Boolean operations on quad meshes

Alex Hernández
Systems and Computer Eng. Program
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil - CEP: 21941-914
Email: ahernandezm@cos.ufrj.br

Claudio Esperança
Systems and Computer Eng. Program
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil - CEP: 21941-914
Email: esperanc@cos.ufrj.br

Nico Pietroni
School of Software
University of Technology Sydney
Sydney, Australia
Email: Nico.Pietroni@uts.edu.au

Paolo Cignoni
Visual Computing Lab
Institute for Computer Science and Technologies
Pisa, Italy - 56124
Email: p.cignoni@isti.cnr.it

***Abstract*—In this work we describe an approach to perform boolean operations between pure quad meshes obtaining as result a pure quad mesh preserving as much as possible the original quadrangulations of the inputs. For this purpose, we solve the boolean operation in a triangular version of the inputs with a robust triangle-based method and then solve a requadrangulation problem on portions of the mesh neighboring the intersection curves of the inputs. Our approach reduces the hard problem of requadrangulation in space to a 2D polygon subdivision problem into patches which are easier to quadrangulate. We propose a method based on partitioning polygon borders into monotonic chains in order to get appropriate patches that can be quadrangulated independently. As far as we know, this is the first work to tackle the problem of computing boolean operations of quad meshes without resorting to a full requadrangulation of the result mesh. Another important goal is to obtain a good edge flow near the intersection curves, a crucial feature for applications relying on quad meshes such as character animation modeling.**

I. Introduction

Boolean or set-theoretic operations are a natural way of constructing complex objects from simpler ones, and is the basis of Constructive Solid Geometry (CSG), where primitive solids are usually implicit objects, which naturally support such operations. On the other hand, Boundary Representation (B-rep) is a modeling paradigm where surfaces are represented by discrete structures such as meshes, which require considerably more effort when used as operands of set-theoretic operations. Since B-rep is very popular in CAD and computer graphics applications, several techniques for computing such operations on meshes have been devised over the years.

This is a non-trivial problem, especially for complex B-rep models as it requires many intersection tests, separating the final surface into pieces and constructing new surfaces out of these pieces. Computing the intersection curves between the input models is a central task and computationally expensive if we demand accuracy. Exact arithmetic and complex techniques were used in several works addressing the issue, as in the Computational Geometry Algorithms Library (CGAL [1]) that supports robust Boolean operations on Nef polyhedra [2], which is considered a seminal reference, despite its high memory cost.

Several approaches were proposed in this area, but all proposals that work with B-rep deal with the connectivity near the intersection curves using triangles. In the case of quad mesh applications, boolean operations must be handled with special care so as to preserve the *edge flow*. The *edge flow* is a kind of discrete surface flow that animators generally set up to follow the principal curvature lines of the surface. A good edge flow is necessary for a model be used for animation. For that reason, simply computing the boolean operation on equivalent triangle meshes and then requadrangulating the result is not a suitable approach. In this work we propose an approach where requadrangulation effort is spent only close to the intersection curves and these are reproduced as faithfully as possible in the finished result.

II. Related works

There are works that use other surface representations in order to deal with the conflicting goals of accuracy, robustness, and efficiency in boolean operations. This is the case of [2] that uses Nef polyhedra representation and is the main robust reference in this area. In fact, most of the time computing is carried out in the conversion of the polygon mesh to the representation in Nef polyhedra. The work of Biermann et al. [3] that uses multiresolution subdivision surfaces and the work of Magalhes et al. [4] that uses a triangle soup representation are other important examples of this sort of approach.

Works handling boolean operations on B-rep's are divided into two groups: those that work with only triangle meshes and those that work with polygonal meshes. On the latter group, we highlight the work of S. Lo, W. Wang [5] that receives as inputs mixed meshes (quadrilaterals and triangles) and also returns mixed meshes. Also worthy of mention is the work of Douze et al. [6], that developed *QuickCSG*, a multiple input polyhedra system for boolean operations which is very fast and robust. They propose a new vertex-centric view of the problem with impressive results. On the former group, we find BSP-based methods like [7] or [8] and the work of Zhou et al. [9] that developed a robust boolean system for triangle meshes using the Winding Number concept, restricting the problem to the class of meshes with a piecewise-constant winding number or PWN meshes. It should be noted that the validation of this method was exhaustive with the online 3D printing repository of 10000 triangle meshes Thingi10K. There is an implementation of this algorithm in the LIBIGL library. Incidentally, this library is used as a tool to compute the boolean operation between the triangular versions of the inputs in our method.

