Computers & Graphics 59 (2016) 13–27

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Booleans of triangulated solids by a boundary conforming tetrahedral mesh generation approach ☆

Zhoufang Xiao [a,b], Jianjun Chen [a,b,*], Yao Zheng [a,b], Jianjing Zheng [a,b], Desheng Wang [c]

[a] Center for Engineering and Scientific Computation, Zhejiang University, Hangzhou 310027, China
[b] School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China
[c] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371, Singapore

ARTICLE INFO

Article history:
Received 7 December 2015
Received in revised form
23 April 2016
Accepted 26 April 2016
Available online 14 May 2016

Keywords:
Boolean
Mesh generation
Boundary recovery
Delaunay triangulation
Intersection

ABSTRACT

A new algorithm is proposed to recast Boolean operations of triangulated solids as a boundary conforming tetrahedral meshing problem. Different from those existing algorithms that merely maintain a conforming surface mesh, the new algorithm maintains a boundary conforming volume mesh at the same time of computing surface intersections. This volume mesh not only provides a background structure in helping the improvement of the efficiency of intersection computations, but also enables the development of a set of efficient and reliable flood-filling type procedures to extract the Boolean outputs. The efficiency and robustness of the proposed algorithm are investigated in further details, and various techniques are suggested to tackle these two issues accordingly. Finally, the performance of the proposed algorithm has been evaluated by performing suitable test cases and the results are compared well with those data obtained by other state-of-the-art codes.

1. Introduction

Boolean operations of triangulated solids are fundamental tasks in computational geometry, computer-aided design, computer graphics, among many other subjects [1–3]. In general, Boolean algorithms can be classified into two major types, according to the approach it adopts for surface repairing, i.e., *rediscretizing the surface* [4–10] and *repairing intersections in-place* [11–25]. The second type of algorithm is preferred in some applications [1–3], because it can ensure geometric accuracy of the output surface. Besides, it can set up the edge-wise and the face-wise mappings between the input and output surfaces so that information attached to the input surface can be easily inherited by the output surface. Those existing Boolean algorithms of repairing intersections in-place [11–25] are usually based on a *bottom-up* procedure, in which it first computes the intersection lines, then imprints these lines onto the surface and uses the chain of intersections lines to bound a surface region, and finally, combines surface regions with similar in/out properties to form the Boolean outputs. In this study, a Boolean algorithm following a very different flowchart to those existing methods is presented. Fig. 1 illustrates the basic flowchart of performing the union operation of three balls, where three major steps are generally involved as:

(1) *Delaunay tetrahedralization of input surface points*. This step starts from a Delaunay tetrahedralization of a box enclosing input surfaces, and then inserts surface points individually into the tetrahedralization by an incremental point insertion scheme [26–29].
(2) *Recovery of lost boundary constraints*. Lost boundary constraints are recovered by inserting *Steiner points* [30–46] at intersection positions of lost constraints and the tetrahedralization. Accordingly, surface and volume elements are subdivided to maintain the mesh conformity. As a result, the intersections and overlaps of input surfaces are repaired in-place and a boundary conforming tetrahedral mesh is produced as output.
(3) *Preparation of the Boolean outputs*. Firstly, the volume elements are classified using a flood-filling algorithm. A set of volume elements that lies inside at least one input surface exactly fills in the volume domain of the union operation of all input surfaces. The output of the union operation is composed of the exterior faces of this volume domain.

The most important step of the proposed algorithm is Step 2, where intersection lines are computed and imprinted to result in a boundary conforming tetrahedral mesh. Unlike existing approaches that create additional spatial decomposition structures to speed up the intersection computations [11], the proposed algorithm utilizes this tetrahedral mesh, as a background structure, to compute the intersections very efficiently. Besides, the existence of

☆This article was recommended for publication by Kai Hormann.
* Corresponding author. Tel.: +86 571 87951883; fax: +86 571 87953167.
E-mail address: chenjj@zju.edu.cn (J. Chen).

http://dx.doi.org/10.1016/j.cag.2016.04.004
0097-8493/

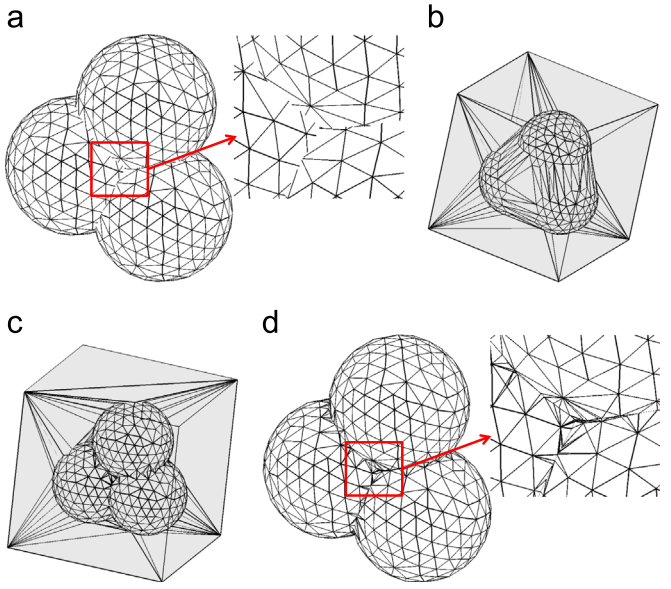

Fig. 1. Illustration for the basic flowchart of the proposed algorithm. (a) The input surfaces (three balls). (b) The Delaunay tetrahedralization after inserting all input points. (c) The boundary conforming tetrahedral mesh. (d) The surface output by the union operation.

