Mesh Reduction with Error Control

Reinhard Klein, Gunther Liebich, W. Straßer

Wilhelm-Schickard-Institut, GRIS, Universität Tübingen, Germany

Abstract

In many cases the surfaces of geometric models consist of a large number of triangles. Several algorithms were developed to reduce the number of triangles required to approximate such objects. Algorithms that measure the deviation between the approximated object and the original object are only available for special cases. In this paper we use the Hausdorff distance between the original and the simplified mesh as a geometrically meaningful error value which can be applied to arbitrary triangle meshes. We present a new algorithm to reduce the number of triangles of a mesh without exceeding a user-defined Hausdorff distance between the original and simplified mesh. As this distance is parameterization-independent, its use as error measure is superior to the use of the L^∞-Norm between parameterized surfaces. Furthermore the Hausdorff distance is always less than the distance induced by the L^∞-Norm. This results in higher reduction rates. Excellent results were achieved by the new decimation algorithm for triangle meshes that has been used in different application areas such as volume rendering, terrain modeling and the approximations of parameterized surfaces. The key advantages of the new algorithm are:

- It guarantees a user defined position dependent approximation error.

- It allows to generate a hierarchical geometric representation in a canonical way.

- It automatically preserves sharp edges.

CR Descriptors: I.3.3[**Computer Graphics**]: Picture/Image Generation, Display algorithms; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling *Curve, surface, and object representations, Object hierarchies* ; **Additional Key Words and Phrases:** hierarchical approximation, model simplification, levels-of-detail generation, shape approximation

1 Introduction and previous work

Triangle meshes are one of the most popular representations of surfaces for computer graphics applications. On the one hand, rendering of triangles is widely supported by hardware and, therefore,

---

[1] Universität Tübingen, Auf der Morgenstelle 10 / C9, 72076 Tübingen, Germany.
E-mail: reinhard@gris.uni-tuebingen.de
http:\\www.gris.uni-tuebingen.de

fast. On the other hand, there is an increasing set of data acquisition techniques which generate triangle meshes as output. Examples are Marching Cubes algorithms, acquisition of digital elevation data, 3D reconstruction from images, or the use of 3D-scanners. However, most of these techniques generate much more triangles than neccessary to represent the given object with a small approximation error. For example typical medical computer tomography or magnetic resonance scanners produce over 100 slices at a resolution of 256 by 256 or up to 1024 by 1024 pixels each. For a sampled data set of the human head at a resolution of 512 by 512 pixels, isosurface extraction of the bones using the *Marching Cubes* algorithm produces a mesh with about 1M triangles. A digital map of Germany with a resolution of 40 meters in North-to-South and West-to-East direction results in about 500M points. Such huge amounts of data lead to problems with data storage and postprocessing programs. Animation and realtime rendering of such datasets is almost impossible even on high performance graphics hardware.

Many techniques were published that aimed at reducing surface complexity. These techniques simplify triangular meshes either by merging elements or by resampling vertices using different error criteria to measure the fitness of the approximated surfaces.

Following, we briefly mention some general and valuable solutions. For some surveys see also [CRS96] and [HG95].

- Coplanar facets merging: coplanar or nearly coplanar data are searched, merged in larger polygons and then retriangulated into fewer simple facets, [HH92, MSS94].

- Mesh decimation: the algorithm uses multiple passes removing on each pass, all vertices that satisfy a distance and a feature angle criterion. The resulting holes are retriangulated [SZL92].

- Mesh optimization: an energy function is evaluated over the mesh and is minimized either by removing/moving vertices or collapsing/swapping edges [HDD+93].

- Point coalescence: this technique subdivides the ambient space into smaller subspaces. Multiple vertices in a subregion are merged into a single vertex using a weighted approximation such as their centeroid. The merged vertices are then reconnected with their neighbours to form a collection of faces [RB93].

- Re-tiling: a new set of vertices is inserted at random on the original surface mesh, and then moved on the surface to be displaced on maximum curvature locations; the original vertices are then iteratively removed. A retiled mesh, built on the new vertices only, is returned [Tur92].

- Multiresolution retiling: the approach uses remeshing, resampling and wavelet parametrization to build a multiresolution representation of the surface from which any approximated representation can be extracted [EDD+95].

