Appeared in *Proceedings of IEEE Visualization '98*, pp. 397-402, 1998.

Progressive Tetrahedralizations

Oliver G. Staadt, Markus H. Gross

Computer Graphics Research Group
Department of Computer Science
Swiss Federal Institute of Technology (ETH) Zurich
e-mail: {staadt, grossm}@inf.ethz.ch

ABSTRACT

This paper describes some fundamental issues for robust implementations of progressively refined tetrahedralizations generated through sequences of edge collapses. We address the definition of appropriate cost functions and explain on various tests which are necessary to preserve the consistency of the mesh when collapsing edges. Although being considered a special case of progressive simplicial complexes [10], the results of our method are of high practical importance and can be used in many different applications, such as finite element meshing, scattered data interpolation, or rendering of unstructured volume data.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – surfaces and object representations.

Keywords: mesh simplification, multiresolution, level-of-detail, unstructured meshes, mesh generation.

1 INTRODUCTION

Progressive meshes [7] and its generalizations to higher dimensions [10] proofed to be an extremely powerful notion for the efficient representation of triangulated geometric objects at different levels-of-detail. Although a general formulation for arbitrary triangulations has already been given in [10], the special case of progressive tetrahedralizations (PT) is of enormous practical importance, since it can be used as a sophisticated representation for a large variety of computations. Finite element discretizations, from where our contribution was motivated, are one example. Here, sophisticated computational methods try to find an optimal balance between refinements of the mesh and of the polynomial degree of the basis functions. Other important applications of progressive tetrahedralizations comprise interpolation and rendering of scattered volume data, where successively refinable methods would definitely improve the performance of existing approaches. A general overview of various mesh simplification methods, including those based on edge collapses, can be found in [6].

However, regardless of the brilliance and simplicity of the idea of edge collapsing, a brute force implementation of the method may rapidly destroy the consistency of the mesh. Various artifacts can be introduced, such as flipped, intersected and degenerated tetrahedra, which in turn may impede any finite element computation. An example is given in Fig. 1. Here two boundary tetrahedra intersect due to an edge collapse in a locally concave mesh region.

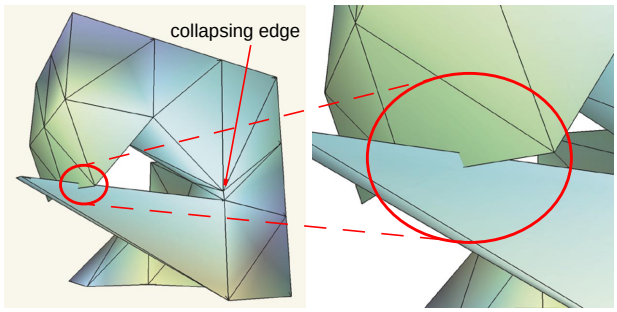


Figure 1: Intersection of two tetrahedra: a) The edges of two non-adjacent tetrahedra bounding the volume intersect while collapsing an edge in a locally concave mesh region. b) Close-up of the intersection (see also CP 1).

In this paper we elaborate on some pitfalls and fallacies people might get caught in when trying to implement the method of edge collapsing for tetrahedral meshes. Specifically, we address the issue of defining appropriate cost functions. Unlike the elegant geometric approach presented in [7], we must account for volume and application specific properties, such as volume preservation, gradient estimation of the underlying data or aspect ratio of the simplex. In addition, we devised a sequence of tests to guarantee a robust and consistent progressive tetrahedralization. Some results obtained on computational fluid dynamics (CFD) data sets illustrate the performance of the PT[1].

The remainder of this paper is outlined as follows:

- In Section 2, we recapitulate previous work that is closely related to our method.
- The main part of this paper (Section 3) elaborates on the core algorithm of our PT scheme and introduces new cost functions for tetrahedral meshes and special tests that guarantee mesh consistency.
- Finally, we describe some implementation issues and demonstrate the method's usefulness on an irregular mesh data set of a turbine blade in Section 4.

2 RELATED WORK

There are numerous schemes for simplifying geometric meshes in computer graphics and scientific visualization, of which Garland and Heckbert provide an extensive overview in [6].

[1] The method is currently implemented as a set of AVS/Express modules and will be made available shortly.

Previous work closely related to our method can be broadly categorized into 2 main classes. Firstly, methods based on edge-collapse/vertex-split transformations and secondly, decimation methods which specifically support simplification of volumetric data.

2.1 Methods based on edge collapsing

Hoppe et al. [9] proposed a triangular mesh simplification scheme, based on edge collapse and vertex split transformations. A set of energy functions is used in order to decide on the sequence of edge collapse transformations, and a sophisticated optimization method, which determines the position of the new vertex, is provided. Hoppe [7] extended this work with support for selective refinement and progressive transmission, as well as with lossy geometric compression [2] and the possibility to interpolate smoothly between different levels of approximation (geomorphs). A more detailed description of progressive meshes is given in Section 3.1.