a boundary conforming tetrahedral mesh enables a *top-down* procedure of preparing the Boolean outputs (i.e., Step 3). It first classifies volume elements into different regions, and then extracts common surface regions and chains of intersection lines that bound these surface regions. As the above procedures access mesh topology only, they are more reliable and efficient than their counterparts that often rely on some kinds of geometrical computations [11–25].

Note that most Boolean algorithms only consider the case of two input components. When more than two components are involved, the Boolean procedure need be called recursively. Taking the input shown in Fig. 1a as an example, such a Boolean procedure needs to first compute the union of two balls, and then add the third ball into the Boolean computation. In each Boolean calling, some computations may be unnecessarily repeated. In contrary, the proposed algorithm treats the input as a whole (Step 2), and its computing efficiency remains stable no matter how many components the same input is separated to. Clearly, when the input geometry contains a large number of components, the proposed algorithm can achieve a higher computing efficiency than those based on recursive procedures.

Nevertheless, because existing boundary recovery algorithms only consider surface inputs containing no intersections and overlaps, more intersection scenarios need be identified and dealt with accordingly when applied these algorithms in Boolean operations. Meanwhile, the efficiency and robustness issues need be investigated carefully.

With respect to the efficiency issue, a key step is to localize the intersection computations [11]. In this study, a set of memory-saving data structures will be introduced to represent the surface and volume meshes, and based on which, the intersection computations are limited in the local parts of the surface and volume meshes where intersections really happen. These efforts enable the developed Boolean algorithm to manage a large mesh input at a comparable speed with those obtained by the fastest Boolean algorithms [25].

With respect to the robustness issue, although recovering prescribed boundary constraints from a tetrahedral mesh is always possible in theory by inserting Steiner points, inconsistent geometric computations may collapse the boundary recovery algorithm in practice. Exact predicates can be introduced to improve the robustness remarkably [47,48]. Nevertheless, because the positions of Steiner points are represented by fix-precision floating-point numbers, exact predicates with these positions as inputs may return an undesirable value and still collapse the algorithm. A solution to this issue is the use of exact arithmetic [16–18]. However, this may complicate the algorithm and slow-down its timing performance considerably. Therefore, we leave this solution for the future work, but introduce two other remedies instead. The first remedy is a *mesh topology improvement scheme*, which is used to transform a mesh that intersects predefined constraints intensively to another one with much fewer intersections, thus remarkably reducing the number of Steiner point insertion (Step 2). The second remedy is to record a mapping between a Steiner point and the intersecting entities from which the Steiner point is generated and a mapping between the original edges and faces and their subdivision results. These auxiliary data structures can help to replace some error-prone geometrical computations by more reliable topological computations.

The remaining sections of the paper are organized as follows. In Section 2, previous studies on boundary recovery algorithms and Boolean algorithms are reviewed, followed by a brief summary of our contributions. Next, the boundary conforming meshing procedure and the *top-down* flowchart that prepares the Boolean outputs are detailed in Sections 3 and 4, respectively. Various examples and their statistics are presented in Section 5. Finally, concluding remarks are given in Section 6.

2. Literature review

2.1. Boundary recovery algorithms

The Delaunay criterion provides a reasonable algorithm to triangulate a given point set. However, boundary constraints could be lost in the resulting mesh, and therefore either *conforming* or *constrained* method is required to recover the lost constraints. For the conforming method, Steiner points are inserted onto the constraints and will not be removed from the resulting mesh; thus, some of the lost constraints can be recovered as concatenations of sub-constraints. For the constrained method, the recovered constraints are the same as the prescribed ones, and no Steiner points are allowed to remain on the constraints.

There is no guarantee to recover an edge or a face from a tetrahedral mesh without adding Steiner points [32]. The typical failing examples are Schönhardt polyhedron [33] and Chazelle polyhedron [34]. Therefore, a robust 3D boundary recovery algorithm must consider how to insert Steiner points [30,31,35–46]. If Steiner points are allowed, the termination problems of both 3D conforming and 3D constrained boundary recovery have already been theoretically resolved [35–37].

Because the existence of Steiner points may have negative impacts on the robustness and efficiency of the boundary recovery procedure, two primary issues regarding Steiner points need further attention. The first issue is where and how to add Steiner points. The second issue is how to remove as many Steiner points as possible.

To resolve the first issue, George et al. [38] proposed an algorithm in the early 1990s that was based on local transformation operators in conjunction with heuristic rules for inserting Steiner points; however, this algorithm suffers from some robustness issues [39]. Weatherill and Hassan [35] first investigated another algorithm that inserts Steiner points directly at the intersection positions of lost boundaries and the mesh. This algorithm is preferred by many researchers [30,31,36,40–42] and also adopted in this study. Another type of algorithm for Steiner point insertion is