The general drawback of all these algorithms is the lack of a common way of measuring the error between the original and the simplified mesh. This has also been pointed out by Cignoni and Rocchini

[CRS96]. Therefore, a general comparison between the above approaches is not easy.

In many cases the relation between the parameters (like the ones used in the algorithms above) and the result of the mesh simplification process is a-priori not obvious for a user. For example, some approaches [SZL92, Tur92] allow the user to define the maximum error that can be introduced in a single simplification step. In this way errors can accumulate and there is no measure for the actual global error.

How further parameters like "feature angles" [SZL92], roughness criteria [Sb94], or decimation rates [SZL92, Tur92] effect the simplification process and the simplified mesh is also not obvious. As the specification of the parameters is difficult, the user in many cases has to run the reduction algorithm several times with different parameters to get a good result.

The mesh simplification process is much easier to control by measuring the distance between the original and simplified mesh. Such distance measures have already been used for mesh simplification in the area of terrain modeling [DFP85, HG95] and the approximation of parameterized surfaces [Kle94, Ke95]. For height fields the distance between the original and the simplified mesh can be measured either as vertical distance from a plane or as distance from the closest point to the polygon. For parameterized surfaces the L^∞-norm is a possible measure.

In addition, the measurement of the distance between original and simplified mesh is neccessary for mesh simplification algorithms that must ensure a certain geometric accuracy between the original and the simplified mesh. An example from medicine is the reconstruction of organs from CT data that need to be replaced by a prosthesis.

The basis of our mesh simplification algorithm is the use of the Hausdorff distance as an appropriate error measure between original and simplified mesh. In contrast to other algorithms like [SZL92] this distance is measured between the reduced and the orignal mesh and *not* between the reduced mesh and some average surface. We shortly define and discuss this distance in the next section.

2   The error metric

The Euclidean distance between a point x and a set $Y \subset \mathbb{R}^n$ is defined by
$$d(x, Y) = \inf_{y \in Y} d(x, y),$$
where $d(.,.)$ is the Euclidean distance between two points in $\mathbb{R}^n$. Using this definition we can define the distance $d_E(X, Y)$ *from* a set X *to* a set Y by
$$d_E(X, Y) = \sup_{x \in X} d(x, Y). \qquad (1)$$

We call this distance one-sided Hausdorff distance between the set X and the set Y. It doesn't define a distance function on the set of all sets of $\mathbb{R}^n$, because it is not symmetric. That means that in general $d_E(X, Y) \neq d_E(Y, X)$.

If, for example, the one-sided Hausdorff distance $d_E(T, S)$ from the simplified mesh S to the original mesh T is less than a predefined error tolerance ϵ, then

$$\forall x \in T \text{ there is a } y \in S \text{ with } d(x, y) < \epsilon.$$

For mesh simplification this condition would be sufficient in many cases. But in some cases the unsymmetry of this distance leads to problems. This can either happen near to the borders of the original mesh or at parts of the mesh that resemble to the border, in the sense that the angle between adjacent triangles along a common edge is very small. An example is the concave blade of a sickle, see Figure 1.
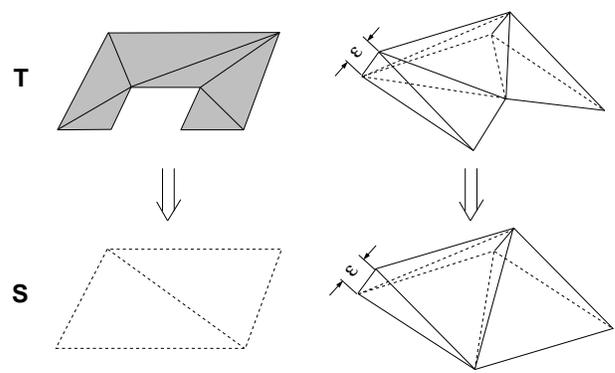

Figure 1: Left: The original nonconvex triangle mesh T is contained within a plane. The simplified mesh S consists of a triangulation of the convex hull of the original mesh and therefore the original mesh is contained in the simplified mesh. Right: The same situation in 3D. $\forall x \in T$ there is a $y \in S$ with $d(x, y) < \epsilon$ but we do not have $\forall y \in S$ there is a $x \in T$ with $d(x, y) < \epsilon$.

