A Short Survey of Mesh Simplification Algorithms

Jerry O. Talton III
University of Illinois at Urbana-Champaign

1. INTRODUCTION

The problem of approximating a given input mesh with a less complex but geometrically faithful representation is well-established in computer graphics. Given the visual complexity required to create realistic-looking scenes, simplification efforts can be essential to efficient rendering. Level-of-detail representations figure prominently in real-time applications such as virtual reality, terrain modeling, and scientific visualization, and as a result there is significant demand for effective algorithms for mesh simplification.

Numerous such algorithms have been proposed, and generically they can be said to vary widely in approach, efficiency, quality, and generality. Some techniques offer efficient processing but produce simplified meshes which are visually undesirable. Others create more pleasing approximations but are slow and difficult to implement. Some algorithms go to great lengths to preserve the topology of the input mesh while others alter it arbitrarily. Many methods are restricted to or only perform well on manifold surfaces. In practice, which algorithm is best suited to perform a given simplification depends heavily on the characteristics of the input mesh and the desired attributes of the approximation.

In this paper, I give a basic overview of the components of some of the most common mesh simplification algorithms and evaluate their strengths and deficiencies. Readers seeking a more detailed survey of the subject are referred to [Garland 1999a] and those looking for a comprehensive treatment might begin with [Luebke et al. 2002].

2. DEFINITIONS

Before we can discuss simplification algorithms we must first define some basic terms. We define a *mesh* exactly as in [Hoppe et al. 1993] to be a pair (V, K) where $V = \left\{\mathbf{v}_i \in \mathbb{R}^3 \mid i \in \{1, \ldots, m\}\right\}$ is a set of vertex positions and K is a simplicial complex representing the connectivity of the mesh. K consists of a set of vertices $\{1, \ldots, m\}$ together with a set of non-empty subsets of the vertices called the simplices of K. Every set consisting of exactly one vertex is a simplex in K, and every non-empty subset of a simplex in K is again a simplex in K. The 0-simplices $\bar{V} = \{i\} \in K$ are called the vertices of the mesh, the 1-simplices $\bar{E} = \{i, j\} \in K$ are called edges, and the 2-simplices $\bar{F} = \{i, j, k\} \in K$ are called faces. We call these vertices, edges, and faces the *elements* of the mesh.

To facilitate later discussion, we also define the *cofaces* of a simplex $s \in K$ by $C(s) = \{s' \in K \mid s \subseteq s'\}$. By this definition, the set of edges and faces incident on

Email: talton@uiuc.edu Address: 201 N. Goodwin Ave., Urbana IL, 61801

Course Notes for CS 598 MJG, October 2004

a vertex form the cofaces of that vertex, and the set of faces adjacent to an edge form the cofaces of that edge.

This definition is particularly nice because it provides an elegant formulation that separates the topological structure of a mesh from its geometric properties while still allowing meshes to be arbitrarily ill-formed. Besides mandating that a mesh be conforming, the only other restriction it imposes is that meshes must be composed of triangular faces, which causes no loss of generality since every planar polygon can be triangulated. This generality is necessary since the meshes created by laser scanning and those designed by artists (the two most common forms of mesh generation) are often severely non-manifold. It is important to keep this fact firmly in mind as we examine simplification strategies and techniques.

One important aspect we have overlooked in our definition is that meshes typically have associated *attributes*. It is not at all uncommon, for example, for each vertex in a given mesh to have an associated normal vector, color value, and set of texture coordinates. We will largely ignore this problem and concentrate instead on geometric simplification. It is worth noting, however, that many simplification algorithms can be logically extended to handle mesh attributes in addition to geometric and topological properties (see [Garland and Heckbert 1998], [Garland and Zhou 2005]) and many others deal with such attributes explicitly (see [Hoppe 1996], [Hoppe 1998], [Cohen et al. 1998]).

3. LOCAL SIMPLIFICATION STRATEGIES

Simplification strategies may be broadly grouped into two categories: local strategies that iteratively simplify the mesh by the repeated application of some local operator, and global strategies that are applied to the input mesh as a whole. Local strategies are by far the most common and so we will concern ourselves primarily with them.