This has been the basis for several other extensions. Xia et al. [16] and Hoppe [8] focused on fast, hierarchical representations, that enable one to efficiently reconstruct selected parts of a triangular progressive mesh. All of these algorithms preserve the topology of the original data, whereas others [10, 3, 13, 12] allow topological changes, such as filling of holes or connecting disconnected parts of a mesh. Popovic et al. [10] devised a general approach for arbitrary dimensional simplicial complexes, which also allows topological changes of the data, in order to further reduce the complexity of the mesh. Unfortunately, they only provide an implementation and examples for the triangular case. Other important types of simplices, such as tetrahedra, are not addressed in detail.

Other work [12, 5, 13] differs from [9] and [7] in the ordering of edge collapse transformations and the optimization method which specifies the new vertex position. Garland et al. [3] propose a very elegant scheme using quadrics for both tasks.

2.2 Vertex removal methods

Various decimation algorithms have been published, which remove single vertices from the mesh and retriangulate the resulting hole. One of the most widely used methods in that category is Schroeder et al. [14].

Fewer methods have been devised that decimate higher dimensional simplicial complexes. Staadt et al. [15] presented a method for simplifying and compressing unstructured triangular and tetrahedral meshes by using hierarchical, wavelet-based decimation schemes and Delaunay tetrahedralization. Grosso et al. [4] use finite element computations to represent triangular and tetrahedral meshes at multiple levels. Cignoni et. al [1] propose a framework which is based on a decimation method and allows one to represent tetrahedral meshes at arbitrary resolution.

3 PROGRESSIVE TETRAHEDRALIZATIONS

For reasons of readability, we recapitulate some basic definitions for progressive meshes [7] and adapt them to progressive tetrahedralizations (PT). General introductions are provided in [7] or [10].

3.1 Background

In PT representations, a tetrahedral mesh with scalar attributes s_i assigned to each vertex $\mathbf{v}_i$ is defined as

$$(M^0, \text{vsplit}_0, \text{vsplit}_1, \ldots, \text{vsplit}_{n-2}, \text{vsplit}_{n-1}) \qquad (1)$$

where M^0 is some coarse base mesh and vsplit_i are vertex split operations to reconstruct the original mesh $M = M^n$ from M^0:

$$M^0 \xrightarrow{\text{vsplit}_0} M^1 \xrightarrow{\text{vsplit}_1} \ldots \xrightarrow{\text{vsplit}_{n-1}} M^n . \qquad (2)$$

Conversely, M^0 is derived from M through a series of edge collapse operations ecol_i which are inverse to vsplit_i:

$$M^n \xrightarrow{\text{ecol}_{n-1}} M^{n-1} \xrightarrow{\text{ecol}_{n-2}} \ldots \xrightarrow{\text{ecol}_0} M^0 . \qquad (3)$$

Each ecol_i replaces an edge e_i with vertices $\mathbf{v}_a$ and $\mathbf{v}_b$ by a new vertex $\bar{\mathbf{v}}_a$. As opposed to some other methods we preserve the topological type of the mesh, that is, all instances of M are homeomorphic. Specifically, we prohibit degenerations of tetrahedra into lower dimensional simplices. The set of tetrahedra sharing e_i will be called $\{icells_i\}$. Thus, an edge split adds the tetrahedra in $\{icells_i\}$ to the list of active elements. Conversely, the set of non-vanishing tetrahedra affected by the associated edge collapse is called $\{ncells_i\}$. Fig. 2 depicts an edge collapse operation in a tetrahedral mesh. All tetrahedra sharing e_i vanish, whereas all tetrahedra sharing only one of the vertices of the edge change in shape.

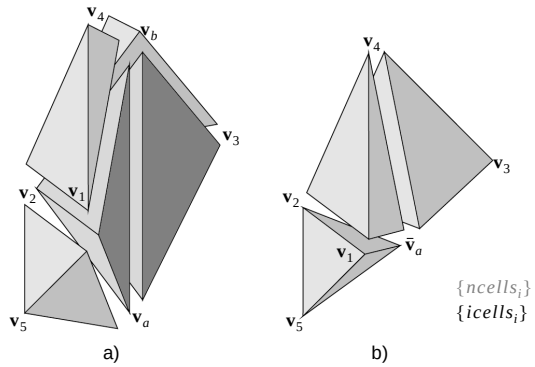


Figure 2: Edge collapse in a tetrahedral mesh: a) Mesh before collapsing edge $(\mathbf{v}_a, \mathbf{v}_b)$. b) Configuration after collapse with resulting vertex $\bar{\mathbf{v}}_a$. (The tetrahedra are shrinked to emphasize the underlying 3-dimensional structure. See also CP 2).

In order to compute a sequence of robust, non-degenerate and consistent meshes, the following aspects have to be considered:

- *Cost functions* which determine the order of ecol operations depending on desired mesh optimization criteria.
- *Sharp* and *feature edges* which should be preserved can be checked during preprocessing.
- *Intersections* and *inversions* of tetrahedra inside and outside of $\{\{icells_i\} \cup \{ncells_i\}\}$, such as the one in Fig. 1, have to be processed at run time.

