

# Simplification of Nonconvex Tetrahedral Meshes

Martin Kraus and Thomas Ertl

Visualization and Interactive Systems Group, Institut für Informatik,  
Universität Stuttgart, Breitwiesenstr. 20–22, 70565 Stuttgart, Germany,  
E-mail: {Martin.Kraus | Thomas.Ertl}@informatik.uni-stuttgart.de

**Summary.** The simplification of nonconvex tetrahedral meshes with the help of edge collapses (edge contractions) is considerably more complex than the corresponding simplification of convex meshes. In particular, edge collapses in nonconvex meshes may cause intersections of cells that are hard to detect, and therefore hard to avoid. More precisely spoken, the problem is quadratic in the number of cells.

However, we show how to reduce the complexity of this problem by employing a preprocessing step, which was originally proposed in order to sort nonconvex meshes. Moreover, 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 of data meshes are often employed. The production of such representations requires a simplification of an original, fine mesh to a coarser mesh, e.g. by resampling a mesh to a lower resolution with less vertices. For unstructured tetrahedral meshes, one way of performing this simplification is to apply a sequence of edge collapses, which are discussed in some detail in Sect. 2. In particular, we will describe problems of edge collapses that are specific to nonconvex tetrahedral meshes.

Our solution to these problems in nonconvex meshes consists of two parts: A preprocessing step – originally proposed by Williams in the context of sorting nonconvex meshes [7] – is briefly described in Sect. 3. The second part – edge collapses in the resultant meshes – is discussed in Sect. 4 with special attention being paid to modifications of the topology of the mesh's boundary. The algorithm is demonstrated with an example in three dimensions in Sect. 5. Section 6 presents our conclusions and plans for future work on this subject.

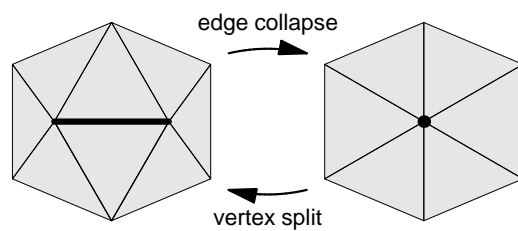
## 2 Background and Related Work

### 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. (Note that the term *edge collapse* in computer graphics corresponds to the term *edge contraction* in computational geometry. We will use the former in

order to be consistent with the majority of our references.) 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 Figs. 1–8 before presenting the method in three dimensions in Figs. 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.



**Fig. 1.** The collapse (contraction) of an edge (*thick line*) to a vertex (*dot*) and the inverse vertex split

The effect of an edge collapse (see Fig. 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*.

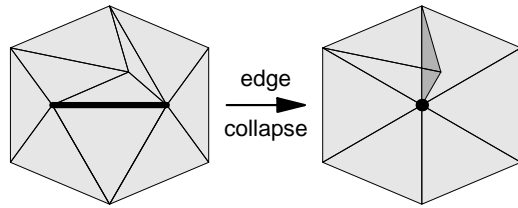
Edge collapses are one of the most powerful tools to simplify triangular or tetrahedral meshes. They can be employed to remove vertices or edges [5] and also to remove triangles or tetrahedra by successive edge collapses [2,6].

Moreover, a sequence of edge collapses can be applied in order to produce hierarchical representations of triangular and tetrahedral meshes as demonstrated in many publications, for example [2,4–6].

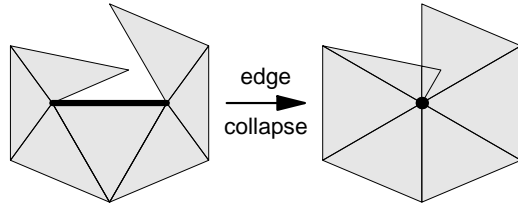
## 2.2 Avoiding Intersections of Cells

As demonstrated in Fig. 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 [2,5,6].

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 Fig. 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.



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



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

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 Figs. 3 and 10.

A naive procedure to avoid such intersections is to test the vertex neighbors of the collapsing edge for intersections with all boundary cells of the mesh. As the worst-case 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 [2] and [5] while the system described in [6] tries to preserve the boundary of the tetrahedral mesh.

