Simplification of Arbitrary Polyhedral Meshes

Shaun D. Ramsey
University of Utah
Salt Lake City, UT, USA
email: ramsey@cs.utah.edu

Martin Bertram
University of Kaiserslautern
Kaiserslautern, Germany
email: bertram@informatik.uni-kl.de

Charles Hansen
University of Utah
Salt Lake City, UT, USA
email: hansen@cs.utah.edu

ABSTRACT: Surface models containing billions of polygons are becoming more frequent in computer graphics. Mesh simplification is necessary for displaying such surfaces at interactive rates. We describe a novel method for simplifying polyhedral meshes while producing multiple levels of detail for progressive transmission and interactive exploration. Unlike previous work on mesh simplification, our method is not restricted to triangle meshes. We propose a highly efficient edge-collapsing algorithm for meshes composed of non-planar multi-sided polygons based on a simple edge-selection strategy.

KEY WORDS: mesh simplification, progressive meshes, multiresolution, level-of-detail.

1 Introduction

Highly detailed geometric models are frequently used in computer graphics to convey realism. Surfaces are mostly represented by polyhedral meshes required for efficient rendering. Meshes reconstructed from precise CAD models or from laser-range scanned objects often contain billions of polygons, making it difficult to display, modify, and transmit such surfaces interactively. It is desirable to simplify models allowing for progressive transmission and real-time rendering. In progressive transmission, a portion of the model is transmitted and displayed, while more information about the mesh is transferred and inserted into the existing model.

Level-of-detail (LOD) techniques [1, 2, 3] are employed to create multiresolution mesh representations for interactive rendering. The amount of detail displayed can be locally adapted to the size of objects on the screen (view-dependent rendering) and to the performance of graphics hardware. Rendering times can thus be reduced with little or no loss in visual accuracy. Geomorphs [4] are used to avoid popping effects when the resolution is changed.

A technique that has been shown to complete these tasks is an edge collapse [5] described by these steps:

1. Given a mesh, choose an edge for collapse.
2. Choose a vertex for the collapse.
3. Replace the edge with the chosen vertex.

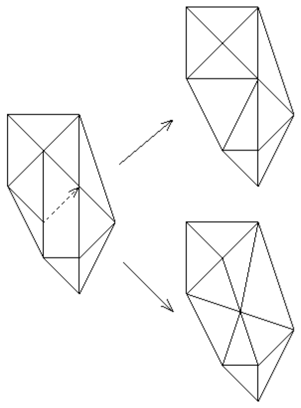

Figure 1. Collapsing an edge (dashed line on the left). Half-edge and full-edge collapses are illustrated to the upper and lower right, respectively.

Triangles incident to the collapsed edge are eliminated. Multi-sided polygons incident to this edge shrink (by removing one edge and one vertex). In the second step, we distinguish between half-edge and full-edge collapses. In a half-edge collapse, the vertex is chosen from the two vertices on the edge. In a full-edge collapse, a new vertex is created, generally as the average of the two vertices, see figure 1, affecting more polygons.

Several simplification techniques have been presented, but none of them deal with arbitrary polyhedral meshes. Before simplifying a polyhedral mesh, each polygon needs to be triangulated, increasing the number of polygons. The task is, however, simplifying the mesh. In the present work, we propose a highly efficient mesh simplification algorithm for meshes containing multi-sided polygons. We demonstrate that a simple edge selection criterion based on normals of adjacent polygons provides visually pleasing results while minimizing computation time. Our algorithm can be used for view-dependent rendering, progressive transmission, and multiresolution editing.

In the next section we review related work. In sections 3 and 4, we present the edge-selection and edge-collapse strategies of our adaptive coarsening algorithm. The inverse operation, splitting vertices for adaptive refinement, is described in section 5. In section 6, we present numerical examples and conclude our work in section 7.

2 Related Work

Turk [6] replaces the vertices on a surface with new points with a curvature-based density. In [7, 8], vertices are removed from the surfaces and the resulting holes are re-triangulated. Rossignac and Borrel [9] place meshes on a grid and cluster groups of vertices that fall into the same region. Lindstrom [10] shows that simple error-estimation produces comparable results to expensive algorithms based on quadratic error metrics.

Multiresolution representations are often used for geometry compression. Lee [11] et al. develop a triangle-quad mesh codec but do not consider arbitrary polygons. In Isenburg's polygon coding method [12], positions are quantized to a coarse level of precision reducing storage time. Kobbelt [13] et al. construct a wavelet-like multiresolution analysis considering local operations on irregular triangle meshes.

