for two types of surgeries. The first is mandibular reconstruction [32]. The second is orthognathic surgery by sagittal split ramus osteotomy (SSRO).

4.1 Mandibular reconstruction

In this surgery, a damaged/missing part of mandible is replaced/filled by (a) section(s) of the fibula bone. To this end, during planning, cuts of the fibula are prepared to fit into the hole prepared in the mandible. In the present case (Fig. 6(c)), a unique section of the fibula is planed to replace a lateral part of the mandible. To cut these two bones, two objects (a cuboid and a bent cuboid, both coloured green in Fig. 6, and called Cut_0 and Cut_1, respectively) have been prepared to model the cutting trajectories of the surgical instrument. The simulation of the cuts is preformed by the subtraction of the cutting objects from the fibula (Fig. 6(b)) and the mandible (Fig. 6(a)).

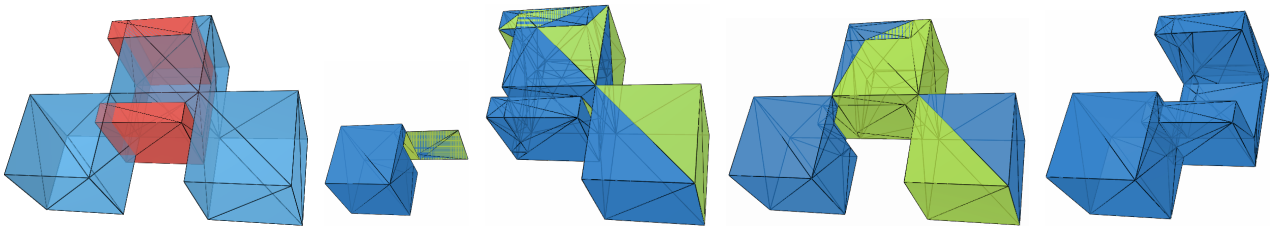
Tables 1 and 2 present the description of results and the processing time for each operation, respectively. After the Boolean operations have been performed using *BORES*, the obtained results match the expected ones. Both results are composed of three shells and do not contain any defects (holes, singularities, or self-intersections). Using *VTK* methods, only the operation on the mandible with implementation 7.0 yields a correct result. All the others are wrong, and the models are corrupted or do not produce output. The results are significantly different even though *BORES* is in the computation time interval given by *VTK* based methods where *VTK** is ahead.

After extraction of targeted shells from the output of the Boolean operations performed by *BORES*, we obtain the expected result (Fig. 7).



(a) *Three cubes* (A in blue) and U (B in red). (b) After merging, we obtain C. (c) $A \cap B$. (d) $A - B$. (e) $B - A$.

Fig. 4 Boolean operations using *BORES* with *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 (figure from [14]).



(a) *Three cubes* (A in blue) and U (B in red). (b) $A \cup B$. (c) $A \cap B$. (d) $A - B$. (e) $B - A$.

Fig. 5 Boolean operations using *VTK* with *Three cubes* (model A in blue) and U (model B in red). For (b)-(e), blue is the outer facing and yellow-green is the inner facing.

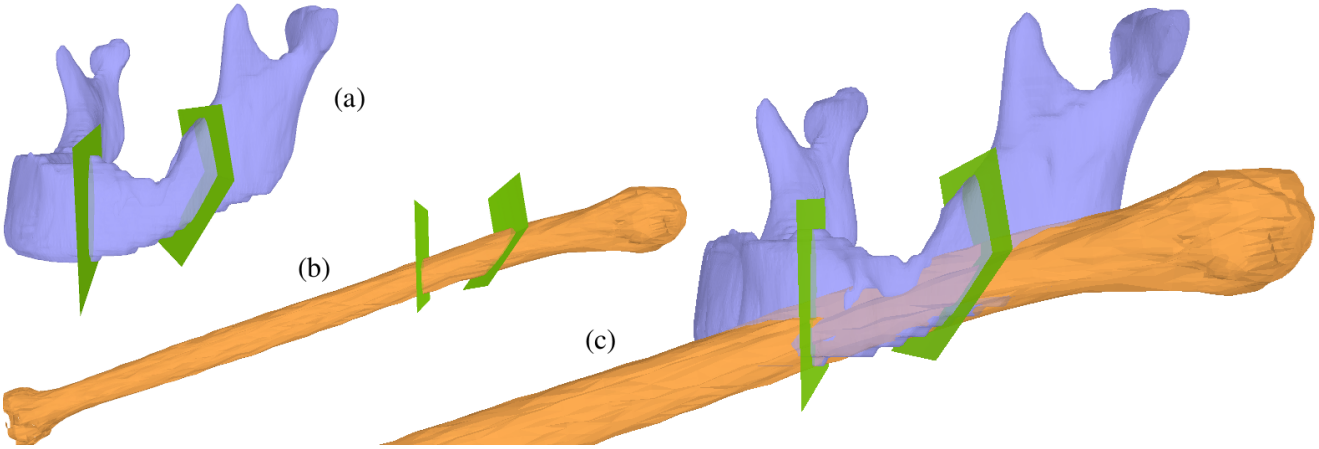


Fig. 6 Positioning of the objects in virtual surgery planning of mandibular reconstruction. (a) The mandible with the cutting shapes. (b) The fibula with the cutting shapes. (c) Global positioning.

Table 1 Properties of the results of the Boolean operations in mandibular reconstruction surgery (information given by Geomagic Studio ®). Where "X" is used when there no output data to analyse. Note: *VTK* based methods producing several duplicated elements, those have been removed to obtain this table.