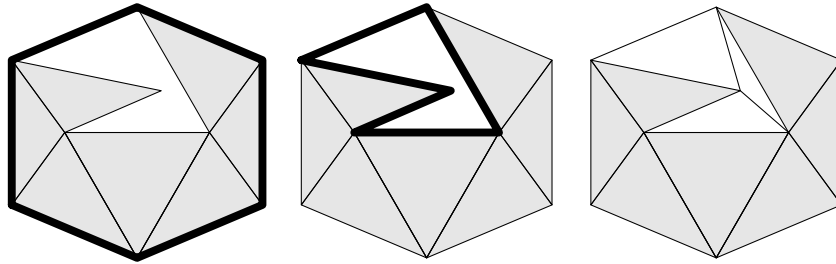
However, performance issues are not the only problem of edge collapses in nonconvex meshes. Additional problems occur in disconnected meshes as edge collapses are obviously not able to join clusters of disconnected meshes in order to simplify them. More generally spoken, it is desirable to modify the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

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

### 3 Convexification of Nonconvex Meshes

In [7] Williams proposed 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.)

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



**Fig. 4.** A step-by-step convexification of the mesh shown in the left-hand side of Fig. 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

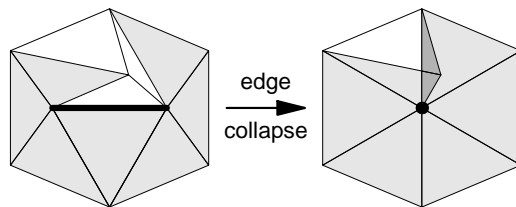
shown in Sect. 4 the number of imaginary cells generated by the triangulation is not relevant in the context of edge collapses. An optimal algorithm (with respect to the number of generated tetrahedra) for the tetrahedralization of nonconvex polyhedra was published in [1].

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 Edge Collapses in Convexified Meshes

### 4.1 Geometric Tests

As mentioned in Sect. 2 and shown in Fig. 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 Fig. 5 for the same edge collapse as in Fig. 3 in a convexified version of the same triangular mesh.

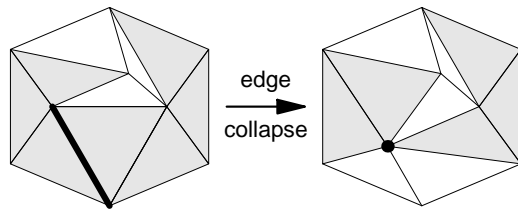


**Fig. 5.** The edge collapse of Fig. 3 in the convexified mesh of Fig. 4 causes an inversion of an imaginary cell (*dark gray*). (Compare also with Fig. 2)

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.



**Fig. 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 Fig. 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 Fig. 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 Fig. 6.)

#### 4.3 Incomplete 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.)

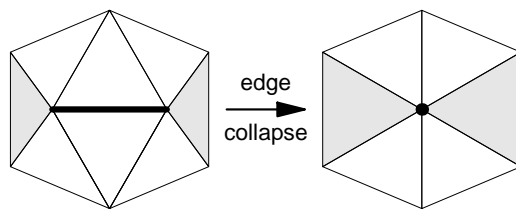
However, not all of these features are always welcome; instead, it is more appropriate to have full control over the modifications of the topology of the mesh's boundary. Here we present two very simple tests for edge collapses in order to avoid

topological changes of the mesh. The tests guarantee the preservation of a certain *edge connectivity*, but do not avoid all possible topological changes. 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.

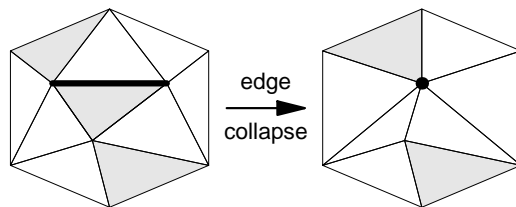
*Def.:* A cell  $T_1$  is *connected* to a cell  $T_2$  of the same type if the cells share a vertex (direct connection), or if  $T_1$  is connected to a third cell  $T_3$  of the same type that is connected to  $T_2$  (indirect connection).

