

Simplification of Nonconvex Tetrahedral Meshes

Martin Kraus, Thomas Ertl

Visualization and Interactive Systems Group
Universität Stuttgart, Germany*

Abstract

We present a new solution to the well-known problems of edge collapses in nonconvex tetrahedral meshes. Additionally, our method is able to handle meshes with topologically non-trivial boundaries and to control the modification of the topology of the mesh's boundary.

1 Introduction

In order to visualize today's huge data sets, hierarchical representations are often employed. The production of such representations of tetrahedral volume meshes requires a simplification of the original mesh. One way of performing this simplification is to apply a sequence of edge collapses, which are discussed in some detail in section 2 with an emphasis on the problems of edge collapses in nonconvex meshes.

Our solution to these problems consists of two parts: A preprocessing step—originally suggested by Peter Williams in the context of sorting nonconvex meshes—is briefly described in section 3.

The second part—edge collapses in the resultant meshes—is discussed in section 4 with special attention being paid to modifications of the topology of the mesh's boundary.

Section 5 presents our conclusions and plans for future work on this subject.

2 Background and Related Work

2.1 Edge Collapses

In the following we will discuss edge collapses in fair tetrahedral meshes, i.e. each face of a tetrahedral cell is either part of the boundary of the mesh or shared by (at most) two cells. Intersections of cells are not allowed; in fact avoiding them is our primary concern. As edge collapses in triangular meshes in two dimensions are very similar to the three-dimensional case of tetrahedral meshes, we will illustrate our method with triangular meshes in Figures 1-8 before presenting the method in three dimensions in Figures 9-13.

It is useful for the description of our method to define some particular terms. We call a tetrahedron a *vertex neighbor* of a vertex if the vertex is shared by the tetrahedron. A tetrahedron is an *edge neighbor* of an edge if the edge is shared, and a *vertex neighbor* of an edge if one of the vertices of the edge is shared. Finally, a tetrahedron is a *face neighbor* of another tetrahedron if the tetrahedra share one face. The definitions of *vertex* and *edge neighbors* can also be applied to triangles in triangular meshes.

The effect of an edge collapse (see Figure 1 for a two-dimensional example) is to remove all edge neighbors of the collapsing edge and to join the two vertices of the collapsing edge in a new vertex. Figure 1 also indicates the inverse operation, which is called a *vertex split*.

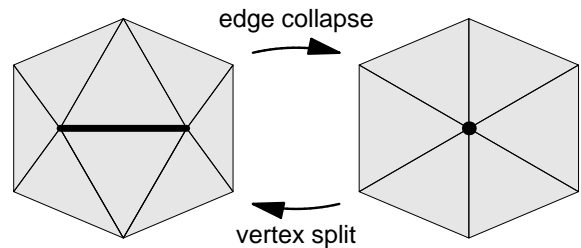


Figure 1: An edge collapse and the inverse vertex split.

Edge collapses are one of the most powerful tools to simplify triangular or tetrahedral meshes. They can be employed to remove vertices or edges (see [4]) and also to remove triangles or tetrahedra by successive edge collapses (see [5]).

Moreover, a sequence of edge collapses can be used to produce hierarchical representations of triangular and tetrahedral meshes as demonstrated in many publications, for example [3, 4, 5].

2.2 Avoiding Intersections of Cells

As demonstrated in Figure 2 an edge collapse can cause an intersection of cells in a triangular or tetrahedral mesh. In order to avoid such self-intersections of a mesh, edge collapses are tested before they are performed (see [4, 5]).

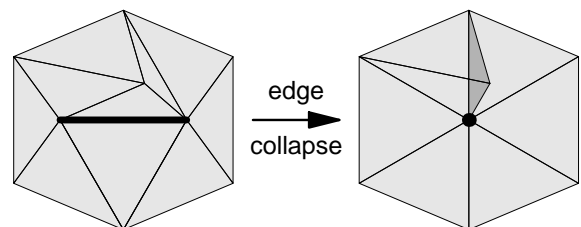


Figure 2: An edge collapse, which causes several intersections of cells and one inversion of a cell (dark gray).

In convex meshes, i.e. meshes the boundary of which are convex polytopes, the test for intersections is particularly simple because any intersection of cells is accompanied by an *inversion* of at least one cell, i.e. a sign flip of the signed volume of a cell. (The inverted cell is marked gray in Figure 2.) Therefore, it is sufficient to test all vertex neighbors of a collapsing edge for inversions in order to avoid self-intersections. This test is *local* as only vertex neighbors are involved.