The remainder of this paper elaborates on the details of these issues.

3.2 Cost Functions

As already mentioned in Section 2, various elegant algorithms [3, 7] based on the ecol/vsplit paradigm used cost functions optimized for triangular surfaces, often accounting for distance measures, triangle shape, and others. In tetrahedral meshes, however, we have to redefine the terms of the cost function considering other features, like volume preservation or gradients. Although many different measures are conveivable to control the simplification process, the following ones yield a good balance between required degrees of freedom and the difficulty of parameter optimization.

Thus, in our setting, for each edge $e_i = (\mathbf{v}_a, \mathbf{v}_b)$, the associated edge collapse operation $\text{ecol}_i(a, b)$: $M^i \leftarrow M^{i+1}$ is assigned the following cost:

$$\Delta E(e_i) = \Delta E_{\text{grad}}(e_i) + \Delta E_{\text{vol}}(e_i) + \Delta E_{\text{equi}}(e_i) \quad . \tag{4}$$

The first term ΔE_{grad} is defined as

$$\Delta E_{\text{grad}}(e_i) = w_{\text{grad}} \cdot |s_a - s_b| \tag{5}$$

and forms a simplified measure for the difference of underlying scalar volume function along the edge e_i. Hence, edges with considerably differing scalar attributes are assigned high costs.

When collapsing edges and removing tetrahedra from the mesh, the overall volume tends to decrease, that is the mesh shrinks down. Therefore, we introduce a second term ΔE_{vol} penalizing volume changes:

$$\Delta E_{\text{vol}}(e_i) = w_{\text{vol}} \cdot \Big(\sum_{T_j \in \{ncells_i\}} (\text{vol}(T_j) - \text{vol}(\bar{T}_j)) + \sum_{T_j \in \{icells_j\}} \text{vol}(T_j) \Big) \quad . \tag{6}$$

T_j denote all tetrahedra in the set of neighborhood cells $\{ncells_i\}$ of e_i and introduced cells $\{icells_i\}$, respectively. $\text{vol}(T_j)$ stands for the volume of T_j and $\bar{T}_j$ is the tetrahedron after the collapse. Note that only simplices in $\{icells_i\} \cup \{ncells_i\}$ can contribute to volume changes.

Especially in FEM applications, it is often required that tetrahedra sustain equilateral shape. ΔE_{equi} can be employed in order to balance the edge length of tetrahedra:

$$\Delta E_{\text{equi}}(e_i) = w_{\text{equi}} \cdot \sum_{T_j \in ncells_i} \Big(\sum_{\{a,b\} \in T_j} (l_{a,b} - m_j)^2 - \sum_{\{a,b\} \in \bar{T}_j} (l_{a,b} - \bar{m}_j)^2 \Big) \tag{7}$$

with edge length $l_{a,b} = |\mathbf{v}_a - \mathbf{v}_b|$ and average edge length $m_j = 1/|T_j| \cdot \sum_{\{a,b\} \in T_j} l_{a,b}$.

Each term can be weighted individually by coefficients w_{grad}, w_{vol} and w_{equi}, respectively, to allow adoption to specific data sets and applications.

Note that the initial mesh M^n will usually be generated from some triangulation scheme. Depending on the application context and the desired mesh features it can be advantageous to include some $\Delta E_{\text{edgelen}}(e_i) = w_{\text{edgelen}} \cdot |\mathbf{v}_a - \mathbf{v}_b|$ into the cost function thereby enforcing short edges to be collapsed earlier.

3.3 Static Tests

Unfortunately, brute force selection of edges according to the cost function from above can introduce mesh inconsistencies, like degeneration, folding, intersection, or loss of individual features.

In order to avoid these types of artifacts, some *static* tests can be carried out prior to building the edge heap. Before we introduce the test criteria we have to define some properties of edges and vertices:

- *sharp (boundary) edge*: an edge e_i is called sharp if it lies on the boundary of the mesh.
- *sharp (boundary) vertex*: a vertex v_i is called sharp if at least one edge incident to v_i is sharp.
- *sharp (boundary) face*: a triangular cell face f_i is called sharp if all its edges are sharp.
- *sharp (boundary) cell*: a tetrahedral cell T_i is sharp if at least one of its faces is sharp.

Sharp edges can be detected efficiently by analyzing all vertices $v_j \in \{icells_i\}$ assuming appropriate data structures to represent the mesh. In a preprocessing step we label all sharp vertices and edges, respectively. Table 1 lists the 5 different cases for combinations of sharp edges and vertices. Only cases 1 and 5 pass the consistency test.

Table 1: The 5 possible cases for combinations of sharp edges and sharp vertices. The examples refer to Fig. 3.