Models cut	Methods	#Singular edges	#Self-inter.	#Holes	#Gaps	#Shells
$(Mandible - Cut_0) - Cut_1$	*BORES*	0	0	0	0	3
$(Fibula - Cut_0) - Cut_1$	*BORES*	0	0	0	0	3
$(Mandible - Cut_0) - Cut_1$	*VTK**	1	2	0	3	3
$(Fibula - Cut_0) - Cut_1$	*VTK**	X	X	X	X	X
$(Mandible - Cut_0) - Cut_1$	*VTK* 7.1	2	45	10	1	4
$(Fibula - Cut_0) - Cut_1$	*VTK* 7.1	X	X	X	X	X
$(Mandible - Cut_0) - Cut_1$	*VTK* 7.0	0	0	0	0	3
$(Fibula - Cut_0) - Cut_1$	*VTK* 7.0	1	40	3	7	7

Table 2 Processing time (in sec) of the Boolean operations in mandibular reconstruction surgery.

Operations	*BORES*	*VTK**	*VTK* 7.1	*VTK* 7.0
$Mandible - Cut_0$	3.196	2.400	5.793	3.852
$(Mandible - Cut_0) - Cut_1$	3.499	1.542	5.498	3.805
$Fibula - Cut_0$	0.638	Failed	Failed	0.674
$(Fibula - Cut_0) - Cut_1$	0.633	Failed	Failed	0.828

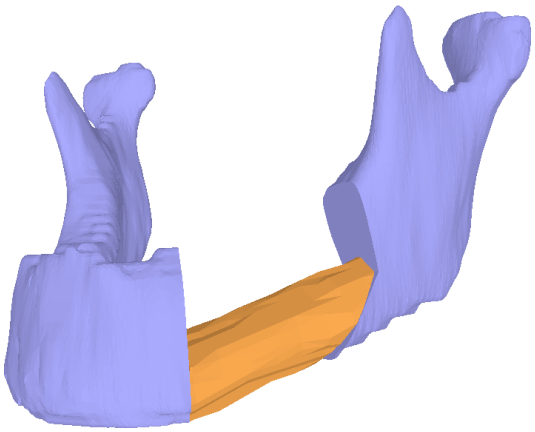
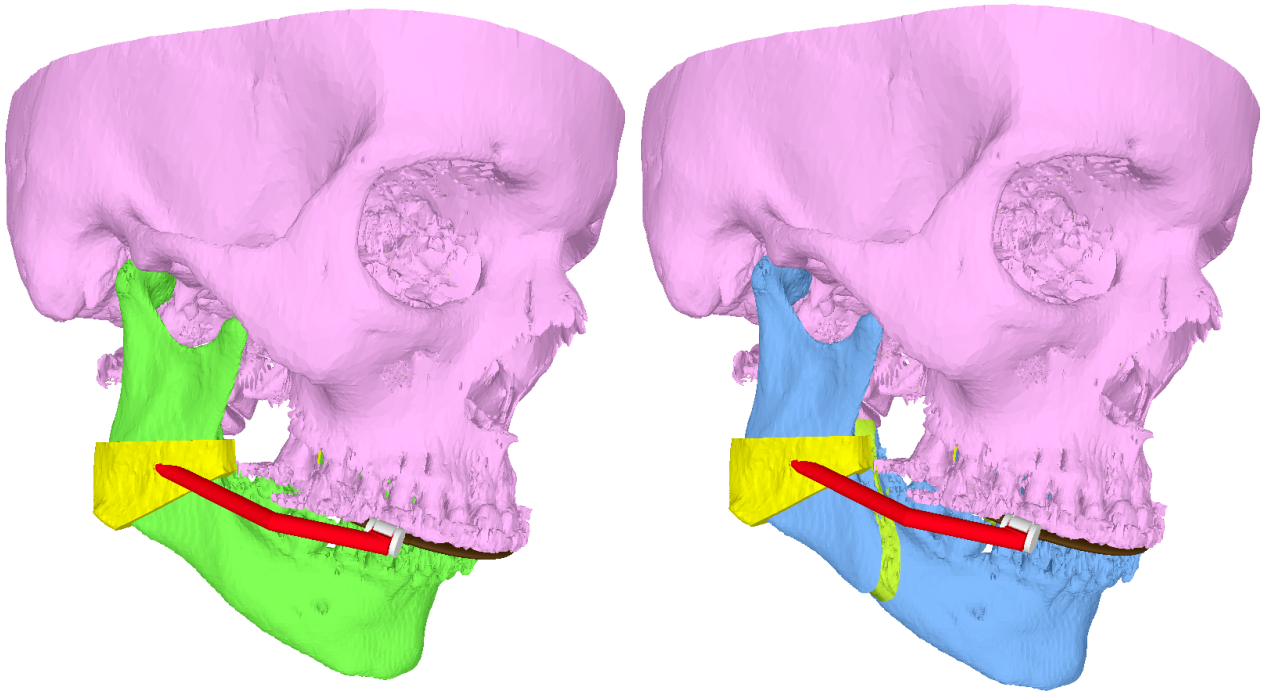


Fig. 7 Planning of positioning of the section of the fibula in the mandible using the result of the Boolean operations by *BORES* method.

4.2 Orthognathic surgery

Orthognathic surgery by SSRO requires occlusal splints to define the various occlusions during the surgical process and condylar positioning devices (CPDs) to fix the positions of the condyles. To this end, [7] proposes a virtual surgery planning using a guide combining simultaneously the occlusal splint (white and brown in Fig. 8) and the CPDs (red and yellow in Fig. 8). This guide is composed of multiple primitive shapes and offsets of partial surfaces of the mandible (Fig. 9). The combination of this guide is accomplished by union operations among its elements. Its adaptation is made by subtractions of the teeth in the occlusal splint and the mandible in the CPD parts. Table 3 reports the processing time of these Boolean operations. For this experience, all operations use theoretical input data. In order to simplify reading, a tag ✓ or ✗ indicates whether the result is correct and without defect or not, respectively. As seen in this table, none of the operations performed by *VTK* based methods generate a correct result. In particular, none of them



(a) Initial position. (b) Final position.

