A General Framework for Mesh Decimation

Leif Kobbelt Swen Campagna Hans-Peter Seidel

Computer Sciences Department, University of Erlangen-Nürnberg
Am Weichselgarten 9, 91058 Erlangen, Germany
kobbelt@informatik.uni-erlangen.de

Abstract

The decimation of highly detailed meshes has emerged as an important issue in many computer graphics related fields. A whole library of different algorithms has been proposed in the literature. By carefully investigating such algorithms, we can derive a generic structure for mesh reduction schemes which is analogous to a class of greedy-algorithms for heuristic optimization. Particular instances of this algorithmic template allow to adapt to specific target applications. We present a new mesh reduction algorithm *which clearly reflects this meta scheme and efficiently generates decimated high quality meshes while observing global error bounds.*

Introduction

In several areas of computer graphics and geometric modeling, the representation of surface geometry by polygonal meshes is a well established standard. However, the complexity of the object models has increased much faster than the through-put of today's graphics hardware. Hence, in order to be able to display and modify geometric objects within reasonable response times, it is necessary to reduce the amount of data by removing redundant information from triangle meshes.

A precise definition of the term *redundancy* in this context obviously depends on the application for which the decimated mesh is to be used. Technically speaking, the most important aspect is the approximation error, i.e., the modified mesh has to stay within a prescribed tolerance to the original data. From an optical point of view, local flatness of the mesh might be a better indicator for redundancy. It is natural that applications as different as rendering and finite element analysis put their emphasis also on the preservation of different aspects in the simplified geometric shape.

In the last years, a host of proposed algorithms for mesh reduction has been applied successfully to level of detail generation [14, 2], progressive transmission [6], and reverse engineering [1]. See [15] for an overview of some relevant literature.

We consider most of the suggested algorithms as generic *templates* leaving the freedom to plug in specific instances of predicates. For example, each algorithm is based on a scalar valued oracle which indicates the degree of redundancy of a particular vertex, edge, or triangle. Depending on the target application, different choices for this oracle are appropriate but this does not affect the algorithmic structure of the scheme.

On the most abstract level, there are two different basic approaches to find a coarser approximation of a given polygonal mesh. The one is to build the new mesh without necessarily inheriting the topology of the original and the other is to obtain the new mesh by (iteratively) modifying the original without changing the topology.

Having a topologically simplified model of the original mesh is useful in applications where the topology itself does not carry crucial information. For example, when rendering remote objects, small holes can be removed without affecting the quality but for a finite element simulation on the same object the holes might be important to obtain reliable results.

In this paper we will analyze *incremental* mesh reduction, i.e., algorithms that reduce the mesh complexity by the iterative application of simple topological operations instead of completely reorganizing the mesh. We will identify the slots where custom tailored predicates or operators can be inserted and will give recommendations when to use which. We then present an original mesh reduction algorithm based on these considerations. The algorithm is fast according to Schroeder's recent definition [17] yet allows global error control with respect to the geometric Hausdorff distance. The scheme is validated in the result section by showing and discussing some examples.

Relevant algorithmic aspects

The topology preserving mesh reduction schemes typically use a simple operation which removes a small submesh and retriangulates the remaining hole. Some schemes use local optimization to find the best retriangulation. To control the decimation process, a scalar valued predicate induces a priority ordering on the set of candidates for being removed. This predicate can be based purely on distance measures between the original and the reduced mesh or it can additionally take local flatness into account.

This macroscopic description matches most of the known

incremental mesh reduction schemes. Due to the overwhelming variety of different algorithms that have been proposed in the literature, there are several authors who attempted to identify important features and classify the different approaches accordingly [16, 15, 3]. We do not want to add another survey but we just give an abridged overview. We will focus on three fundamental ingredients that are necessary (and sufficient) to build your own mesh reduction algorithm. The ingredients are a *topological operator* to modify the mesh locally, a *distance measure* to check whether the maximum tolerance is not violated, and a *fairness criterion* to evaluate the quality of the current mesh.

Topological operators

The classical scheme of [18] removes a single vertex $\mathbf{v}$ and retriangulates its crown. Thus, in every step, a patch of n triangles (the valence of $\mathbf{v}$) is replaced by a new patch with $n-2$ triangles. In general, a local edge-swapping optimization is necessary to guarantee a reasonable quality of the retriangulated patch.

In [6], edges $\overline{\mathbf{p}\,\mathbf{q}}$ are collapsed into a new vertex $\mathbf{r}$ which removes two triangles from the mesh. This operation can also be understood as submesh removal and retriangulation. In this case the local connectivity of the retriangulation is fixed but the optimal location for $\mathbf{r}$ is determined by a local energy minimization heuristic.

We could cut out larger submeshes from the original mesh but this would require a more sophisticated treatment of special cases.