Local simplification strategies are generally greedy. Typically, they define some mesh operation S that, when applied to a mesh M, acts on a small collection of its elements and produces a new mesh $\bar{M}$ with fewer elements. By repeated application of S, a mesh may be simplified arbitrarily. In order to determine the mesh elements to which S should be applied on a given iteration, S may be associated with an *error function* (or *cost function*) that measures the amount of error the operation will introduce into the approximation. By computing the error associated with every possible application of S at a particular iteration, the algorithm can apply the one with minimal cost. This type of heuristic is quite reasonable for simplification problems, and in practice these methods work well.

One nice property of local iterative algorithms is that they allow the user to specify the desired attributes of the target approximating mesh with a high degree of precision. For example, the user may allow the algorithm to run until the mesh contains k faces, or until the error at a given vertex exceeds some threshold ϵ. Global strategies, in contrast, are less amenable to this type of specificity.

3.1 Vertex Decimation

Vertex decimation, first proposed in [Schroeder et al. 1992], operates on a single vertex by deleting that vertex and re-tessellating the resulting hole. Typically some classification scheme based on the adjacency information of the selected vertex is

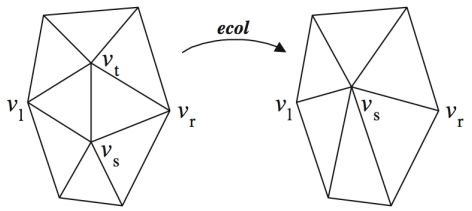

Fig. 1. The edge contraction operation [Hoppe 1993].

used to determine the manner in which this re-tessellation proceeds. Because such classifications are fundamentally topological in nature, this re-tessellation procedure need not alter the topology of the input mesh. In addition, vertex decimation algorithms do not require their input to be manifold, although they are generally incapable of simplifying around non-manifold vertices.

The error associated with a particular decimation typically depends on the classification of the vertex being decimated. For the general case of a manifold vertex $\{i\}$ not near a boundary, [Schroeder et al. 1992] consider the set of planes formed by the triangles in $N(i)$ and compute a single approximating plane based on the area-weighted average of the triangle normals $\mathbf{n}_k$, centers $\mathbf{x}_k$, and areas A_k:

$$\mathbf{n} = \frac{\sum_k A_k \mathbf{n}_k}{\sum_k A_k} \qquad \hat{\mathbf{n}} = \frac{\mathbf{n}}{|\mathbf{n}|} \qquad \mathbf{x} = \frac{\sum_k A_k \mathbf{x}_k}{\sum_k A_k}$$

They then calculate the distance from vertex $\{i\}$ to this plane by $|\hat{\mathbf{n}} \cdot (\mathbf{v}_i - \mathbf{x})|$ and define this quantity to be the cost associated with the decimation. This metric ensures that vertices in smooth regions will be decimated before vertices that define sharp features.

In practice, simplification schemes based on vertex decimation excel at eliminating extraneous geometry, such as is typically found in meshes generated by the marching cubes algorithm described in [Lorensen and Cline 1987], where large planar regions may be subdivided into many redundant triangles. [Schroeder 1997] implemented a robust vertex decimation algorithm in the Visualization ToolKit, which is freely available from http://public.kitware.com/VTK/.

3.2 Edge Contraction

Edge contraction, originally proposed in [Hoppe et al. 1993], is the most common simplification operation. An edge contraction operates on a single edge $\{i, j\}$ and contracts that edge to a single vertex $\{h\}$, updating all edges previously incident on $\{i\}$ and $\{j\}$ to reference $\{h\}$. On a manifold mesh without boundary, each edge contraction will collapse precisely two faces (see Fig. 1). Edge contractions can alter the topology of a mesh, since repeatedly contracting all the edges around a hole will eventually close it. In addition, edge contractions may in principle be applied indiscriminately to edges containing non-manifold vertices.