Lounsbery [14] and Eck [15] describe a wavelet approach for simplifying meshes. Since their approach uses meshes with regular subdivision connectivity, the represented surfaces need to be re-triangulated. Edge collapses are widely used [4, 5, 16, 17, 18, 19] due to their smooth transitions and quality of simplification. As such, we have chosen to extend this form of simplification to arbitrary meshes.

Kobbelt [20] used simplification to produce a re-parameterization of surfaces. A coarse base mesh is subdivided regularly and the points of the regular mesh are projected back onto the coarse surface. Similar approaches have been used in MAPS [21] and Bertram's work on isosurfaces [22]. Our simplification method can be used for re-parameterization of surfaces, as well.

Meshes can be constructed by a variety of techniques, like scanning real-life models, sampling free-form surfaces, and extracting isosurfaces. Surface models are thus defined in a variety of ways, mostly using meshes composed of triangles and multi-sided polygons [4, 23, 24, 25]. Meshes containing pentagons, hexagons and quadrilaterals are common in many modeling packages [12]. Our algorithm is capable of processing the most general type of meshes.

3 Edge Selection

The first step in the simplification process is to choose an edges to be collapsed. Our selection algorithm is based on the deviation of the normals in the resulting mesh when an edge is removed. Consider an edge for removal, create a new mesh with this edge collapsed and then compare normals in the new and old mesh:

$$cos\ \alpha \ = \ \mathbf{n_1} \cdot \mathbf{n_2}. \tag{1}$$

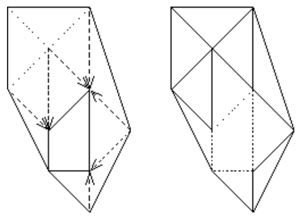

Figure 2. A multi-sided polygon is preserved (left) or shrunk into a triangle (right) by collapsing one of the dotted edges. Arrows denote half-edge collapses.

If the dot product between the normals of the corresponding polygons in the two meshes is above a certain threshold, $cos\ \alpha \geq \varepsilon$, then allow the collapse. Using a threshold close to one, say $\varepsilon = 0.99$, will eliminate all polygons that are nearly co-planar.

Polygon normals are simply computed by averaging the normals of incident vertices. In the case that vertex normals are not available, we estimate the polygon normal by averaging the cross products of each pair of adjacent edges of the polygon:

$$\frac{1}{n}\sum_{i=1}^{n}(p_{i-1} - p_i) \times (p_{i+1} - p_i) \tag{2}$$

using indices modulo n where n is the number of edges. For triangles, the normal is computed from a single cross product. In the case of quadrilaterals, a cross product of the diagonals provides a sufficient estimate.

Finding the best choice for a single edge collapse in the entire mesh would be very inefficient. Hence, we sweep through the mesh and select all *independent* edges, that can be collapsed such that the normal deviation remains below the prescribed threshold ε. Two edges are considered to be *independent*, if they do not belong to a common polygon. A set of independent edges can easily be determined by marking the incident polygons of a selected edge and by processing only edges of polygons that have not been marked. After each sweep, the selected edges are collapsed and the entire process can be repeated until a desired resolution is obtained.

To obtain a finer granularity of levels (or to obtain a larger number of levels), we randomly select a set of independent edges that are eligible for collapse. The maximal number of collapsed edges for each level can be defined by the user. To obtain uniform levels of resolution, a certain percentage of all edges is selected for collapse in every sweep. A greater number of levels allows to adapt more smoothly to a required resolution.

When simplifying a mesh, the user decides whether multi-sided polygons should be preserved or simplified as soon as possible. When this choice has been made, the algorithm selects edges accordingly. This way, the coarser

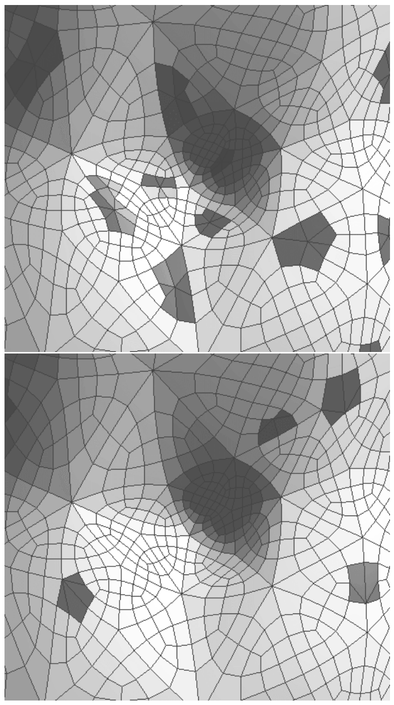

Figure 3. Comparison of full-edge collapses (top) with half-edge collapses (bottom) in the venus mesh.

meshes are either composed of multi-sided polygons, or converted into pure triangle meshes after a few passes, see figure 2.