Fig. 8 Initial and final positions with guides. This figure shows the initial and final phases of planning for the Boolean operations required to generate the splints. **Legend.** pink: *Maxilla*, green (a): initial mandible ($Mand_{Init}$), blue (b): final mandible (after SSRO and moving) ($Mand_{Final}$ for the teeth part and $Cond_{R/L}$ for the condylar sections), yellow: plates ($Plate_{R/L}$), red: frames ($Frame_{R/L}$), white: clips, dark brown: splints, and yellow-green (b): inner side of faces.

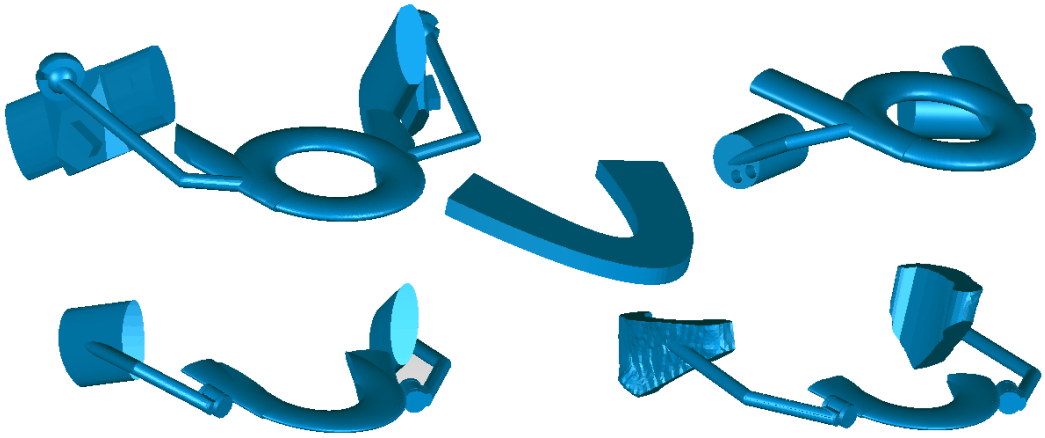


Fig. 9 Examples of guides for orthognathic surgery.

return a result without defects, even after merging duplicate vertices. In addition, this test shows up the high instabilities of *VTK* 7.1 and *VTK** with high rate of failure. Moreover, *BORES* has performed Boolean operations as yielding the correct result when the input configuration was consistent and warns of the impossibility of the operation by targeting the issue (Fig. 10). We note that, for *VTK* 7.0 and 7.1, since these methods are not handling holes, watertight models have been used and these methods have been excluded for $Plate_{R/L} - Cond_{R/L}$ tests. These tests have mainly been built to analyse the result of operations with holes. As this experiment used bigger data than the previous one, computation time increased, highlighting the difference in performance between the methods. *BORES* has a processing time significantly shorter than that of *VTK* 7.0 and 7.1 (about two or three times shorter) and close, but longer than that of *VTK**.

To have a simple overview of the defects from the operations performed by *VTK* methods, the process of inserting the negative surfaces of upper teeth into the neutral *Splint*

Table 3 Processing time (in sec) of the Boolean operations in orthognathic surgery.

Operations	*BORES*	*VTK**	*VTK* 7.1	*VTK* 7.0
$Plate_R - Mand_{Init}$	7.071 ✓	Failed	Failed	21.499 ✗
$Plate_L - Mand_{Init}$	7.401 ✓	7.071 ✗	25.317 ✗|✓	18.855 ✗|✓
$Frame_{R_0} \cup Frame_{R_1}$	1.426 ✓	Failed	Failed	2.752 ✗
$(Frame_{R_0} \cup Frame_{R_1}) \cup Frame_{R_2}$	0.585 ✓	Failed	Failed	0.589 ✗
$Frame_{L_0} \cup Frame_{L_1}$	2.161 ✓	1.043 ✗	4.350 ✗	3.286 ✗
$(Frame_{L_0} \cup Frame_{L_1}) \cup Frame_{L_2}$	1.856 ✓	Failed	Failed	2.287 ✗
$(Plate_R - Mand_{Init}) \cup Frame_R$	1.850 ✓	Failed	Failed	3.618 ✗
$(Plate_L - Mand_{Init}) \cup Frame_L$	2.583 ✓	1.180 ✗	4.350 ✗	4.317 ✗
$Plate_R - Cond_R$	$Error_1$	Failed	Holey test	Holey test
$Plate_L - Cond_L$	$Error_1$	Failed	Holey test	Holey test
$Splint - Maxilla$	25.485 ✓	21.467 ✗	126.363 ✗	98.309 ✗
$(Splint - Maxilla) - Mand_{Init}$	11.785 ✓	Failed	Failed	42.980 ✗
$(Splint - Maxilla) - Mand_{Final}$	10.143 ✓	Failed	Failed	44.990 ✗
$Splint_{Init} \cup Clips$	3.803 ✓	Failed	Failed	4.565 ✗
$Splint_{Final} \cup Clips$	3.602 ✓	Failed	Failed	4.836 ✗

- $Frame_{L/R} = \sum_i Frame_{L_i/R_i}$, where $Frame_{L_i/R_i}$ is the i-th component of the $Frame_{R/L}$.
- $Splint_{Init} = (Splint - Maxilla) - Mand_{Init}$.
- $Splint_{Final} = (Splint - Maxilla) - Mand_{Final}$.
- $Error_1$: An error has been detected. A bordering edge is colliding with the inner part of the surface.
- ✓: Result correct and without defects.
- ✗: Result incorrect and/or with defects.
- ✗|✓: Correct result and without defects after removal of small shells and duplicate vertices and faces.