However, if a collapsing edge in a nonconvex mesh has vertex neighbors that are cells at the boundary (i.e. one of the faces of the cell is part of the boundary of the mesh) then the edge collapse can cause self-intersections of the mesh without causing an inversion of a cell as shown in Figures 3 and 10.

*Universität Stuttgart, IfI, Abt. VIS, Breitwiesenstr. 20-22, 70565 Stuttgart, Germany; E-mail: {Martin.Kraus | Thomas.Ertl}@informatik.uni-stuttgart.de.

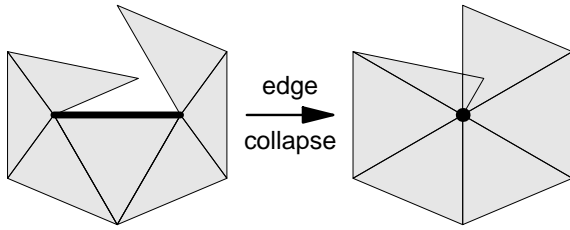


Figure 3: An edge collapse, which causes an intersection of two cells without causing an inversion of any cell.

A naive procedure to avoid such intersections is to test the vertex neighbors of the collapsing edge for intersections with all other cells of the mesh. As the time complexity of this *global* test depends linearly on the number of cells in the whole mesh, it is usually too expensive to be performed without additional auxiliary data structures. A more elaborated implementation of this test is discussed in [4] while the system described in [5] tries to preserve the boundary of the tetrahedral mesh.

However, performance issues are not the only problem of edge collapses in nonconvex meshes. Additional problems occur in disconnected meshes as edge collapses are obviously not able to join clusters of disconnected meshes in order to simplify them. More generally spoken, it is desirable to modify the topology of the mesh's boundary in a controlled way when performing edge collapses.

Before presenting our approach to solve these problems in section 4, we describe the necessary preprocessing step in the following section.

3 Convexification of Nonconvex Meshes

More than eight years ago Peter Williams proposed in [6] to convert nonconvex meshes to convex meshes by triangulating all voids and cavities and marking the cells generated by this triangulation as *imaginary*. (*Virtual* is today's more fashionable word for the same idea.)

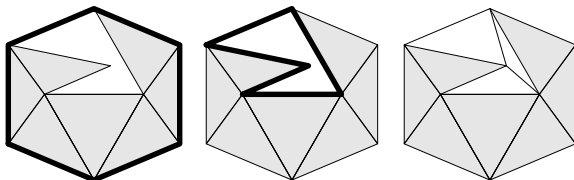


Figure 4: A step-by-step convexification of the mesh shown in the left-hand side of Figure 3. From left to right: the convex hull (thick line), the nonconvex polygon (thick line) between the convex hull and the boundary of the mesh, and the mesh together with imaginary cells (white) generated by the triangulation of the nonconvex polygon.

Figures 4 and 11 summarize the basic steps of this process. Firstly, the convex hull of the mesh is computed; then all voids and cavities are identified and triangulated; finally, the new imaginary cells are attached to the existing mesh. (As will be shown in section 4 the number of imaginary cells generated by the triangulation is not relevant in the context of edge collapses. An optimal algorithm (with respect to the number of generated tetrahedra) for the tetrahedralization of nonconvex polyhedra was published in [1].)

We call this preprocessing step a *convexification* of a nonconvex mesh as it allows us to apply (slightly modified) algorithms for convex meshes to a nonconvex mesh. (In this sense the convexification might be called a *meta-algorithm*.) The following section shows how to overcome problems of edge collapses introduced by nonconvexities (including non-trivial topologies of the boundary) with the help of this preprocessing step.

4 Simplification of Convexified Meshes

4.1 Geometric Tests

As mentioned in section 2 and shown in Figure 3, edge collapses in nonconvex meshes can cause intersections of cells without causing an inversion of any cell. In convexified meshes, however, such edge collapses will always cause an inversion of at least one imaginary cell as shown in Figure 5 for the same edge collapse as in Figure 3 in a convexified version of the same triangular mesh.