CASE	e	v_a	v_b	PERMISSIBLE	EXAMPLE
1				yes	$(\mathbf{v}_3, \mathbf{v}_4)$
2		*sharp*		optional[a]	$(\mathbf{v}_1, \mathbf{v}_{14})$
3			*sharp*	optional[a]	$(\mathbf{v}_3, \mathbf{v}_2)$
4		*sharp*	*sharp*	no[b]	$(\mathbf{v}_8, \mathbf{v}_{10})$
5	*sharp*	*sharp*	*sharp*	yes[c]	$(\mathbf{v}_0, \mathbf{v}_1)$

a: cases 2 and 3 induce "dents" on the boundary surface which may not be desired.
b: case 4 introduces degenerated cells, since $\mathbf{v}_9$ is not deleted.
c: a sharp edge always implies that its vertices are sharp. Simplifications of the boundary surfaces are allowed.

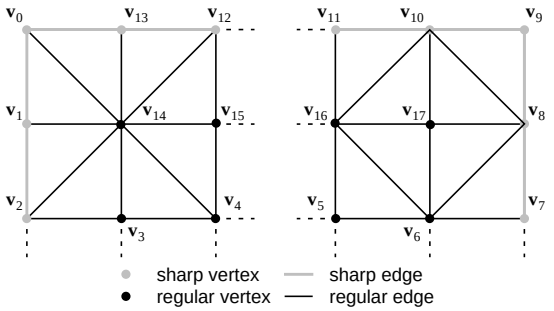


Figure 3: Naming conventions used for static consistency tests. For simplicity, the examples are depicted in 2D.

3.4 Dynamic Tests

Unfortunately, not all inconsistencies can be fixed with the static tests from above. Some severe problems arise *dynamically* while performing individual ecol operations and further tests are required on the fly.

The normal flipping heuristic from [12] can be generalized in order to circumvent folding or self-intersection of tetrahedra. This can easily be implemented by analyzing the volume of all $T_i \in \{ncells_i\}$ before and after the collapse. Recall that the volume of a tetrahedron is defined by the parallelepipedial product of its 3 edges e_i, e_j and e_k:

$$\frac{1}{6}[e_i, e_j, e_k] = \frac{1}{6}\langle e_i \times e_j, e_k \rangle . \qquad (8)$$

If the volume of one of the neighboring tetrahedra becomes negative, tetrahedral folding occurs. In this case the edge fails the consistency test. This test also avoids degenerate cells by setting a lower volume threshold to be retained after the collapse.

We start from the following observation: edge collapses can cause global intersections of tetrahedra, a simple example of which is shown in Fig. 1. This requires additional testing. If the set $\{\{icells_i\} \cup \{ncells_i\}\}$ contains no sharp edge, it's boundary forms a polytope entirely wrapping the edge. A collapse of the edge, however, does not affect the boundary of the polytope, whose disjoint triangulation is given by the tetrahedra $\bar{T}_j \in \{ncells_i\}$. Thus, intersections can only occur with sharp cells and we can restrict the intersection tests to the mesh boundary.

Fig. 4a depicts the top view of a tetrahedral mesh where T_j is a sharp cell that is close to, but not intersecting the boundary of the mesh. Let e_i be the edge to be collapsed next and let vertex $\mathbf{v}_b$ be closer to the viewpoint than $\mathbf{v}_a$. The situation after the collapse where T_j is intersecting two faces of $\{ncells_i\}$ is depicted in Fig. 4b.

In essence, we have to perform triangle–triangle intersection tests [11] in case of sharp edges or vertices which can be carried out as follows:

First, we define the set of triangles $\{f_k\}$ containing all sharp faces of tetrahedra $T_j \in \{ncells_i\}$. These are the faces which can change after ecol_i since they all share the new vertex $\bar{\mathbf{v}}_a$. Thus they are our prime candidates for intersection with other sharp faces.

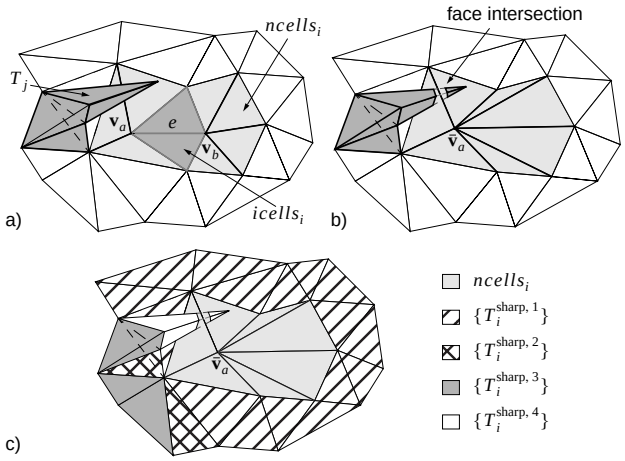


Figure 4: Tetrahedron T_j intersects two faces of $\{f_k\}$ after collapsing edge e_i into vertex $\bar{\mathbf{v}}_a$. a) before collapse. b) after collapse. c) Traversal of the mesh $\{T_i^{\text{sharp},\,s}\}$ for iteration steps 1–4.