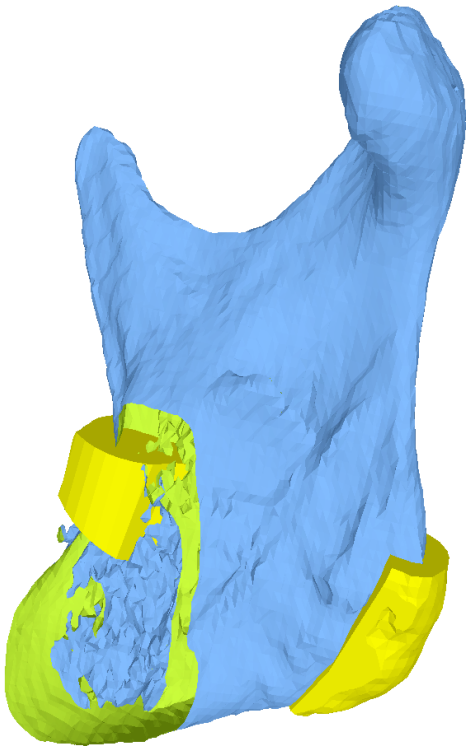
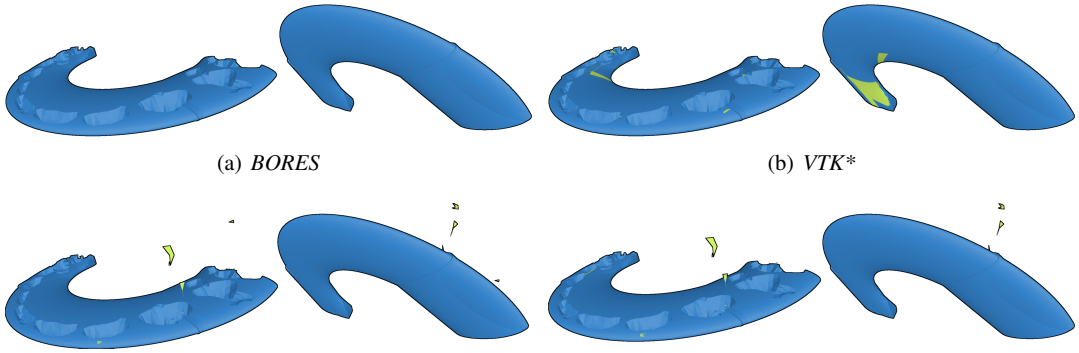


Fig. 10 Example of inconsistent input to a Boolean operation according the requirements of *BORES* method. Yellow: the right plate $Plate_R$. Blue: the right condyle $Cond_R$ after SSRO. Yellow-green: the inner side of faces. The bordering edges of the $Cond_R$ surface intersect the inner part of the surface of the plate. In that case, the Boolean operation cannot generate a correct result.

model is presented in Fig. 11. In this example, we can see that even in this first tooth surfaces extraction, *VTK* methods create multiple defects that accumulate through the series of operations resulting in the dislocation of the objects or drive to the impossibility of performing a result.

4.3 Experiments

In addition to the previous experiments, the two methods *BORES* and *VTK** (with *VTK* 7.0 and 7.1 as control methods of *VTK**), have been tested on several surgery plannings. These tests consist of seven plannings in mandibular reconstruction and seven in orthognathic surgery (including creation of cutting guides). The orthognathic surgery used data in [7] and [10]. As in Section 4.1, these plannings consist of initial and final positions of mandible and/or maxilla. Using this positioning, guides are created by union operations between the components of the guides and difference operations between the guides and the bone components. However, as guides are shaped on the patient's anatomy and these surgeries have been performed for a long time, the guide shapes and their structure have evolved as well. The guides are mainly composed of three elements: the splint, the frames and the plates (as seen in Section 4.2). Plates have changed shape several times and were generated by synthetic shapes (transformed cylinder) or by using offsets of bone surface. The frames linking the plates and the splint can be clipped or merged in one of the two extremities. Due to this variety of procedures for generating guides, we



(a) *BORES* (b) *VTK** (c) *VTK* 7.1 (d) *VTK* 7.0

Fig. 11 Comparison between *BORES*, *VTK**, *VTK* 7.1 and *VTK* 7.0 in the incrustation maxilla teeth into the neutral splint (*Splint* − *Maxilla*). Defects (after merging duplicate vertices for *VTK* methods): (a) 0 hole, 0 self-intersection, and 0 gap. (b) 5 holes, 34 self-intersections, and 89 gaps. (c) 17 holes, 26 self-intersections, and 109 gaps. (d) 14 holes, 104 self-intersections, and 163 gaps.

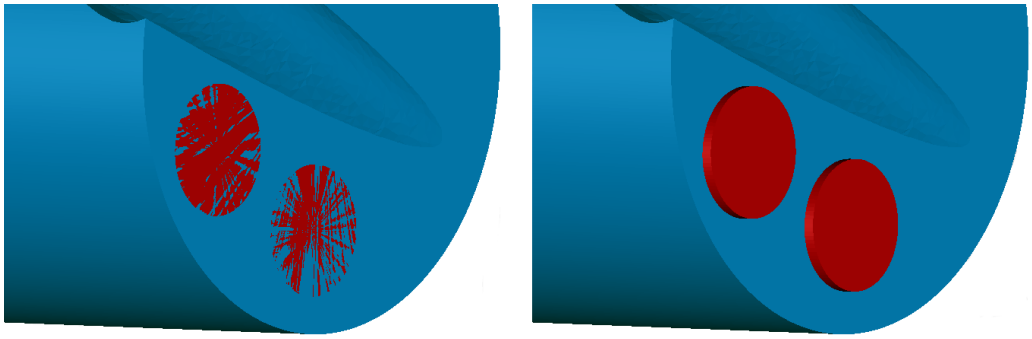


Fig. 12 In order to avoid coplanar collisions as much as possible, hole cylinders have been lengthened in the planning of orthognathic surgery 05 (Left: original data. Right: after modifications).