For a given contraction $\{i, j\} \rightarrow \{h\}$, it is not immediately clear what value should be assigned to $\mathbf{v}_h$, i.e. where the resulting vertex should be placed. Obvious choices such as $\mathbf{v}_h = \mathbf{v}_i$, $\mathbf{v}_h = \mathbf{v}_j$, or $\mathbf{v}_h = (\mathbf{v}_i + \mathbf{v}_j)/2$ are convenient, but can

Fig. 2. A sequence of approximations generated by [Garland and Heckbert 1997].

easily be shown to be non-optimal. Rather than placing $\mathbf{v}_h$ arbitrarily, it is sensible instead to consider the error function associated with the contraction operation and attempt to minimize its value over the space of possible vertex placements. Once the optimal target position has been computed for every possible contraction, the contraction with the smallest associated error can be selected and applied.

Obviously this minimization process will be completely dependent on the choice of error function, so many such functions have been proposed in the literature. [Hoppe 1996] described a complex metric involving four separate terms, the first measuring the distance of $\mathbf{v}_h$ to the original mesh, the second penalizing contractions that fail to preserve the mesh's sharp features, the third accounting for the accuracy of the mesh's scalar attributes, and the last a spring term intended to regularize the minimization problem. This metric gives very nice results, but computing the optimal target position for a given contraction is a non-linear problem and is in practice very inefficient (not to mention difficult to code). In addition, the formulation of this error function essentially restricts its application to manifold meshes, even though edge contractions themselves impose no such restrictions.

[Ronfard and Rossignac 1996] proposed a simpler measure of error. Given a contraction $\{i, j\} \rightarrow \{h\}$ they define the local geometric error to be the maximum squared distance between $\mathbf{v}_h$ and the planes defined by the triangles in $C(i) \bigcup C(j)$. Their algorithm initially sets $\mathbf{v}_h = \mathbf{v}_i$, and then a relaxation step occurs to reposition the vertex more optimally. To avoid error propagation, each new vertex inherits the plane equations from the cofaces of the two merged vertices when a contraction is performed. This strategy penalizes movement away from the surface of the mesh and automatically preserves a mesh's sharp features.

[Garland and Heckbert 1997] observed that, given a single plane $(\mathbf{n}, d)$, one can express the squared distance from the plane to a point $\mathbf{x}$ by:

$$Error(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} + 2\mathbf{b}^T\mathbf{x} + c \tag{1}$$

where the collection $(A, \mathbf{b}, c) = (\mathbf{n}\mathbf{n}^T, d\mathbf{n}, d^2)$ is the *fundamental quadric* of the plane $(\mathbf{n}, d)$. Furthermore, they showed that quadrics are additive: given a set of planes $\{(\mathbf{n}_k, d_k)\}$ evaluating (1) with $(A, \mathbf{b}, c) = (\sum \mathbf{n}_k\mathbf{n}_k^T, \sum d_k\mathbf{n}_k, \sum d_k^2)$ yields the sum of the squared distances between $\mathbf{x}$ and each of the planes. Since (1) is a quadratic function, it is well know that its minimum will occur at $\mathbf{v}_h = -A^{-1}\mathbf{b}$ with associated minimum value $-\mathbf{b}^T A^{-1}\mathbf{b} + c$.

These properties allow for an extremely efficient iterative edge contraction algorithm that works on non-manifold geometry, produces high-fidelity approximations, and requires progressively less memory with each iteration. When a contraction $\{i, j\} \rightarrow \{h\}$ is performed, the quadric associated with the new vertex $\{h\}$ is simply the sum of the quadrics associated with $\{i\}$ and $\{j\}$. [Heckbert and Garland

1999] showed that, under the L_2 metric, this strategy produces provably optimal triangulations in the limit as the number of triangles goes to infinity and their area goes to zero. [Garland 1999b] presented a more complete description of the algorithm, and [Garland and Zhou 2005] described a generalized version for higher dimensions. In practice, this method of simplification is probably the best tradeoff between computational efficiency, geometric fidelity, and topological generality.