A nice property of the basic vertex-removal and edge-collapse operators is that consistency preservation is easy to guarantee. We just have to check the injectivity of the crown of the vertex $\mathbf{v}$ or the edge $\overline{\mathbf{p}\,\mathbf{q}}$ respectively. The rejection of all operations that would lead to complex vertices or edges is the reason why most incremental schemes do not change the global topology of a mesh.

Our observation when testing different reduction schemes on a variety of meshed models is that the underlying topological operator on which an algorithm is based does not have a significant impact on the results. The quality of the resulting mesh turns out to be much more sensitive to the criteria which decide *where* to apply the next reduction operation. Hence, we recommend to make the topological operator itself as simple as possible, i.e., by eliminating all geometric degrees of freedom.

Concluding from these considerations, we suggest the use of what we call the *half-edge collapse*. A common way to store orientable triangle meshes is the half-edge structure [13] where an undirected edge $\overline{\mathbf{p}\,\mathbf{q}}$ is represented by two directed halves $\mathbf{p} \to \mathbf{q}$ and $\mathbf{q} \to \mathbf{p}$. Collapsing the half-edge $\mathbf{p} \to \mathbf{q}$ means to pull the vertex $\mathbf{q}$ into $\mathbf{p}$ and to remove the triangles that have become singular.

This topological operator's major advantage is that it does not contain any unset degrees of freedom which would have to be determined by local optimization. If we treat the two half-edge mates as separate entities then the only decision is whether a particular collapse is to be performed or not. Moreover, the reduction operation does not "invent" new geometry by letting some heuristic decide about the position of $\mathbf{r}$. The vertices of the decimated mesh are always a proper subset of the original vertices. The half-edge collapse can be understood as a vertex removal without the freedom of chosing the triangulation or as an edge collapse without the freedom of setting the position of the new vertex.

Figure 1 shows the submeshes involved in the basic topological operations.

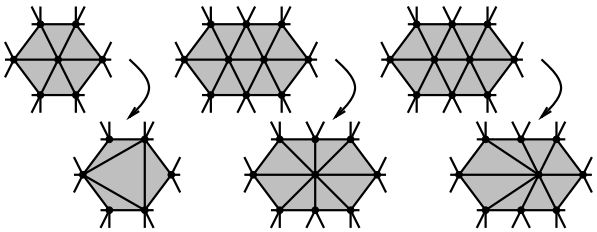

Figure 1: Vertex-removal, Edge-collapse, and Half-Edge-Collapse.

Distance measures

In most technical applications there is a predefined tolerance which bounds the maximum deviation of the approximating mesh from the original one. To make a reduction scheme useful in practical real-world applications it has to guarantee such tolerance globally over the whole surface. All schemes that base the decision whether vertices are redundant or not merely on the local curvature can never satisfy this requirement. In general there is an obvious trade-off between tightness and computational complexity of the geometric error estimation.

A straight forward solution to (over-) estimate the current error during the reduction process is to compute the deviation of the submesh T_i' that replaces the mesh T_i in the ith step of the reduction and to accumulate these contributions locally [17, 3]. In general, this leads to a very coarse but conservative estimate of the true error. Accumulating not only the pure distance but an error quadric bounding the region of allowable deviation, leads to much better results [5].

Several authors try to estimate the true geometric deviation of the two meshes in a more sophisticated manner by computing the two-sided Hausdorff distance between the meshes [9]. One way to cope with the computational complexity and the many special cases, is to scatter sample points on both meshes and compute Euclidian distances between pairs of points [8]. Alternatively, both one-sided Hausdorff distances can be estimated by computing distances between the scattered points on one mesh and the triangles of the other [3].

A very interesting but also rather complex approach (if computed exactly) is the construction of offset-meshes which enclose a *simplification envelope* around the surface [4]. Since the correct computation of the offset-meshes is quite complicated, simple heuristics have to be applied to obtain reasonable and conservative approximations.

In rendering applications, more specialized reduction schemes base the error metric not on intrinsic geometric measures in object space but (dependent on the current view) on visual measures in screen space [7, 12].

Although the two-sided Hausdorff distance fits the intuitive notion of the deviation of one geometric object from the other very well, it is not appropriate for most of the typical input data to mesh reduction algorithms.

The reason for this is that in general only the *vertices* of the given mesh represent actually measured points. This is true for laser-range scanned data and data obtained from mechanical probing. The initial triangulation that recovers the neighborhood relations in the input to our reduction algorithm has typically been generated by a preprocessing algorithm which itself is based on heuristic decisions (and not on specific knowledge about the object). Hence, there is no point in approximating the whole piecewise linear surface but it is enough to approximate the discrete data points themselves. As a consequence, the one-sided Hausdorff distance between the discrete set of data points and the current decimated mesh matches the intended concept of geometric deviation best.