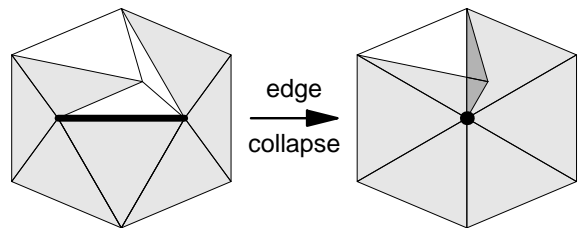


Figure 5: The edge collapse of Figure 3 in the convexified mesh from Figure 4 causes an inversion of an imaginary cell (dark gray). (Compare also with Figure 2.)

Therefore, convexified meshes allow us to test for self-intersections of cells by simply testing all vertex neighbors (including imaginary cells) of the collapsing edge for sign flips of the signed cell volume, which is a local, geometric test as in the case of convex meshes. Thus, the total number of new imaginary cells generated by the convexification is not relevant for the efficiency of this test.

4.2 Preservation of the Convex Hull

Not only are edge collapses in nonconvex meshes more complicated than in convex meshes, they can also transform a convex mesh into a nonconvex mesh.

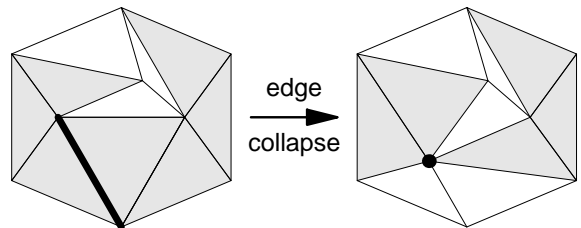


Figure 6: An edge collapse, which could generate a nonconvexity. Two new imaginary cells are inserted in order to preserve the original convex hull.

An example is depicted in Figure 6, which also shows our solution: Instead of recomputing the convex hull (a global operation if implemented naively), we insert imaginary cells between the new vertex and the convex hull in order to preserve the original convex hull. (The effect can also be seen at the bottom of Figure 12.)

This is an efficient, local operation. However, it will also insert some new edges; therefore, a simplification process might run into an endless loop by collapsing edges which are instantly reconstructed by the insertion of imaginary cells. In order to avoid this problem, a simple test for edge collapses has to be added. Edge collapses are avoided if the following three conditions are met: All edge neighbors of the collapsing edge are imaginary, one vertex is part of the boundary of the mesh, and all vertex neighbors of this vertex are imaginary. (For an example see the edge between the new imaginary cells in the right-hand side of Figure 6.)

4.3 Topology Preservation

Edge collapses in convexified meshes are considerably more powerful than edge collapses in the original meshes. For example, disconnected meshes can be joined in order to be simplified, tunnels in the boundary of a mesh can be closed, bridges between meshes can be broken, etc. (Figure 12 shows a new connection between originally disconnected parts of the mesh in the top-right corner and a disconnection of cells at the bottom.)

However, not all of these features are always welcome; instead, it is more appropriate to have full control over the modifications of the topology of the mesh's boundary. Here we present two very simple tests for edge collapses in order to avoid topological changes of the mesh. Both tests involve only vertex neighbors of the collapsing edge. Before presenting these topological tests, we have to define some basic terms.

Def.: The *type* of a cell is either imaginary or non-imaginary.

Def.: A cell T_1 is *connected* to a cell T_2 of the same type if the cells share a vertex (direct connection), or if T_1 is connected to a third cell T_3 of the same type that is connected to T_2 (indirect connection).

Def.: Two cells are *disconnected* if they are not connected.

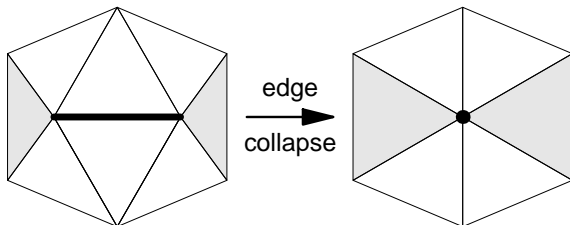


Figure 7: An edge collapse, which connects two non-imaginary cells (gray).

Figure 7 shows an example of an edge collapse that establishes a new connection between two non-imaginary cells. In general, the collapse of an edge e between two vertices v_1 and v_2 can connect two previously disconnected cells if all of the following three conditions are met: All of the edge neighbors of e are of the same type t ; at least one of the vertex neighbors of v_1 is not of type t ; and at least one of the vertex neighbors of v_2 is not of type t . This is a necessary condition; therefore, it is sufficient to avoid edge collapses that fulfill it in order to avoid new connections between cells. The test is the same for triangular and tetrahedral meshes.