In order to avoid testing these faces against all other sharp tetrahedra, we propose the following iterative method: The algorithm starts from an initial set $\{T_i^{\text{sharp}}\}$ containing the subset of all sharp tetrahedra in the direct neighborhood of $ncells_i$.

We take the first element of $\{T_i^{\text{sharp}}\}$ and test for intersection with all faces in f_k. If the test fails and no intersection occurs, we label the tetrahedron as visited and proceed to the next element of $\{T_i^{\text{sharp}}\}$. Otherwise we can abort the test and reverse the current ecol_i operation. The restriction to the sharp faces of each tetrahedron simplifies the intersection test, since many cell have only one sharp face (see Fig. 4). After probing all cells, we replace all visited tetrahedra in $\{T_i^{\text{sharp}}\}$ with their non-visited neighboring sharp cells and thereby traverse the mesh. The $\{T_i^{\text{sharp},\,s}\}$ are shown in Fig. 4c for different iteration steps.

The following pseudo code summarizes the principle steps of the intersection test:

```
intersection_test {
    // Tsharp[i,m]: sharp cells T_i at iteration s
    // f[k]: sharp faces in ncells_i
    // calculate_intersections: triangle-triangle
    // intersection test
    // S: maximum iteration level
    s = 0;
    while (Tsharp[i,s] not empty && s < S){
        forall f[i] in Tsharp[i,s]
            forall f[k]
                calculate_intersections(f[i], f[k]);
        update Tsharp[i,s];
        s++;
    }
}
```

Note that the test asymptotically traverses all sharp faces of the mesh and consumes time proportional $O(F \cdot B)$, where F stands for the number of elements in $\{f_k\}$ and B for the overall number of sharp faces in the data set. As we will demonstrate in Section 4, one can restrict the iteration to an upper bound in practice.

3.5 Edge Collapse

After calculating the cost of each edge and determining the static conditions of the corresponding edge collapse operation, we build a heap with all remaining edges sorted according to their associated costs. In order to carry out the edge collapse ecol_i, we pop the topmost edge from the heap, check for intersections, construct a vsplit_i record for reconstruction, and update all edges on the heap which are in $\{ncells_i\}$. This process is repeated until the heap is empty or the desired number of collapses has been reached. Although we implemented various schemes for the optimal positioning of $\bar{\mathbf{v}}_a$, such as stochastic optimization, we found that the halfway between $\mathbf{v}_a$ and $\mathbf{v}_b$ is a good choice.

4 RESULTS

Our implementation is currently integrated into AVS/Express. Although in this paper we focus on specific issues of tetrahedral meshes, our AVS/Express modules can handle triangular meshes as well.

For the following investigations, an irregular mesh of a turbine blade was selected. The original data set consists of 576,576 tetrahedra with scalar node data representing pressure between the blades. Fig. 3 shows results with various levels of reconstruction, different settings of the cost functions, and extracted isosurfaces, both for the original mesh and for a selected subset with only one blade. Table 2 lists the performance statistics and parameter settings used to generate the example meshes. M^n denotes the original number of tetrahedra and M^i is the number of tetrahedra of the reconstructed mesh. ΔE_{grad}, ΔE_{vol}, and ΔE_{equi} indicate the individual cost function terms used for the example. *Time* shows the computation time for a full mesh collapse, *i* the number of vsplits, and *Hits* the number of dynamically detected intersections ($s = 2$). The performance of our algorithms was measured on an Indigo2 R10000@195MHz. The reconstruction time for the examples was between 0.1s and 0.8s, depending on the number of splits.

Table 2: Parameter settings and performance statistics for the results shown in Fig. 3.

Fig.	M^n	M^i	E_{grad}	E_{vol}	E_{equi}	Time	i	Hits
a)	576,576	117,139	✓	✓	✓	4h57'	15,000	40,415
b)	78,624	8,317	✓	✓	✓	34'23"	1,000	6,040
c)	78,624	13,327	✓	✓	✓	34'23''	2,000	6,040
d)	78,624	13,327	✓	✓	✓	34'23''	2.000	6,040
e)	78,624	10,554	✓	×	×	19'26''	1,000	2,386
f)	78,624	7,440	×	✓	×	41'58''	500	7,919
g)	78,624	8,169	×	×	✓	39'35''	1,000	4,798

Fig. 3a shows isosurfaces of the blades, extracted from M^{15000} at isovalue 5.0. A slice through the mesh is displayed in order to depict the irregularity of the mesh reconstruction. Note especially the high quality of the isosurfaces which confirms the effectiveness of ΔE_{grad}. A subset of the mesh with one blade is shown in Fig. 3b-c for two different resolutions. In order to balance the individual terms of ΔE, the weights were set to $w_{\text{grad}} = 1$, $w_{\text{vol}} = 500$, and $w_{\text{equi}} = 200$, respectively.