3.3 Appearance-Preserving Simplification

A completely different approach to determining the error associated with a given simplification operation was described in [Cohen et al. 1998]. Their metric, which is appearance-based, measures the amount of deviation caused by the operation in the screen-space (pixelized) representation of the mesh. Their algorithm functions by decoupling surface position from color and curvature information and storing the latter two quantities in texture and normal maps. A traditional geometric simplification algorithm can then be employed to filter the surface position, while a hardware-based approach is used to filter the color and normal information, resulting in simplified representations that are nearly visually indistinguishable from the original. However, the algorithm requires a parameterization of the mesh in order to function, which, if not already present in the form of texture coordinates, must be computed. [Lindstrom and Turk 2000] described a more general image-based simplification strategy that does not require specialized hardware or software algorithms.

4. GLOBAL SIMPLIFICATION STRATEGIES

As we previously mentioned, global strategies are far from prevalent in the simplification literature. Nonetheless, they are worth a brief examination.

4.1 Vertex Clustering

The method of vertex clustering was originally proposed by [Rossignac and Borrel 1993] to handle meshes of arbitrary topological structure. In their algorithm, each vertex in the input mesh is assigned a weight based on its perceptual importance: vertices adjacent to triangles with large faces and those in areas of high curvature are weighted more heavily than vertices in smooth regions adjacent to smaller triangles. Next, a bounding box is placed around the mesh and subdivided into a three-dimensional grid. Finally, all the vertices in a given grid cell are clustered to the position of the vertex with maximum weight. Degenerate geometry may then be removed, although the authors proposed a novel visualization of certain degenerate faces and edges in an effort to enhance the recognizability of their simplified meshes. The algorithm is extremely efficient and simple to implement, and the level of simplification may be controlled (although with some difficulty) by choosing the resolution of the overlaid grid. However, vertex clustering can drastically alter the topology of the input mesh in an unpredictable manner, and in practice does not produce very faithful geometric approximations at low levels of detail.

4.2 Shape Approximation

[Cohen-Steiner et al. 2004] cast mesh simplification as a global optimization problem. They employ a variational partitioning scheme to segment the input mesh into

Fig. 3. Variational shape approximation from [Cohen-Steiner et al. 2004].

a set of non-overlapping connected regions, and then fit a locally-approximating plane (or *shape proxy*) to each one. The vertices on the original mesh that coincide with the intersection of three or more shape proxies are retained and re-triangulated. An iterative edge contraction process is subsequently applied to eliminate excess geometry and produce the resulting simplified mesh. Although the process is not particularly efficient, it produces approximations that are slightly more accurate than those in [Garland and Heckbert 1998] as measured by the authors' proposed $L_{2,1}$ metric.

5. RELATED TOPICS

While not simplification strategies in and of themselves, there are a two related topics that bear mentioning.

5.1 Progressive Meshes

As we have already seen, the simplification strategy proposed in [Hoppe 1996] is less practical than more recent methods. Nonetheless, another contribution of the paper is still extremely significant. A *progressive mesh* stores an arbitrary mesh $\bar{M}$ as a much coarser mesh M^0 together with a series of n records indicating how to incrementally refine M^0 back into the original mesh $M^n = \bar{M}$. Each of these n records stores a *vertex split* which is the inverse of an edge contraction. These splits and contractions may be used to smoothly transform between any of the mesh's representations $M^i \longleftrightarrow M^j$ either by performing each split (or contraction) incrementally or by simultaneously moving many vertices. Progressive meshes are implemented in Microsoft's DirectX API and are extremely useful for view-dependent level-of-detail applications [Hoppe 1997].