Finally, we must also consider works about quadrangulation of meshes. In this field, there are several types of approaches. For our purposes we focus on quadrangulators of surface patches that solve the problem by restricting the quadrangulation effort to selected regions of the mesh, so that the final quadrangulation is the union of the quadrangulated patches.

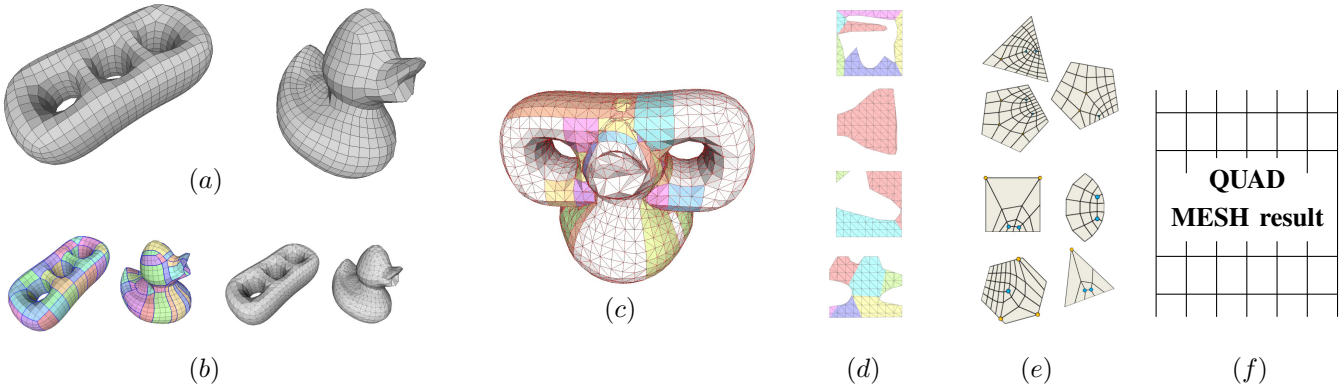


Fig. 1. Steps to compute the union operation between two quad meshes: (a) Input quad meshes. (b) For each mesh, a partition into 4-sided patches is computed and an equivalent triangle mesh is computed. (c) Intersection curves are computed and segmented with respect to the patches they cut. (d) Each cut patch and corresponding intersection curve segments are brought to the 2D parametric space inherited from the quadrangulation, resulting in a set of polygonal regions. Each polygonal region is then subdivided into "well-shaped" polygonal parts whose boundaries contain a small number of monotonic chains. Intuitively, these parts can then be requadrangulated independently. (e) Patterns used to quadrangulate each part. (f) Requadrangulated parts are unprojected back to the original mesh space and stitched together with the uncut patches (this step is still being implemented).

Bessmeltsev et al. [10] developed a system for reconstruction of quad dominant meshes from an input curve network. An important characteristic of this method is that it infers both geometry and topology from closed 3D paths. The works of Takayama et al. [11] and Marcias et al. [12] are aimed at retopologizing a triangle mesh. Both focus the attention on the problem of quadrangulating a 2D $n-$sided patch with good results. The first uses a pattern-based approach and the second uses a trained database of quadrangular patches. Since we expect to produce triangular patches that are relatively small with respect to the input meshes, we propose using Takayama's method. While Marcias' method is generally better for controlling the edge flow, in our case, the edge flow is given by setting the intersection curves as boundaries of patches.

III. Methodology

The entire work-flow of our proposal can be seen in the Figure 1. We use a number of consolidated techniques in order to deal with the hard problems found during the implementation. These steps are discussed in detail below. The next step is to quadrangulate only the patches crossed by intersection curves. Requadrangulating these patches assuming the intersection curves as boundaries has two effects: (1) forces the edge flow to be orthogonal to the intersection curves for small patches, and (2) minimizes the disturbance of the edge flow along other boundaries.

A. Partition in 4-sided patches

This part is aimed at constructing a 1-to-1 mapping of the original surface onto a set of *patches* with rectangular topology, i.e., where all vertices have degree 4. We use a simple approach working directly on the quad mesh topology by searching irregular points, separatrix lines and tracing the flow lines. The main purpose of this step is to establish subdomains on the mesh that can be parameterized along two directions.