To handle such cases we use the Hausdorff distance. It is defined by
$$d_H(X, Y) = \max(d_E(X, Y), d_E(Y, X)). \qquad (2)$$

In contrast to the one-sided Hausdorff distance it is symmetric and we have
$$d_H(X, Y) = 0 \Longleftrightarrow X = Y.$$

If the Hausdorff distance between the original triangulation T and the simplified triangulation S is less than a predefined error tolerance ϵ, then

$$\forall x \in T \text{ there is a } y \in S \text{ with } d(x, y) < \epsilon$$

and

$$\forall y \in S \text{ there is a } x \in T \text{ with } d(x, y) < \epsilon.$$

Therefore, the Hausdorff distance between the original and simplified triangulation is the one a user would intuitively think of.

It is worthwhile to mention that for *any* parameterized surface $S : \mathbb{R}^2 \supset \Omega :\longrightarrow \mathbb{R}^3$ that is approximated by a piecewise linear surface $T :: \mathbb{R}^2 \supset \Omega :\longrightarrow \mathbb{R}^3$ we always have

$$d_H(S, T) \leq ||S - T||_\infty = \sup_{u \in \Omega} ||S(u) - T(u)||.$$

For this reason, using the Hausdorff distance for error measurement results in higher reduction rates using the same error tolerance.

3   The algorithm

The algorithm is a typical mesh simplification algorithm, that is it starts with the original triangulation T and successively simplifies it: It removes vertices and retriangulates the resulting holes until no further vertices can be removed from the simplified triangulation S without exceeding a predefined Hausdorff distance between the original triangulation and the simplified one.

The main problem of the algorithm is how to compute the Hausdorff distance

$$d_H(S, T) = \max(d_E(S, T), d_E(T, S)) \qquad (3)$$

between the original and simplified mesh. While, in general, this is a very complicated task, it can be easily solved in the case of an
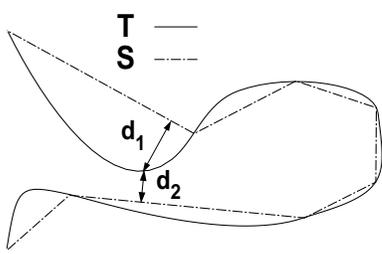
Figure 2: Although the distance d_2 is less than distance d_1, d_1 is measured due to the topological correspondance.

Figure 3: If the vertex v is removed only the potential errors of neighbour vertices have to be updated.

iterative simplification procedure. The idea is to keep track of the actual Hausdorff distance between the original and simplified mesh and of the correspondence between these two meshes from step to step. This correspondance is the clue to compute the Hausdorff distance between the two meshes. It allows for every $x \in S$ in the new triangulation to find the triangles of the original triangulation that contain the $y \in T$ nearest to x, that is $d(x, y) \leq d(x, z), \forall z \in U \subset T$ and vice versa. $U \subset T$ needs to be carefully chosen in order to avoid measuring the distance to topologically not neighbouring parts of T, see Figure 2. Note that keeping track of the correspondance and the Hausdorff distance $d_H(S, T)$ is a local operation because only a small part of the simplified triangulation changes in each step. Points of the simplified triangulation that may change the Hausdorff distance d_H must belong to the modified area of the simplified triangulation. The calculation of the new distance is restricted to that area. After each step we know the Hausdorff distance between the original and simplified mesh. Based on this information a multiresolution representation of the model can be built.

A further idea of the new algorithm is to compute and update an error value for every single vertex of the simplified mesh. This value describes the *potential error*, that is the Hausdorff distance that would occur if a certain vertex was removed. In each step we actually eliminate one of the vertices whose removal causes the smallest potential error. At the beginning of the algorithm the original and simplified triangulation coincide. For every single vertex the potential error is computed and all vertices are stored into a list L in ascending order according to their potential errors. If a vertex is actually removed from the current simplified triangulation this list is updated. Because of the ordering of the list, the vertex that should be removed next is placed at its beginning. There are two cases where the removal of a vertex would not make sense: First so-called complex vertices, see [SZL92], and second vertices for which the retriangulation of the resulting hole may lead to topological problems. These situations are detected by topological consistency checks, see [Tur92]. In both cases the potential error is set to infinity.