Although an edge may pass the normal test, it may still be ineligible for removal in our algorithm if this would modify surface topology. For example, we cannot collapse an edge belonging to a tetrahedron. An edge collapse can only be performed when the stencil of vertices surrounding this edge is non-degenerate (i.e., all these vertices are different). Special rules apply to boundary edges.

4 Edge Collapse

An edge selected for collapse can be removed in two ways, choosing either a full-edge collapse or a half-edge collapse, see figure 1. These two edge collapse techniques differ only in how they choose the vertex. Half-edge collapses should be used when original vertices should remain on the simplified mesh. They are preferably used since they affect a smaller portion of the mesh than full-edge collapses. The latter minimizes the deviation of the new vertex from the endpoints. Lower deviation reduces popping artifacts while transitioning between two levels of resolution.

In both cases, multi-sided polygons may be affected, depending on the type of edge selection technique used. In triangle meshes, collapses cause two triangles to be deleted as shown in figure 1. In our technique an edge that is part

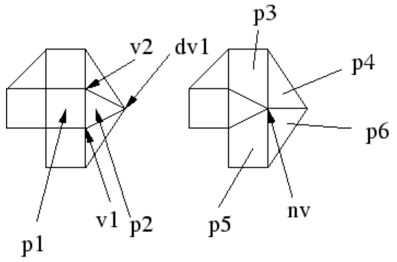

Figure 4. Information used for reconstructing a collapsed edge. v1 and v2 are vectors storing the locations of the endpoints before the collapse. nv is the id of the new vertex. p1 and p2 are the polygons involved in the actual collapse. p3 and p4 are altered polygons which originally pointed to v2. p5 and p6 are altered polygons which originally pointed to v1. Since polygon p2 is deleted in the edge collapse, we also store the position of vertex dv1.

of a multi-sided polygon may be collapsed to produce a polygon with fewer vertices, as demonstrated in figure 1. Figure 3 illustrates an example of these collapses in a real mesh. The algorithm for full-edge and half-edge collapses outlined above removes triangles incident with a collapsed edge and decreases the number of vertices in multi-sided polygons.

5 Vertex Splits

To allow for progressive transmission, vertex splits are used. A vertex split simply inverts the process of an edge collapse. Vertex splits allow for geomorphs that smoothly transition between meshes at different levels of detail. For each vertex emanating from an edge collapse, this edge can be recovered from the information about the mesh stored at the time of edge collapse. The final result will be a coarse mesh combined with the necessary information to recreate the original mesh.

When performing an edge collapse, we record the information necessary to perform the vertex split. This information includes the vertices of the collapsed edge and the vertex identification to which the edge collapsed. It includes references to deleted triangles, and the id of the third vertex. Lastly, the record should include a list of polygons in which there was a vertex change, see figure 4.

If this information is stored in reversed order (last collapsed to first), then the original mesh can be reconstructed from the coarsest mesh by performing vertex splits. In this way, the simplified mesh can be transmitted, displayed and successively updated according to the recorded information.

no. edges	original	10	50	70	final
tri's	9331	10273	11412	10790	8930
quad's	12164	10721	6294	4678	2738
5	27	24	8	5	3
6		1		2	
7				6	
8	4	3			4
9				3	8
10			2	6	9
11		1	5	9	9
12	64	63	57	46	26

Table 1. This table describes the number of polygons (of various sizes) at various stages of simplification. In this case, we used the crocodile mesh due to its interesting face sizes and $\varepsilon = 0.99$.

model	original	simplified	time
dragon	871414	48584	42.900
bunny	69451	6958	4.110
crocodile	21590	11695	3.670
venus	4254	711	0.310

Table 2. This table describes the entire time it takes to simplify a mesh. This includes time to read the mesh from disk and to place it into memory. The original column describes the number of polygons in the original mesh. The simplified column describes how far the mesh was simplified when using a threshold of $\varepsilon = 0.99$. All times are in seconds. In these images, one percent of the original total edges are searched before an edge for collapse is chosen. We note that it is even more efficient to collapse multiple edges in one pass.