This error measure also corresponds to the standard setting in scattered data approximation. Generally, surfaces (the decimated mesh in our case) are fitted to scattered points in space. There is no reason why connecting the data points and additionally taking into account the continuum of points on this piecewise linear (pre-) reconstruction of the surface should lead to better results.

One positive effect of approximative two-sided Hausdorff distance based error metrics is that it has a stabilizing influence on the intermediate optimization performed during the reduction. This however can also be achieved by using a proper fairness criterion (cf. next section) with the additional advantage that weight parameters are provided which have an intuitively predictable effect on the result.

Fairness criteria

Several algorithms in this field do not clearly distinguish between the *approximation error* that is introduced by one reduction step and the effect on the *fairness quality* of the resulting mesh. Both are usually combined in the predicate by which potential reduction steps are rated. Analyzing the nature of the problem and the incremental approaches to its solution reveals deeper insight.

Formalizing the mesh reduction problem as a scattered data problem, our goal is to construct a surface $\mathcal{S}$ whose maximum distance to the given data points $\mathcal{P}$ does not violate a prescribed tolerance

$$\|\mathcal{S} - \mathcal{P}\|_\infty \;\leq\; \varepsilon. \tag{1}$$

In our setting, the surface $\mathcal{S}$ is a triangle mesh with vertices $V(\mathcal{S})$ being a (minimal) subset of $\mathcal{P}$. Since a globally optimal solution for this problem is very difficult to find, we have to be satisfied with a local minimum.

From this point of view, incremental mesh reduction schemes appear as *greedy* algorithms for the optimization problem. Just like for classical knapsack problems [19], we have an *objective function* which is the number of removed vertices and we have a *capacity function* which reflects how far from exhausting the maximum tolerance the current decimated mesh still is (the remaining "unused" tolerance). In fact, most mesh reduction schemes apply those reduction steps first which cause the minimum waste of capacity (i.e., the minimum increase in approximation error) and this would be an optimal greedy-decision since it provides the most benefit for minimal investment. Notice that every reduction step removes a constant number of vertices (usually one).

However, the mesh reduction problem is more complex than the knapsack problem. The reason for this is that in the iterative algorithm, future reduction steps depend on earlier decisions. Hence, the simple greedy approach cannot lead to a globally optimal solution. This is the point when fairness criteria are introduced to steer the algorithm. We can exploit meta-knowledge about the problem, i.e., the knowledge that the points to be approximated lie on a reasonably smooth surface and that the coarse approximation should also be as smooth as possible. This is why the ordering of the potential reduction steps according to local curvature leads to better results compared to the pure approximation error minimization. In other words, the use of a fairness oracle turns the plain downhill decision of the greedy algorithm into an "educated guess".

From this abstract point of view, we can clearly distinguish between the different functions that control an incremental mesh reduction algorithm: we have the *cost function* measuring the remaining complexity of the mesh, we have a *capacity function* enforcing the prescribed tolerance, and we have a *guidance predicate* rating the potential reduction steps according to the extent to which the reduction would affect the fairness of the decimated mesh.

Algorithmically, we have to implement a binary oracle (yes/no) which checks whether a particular reduction step would violate the maximum tolerance. This oracle merely decides whether a given vertex or edge belongs to the *candidate set* of legal reduction steps. Within the candidate set, we assign priorities according to the fairness predicate. In each greedy-step we pick that reduction step which improves the fairness most or at least decreases it least.

By applying such algorithms to a highly detailed noisy mesh (cf. Figure 2), we can actually see how the mesh is smoothed out first (noise removal increases fairness) and then slowly degenerates (decreasing fairness).

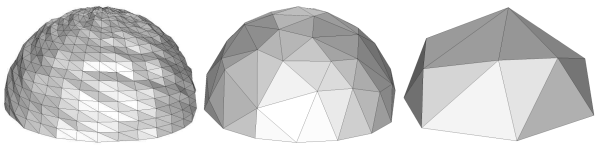

Figure 2: Fairness optimization during iterative mesh reduction.

We now turn our attention to the explicit definition of the fairness functional. This functional obviously depends on the application for which the decimated mesh is to be used. We will try to systematically develop a generic functional anyway which can be adapted to particular objectives by weighting factors.

The topological reduction operators modify the surface locally. Hence, we have to base our fairness functional on local surface properties. From differential geometry we know that the first and second fundamental form characterize the behavior of a surface sufficiently well for most applications. Here, the first fundamental form accounts for the local distortion within a parameterized surface, i.e. the mapping of lengths and angles, while the second fundamental form provides complete information about the local curvatures.