This strategy of implicit sorting preserves sharp edges in the original triangulation, see Figure 10.

3.1 Description of the algorithm

In the algorithm we first concentrate on the one-sided Hausdorff distance $d_E(T, S)$ from the original to the simplified mesh. After the realization of this one-sided distance, it is relatively easy to calculate the full Hausdorff distance, if neccessary.

3.1.1 The main loop

After building the list L, the triangulation is simplified through an iterative removement of vertices, one at a time: At each iteration, the vertex on top of the list L is removed from the list and from the actual triangulation, provided that its corresponding potential error is
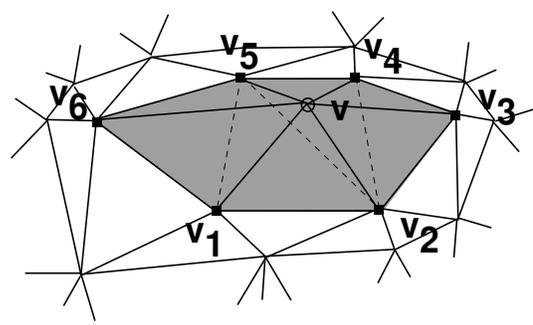
smaller than the predefined maximum Hausdorff distance ϵ. Otherwise it is not possible to remove an additional vertex while keeping the distance between simplified and original triangulation smaller than ϵ and we are finished.

If we remove a vertex v from the triangulation its adjacent triangles are removed and the remaining hole is retriangulated. For this purpose the adjacent vertices are projected into a plane similar to the algorithm of Turk [Tur92]. If the corresponding polygon in the plane does not self-intersect, the polygon is triangulated using a constrained Delaunay triangulation. The use of the Delaunay triangulation is not essential but we found that it produces better reduction results than an arbitrary triangulation of the polygon. In addition, for all neighbouring vertices $v_1, \cdots, v_n, n \in \mathbb{N}$ of v the potential errors need to be updated, see Figure 3. The vertices have to be removed from the list L and reinserted into L according to their new potential error. Note that this can be done in $O(\log r)$ time, where r is the number of remaining vertices in the reduced mesh.

3.1.2 Calculation of the potential error

One of the crucial parts in the algorithm is the computation of the potential error of a vertex, because in this step not only the distances between vertices have to be computed but the distance d_E between all points of the two triangulations. To simplify things we use

$$d_E(T, S) = \max_{t \in T} d(t, S)$$

instead of d_E. If none of the neighbour vertices of a single vertex has already been removed from the original triangulation, it is clear how to calculate the potential error of that vertex: Let $t_i, i = 1..n$ be the set of removed triangles and $s_j, j = 1..m$ the set of new triangles produced during the retriangulation and v the vertex. Note that $m = n - 2$ in the general case and $m = n - 1$ if the removed vertex was a border vertex. To calculate $d_E(T, S)$ it is sufficient to calculate

$$d_E(\{t_i\}_{i=1..n}, \{s_j\}_{j=1..m}),$$

see Figure 4.

Yet after some simplification steps there are triangles $t_k, k \in K$ in the original mesh with vertices that do no longer belong to the simplified mesh, see Figure 5. The straightforward way to calculate the maximum distance

$$d_E(t_k, S) = \sup_{x \in t_k} d(x, S)$$

for these triangles is not realizable to the whole simplified triangulation S. To solve this problem we store for each already removed vertex v of the original triangulation T the triangle $s \in S$ that has the smallest distance to v. Vice versa we store for each triangle $s \in S$ all vertices that reference s as the triangle with smallest distance,
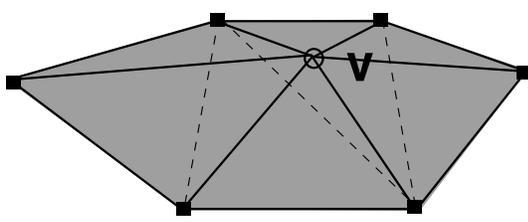
Figure 4: None of the neighbour vertices of a vertex are removed. In this case it is clear which triangles of the old and new triangulation have to be considered to calculate the potential error that would arise if the vertex was removed.
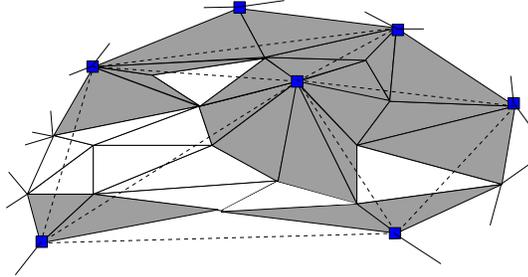