6 Results

We described a method to simplify arbitrary meshes using edge collapse techniques. Edge selection depends upon the deviation of polygon normals after an edge collapse. The user may also tune the edge selection process according to a particular mesh or need. An example of extreme mesh simplification using the dragon data set from Stanford is shown in figure 5.

When performing an edge collapse, the cosine of the angle between polygon normals is considered. This polygon normal threshold can be modified to give a desired result. The original algorithm consisted of a threshold of 0 which satisfies the condition that no polygons may flip normals in an edge collapse. By raising this threshold level, it was observed that it is a useful tool to control the simplification process. A value of 0.99 typically gives desirable results. A comparison of different thresholds is provided in figure 6.

In our algorithm, edge selection can be done by viewing a subset of possible edges for edge collapse and choosing the edge with the least deviation according to our heuristics. It is possible to scan the entire mesh for the best edge collapse, but we have discovered that it is only necessary to view many possible edge collapses if the threshold is low. As such, for a high threshold, a collapse can occur at the first valid edge. For low thresholds, searching a higher number of edges yields a better result. Figure 7 suggests that the algorithm produces similar results when the edges to be collapsed are chosen either from a large or from a small set of candidates.

We also ensure that only independent edges (not sharing a polygon) can be collapsed in one pass. This is simply implemented by marking the polygons affected by a collapse and by excluding their edges from any further collapse in the same pass. We found that it was interesting to track the number of different types of polygons in our meshes. The crocodile surface consists of polygons with up to twelve edges. Table 1 contains a table which describes how many of each type of polygon were present in the mesh at different points in the simplification process.

Due to the simple criterion for selecting edges, our mesh simplification method is highly efficient. When searching one percentage of the total mesh for a valid edge, we achieved the times found in table 2. If necessary, our simplification algorithm can be faster by lowering the percent of the total mesh searched for a valid edge.

7 Conclusions and Future Work

We have developed a simplification method processing meshes composed of triangles and multi-sided polygons. We obtain visually pleasing results using a simple edge-selection criterion based on the deviation of surface normals estimated from the mesh. Our level-of-detail algorithm can be used for progressive transmission and interactive exploration of large-scale meshes.

We are planning to use our algorithm in selective refinement and multiresolution editing applications. Similar algorithms based on pure triangle meshes have been devised earlier [4, 26, 27]. Future work will be directed at the preservation of feature lines and the corresponding segmentation and parameterization of surfaces.

Acknowledgments

At this point we would like to acknowledge Stanford 3D Scanning Repository and Avalon for the meshes used in this paper. A special thanks to the Utah Graphics Group for making models available to us.

References

[1] Clark, J. Hierarchical geometric models for visible surface algorithms, In *Communications of the ACM 19*, 1976, pp. 547–554.

[2] Funkhouser, T. A., and Séquin, C. H. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments, In *Proceedings of ACM SIGGRAPH 93*, 1993, pp. 247–254.

[3] Rosenfeld, A., Ed. *Multiresolution Image Processing and Analysis*, Springer, Berlin, 1984.

[4] Hoppe, H. Progressive meshes, In *Proceedings of ACM SIGGRAPH 96*, 1996, pp. 99–108.

[5] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. Mesh optimization, In *Proceedings of ACM SIGGRAPH 93*, 1993, pp. 19–26.

[6] Turk, G. Re-tiling polygonal surfaces, In *Proceedings of ACM SIGGRAPH 92*, 1992, pp. 55–64.

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

[8] Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Jr., F. P. B., and Wright, W. Simplification envelopes, In *Proceedings of ACM SIGGRAPH 96*, 1996, pp. 119–128.

[9] Rossignac, J., and Borrel, P. Multi-resolution 3D approximations for rendering complex scenes, Modeling in Computer Graphics: Methods and Applications, 1993, pp. 455–465.

[10] Peter Lindstrom and Greg Turk. Evaluation of Memoryless Simplification, *IEEE Transactions on Visualization and Computer Graphics 5*, 2 (April - June), 1999, pp. 98–115.

[11] Lee, H., Alliez, P., and Desbrun, M. Angle-Analyzer: A Triangle-Quad Mesh Codec, *Proceedings of Eurographics 02*, 2002, pp. 383–392.

[12] Isenburg, M., and Alliez, P. Compressing Polygon Mesh Geometry with Parallelogram Prediction, In *IEEE Visualization 2002*, 2002, pp. 141–146.