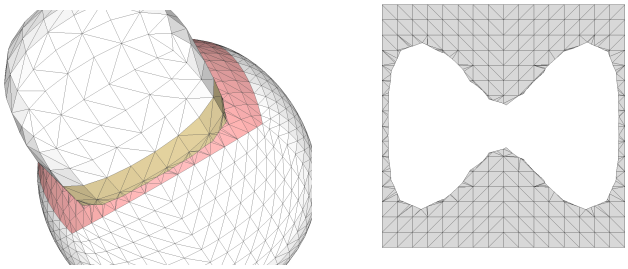


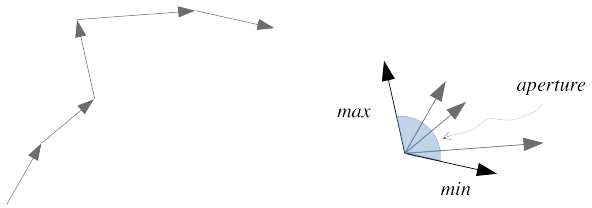
Fig. 2. Parameterization example. (a) Union operation of the models. (b) Parameterization of the red patch in (a).

B. Boolean operation

Although have implemented a naive algorithm that solves the problem for well-behaved meshes, our current prototype now uses the LIBIGL implementation of the method of Zhou et al [9] in order to ensure robustness. For converting the quad meshes into triangle ones we simply divide each quad into two triangles but keep track of the vertex indices for future computations.

C. Parameterization

At this point, we have a complete mesh with the result of the boolean operation. More importantly, this mesh also contains information about (1) which parts originate from each of the two input models, and (2) the new edges introduced by the operation, i.e., the intersection curves. Due to the preprocessing of the input quad meshes, the requadrangulation



(a) tight monotone chain (b) cone aperture

Fig. 3. Determining the abstract cone of a chain

work can be done independently for each original 4-sided patch that was crossed by an intersection curve. This allows us to solve each problem topologically on a plane (Figure 2). We use the natural parameterization of each patch and interpolate the intersection curve coordinates.

D. Splitting patches into semi-regular parts

This is the key part of our approach. We attempt to subdivide the regions present in each patch of interest (like the red patch in Figure 2) into parts that can be quadrangulated. For this purpose, we subdivide the boundary of the polygons (possibly with holes) into monotone chains that will be candidate sides for the patch subdivision. These monotone chains are akin to "straight sides" that can be used as support for consecutive quads along a flow line of the quadrangulation.

1) Partition into tight monotone chains: We define a *tight monotone chain* as a chain of edges or poly-line segment such that the beam of vectors corresponding to the chain edges define an abstract cone with aperture less than a given threshold. An *abstract cone* is defined by two vectors that we call the min and max directions. The angle between these is called the cone's *aperture*, which can be enlarged by updating min and max. These concepts are illustrated in Figure 3. We proceed to subdivide the regions' boundaries following the steps:

a) Polygons with holes are converted into simple polygons by connecting each hole with the exterior boundary.
b) Starting at a feature vertex of the polygon boundary (the vertex with smallest internal angle), follow the boundary to compute tight monotone chains. Initially, an abstract cone with zero aperture is built with min and max set to first directed edge. Walking around the circulation, consecutive edges are added to the cone while corresponding vertices are added to the chain until the cone's aperture becomes larger than the threshold (we use 60 degrees in our experiments). The process is repeated until the first vertex is reached.

2) Diagonal ranking: The ranking process is addressed to choose the best *diagonal* connecting one endpoint of a monotone chain to another vertex of the polygon. A line segment is a diagonal if it does not intersect the boundary of the polygon other than at the endpoints. This step ranks the diagonals according to the following criteria:

- Length of the diagonals – smaller diagonals are better.
- Angle deviation from the bisector of the vertex – diagonals that split the internal angles at each endpoint roughly in the middle are better.
- Balance between the number of chains of the result – diagonals that split the polygon boundary into two parts with n and m chains where $n \simeq m$ are better .

We average these criteria giving preference to the angle deviation. Once the best diagonal is selected, the polygon is split and each of the two parts are recursively split until each part's boundary has no more than 5 monotone chains.

E. Quadrangulation of the patches

With the patches found in the before step, we proceed to use the method of Takayama et al. [13], take care to respect the method's requirements. In particular, each patch must contain an even number of edges. In order to ensure this, we have two options: adding points into the diagonals and solving an integer programming problem, or adding vertices to the intersection curves and extending to the triangle versions of the inputs. Currently, our implementation adopts the latter option.