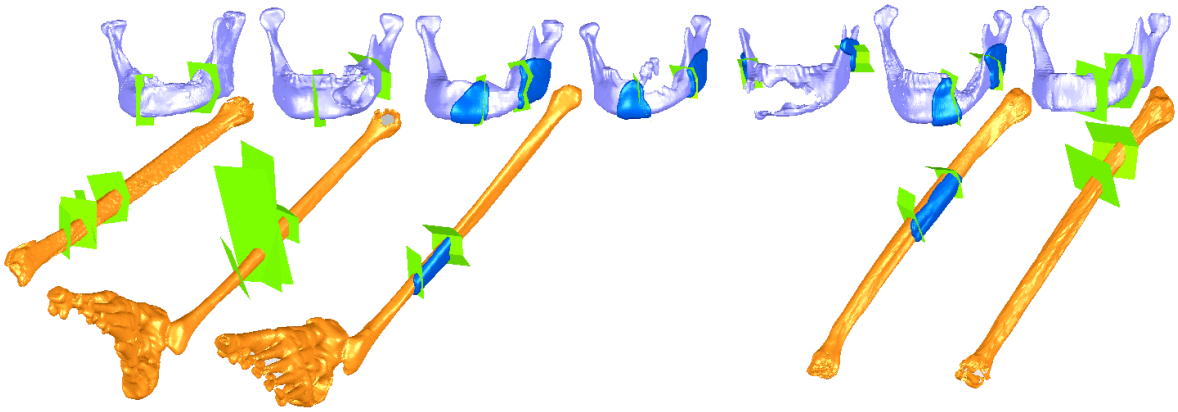


Fig. 13 Examples of surgery plannings of mandibular reconstruction surgery using the fibula bone.

have a high variability of mesh characteristics for performing the Boolean operations. Orthognathic surgery uses two bones (fibula and mandible), offsets the surfaces that will be used to create guides (called *GFi* and *GMi*), and cutting objects (seen in Section 4.1 and called *CFi* and *CMi*). While bone surfaces and offsets are reconstructed data, all cutting shapes are synthetic and their triangulation has been performed to fit in with their edges (Fig. 13). Since *VTK** cannot theoretically support coplanar collisions, a modification of the original data of orthognathic surgery 05 has been made to avoid obvious coplanar collisions (Fig. 12) and singular edges have been removed. We note that the initial data are free of self-intersections and degenerate faces.

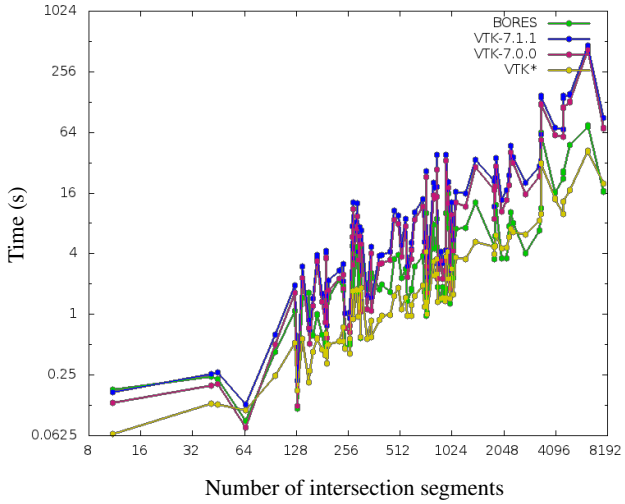


Fig. 14 Time processing according to the number of intersection segments (using scales in log_2).

5 Results and discussion

Using the data presented above, we obtain 155 Boolean operations. For all these operations, the output should be without holes or self-intersections. The evaluation of implementations (*BORES*, *VTK**, *VTK* 7.1 and 7.0) has been performed by running each Boolean operation 10 times to obtain the mean processing time. For the result of each operation, the output has been analysed to report the number of self-intersections and bordering edges. Furthermore, each planning is a sequence of Boolean operations. This implies that the result of an operation may be the input of a subsequent operation. In order to avoid the accumulation of defects and have a correct observation of each operation, these operations have been initially run as a sequence of the tested method and subsequently by using theoretical inputs.

Speed By their principles, all tested methods devote most of the processing time to the computation of collisions. Therefore, their processing time depends on the number of colliding faces. Figure 14 shows the processing time of the four implementations according to the number of intersection segments, when all methods generate a result. These curves show that under 0.5 s, the four implementations are similar, with a slight advantage to *VTK* 7.0 and *VTK**. However, over 0.5 s, *VTK** is significantly faster than the others, then *BORES* than *VTK* 7.0 and 7.1. On average *BORES* and *VTK** are 5 times and up to 10 times faster than *VTK* 7.0 and 7.1. In addition, the *VTK* 7.1 and *VTK** methods spent more than 1 h for two tests (mandibular reconstruction surgery 03: $GM1 - CM1$ and orthognathic surgery 06: $Plate_{R_0} \cup Plate_{R_1}$).