[13] Kobbelt, L., Campagna, S., Vorsatz, J. and Seidel, H. Interactive Multi-Resolution Modeling on Arbitrary Meshes, In *Proceedings of SIGGRAPH 98*, 1998, pp. 105–114.

[14] Lounsbery, M., DeRose, T. D., and Warren, J. Multiresolution analysis for surfaces of arbitrary topological type, *ACM Transactions on Graphics 16*, 1 (January), 1997, pp. 34–73.

[15] Eck, M., DeRose, T. D., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. Multiresolution analysis of arbitrary meshes, In *Proceedings of ACM SIGGRAPH 95*, 1995, pp. 173–182.

[16] Garland, M., and Heckbert, P. S. Surface simplification using quadric error metrics, In *Proceedings of ACM SIGGRAPH 97*, 1997, pp. 209–216.

[17] Hoppe, H. View-dependent refinement of progressive meshes, In *Proceedings of ACM SIGGRAPH 97*, 1997, pp. 189–198.

[18] Popović, J., and Hoppe, H. Progressive simplicial complexes, In *Proceedings of ACM SIGGRAPH 97*, 1997, pp. 217–224.

[19] Ronfard, R., and Rossignac, J. Full-range approximation of triangulated polyhedra, In *Proceedings of Eurographics 96*, 1996, pp. 67–76.

[20] Kobbelt, L., Vorsatz, J., Labsik, U., Seidel, H. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces, In *Proceedings of Eurographics '99*, 1999, pp. 119–129.

[21] Lee, A. Sweldens, W., Schroder, P., Cowsar, L., and Dobkin, D. MAPS: Multiresolution Adaptive Parameterization of Surfaces, In *Proceedings of ACM SIGGRAPH 98*, 1998, pp. 95–104.

[22] Bertram, M., Duchaineau, M., Hamann, B., Joy, Kenneth. Bicubic subdivision-surface wavelets for large-scale isosurface representation and visualization, In *IEEE Visualization 2000*, 2000, pp. 389–396.

[23] Trotts, I. J., Hamann, B., Joy, K. I., and Wiley, D. F. Simplification of tetrahedral meshes, In *IEEE Visualization '98*, 1998, pp. 287–296.

[24] Cignoni, P., Costanza, D., Montani, C., Rocchini, C., and Scopigno, R. Simplification of tetrahedral meshes with accurate error evaluation, In *IEEE Visualization 2000*, 2000, pp. 85–92.

[25] Zhao, P., and Teh, H. C. Rational bicubic simple quadrilateral mesh surfaces, *The Visual Computer 11*, 8, 1995, pp. 401–418.

[26] Zorin, D., Schröder, P., and Sweldens, W. Interactive multiresolution mesh editing, In *Proceedings of ACM SIGGRAPH 97*, 1997, pp. 259–268.

[27] Lee, S. Interactive multiresolution editing of arbitrary meshes, *Computer Graphics Forum 18*, 3 (September), 1999, pp. 73–82.