In Fig. 3d, the mesh is cut to render its internal structure. In order to emphasize the influence of individual cost function terms we computed Fig. 3e-g.We observe that each of the energy terms stands for a specific feature. ΔE_{grad}, for instance in Fig. 3e reconstructs the blade region very well, but violates volume preservation and produces poorly shaped simplices. As expected, ΔE_{vol} sustains the volume, whereas ΔE_{equi} preserves the equilateral shape of the tetrahedra.

Usually, it can not be guaranteed that all intersections are detected with $s = 2$. For the above example the total number of intersection hits with $s \to \infty$ is 6,116, which means that 98.8% of the intersections have already been detected at iteration level 2.

5 CONCLUSIONS AND FUTURE WORK

We have presented a technique for generating progressive tetrahedralizations, especially emphasizing on problems such as intersections or degenerations. The cost functions that we have proposed are well suited for a wide range of volume data sets in different applications areas. Although tetrahedral mesh decimation is a complex task, we have implemented a fast and efficient method that avoids most of the pitfalls of tetrahedral meshing.

Future work will be directed towards combining the optimization of vertex placement after edge collapse with avoiding selfintersection and folding. By solving a constraint linear optimization problem [11], we can determine a polytope, in which the new vertex can be placed intersection-free. We would also like to enhance error analysis in order to quantify geometric and data approximation errors.

ACKNOWLEDGMENTS

This research was supported in parts by the ETH research council under grant No. 41–2642.5 and the Schlumberger Stichting Fund. Our special thanks to Marcelo Vetter for implementing parts of the algorithms.

REFERENCES

[1] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. "Multiresolution representation and visualization of volume data." *IEEE Transactions on Visualization and Computer Graphics*, 3(4):352–369, Dec. 1997.

[2] M. F. Deering. "Geometry compression." In R. Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 13–20. ACM SIGGRAPH, Addison Wesley, Aug. 1995. held in Los Angeles, California, 06-11 August 1995.

[3] M. Garland and P. S. Heckbert. "Surface simplification using quadric error metrics." In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 209–216, Aug. 1997.

[4] R. Grosso, C. Lürig, and T. Ertl. "The multilevel finite element method for adaptive mesh optimization and visualization of volume data." In *Proceedings of IEEE Visualization '97*, pages 387–394, 1997.

[5] A. Guéziec. "Surface simplification inside a tolerance volume." Technical Report RC 20440, IBM T. J. Watson Research Center, 1996.

[6] P. Heckbert, J. Rossignac, H. Hoppe, W. Schroeder, M. Soucy, and A. Varshney. "Course no. 25: Multiresolu-