These concepts from differential geometry can be transferred to the discrete setting of triangular meshes [11]. Either local low order polynomial interpolants [20, 10] or geometric analogies lead to semantically equivalent characterizations.

Assigning parameter values (u_i, v_i) to the direct neighbors $\mathbf{p}_i$ of a vertex $\mathbf{p}$ allows us to construct a local least squares fitting quadratic polynomial. The coefficients of this polynomial can be considered as Taylor coefficients of a local expansion. Having derivative information up to the second order is enough to reconstruct the fundamental forms locally. This provides access to approximations of all relevant geometric properties.

Since local interpolation requires the solution of a (small) linear system, we can optimize the performance of the reduction algorithm by using geometrical analogies to estimate the local fairness of first and second order. For example the roundness of the triangles, i.e., the ratio of inner circle radius to the longest edge can be used to estimate the local distortion and the dihedral angles between adjacent triangles to measure the local curvature.

Combining these local surface properties into one functional

$$F(\mathcal{S}) \;:=\; \sum_{\mathbf{p}\,\in\,\mathcal{S}} \alpha\; E(\mathbf{p}) + \beta\; R(\mathbf{p}) + \gamma\; S(\mathbf{p})$$

allows to put more emphasis on either the local approximation error $E(\mathbf{p})$ (function value), the local distortion $R(\mathbf{p})$ (1st order derivatives), or the local curvature $S(\mathbf{p})$ (2nd order derivatives) by simply adjusting the weight coefficients accordingly. For example, if the reduced mesh should preserve as much detail as possible with a prescribed number of triangles then α should be set to a large value (and the tolerance to infinity). A large β leads to reduced meshes which are suitable for finite element analysis since better conditioned triangles are preferred. If γ is the leading coefficient then the meshes will be optimized with respect to outer fairness which is important for rendering and display.

Of course, the right choice for the weight coefficients α, β, and γ strongly depends on the initial mesh since $F(\mathcal{S})$ is not even invariant with respect to scaling. If the ranking of the different fairness aspects should be guaranteed independently of the object to be decimated then *cascading* the fairness functional is more appropriate: to compare two potential reduction operations, we can first compare their impact on the sum of the surrounding dihedral angles $S(\mathbf{p})$. If those are approximately equal then we compare the average roundness of the adjacent triangles $R(\mathbf{p})$ and so on.

The example in Figure 3 clearly shows that the quality measure $S(\mathbf{p})$ (middle) makes the algorithm produce meshes that closely follow the outer geometry but do not have "round" triangles. Increasing the weight for $R(\mathbf{p})$ causes the resulting mesh to become more stable but larger jumps of the normal vector might occur.

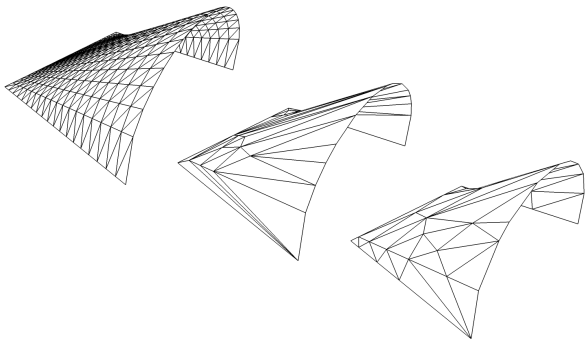

Figure 3: The mesh on the left is decimated by using a fairness functional which punishes large dihedral angles (middle) or strongly distorted triangles (right). Notice how the minimization of discrete curvature causes the triangles to stretch along the lines of minimal curvature.

In addition to these geometric fairness measures there is also the possibility of taking color information into account which might be associated with the triangles or the vertices [6].

A new mesh reduction algorithm

According to our analysis of the relevant components in incremental mesh reduction schemes, we designed our

own algorithm to verify our recommendations. The result is an effective and efficient algorithm which allows fairness control by intuitive parameters.
As the topological operation we use the half-edge collapse. The reason for this decision is that we want to strictly separate the topological operation from the geometrical aspects of fairness and approximation error. Mixing the straight greedy paradigm with local optimizations (in more flexible operators) does neither improve the performance of the algorithm nor does it lead to better results in general. Another reason why we prefer the half-edge collapse is that no new points are generated. This makes progressive transmission of meshes more effective and is crucial for integrated level of detail extraction.
In a preprocessing phase all potential half-edge collapses have to be evaluated and ranked according to the fairness criterion. Only the legal collapses are stored in a priority queue in the order of decreasing fairness. Obviously, after the execution of a collapse operation this queue has to be updated by locally reevaluating the potential collapses.
The estimation of the one-sided Hausdorff distance between the original data points and the decimated mesh is computationally very expensive. In fact, the scheme spends most of the time computing distances between points and triangles. To speed up the process, we exploit the fact that vertices are removed iteratively by simple half-edge collapses.
When the edge $\overline{\mathbf{p}\,\mathbf{q}}$ is collapsed into the vertex $\mathbf{p}$, we have to compute the distance of $\mathbf{q}$ to the resulting mesh. In order to reduce the complexity of this operation we restrict the area where we search for the minimum distance to that submesh which is affected by the edge collapse (cf. Figure 4). This overestimates the true minimum distance in general but in the vast majority of the possible configurations it will give the true minimum distance. After the closest triangle has been identified, we store the vertex in a list associated with this triangle.