Edge collapses are also able to disconnect cells; an example is presented in Figure 8. In order to formulate a test for disconnecting edge collapses, we need one more definition:

Def.: An edge neighbor T of type t of an edge e is *isolated* if none of the faces of T that do not share e are shared by a face neighbor of T of type t .

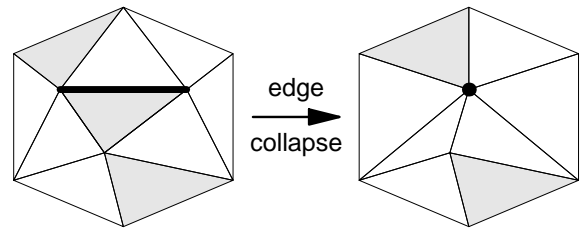


Figure 8: An edge collapse, which disconnects two non-imaginary cells (gray).

Using this definition we can state that the collapse of an edge e can disconnect two previously connected cells if at least one of the edge neighbors of e is isolated. This is again a necessary condition, which works for triangular and tetrahedral meshes. (Note that the faces of a triangular cell are its edges.)

These two topological tests allow us to avoid new connections and/or disconnections simply by avoiding edge collapses that fulfill the conditions stated above. An example with a topological non-trivial mesh is given in Figure 13, which should be compared with Figure 12, where these tests were not applied.

5 Conclusions and Future Work

An idea of Peter Williams to convert nonconvex into convex meshes was successfully applied to solve the problems of edge collapses in nonconvex tetrahedral meshes with topologically non-trivial boundaries, which may be non-manifolds. Intersections of cells, new non-convexities of the boundary, and modifications of the topology are identified and avoided by efficient, local tests. Therefore, the particular problems of simplifying nonconvex meshes by edge collapses are in principle solved.

However, many problems are still open: an efficient and robust computation of the convex hull of an arbitrary tetrahedral mesh; an efficient and robust computation of the tetrahedralization of an arbitrary polyhedron; more elaborated definitions of topology preserving edge collapses (see [2]); and, of course, applications to real-world data sets.

References

- [1] Bernard Chazelle and Leonidas Palios. Triangulating a Nonconvex Polytope. *Discrete & Computational Geometry*, 5:505-526, 1990.
- [2] Tamal K. Dey, Herbert Edelsbrunner, Sumanta Guha, and Dmitry V. Nekhayev. Topology Preserving Edge Contraction. *Publ. Inst. Math. (Beograd) (N.S.)*, 66:23-45, 1999.
- [3] Jovan Popović and Hugues Hoppe. Progressive Simplicial Complexes. In *Computer Graphics Proceedings (SIGGRAPH '97)*, pages 217-224, 1997.
- [4] Oliver G. Staadt and Markus H. Gross. Progressive Tetrahedralizations. In *Proceedings of IEEE Visualization '98*, pages 397-402, 1998.
- [5] Issac J. Trotts, Bernd Hamann, Kenneth I. Joy, and David F. Wiley. Simplification of Tetrahedral Meshes. In *Proceedings of IEEE Visualization '98*, pages 287-295, 1998.
- [6] Peter L. Williams. Visibility Ordering Meshed Polyhedra. *ACM Transactions on Graphics*, 11(2):103-126, 1992.

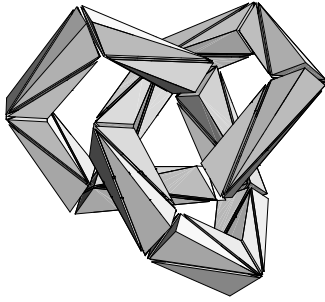


Figure 9: The nonconvex tetrahedral mesh that is the starting point for the calculations depicted in Figures 10-13.

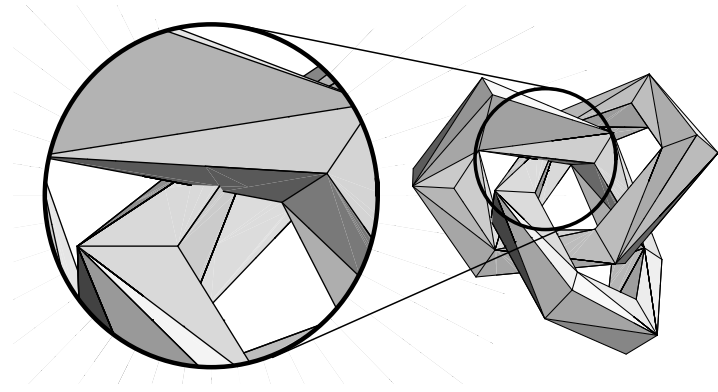


Figure 10: The tetrahedral mesh from Figure 9 after one edge collapse, which causes two self-intersections of the mesh. One of the self-intersections is shown in detail.