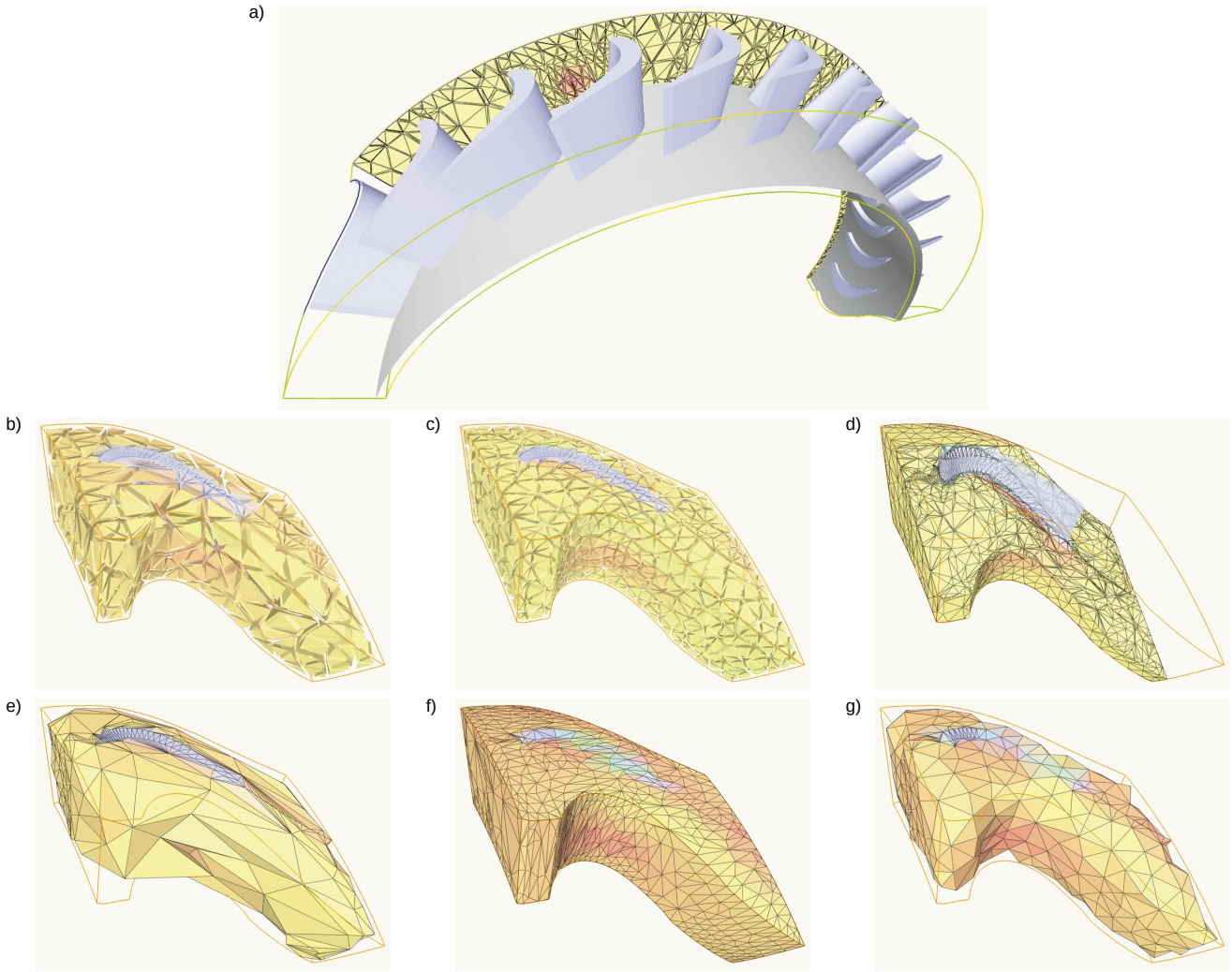


Figure 5: PT representations of an irregular turbine blade mesh. a) Extraction of isosurfaces of the turbine blades using marching tetrahedra. b)-c) Part of the data set at different reconstruction levels with shrinked tetrahedra. d) Part of the mesh is cut to render its internal features. e) PT with ΔE_{grad}. f) ΔE_{vol}. g) ΔE_{equi} only. Parameter settings and performance statistics are listed in Table 2. (See also CP 3. Data set courtesy of AVS Inc.)

tion surface modeling." In *Course Notes for SIGGRAPH '97*. ACM SIGGRAPH, 1997.

[7] H. Hoppe. "Progressive meshes." In H. Rushmeier, editor, *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 99–108, Aug. 1996.

[8] H. Hoppe. "View-dependent refinement of progressive meshes." In *Proceedings of SIGGRAPH '97*, pages 189–198, 1997.

[9] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Mesh optimization." In *Proceedings of SIGGRAPH '93*, pages 19–26, Aug. 1993.

[10] J. Popovic and H. Hoppe. "Progressive simplicial complexes." In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 217–224, Aug. 1997.

[11] F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer, New York, 1985.

[12] R. Ronfard and J. Rossignac. "Full-range approximation of triangulated polyhedra." In *Proceedings of EUROGRAPHICS '96*, pages C67–C76, 1996.

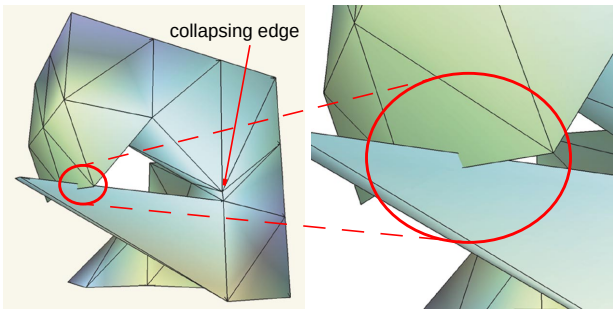
[13] W. Schroeder. "A topology modifying progressive decimation algorithm." In *Proceedings of IEEE Visualization '97*, pages 205–212, 1997.

[14] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. "Decimation of triangle meshes." In *Proceedings of SIGGRAPH '92*, pages 65–70. ACM SIGGRAPH, 1992.

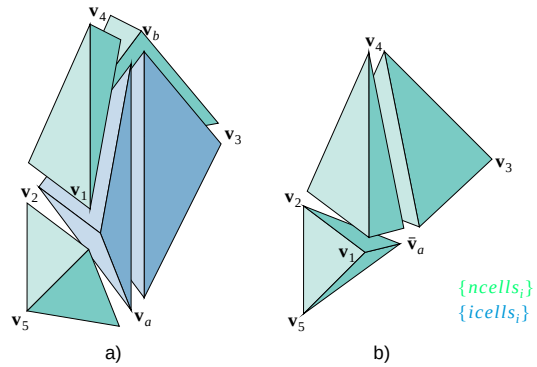
[15] O. G. Staadt, M. Gross, and R. Weber. "Multiresolution compression and reconstruction." In *Proceedings of IEEE Visualization 1997*, pages 337–346. IEEE, 1997.

[16] J. C. Xia, J. El-Sana, and A. Varshney. "Adaptive real-time level-of-detail-based rendering for polygonal models." *IEEE Transactions on Visualization and Computer Graphics*, 3(2):171–183, 1997.