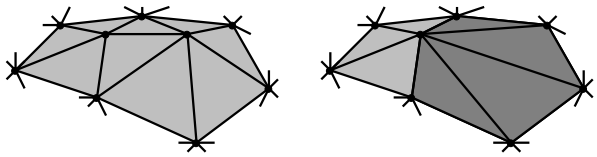

Figure 4: Hausdorff-distances of removed vertices are only computed to the local region which is directly involved in an edge collapse (dark grey).

Now consider an arbitrary reduction step $\overline{\mathbf{p}\,\mathbf{q}} \mapsto \mathbf{p}$ during the iteration. All triangles that are adjacent to $\mathbf{q}$ have a list of associated vertices. In order to recompute the Hausdorff-distance after the collapse, all those vertices have to be redistributed among the remaining triangles. Again, we restrict our search to the modified region of the mesh. The very rare cases in which another triangle happens to lie closer to an original data point are ignored for the sake of simplicity.
Since more and more vertices are removed from the mesh, this redistribution becomes more and more complex. Hence, we have to optimize this step in order to achieve reasonable performance. Since we want to find the minimum distance of a set of points from a *fan* of triangles we can exploit the special configuration.
The midplane between adjacent triangles, i.e. the plane which is spanned by the common edge and the average normal vector, splits the space into two half-spaces. To decide which triangle is closer to a given vertex, we just have to check on which side of this plane the vertex lies.
The whole fan of triangles among which the cloud of vertices has to be distributed defines a pencil of midplanes (cf. Figure 5). The most efficient way to distribute the vertices is to check each against one plane after the other until the right slot is found. Of course this algorithm is not waterproof. Special configurations of mutually intersecting planes might cause wrong assignments but this will only lead to conservative overestimations of the error and hence the scheme will not fail to observe the maximum tolerance. In practical experiments with many different meshes, this simplified distance computation never lead to severe problems. Figure 6 shows the distance vectors computed by our algorithm for a simple example. The scheme fails to be correct only in very extreme configurations and then it overestimates the error, hence the scheme behaves more carefully in dangerous regions.

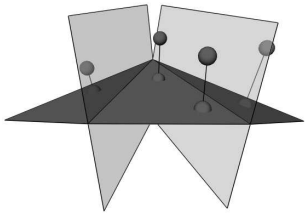

Figure 5: A pencil of planes dividing the space into slots where sample points are assigned to the corresponding triangle.

Special attention has to be paid to boundary vertices where the one-sided Hausdorff-distance fails to be a proper model for the intuitive geometric intent. On the boundaries we have to prevent vertices from sliding into the inner region of the surface. Although this would not violate the maximum tolerance, it ignores the additional knowledge about the local topology. Hence, for boundary vertices we do not compute distances to triangles but take distances to the decimated boundary polygon (cf. Figure 6).
The same argument holds for inner surface vertices which lie on a feature line. These are, in some sense, singular curves on surfaces and hence should be treated as such, i.e. deviations from such lines *within* the surface matter. This is very similar to the treatment of discontinuities in the color attributes.

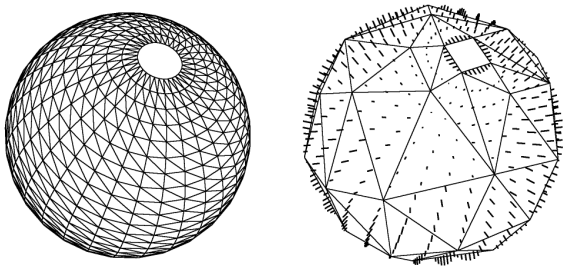

Figure 6: The restricted Hausdorff-distance coincides with the true minimum distance in most cases. In the right picture the distance vectors from the original data points to the triangles of the reduced mesh are shown. Notice the different distance definition for the boundary vertices.

Results

We show several decimated meshes generated by our algorithm. Figure 7 demonstrates the effectiveness of our scheme. The original mesh is reduced to about 0.5 % of the original data. The corresponding computing times are given in Table 1. All times are benchmarked on a SGI, R10000, 195 MHz.