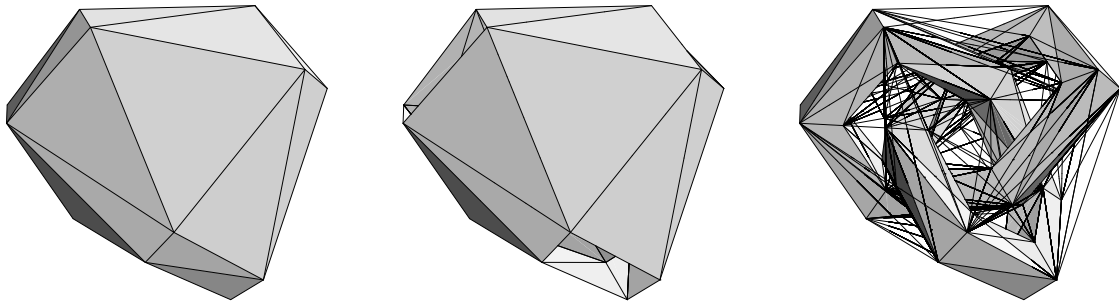


Figure 11: (Intermediate) results of the convexification of the mesh shown in Figure 9. From left to right: the convex hull of the mesh; the nonconvex polyhedron between the convex hull and the boundary of the mesh, which has to be tetrahedralized; and the mesh together with imaginary tetrahedra generated by the tetrahedralization (only edges of imaginary cells are shown).

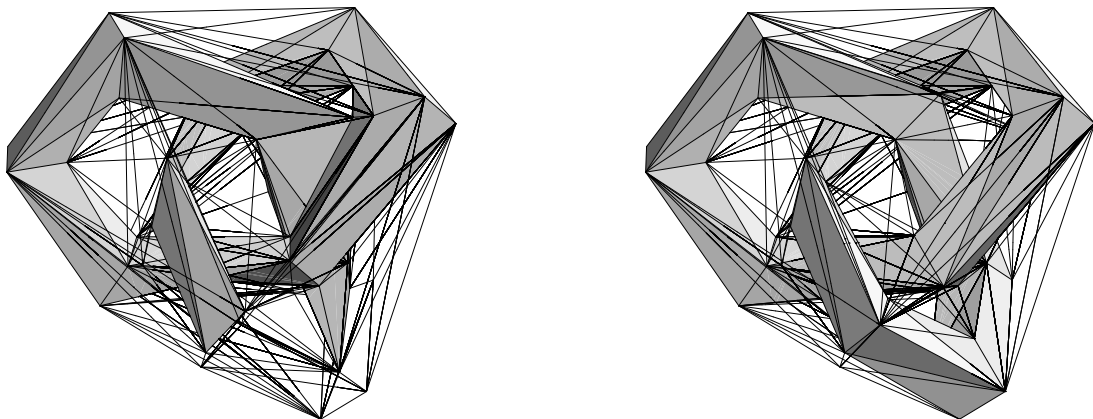


Figure 12: The result of a simplification of the convexified mesh depicted in Figure 11 without topological tests. The geometric tests guarantee that there are no self-intersections and hamper further edge collapses.

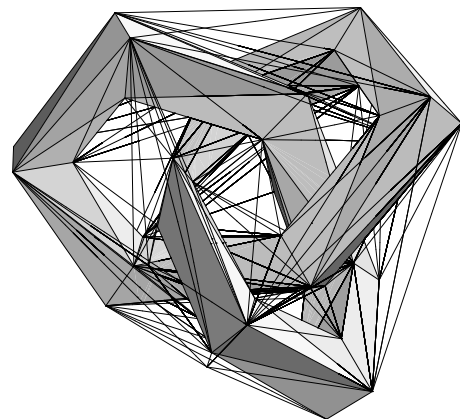


Figure 13: Topological tests in a simplification of the mesh from Figure 11 guarantee the preservation of the connectivity. Therefore, the simplification process is halted earlier than without these tests.