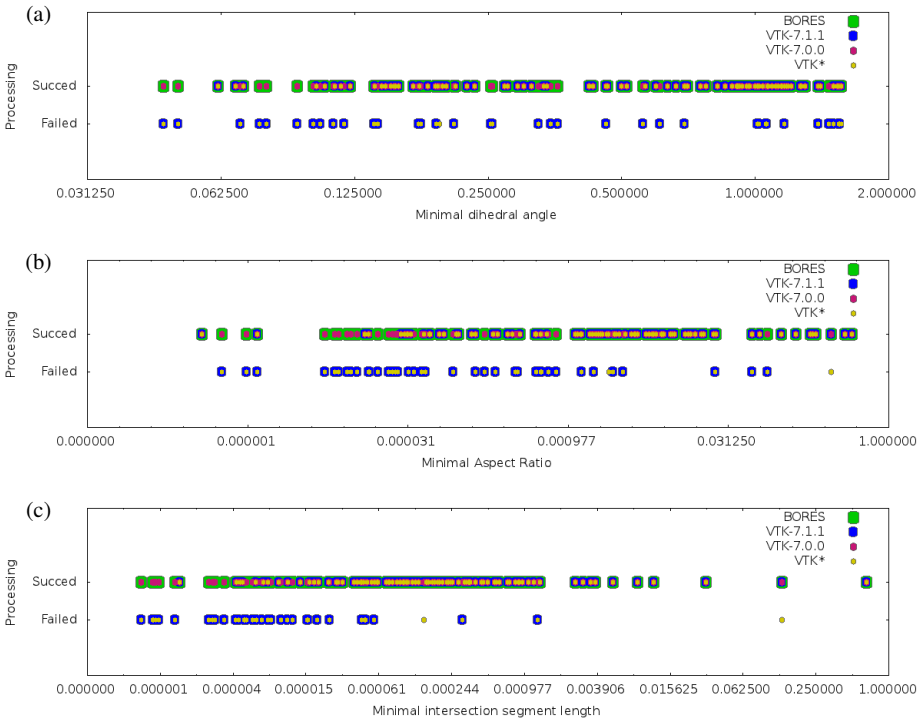


Fig. 15 Plots of successes/fails according to (a) the minimal dihedral angle between colliding faces, (b) the aspect ration of colliding faces, and (c) the minimal length of intersection segments (using x-scale in log_2).

Robustness The robustness of the implementations can be evaluated by the tests using the theoretical inputs. These tests yielded successes (the test generated an output) of

- $155/155 = 100\%$ for *BORES*,
- $99/155 \simeq 63.87\%$ for *VTK**,
- $101/155 \simeq 65.16\%$ for *VTK* 7.1, and
- $155/155 = 100\%$ for *VTK* 7.0.

In terms of successful input/output, the results were

- $155/155 = 100\%$ for *BORES*,
- $89/155 \simeq 57.42\%$ for *VTK**,
- $91/155 \simeq 58.71\%$ for *VTK* 7.1, and
- $147/155 \simeq 94.84\%$ for *VTK* 7.0.

VTK 7.1 and *VTK** present a higher sensitivity than them predecessor (7.0). Its failing cases terminate with a "segmentation fault". The principal sensitivity factors of approaches using directly the mesh are the dimensions of colliding triangles, length of the intersection segments, and dihedral angles between colliding faces. In order to find a correspondence between these factors and the sensitivity of *VTK* 7.1 and *VTK**, Fig. 15 shows successes/fails of the studied methods according to different factors. Even though (b) shows a slight impact of the aspect ratio on the success of *VTK* 7.1 and *VTK**, (c) highlights its sensitivity to the length of colliding segments, with a small percentage of success when the minimal colliding segment is under 10^{-5}.

Quality The quality of the Boolean operations can be measured by counting the number of self-intersections and bordering edges of the output. We recall that all outputs should be without holes. These quantifications are shown in Fig. 16. While *BORES* does not generate any defects, *VTK*-based methods create several defects whose number is related to the number of colliding faces between the two inputs, when we look at the direct output. After merging duplicate vertices and faces, these defects decrease considerably, but still remain. In addition, we can see that, even though the two control versions of *VTK* (7.0 and 7.1) do not create exactly the same number of defects, they are comparable, whereas *VTK** creates more duplicate vertices but less floating shells (bodies).

Discussion In our experiments, *VTK** showed that it performs Boolean operations significantly faster than the previ-