Figure 9 shows a whole sequence of decimated meshes. It demonstrates the effect of the different fairness functionals. Using the dihedral angle criterion (middle row) causes a concentration of the triangles in regions of high curvature where the triangles stretch along the minimum curvature direction. Flat regions are strongly decimated without caring about the occurrence of long thin triangles. Optimizing the aspect ratio of the triangles instead leads to a more equalized distribution of the triangles and avoids extreme configurations (lower row). However, the number of remaining triangles for the same tolerance is larger.

As we expect, the choice of the fairness criterion has some impact on the obtainable degree of reduction for a given tolerance (cf. Table 2). Ordering the potential edge collapses according to the dihedral angle criterion (order 2) typically leads to the least number of triangles. This effect can be interpreted as the "educated guessing" which pushes the downhill algorithm into the right direction. Ordering the reduction candidate set according to the aspect ratio of the triangles (order 1) usually allows even less reduction than the pure ordering by increasing approximation error (order 0).

A casual attempt to explain this behavior is the following: all the objects we used to test our algorithm represented (piecewise) smooth objects. Hence, the affinity of the algorithm to preserve the local flatness of the surfaces by minimizing dihedral angles is a suitable model for the data. The roundness of the triangles, however, is a bad model since the tight approximation of a smooth surface by equilateral (and equally sized) triangles works well only for very special geometries, e.g., for spheres.

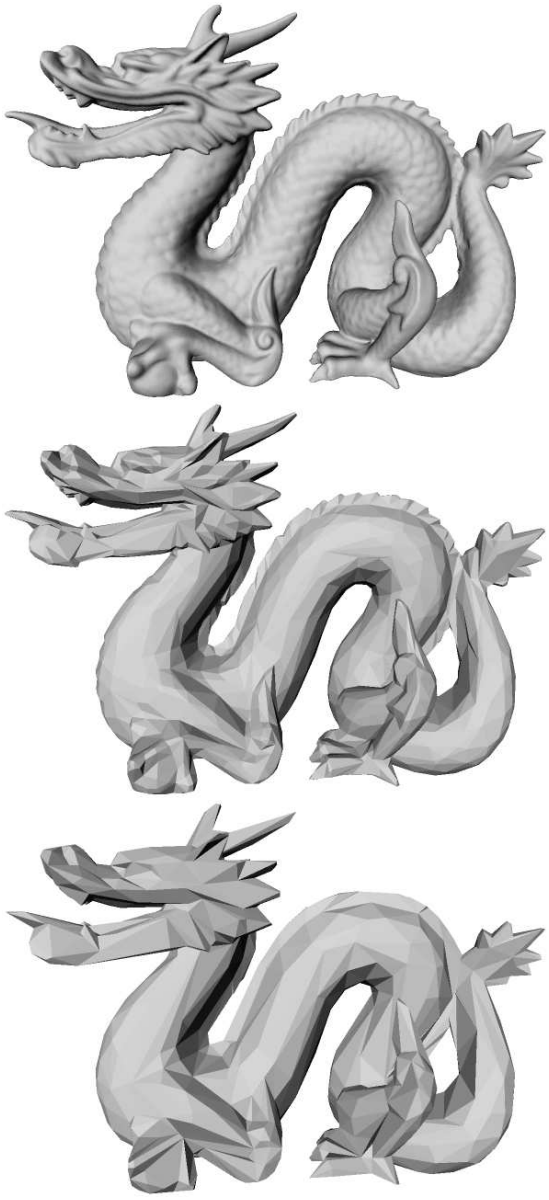

Figure 7: Reduced meshes derived from a rather complex model (top, 871,414 triangles). The absolute approximation tolerances are $5 \cdot 10^{-4}$ (middle, 7,402 triangles) and 10^{-3} (bottom, 4458 triangles). Cf. Table 1 for the bounding box size.

Hence this guidance function is not chosen optimally (cf. Figure 8).

The majority of the computational costs of the algorithm is due to the fact that updating the fairness and the deviation of the current mesh requires a local recalculation of the corresponding values after each edge collapse. Reduction schemes which base the redundancy oracle on conservative local overestimations are much faster but tend to waste a significant amount of tolerance. Cf. the

(order 0)	bounding box	#Δ orig.	#Δ reduced	ε prescribed	ε actual	time (sec.)
dragon	.20×.14×.09	871,414	7,402	.0005	.000499	584.9
dragon	.20×.14×.09	871,414	4,458	.001	.000998	666.4
bunny	.15×.15×.15	69,473	1,182	.001	.000998	35.3
mechpart	81× 50× 26	7,942	438	.5	.49646	2.64

Table 1: CPU times and reduction rates for different mesh models. Notice the tight satisfaction of the prescribed error bounds.