IV. Results

Our prototype uses the Visualization and Computer Graphics Library (VCGlib [14]) for the geometry tasks and the Qt platform in C++ 11. Also we use the LIBIGL library for the triangle boolean operations. We have yet to finish the implementation, but we have already achieved good results with our 2D subdivision method in a number of experiments. We present some cases of the subdivision on the Figure 4. The integration of Takayama's method is still pending.

V. Conclusion and Future works

We have presented an approach to perform boolean operations between quad meshes and returns another quad mesh preserving as much as possible the original quadrangulations. Additionally, the resulting mesh should present a good edge flow near the intersection curves. This is a useful tool for modelers who work with quad meshes already processed for animation. This demand can be seen in forums of 3d artists that, at this moment, solve that problem using assisted retopologizers such as Quad Draw Tool of Maya [15], Topogun [16], RetopoFlow [17], MODO retopology [18], 3DCOAT retopology [19], etc.

A disadvantage of our method, however, is that the method of Zhou et al. [9] for computing the boolean operation of triangle meshes requires that the inputs be closed surfaces. As future work, we intend to improve our boolean operation system to handle open surfaces without losing the robustness. At the moment, the work is in the final implementation phase which will require a few more months before it can be presented to the academic community.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

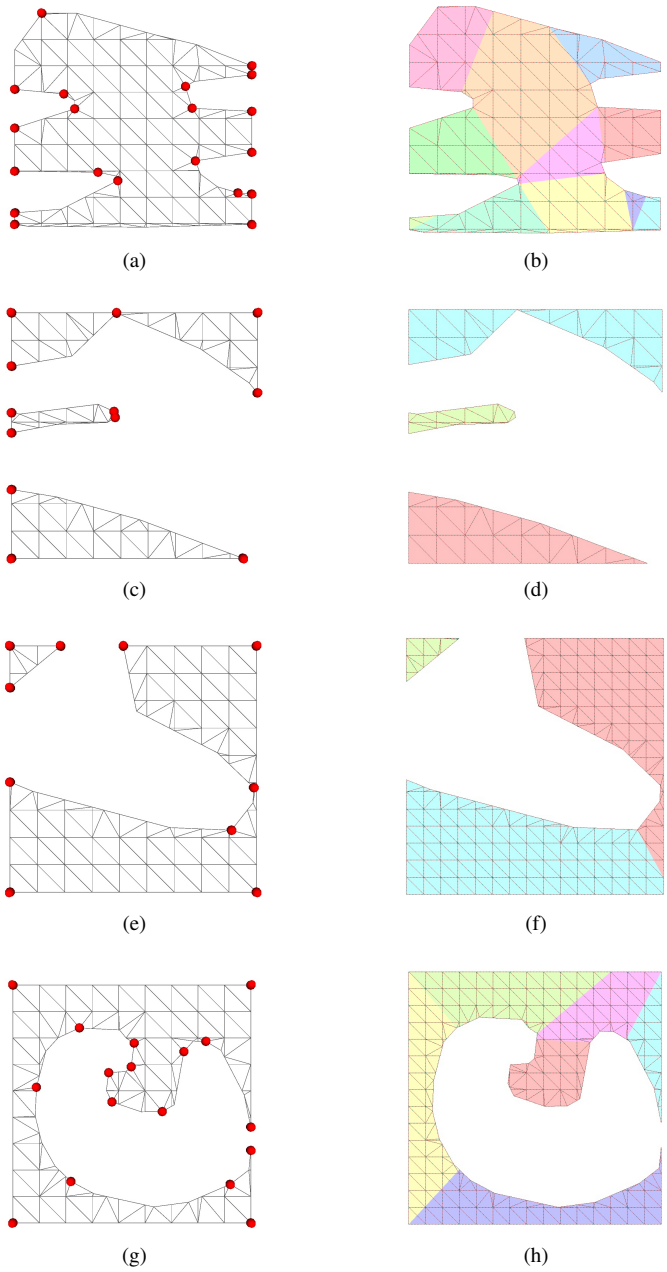


Fig. 4. 2D patch subdivision results. Our method tries to partition a complex 2d shape into subpatches, each having a border composed of a small number of monotonic chains. The left column (a, c, e ,g) shows the vertices where the border of several patches were split into a number of monotonic chains. The right column (b, d, f, h) shows the choice of diagonals used to obtain the final subpatches (colored regions).