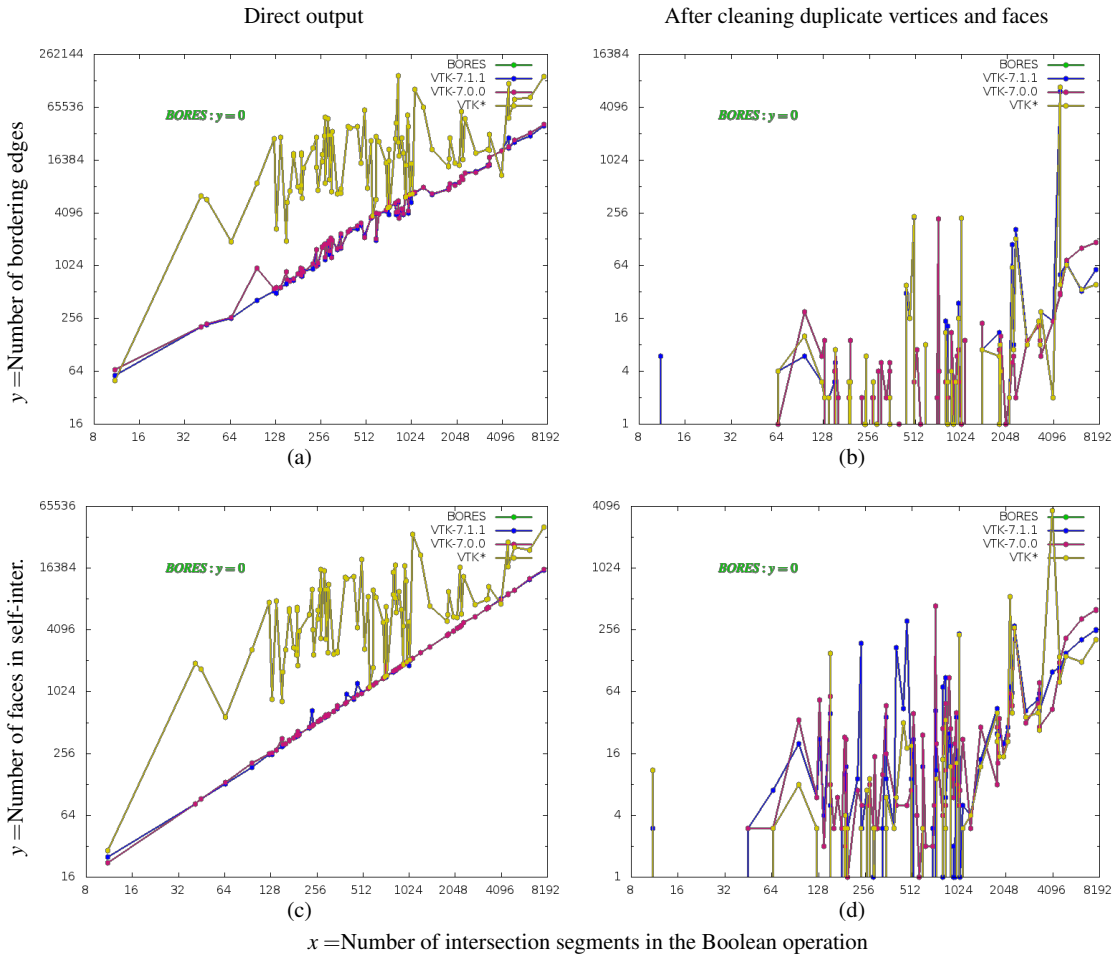


Fig. 16 Plots of defects ((a-b) bordering edges and (c-d) self-intersections) in relation to the number of colliding faces (using scales in log_2).

ous versions of *VTK* (*VTK* 7.0 and 7.1). However, *VTK** was not robust (like *VTK* 7.1) and the resulting meshes were full of defects. The proposed method, *BORES*, proved to be robust resulting in meshes without any defects in all Boolean operation tests. Even though *BORES* was slightly slower than *VTK**, it was faster than the other *VTK* methods. The computation time of the classification by *BORES* and *VTK** was negligible considering the time of intersection and remeshing. Because the *VTK* methods remesh the colliding faces without taking into account the neighbourhood, they often create duplicate vertices. This topological loss and change of the surface are one of the reasons of the defects in the resulting meshes by the *VTK* methods.

6 Conclusion

Boolean operations are essential in computer-aided design and especially for the creation of guides in virtual surgery planning. Therefore, several methods for handling them have been proposed in the literature. However, the complex context of virtual surgery planning, which simultaneously uses surface acquisition, surfaces extracted from volume data, and synthetic surfaces, imposes certain requirements that not met by most existing methods. Among the existing methods, two methods have been selected and compared, namely *BORES* [14] and *VTK** (based on [15, 16]). These two methods work directly on the mesh data structure. This allows them to handle holes. Moreover, they consist of two similar main steps: merging and classification. However, the methods differ in the strategies used in these two steps. In the first step, *VTK** computes exclusively non-coplanar collisions, whereas *BORES* computes all types of collisions and merging duplications. In the second step, *VTK** assumes that all intersection edge series are composed of two pairs of surfaces, whereas *BORES* uses an angular classification allowing for complex configurations and detection of inconsistencies.

BORES has demonstrated its efficiency, accuracy, and robustness on the entire set of data proposed in this paper. In contrast, *VTK** has shown high instability and its weakness to handle complex configurations, such as singular edges and coplanar collisions. However, the *BORES* method is limited to colliding meshes. In future works, the method would be extended to handle floating shells (bodies), while preserving its flexibility and robustness.

Statements

Funding: This study was funded 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 research was supported in part by the KIST institutional program (Grant Number 2E26276, 2E26880).

Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human participants performed by any of the authors.

References

1. Kim Y, Kim L, Lee D, Shin S, Cho H, Roy F, Park SH (2015) Deformable mesh simulation for virtual laparoscopic cholecystectomy training. The Visual Computer, 31:485–495
2. Tay C, Khajuria A, Gupte C (2014) Simulation training: A systematic review of simulation in arthroscopy and proposal of a new competency-based training framework. International Journal of Surgery, 12:626 – 633. ISSN 1743-9191. doi:http://dx.doi.org/10.1016/j.ijsu.2014.04.005
3. Stirling ER, Lewis TL, Ferran NA (2014) Surgical skills simulation in trauma and orthopaedic training. J Orthop Surg Res, 9:126. ISSN 1749-799X. doi:10.1186/s13018-014-0126-z. 25523023[pmid]
4. Leiggener C, Messo E, Thor A, Zeilhofer HF, Hirsch JM (2009) A selective laser sintering guide for transferring a virtual plan to real time surgery in composite mandibular reconstruction with free fibula osseous flaps. International journal of oral and maxillofacial surgery, 38:187–192. ISSN 1399-0020. doi:10.1016/j.ijom.2008.11.026. PMID: 19179046
5. Béziat JL, Babic B, Ferreira S, Gleizal A (2009) [justification for the mandibular-maxillary order in bimaxillary osteotomy]. Revue de stomatologie et de chirurgie maxillo-faciale, 110:323–326. ISSN 1776-257X. doi:10.1016/j.stomax.2009.09.009. PMID: 19939425
6. Metzger MC, Hohlweg-Majert B, Schwarz U, Teschner M, Hammer B, Schmelzeisen R (2008) Manufacturing splints for orthognathic surgery using a three-dimensional printer. Oral surgery, oral medicine, oral pathology, oral radiology, and endodontics, 105:e1–7. ISSN 1528-395X. doi:10.1016/j.tripleo.2007.07.040. PMID: 18230371
7. Laurentjoye M, Charton J, Boileau M (2015) Orthognathic mandibular osteotomy and condyle positioning: update and innovation. L'Orthodontie Française, 86:73–81
8. Frisardi G, Chessa G, Barone S, Paoli A, Razionale A, Frisardi F (2011) Integration of 3d anatomical data obtained by ct imaging and 3d optical scanning for computer aided implant surgery. BMC medical imaging, 11:1
9. Nakao M, Aso S, Imai Y, Ueda N, Hatanaka T, Shiba M, Kirita T, Matsuda T (2016) Automated planning with multivariate shape descriptors for fibular transfer in mandibular reconstruction. IEEE Transactions on Biomedical Engineering, PP:1–1. ISSN 0018-9294. doi:10.1109/TBME.2016.2621742
10. Laurentjoye M, Charton J, Desbarats P, Montaudon M (2014) Mandibular surgery planning and 3d printed splint design. International Journal of Computer Assisted Radiology and Surgery, 9 (Suppl 1):S253–54
11. Polley JW, Figueroa AA (2013) Orthognathic positioning system: Intraoperative system to transfer virtual surgical plan to operating field during orthognathic surgery. Journal of Oral and Maxillofacial Surgery, 71:911 – 920. ISSN 0278-2391. doi: http://dx.doi.org/10.1016/j.joms.2012.11.004