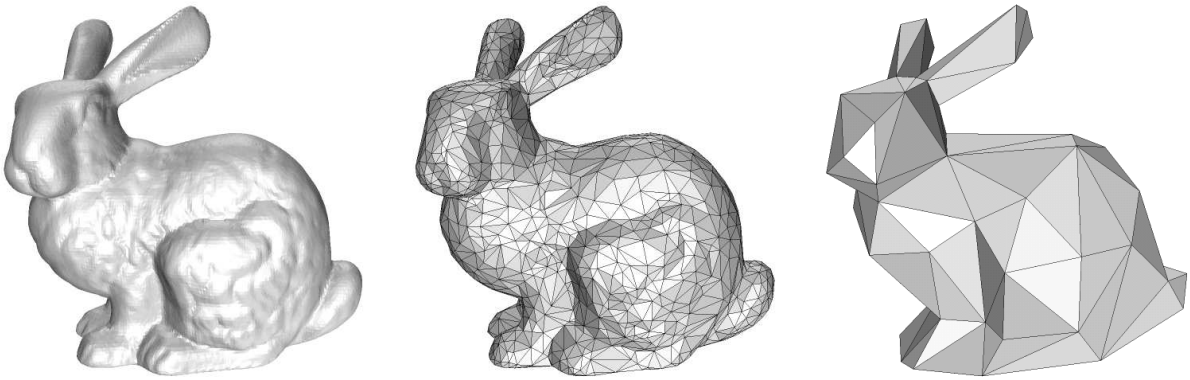

Figure 8: The reference model for mesh reduction algorithms. For moderate reduction (middle) the roundness criterion leads to the best results, for extreme reduction (right) the flatness criterion yields better results since the coarse mesh adapts better to the original geometry.

	#Δ	ε actual	time (sec.)
order 0, $\varepsilon = .05$	2,089	.0498	1.82
order 0, $\varepsilon = .1$	1,500	.0999	2.03
order 0, $\varepsilon = .25$	837	.2492	2.37
order 1, $\varepsilon = .05$	2,555	.0499	1.98
order 1, $\varepsilon = .1$	1,784	.0998	2.26
order 1, $\varepsilon = .25$	872	.2492	2.76
order 2, $\varepsilon = .05$	1,952	.0497	5.2
order 2, $\varepsilon = .1$	1,357	.0991	6.04
order 2, $\varepsilon = .25$	704	.2491	7.51

Table 2: Reduction rates and running times for the mechanical part model (7,942 triangles) with different fairness criteria and error bounds. The computational costs increase with the order of the fairness functional since larger regions have to be updated after each reduction step.

	#Δ	ε actual	time (sec.)
order 0, $\varepsilon = .0001$	14,340	.000099	19.8
order 0, $\varepsilon = .0005$	2,697	.000498	30.9
order 0, $\varepsilon = .001$	1,182	.000998	35.3
order 1, $\varepsilon = .0001$	16,328	.000099	20.2
order 1, $\varepsilon = .0005$	3,227	.000499	33.2
order 1, $\varepsilon = .001$	1,505	.000999	38.4
order 2, $\varepsilon = .0001$	12,843	.000099	43.7
order 2, $\varepsilon = .0005$	2,359	.000499	86.4
order 2, $\varepsilon = .001$	1,019	.000999	103.2

Table 3: Reduction rates and running times for the Stanford bunny model (69,473 triangles) with different fairness criteria and error bounds.

tables 1, 2, and 3 where prescribed error tolerances and actually occurring maximum errors are given.

Conclusions

We presented a detailed analysis of a generic incremental mesh reduction framework putting this specialized problem into the more general context of greedy optimization. We identified the relevant components and justified our recommendations for the choice of such predicates and operators. A constructive proof for our claim is given by the new reduction algorithm which we derived accordingly.

The scheme in our current implementation generates very good meshes according to the setting of the weight coefficients in our fairness functional. Separating the tolerance measure from the fairness measure makes the impact of the coefficients on the result predictable and accessible to the practical user. Although the scheme computes the global Hausdorff distance between the data points and the decimated mesh it still achieves a performance which meets the Schroeder-bound [17] of 10^8 removed triangles per day.