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



**Fig. 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.



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

Edge collapses are also able to disconnect cells; an example is presented in Fig. 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  $t$ .

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 in three dimensions will be presented in the Sect. 5.

#### 4.4 Complete Topology Preservation

In the preceding subsection, we were only concerned with a certain edge connectivity. It is possible to extend this concept to faces of cells by defining a corresponding *face connectivity*:

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

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

*Def.:* An edge neighbor  $T$  of type  $t$  of an edge  $e$  is *actively isolated* if all of the faces of  $T$  that do not share  $e$  are shared by face neighbors of  $T$  that are not of type  $t$ .

The collapse of an edge  $e$  can face connect previously face disconnected cells if at least one of the edge neighbors of  $e$  is actively isolated.

It is necessary to test all edge neighbors of the collapsing edge for face disconnections. This is a sufficient test because face disconnections of cells that are not edge neighbors of the collapsing edge imply the existence of a face disconnection between edge neighbors of the collapsing edge.

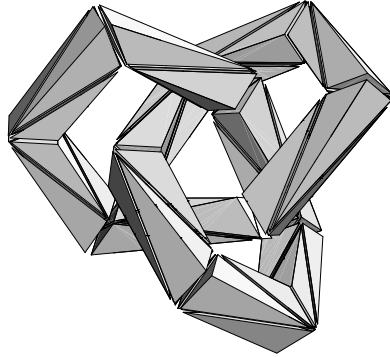
This is again a simple, local test, which is easily implemented. However, even the combined tests for edge and face connectivity do not preserve the topology completely. Fortunately, a rigorous solution to this problem has been published by Dey et al. in [3]. In fact, our tests turn out to be special cases of the *Link Conditions* for 3-complexes. (See Theorem C in [3].)

## 5 Example in Three Dimensions

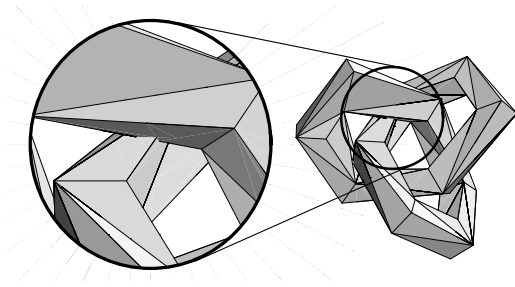
After presenting our algorithm in Sects. 3 and 4, we will now briefly demonstrate our algorithm in three dimensions. Figure 9 shows the topological non-trivial mesh that is the starting point for the calculations depicted in Figs. 10–13

The problem of cell intersections caused by edge collapses is illustrated in Fig. 10: An edge collapse in the original mesh of Fig. 9 has caused two self-intersections of the nonconvex mesh. (One of these self-intersections is magnified.)

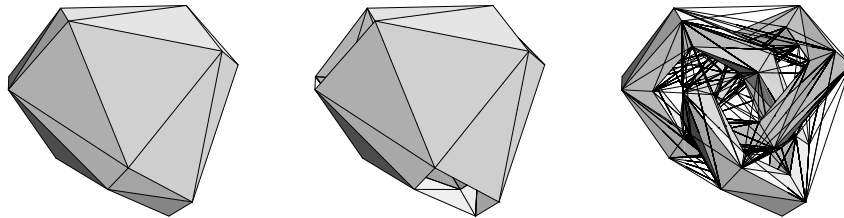
In analogy to Fig. 4, the convexification of the trefoil mesh is depicted in Fig. 11. In order to simplify the original mesh, edge collapses have to be performed in the resultant, convexified mesh.