12. Lin HH, Chang HW, Lo LJ (2015) Development of customized positioning guides using computer-aided design and manufacturing technology for orthognathic surgery. International Journal of Computer Assisted Radiology and Surgery, 10:2021–2033. ISSN 1861-6429. doi:10.1007/s11548-015-1223-0
13. Aboul-Hosn Centenero S, Hernaindez-Alfaro F (2012) 3D planning in orthognathic surgery: CAD/CAM surgical splints and prediction of the soft and hard tissues results - our experience in 16 cases. Journal of cranio-maxillo-facial surgery: official publication of the European Association for Cranio-Maxillo-Facial Surgery, 40:162–168. ISSN 1878-4119. doi:10.1016/j.jcms.2011.03.014. PMID: 21458285
14. Charton J, Kim L, Kim Y (2017) Boolean operations by a robust, exact, and simple method between two colliding shells. Journal of Advanced Mechanical Design, Systems, and Manufacturing, Special Issue on the 7th Asian Conference on Design and Digital Engineering. Accepted for publication in September 2017
15. Updegrove A, Wilson NM, Shadden SC (2016) Boolean and smoothing of discrete polygonal surfaces. Advances in Engineering Software, 95:16 – 27. ISSN 0965-9978. doi:http://dx.doi.org/10.1016/j.advengsoft.2016.01.015
16. Mei G, Tipper JC (2013) Simple and robust boolean operations for triangulated surfaces. CoRR, abs/1308.4434
17. Quammen C, Weigle C, II RT (2011) Boolean operations on surfaces in vtk without external libraries. The VTK Journal, 797:1
18. Thibault WC, Naylor BF (1987) Set operations on polyhedra using binary space partitioning trees. ACM SIGGRAPH computer graphics. ACM, vol. 21, pp. 153–162
19. Granados M, Hachenberger P, Hert S, Kettner L, Mehlhorn K, Seel M (2003) Algorithms - ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, chap. Boolean Operations on 3D Selective Nef Complexes: Data Structure, Algorithms, and Implementation. ISBN 978-3-540-39658-1, pp. 654–666. doi:10.1007/978-3-540-39658-1_59
20. Chen Y (2007) Robust and accurate boolean operations on polygonal models. ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. ASME, vol. 2, pp. 357–369. doi:10.1115/DETC2007-35731
21. Bernstein G, Fussell D (2009) Fast, exact, linear booleans. Proceedings of the Symposium on Geometry Processing. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SGP '09, pp. 1269–1278
22. Campen M, Kobbelt L (2010) Exact and robust (self-)intersections for polygonal meshes. Comput Graph Forum, 29:397–406
23. Hachenberger P, Kettner L (2016) 3d boolean operations on nef polyhedra. CGAL User and Reference Manual, CGAL Editorial Board. 4.8 edn.
24. Bernstein G (2007). Cork. https://github.com/gilbo/cork
25. Chen M, Chen XY, Tang K, Yuen MMF (2010) Efficient boolean operation on manifold mesh surfaces. Computer-Aided Design and Applications, 7:405–415. doi:10.3722/cadaps.2010.405-415
26. Feito F, Ogayar C, Segura R, Rivero M (2013) Fast and accurate evaluation of regularized boolean operations on triangulated solids. Computer-Aided Design, 45:705 – 716. ISSN 0010-4485. doi:http://dx.doi.org/10.1016/j.cad.2012.11.004
27. Schifko M, Jüttler B, Kornberger B (2010) Industrial application of exact boolean operations for meshes. Proceedings of the 26th Spring Conference on Computer Graphics. ACM, New York, NY, USA, SCCG '10. ISBN 978-1-4503-0558-7, pp. 165–172. doi:10.1145/1925059.1925089
28. Pavić D, Campen M, Kobbelt L (2010) Hybrid booleans. Computer Graphics Forum, 29:75–87. ISSN 1467-8659. doi:10.1111/j.1467-8659.2009.01545.x
29. Antonio F (1992) Graphics gems iii. Academic Press Professional, Inc., San Diego, CA, USA, chap. Faster Line Segment Intersection. ISBN 0-12-409671-9, pp. 199–202
30. Möller T (1997) A fast triangle-triangle intersection test. Journal of Graphics Tools, 2:25–30
31. Shewchuk JR (1997) Delaunay Refinement Mesh Generation. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania. Available as Technical Report CMU-CS-97-137
32. Leiggener C, Messo E, Thor A, Zeilhofer HF, Hirsch JM (2009) A selective laser sintering guide for transferring a virtual plan to real time surgery in composite mandibular reconstruction with free fibula osseous flaps. International Journal of Oral and Maxillofacial Surgery, 38:187 – 192. ISSN 0901-5027. doi:http://dx.doi.org/10.1016/j.ijom.2008.11.026