5.2 Simplification Envelopes

[Cohen et al. 1996] described *simplification envelopes* as a generalization of offset surfaces intended to assist other simplification algorithms. Simplification envelopes allow the generation of approximations that are guaranteed not to deviate from the original surface by more than some ϵ while preserving global topology. An outer envelope is formed by displacing each vertex along its normal by some δ and an inner envelope is formed by displacing by $-\delta$. The value of δ is adaptively updated in areas of high curvature to ensure that the envelopes do not self-intersect. Once calculated, a simplification algorithm can guarantee a tight bound on the fidelity of its representations by ensuring that no simplification operation causes a vertex or

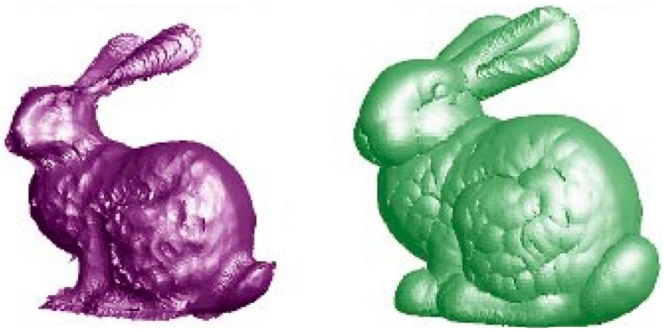

Fig. 4. Inner and outer simplification envelopes from [Cohen et al. 1996].

edge to move outside the region defined by the envelopes. In practice simplification envelopes can be quite useful, but their construction requires the input mesh to be an orientable manifold and their careful preservation of topology and avoidance of self-intersection limits their ability to assist in drastic simplifications. [Zelinka and Garland 2002] described a more general method called *permission grids* that provides tight error bounds on arbitrary triangulated meshes while allowing topological changes during simplification.

REFERENCES

Cohen, J., Olano, M., and Manocha, D. 1998. Appearance-perserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM Press, 115–122.

Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F., and Wright, W. 1996. Simplification envelopes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM Press, 119–128.

Cohen-Steiner, D., Alliez, P., and Desbrun, M. 2004. Variational shape approximation. *ACM Trans. Graph. 23,* 3, 905–914.

Garland, M. 1999a. Multiresolution modeling: Survey and future opportunities.

Garland, M. 1999b. Quadric-based polygonal surface simplification. Ph.D. thesis, Carnegie Mellon University.

Garland, M. and Heckbert, P. S. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 209–216.

Garland, M. and Heckbert, P. S. 1998. Simplifying surfaces with color and texture using quadric error metrics. 263–270.

Garland, M. and Zhou, Y. 2005. Quadric based surface simplification in any dimension. *ACM Trans. Graph. 24,* 2.

Heckbert, P. and Garland, M. 1999. Optimal triangulation and quadric-based surface simplification. *Journal of Computational Geometry: Theory and Applications 14,* 1-3 (November), 49–65.

Hoppe, H. 1996. Progressive meshes. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. *Computer Graphics*.

Hoppe, H. 1997. View-dependent refinement of progressive meshes. *Computer Graphics 31,* Annual Conference Series, 189–198.

Hoppe, H. 1998. Efficient implementation of progressive meshes. *Computers and Graphics 22,* 1, 27–36.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. 1993. Mesh optimization. *Computer Graphics 27,* Annual Conference Series, 19–26.

Lindstrom, P. and Turk, G. 2000. Image-driven simplification. *ACM Trans. Graph. 19,* 3, 204–241.

Lorensen, W. E. and Cline, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. ACM Press, 163–169.

Luebke, D., Watson, B., Cohen, J. D., Reddy, M., and Varshney, A. 2002. *Level of Detail for 3D Graphics*. Elsevier Science Inc.

Ronfard, R. and Rossignac, J. 1996. Full-range approximations of triangulated polyhedra. In *Proceedings of Eurographics*. Vol. 15. C–67.

Rossignac, J. and Borrel, P. 1993. Multi-resolution 3d approximations for rendering complex scenes. In *Geometric Modeling in Computer Graphics*, B. Falcidieno and T. Kunii, Eds. Springer Verlag, Genova, Italy, 455–465.

Schroeder, W. J. 1997. A topology modifying progressive decimation algorithm. In *IEEE Visualization*. 205–212.

Schroeder, W. J., Zarge, J. A., and Lorensen, W. E. 1992. Decimation of triangle meshes. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. ACM Press, 65–70.

Zelinka, S. and Garland, M. 2002. Permission grids: Practical, error-bounded simplification. *Transactions on Graphcs 21,* 2 (April).