**Fig. 9.** A tetrahedral mesh with the shape of a trefoil knot



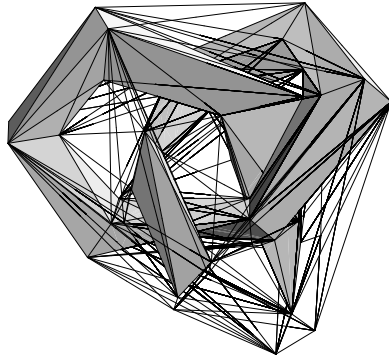
**Fig. 10.** The tetrahedral mesh of Fig. 9 after one edge collapse



**Fig. 11.** (Intermediate) results of the convexification of the mesh shown in Fig. 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)

As mentioned, the convexification requires the computation of the convex hull of the mesh, the computation of the space between the mesh's boundary and its convex hull, and the tetrahedralization of this space. The most difficult problem of these three steps is the tetrahedralization. Although a general algorithm for this problem exists [1], we did not implement it yet, but decided to use a strongly simplified variant of this algorithm in order to process our specific example.

Figure 12 shows a simplified mesh after several edge collapses. The employed geometric tests for cell inversions guarantee that there are no self-intersections and hamper further edge collapses. Note that more edge collapses were possible, if the new vertex is not placed in the center of the collapsed edge. However, we did not exploit this freedom in our example.

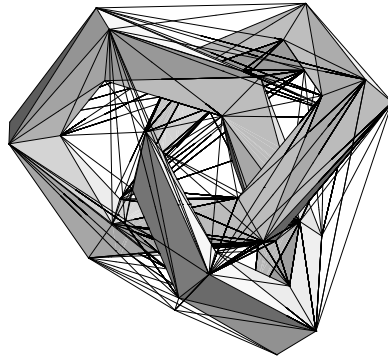


**Fig. 12.** The result of a simplification of the convexified mesh depicted in Fig. 11 without topological tests

The topology of the mesh in Fig. 12 is obviously different from the original mesh as the knot was disconnected. Topological tests in a simplification of the convexified mesh of Fig. 11 guarantee the preservation of edge connectivity as shown in Fig. 13; therefore, the simplification process is halted earlier than without these tests. Again, further simplification steps are possible with more general edge collapses.

## 6 Conclusions

As we have demonstrated, the idea of Williams to convert nonconvex into convex meshes can be successfully applied to solve the problems of edge collapses in non-convex tetrahedral meshes with topologically non-trivial boundaries, which may be non-manifolds. Intersections of cells, new nonconvexities 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.



**Fig. 13.** Same as Fig. 12 with additional topological tests

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; the integration of complete topology preservation into our system; and, of course, applications to real-world data sets.

## References

1. Chazelle B., Palios L. (1990) Triangulating a Nonconvex Polytope. *Discr Comp Geom* 5:505–526
2. Cignoni P., Costanza D., Montani C., Rocchini C., Scopigno R. (2000) Simplification of Tetrahedral Meshes with Accurate Error Evaluation. In: Ertl T., Hamann B., Varshney A. (Eds.) *Visualization 2000*, Salt Lake City, Utah, USA, October 8–13, 2000. IEEE Computer Society Press, Piscataway, and ACM Press, New York, 85–92
3. Dey T.K., Edelsbrunner H., Guha S., Nekhayev D.V. (1999) Topology Preserving Edge Contraction. *Publ Inst Math (Beograd)* 66:23–45
4. Popović J., Hoppe H. (1997) Progressive Simplicial Complexes. In: Whitted T., Mones-Hattal B. (Eds.) *SIGGRAPH 97*, Los Angeles, California, USA, August 3–8, 1997. ACM Press, New York, 217–224
5. Staadt O.G., Gross M.H. (1998) Progressive Tetrahedralizations. In: Evert D., Hagen H., Rushmeier H. (Eds.) *Visualization '98*, Research Triangle Park, North Carolina, USA, October 18–23, 1998. IEEE Computer Society Press, Piscataway, and ACM Press, New York, 397–402
6. Trotts I.J., Hamann B., Joy K.I., Wiley D.F. (1998) Simplification of Tetrahedral Meshes. In: Evert D., Hagen H., Rushmeier H. (Eds.) *Visualization '98*, Research Triangle Park, North Carolina, USA, October 18–23, 1998. IEEE Computer Society Press, Piscataway, and ACM Press, New York, 287–295
7. Williams P.L. (1992) Visibility Ordering Meshed Polyhedra. *ACM Trans Graph* 11(2):103-126