REFERENCES

[1] The CGAL Project, *CGAL User and Reference Manual*, v 4.11 ed. CGAL Editorial Board, 2017, http://doc.cgal.org/4.11/Manual/packages.html.

[2] P. Hachenberger and L. Kettner, "3d boolean operations on Nef polyhedra," in *CGAL User and Reference Manual*, 4th ed., C. E. Board, Ed., 2016. [Online]. Available: http://doc.cgal.org/latest/Nef_3/index.html

[3] H. Biermann, D. Kristjansson, and D. Zorin, "Approximate boolean operations on free-form solids," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 185–194. [Online]. Available: http://doi.acm.org/10.1145/383259.383280

[4] S. V. G. Magalhães, W. R. Franklin, and M. V. A. Andrade, "Fast exact parallel 3d mesh intersection algorithm using only orientation predicates," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL'17. New York, NY, USA: ACM, 2017, pp. 44:1–44:10. [Online]. Available: http://doi.acm.org/10.1145/3139958.3140001

[5] S. Lo and W. Wang, "An algorithm for the intersection of quadrilateral surfaces by tracing of neighbours," *Computer Methods in Applied Mechanics and Engineering*, vol. 192, no. 2021, pp. 2319 – 2338, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045782503002408

[6] M. Douze, J.-S. Franco, and B. Raffin, "QuickCSG: Arbitrary and Faster Boolean Combinations of N Solids," Inria - Research Centre Grenoble – Rhône-Alpes ; INRIA, Research Report RR-8687, Mar. 2015. [Online]. Available: https://hal.inria.fr/hal-01121419

[7] G. Bernstein and D. Fussell, "Fast, exact, linear booleans," in *Proceedings of the Symposium on Geometry Processing*, ser. SGP '09. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2009, pp. 1269–1278. [Online]. Available: http://dl.acm.org/citation.cfm?id=1735603.1735606

[8] D. Pavi, M. Campen, and L. Kobbelt, "Hybrid booleans," *Computer Graphics Forum*, vol. 29, no. 1, pp. 75–87, 2010. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2009.01545.x

[9] Q. Zhou, E. Grinspun, D. Zorin, and A. Jacobson, "Mesh arrangements for solid geometry," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 39:1–39:15, Jul. 2016. [Online]. Available: http://doi.acm.org/10.1145/2897824.2925901

[10] M. Bessmeltsev, C. Wang, A. Sheffer, and K. Singh, "Design-driven quadrangulation of closed 3d curves," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 178:1–178:11, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366197

[11] K. Takayama, D. Panozzo, and O. Sorkine-Hornung, "Pattern-based quadrangulation for N-sided patches," *Computer Graphics Forum (proceedings of EUROGRAPHICS Symposium on Geometry Processing)*, vol. 33, no. 5, pp. 177–184, 2014.

[12] G. Marcias, K. Takayama, N. Pietroni, D. Panozzo, O. Sorkine-Hornung, E. Puppo, and P. Cignoni, "Data-driven interactive quadrangulation," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 65:1–65:10, Jul. 2015. [Online]. Available: http://doi.acm.org/10.1145/2766964

[13] K. Takayama, D. Panozzo, A. Sorkine-Hornung, and O. Sorkine-Hornung, "Sketch-based generation and editing of quad meshes," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 97:1–97:8, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461955

[14] P. Cignoni and F. Ganovelli, *Visualization and Computer Graphics Library*, v 1.0.1 ed. Visual Computing Lab of the Italian National Research Council-ISTI, 2016, http://vcg.isti.cnr.it/vcglib/.

[15] A. Inc, "Maya quad draw tool," 2017, https://knowledge.autodesk.com/support/maya-lt/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/MayaLT/files/GUID-20DEA0B6-C090-49EA-98AE-172F1C382A05-htm.html. Accessed: 2017-11-17.

[16] S. P. SRL, "Topogun," 2017, http://www.topogun.com. Accessed: 2017-11-17.

[17] C. Cookie, "RetopoFlow-retopology tools for blender," 2017, https://blendermarket.com/products/retopoflow/. Accessed: 2017-11-17.

[18] T. F. V. Limited, "Modo automatic retopology tool," 2017, https://help.thefoundry.co.uk/modo/content/help/pages/modeling/edit_geometry/auto_retopo.html. Accessed: 2017-11-17.

[19] 3D-Coat, "3D-Coat retopo tools," 2017, http://3dcoat.com/manual/retopo/238-retopotools/. Accessed: 2017-11-17.