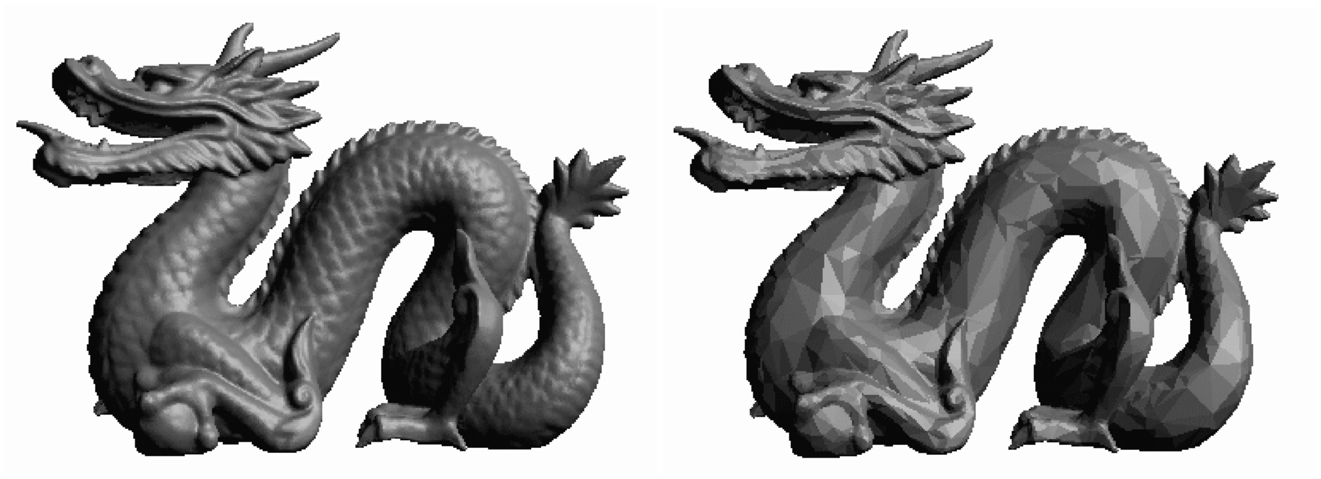

Figure 5. Simplification of the Stanford dragon, composed of 871,414 polygons (left) down to 48,124 polygons (right). All important features at face, tail, and ridges are still visible.

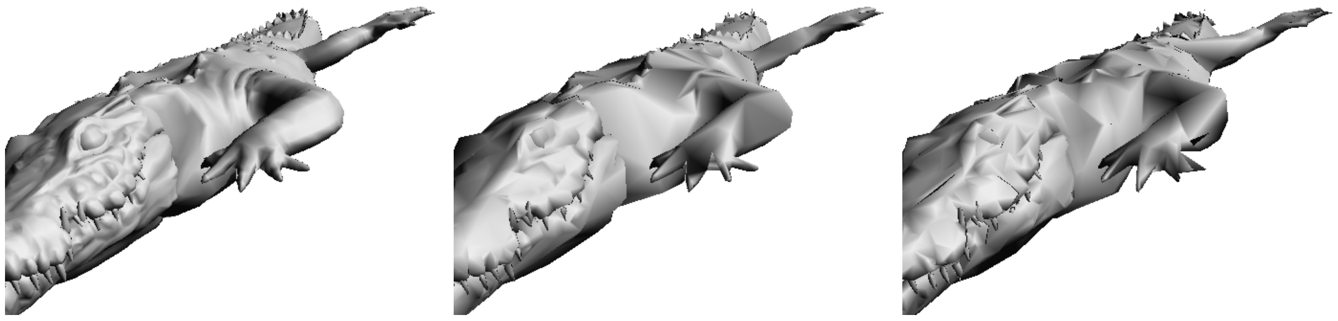

Figure 6. Simplification of the crocodile data set (left, full resolution) down to 5600 polygons using different thresholds $\varepsilon = 0.95$ (middle) and $\varepsilon = 0$ (right). Note that more features are preserved using the greater threshold in the middle image using the same number of triangles as in the right image.

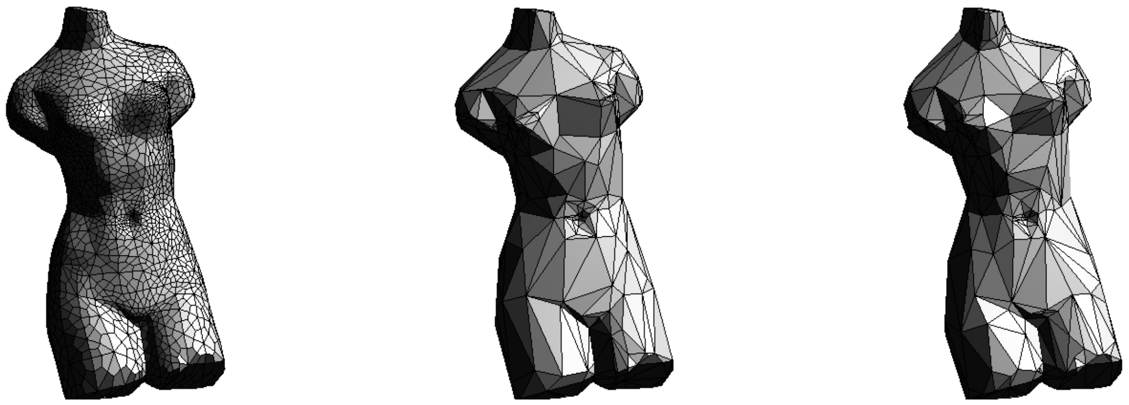

Figure 7. Simplification of the venus data set (left, full resolution) down to 2000 polygons. Each collapsed edge was selected as the best fit from 42 edges (middle) and from 425 edges (right). Both strategies provide similar results.