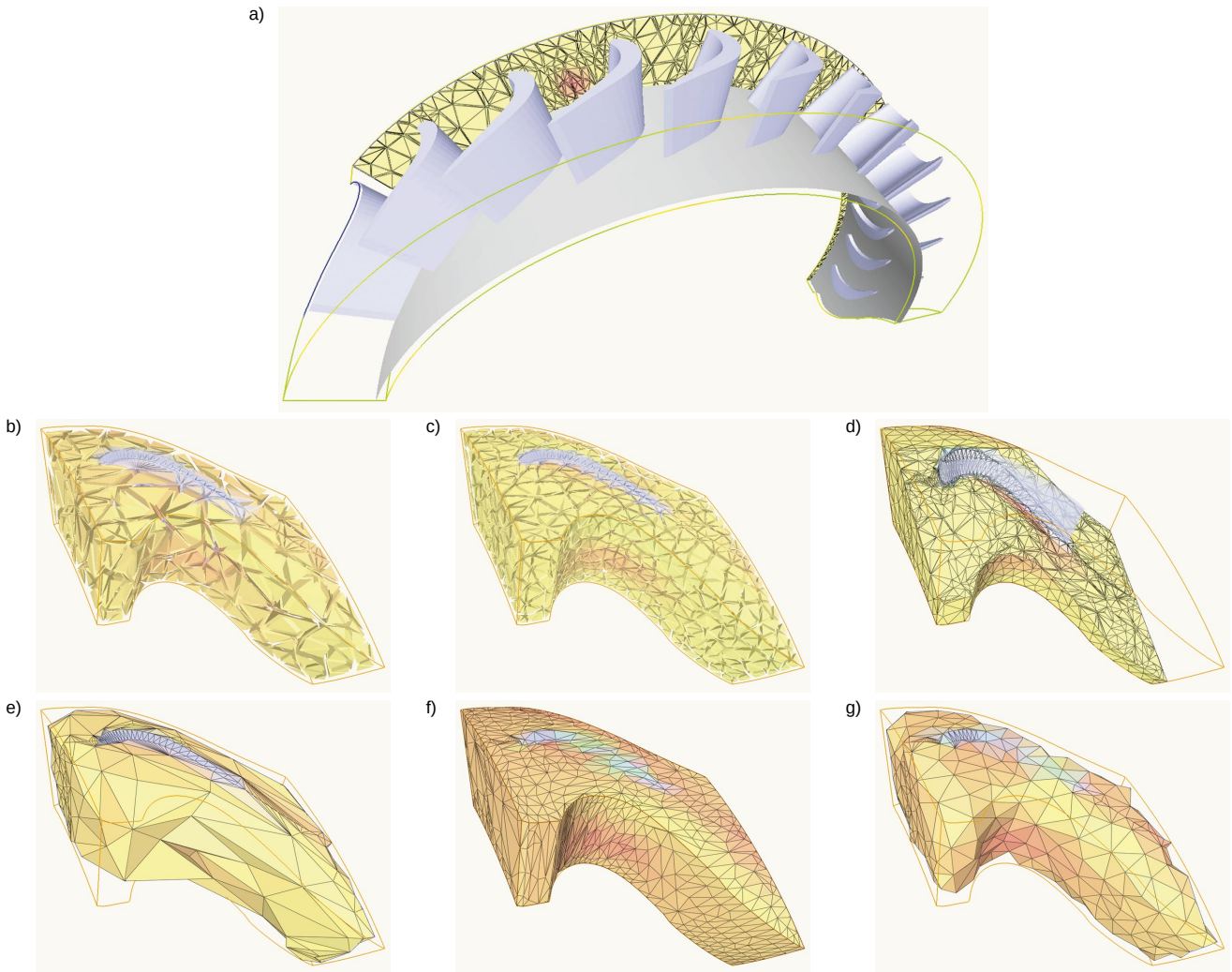
Appeared in *Proceedings of IEEE Visualization '98*, pp. 397-402, 1998.



CP 1: Intersection of two tetrahedra: a) The edges of two non-adjacent tetrahedra bounding the volume intersect while collapsing an edge in a locally concave mesh region. b) Close-up of the intersection.



CP 2: Edge collapse in a tetrahedral mesh: a) Mesh before collapsing edge $(\mathbf{v}_a, \mathbf{v}_b)$. b) Configuration after collapse with resulting vertex $\bar{\mathbf{v}}_a$. (The tetrahedra are shrinked to emphasize the underlying 3-dimensional structure.)



CP 3: PT representations of an irregular turbine blade mesh. a) Extraction of isosurfaces of the turbine blades using marching tetrahedra. b)-c) Part of the data set at different reconstruction levels with shrinked tetrahedra. d) Part of the mesh is cut to render its internal features. e) PT with ΔE_{grad}. f) ΔE_{vol}. g) ΔE_{equi} only. Parameter settings and performance statistics are listed in Table 2. (Data set courtesy of AVS Inc.)