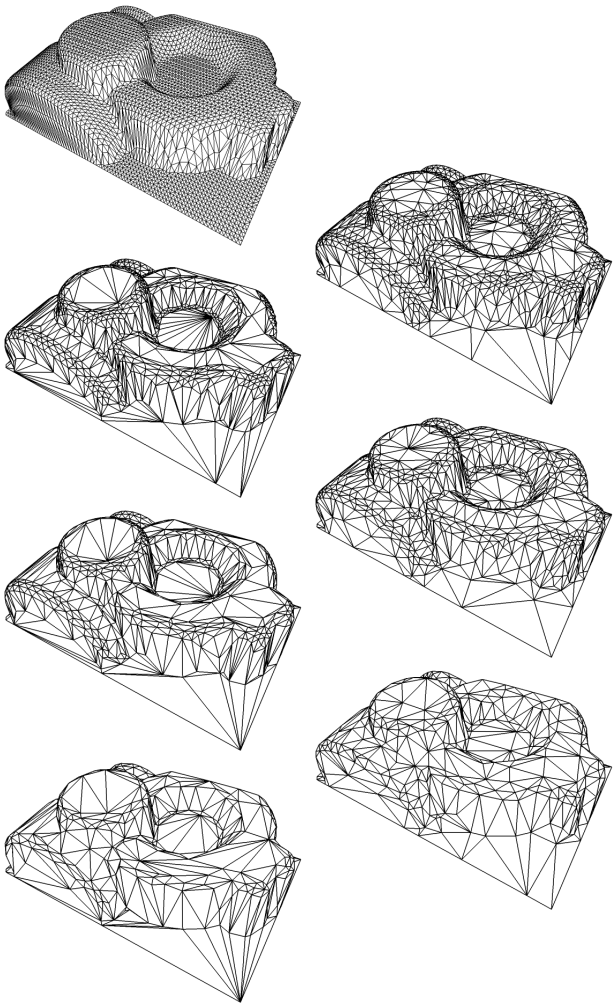

Figure 9: Several meshes obtained from the original model (upper left) by applying our new algorithm with different absolute error tolerances (top to bottom: $\varepsilon = 0.05, 0.1, 0.25$) and fairness criteria (left: flatness, right: roundness). Cf. Table 2.

Carefully checking where the computation time is spent during the reduction algorithm reveals that measuring the Hausdorff-distance is still the most costly part. To evaluate the fairness criterion does not seem to be a critical factor. Hence, we could think of more sophisticated fairness functionals without much impact on the performance.
Since, due to the use of half-edge collapses, the set of vertices in the reduced mesh is always a proper subset of the original data points, progressive transmission of the mesh is very effective. In fact, the incremental reduction can be understood as the conversion of the original mesh into a *progressive mesh* representation and hence the proposed algorithm also serves as a device to generate a complete set of level of detail approximations to the original mesh.

References

[1] Chandrajit Bajaj, Fausto Bernardini, J. Chen, and D. Schikore. Triangulation-based 3d reconstruction methods. In *13th ACM Symposium on Computational Geometry*, 1997.

[2] Rikk Carey and Gavin Bell. *The Annotated VRML 2.0 Reference Manual*. Addison-Wesley, 1997.

[3] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution Decimation Based on Global Error. *The Visual Computer*, 13:228–246, 1997.

[4] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Grek Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. Simplification Envelopes. In *SIGGRAPH '96*, pages 119–128.

[5] Michael Garland and Paul S. Heckbert. Surface Simplification Using Quadric Error Metrics. In *SIGGRAPH '97*, pages 209–218.

[6] Hugues Hoppe. Progressive Meshes. In *SIGGRAPH '96*, pages 99–108.

[7] Hugues Hoppe. View-Dependent Refinement of Progressive Meshes. In *SIGGRAPH '97*, pages 189–198.

[8] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh Optimization. In *SIGGRAPH '93*, pages 19–26, 1993.

[9] Reinhard Klein, Gunther Liebich, and W. Straßer. Mesh Reduction with Error Control. In *IEEE Visualization '96, Conference Proceedings*, pages 311–318.

[10] Leif Kobbelt. Variational Design with Parametric Meshes of Arbitrary Topology. submitted for publication, 1997.

[11] Leif Kobbelt. Discrete Fairing. In *Proc. of the 7th IMA Conf. on the Mathematics of Surfaces*, 1997.

[12] David Luebke and Carl Erikson. View-Dependent Simplification of Arbitrary Polygonal Environments. In *SIGGRAPH '97*, pages 199–208.

[13] Martti Maentylae. *An Introduction to Solid Modeling*. Computer Sciene Press, 1988.

[14] Rémi Ronfard and Jarek Rossignac. Full-Range Approximation of Triangulated Polyhedra. In *Proceedings of Eurographics '96*, pages C67–C76.

[15] Jarek Rossignac. Simplification and Compression of 3D Scenes, 1997. Tutorial Eurographics '97.

[16] Will Schroeder. Polygon Reduction Techniques. IEEE Visualization '95.

[17] William J. Schroeder. A Topology Modifying Progressive Decimation Algorithm. In *IEEE Proceedings Visualization '97*, pages 205–212.

[18] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of Triangle Meshes. In *SIGGRAPH '92*, pages 65–70.

[19] Robert Sedgewick. *Algorithms*. Addison-Wesley, 1983.

[20] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In *SIGGRAPH '94*, pages 247–256.