Figure 5: For the white triangles of the original triangulation (solid lines) it is a priori not clear to which triangles of the simplified triangulation (dashed) distances have to be computed.

see Figure 6. This information is updated in each iteration step and suffices to calculate $d_E(T, S)$. Let $s_{il} \in S_l$, $i = 1..n$ be the set of removed triangles and $s_{jl+1} \in S_{l+1}$, $j = 1..m$ the set of new triangles produced during the step from triangulation S_l to S_{l+1} and $\{v_k \in V\}$ the set of vertices of the original triangulation that are already removed. Furthermore each v_k must be nearest to one of the removed triangles $s_{il} \in S_l$. For all triangles of the original triangulation T incident to one of the vertices v_k the distance to S_{l+1} is calculated. It is sufficient to calculate the distances between triangles of the original triangulation and a subset $\tilde{S} \subset S_{l+1}$, where $\tilde{S}$ contains the newly created triangles of S_{l+1} and the triangles of S_{l+1} sharing at least one point with the newly created ones. This is justified by

$$d_E(t, S) \leq d_E(t, \tilde{S}).$$

Note that this is a local procedure and that this data structure not only enormously accelerates the distance computation but also ensures that the distance measure is always calculated to the correct part of the simplified mesh, i.e. the distance measurement respects the topology.

3.1.3 Distance from a triangle t to the simplified triangulation S

The maximum of the distances from all three vertices of a triangle t in the original mesh to the simplified mesh is not always an upper bound for the distance from a triangle t to the simplified triangulation S. If the smallest distances from the vertices of the triangle t exist to different triangles of the simplified mesh S, the distance from t to S may occur between a point on the border or even inside of the triangle t and a point somewhere on the simplified triangulation S. We distinguish two cases:

- The triangle t of the original triangulation has no vertex in common with the simplified triangulation S.
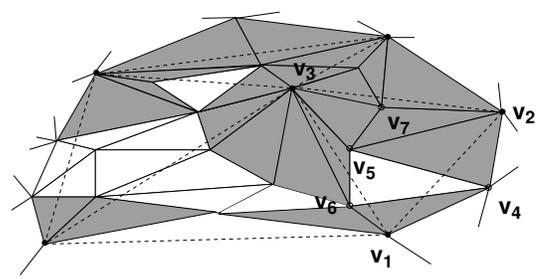

Figure 6: For every already removed vertex in the original triangulation we keep the triangle in the simplified triangulation that is nearest to the vertex itself. For example the vertices v_4, v_5, v_6 and v_7 store $\Delta(v_1, v_2, v_3)$. Vice versa the triangle $\Delta(v_1, v_2, v_3)$ stores the vertices v_4, v_5, v_6, v_7.
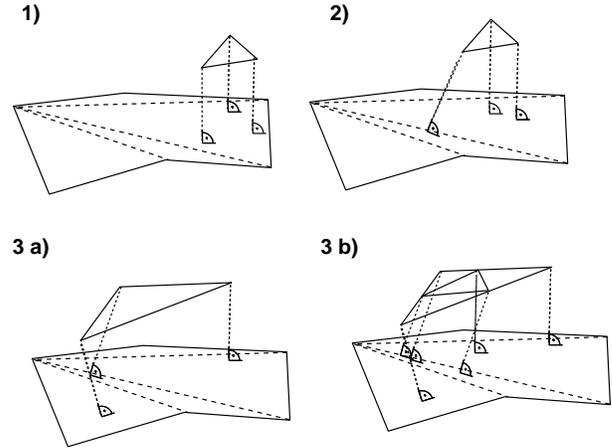

Figure 7: In the Figures 1), 2), and 3) the vertices of the triangle t have smallest distances to one, two, or three different triangles of the simplified triangulation S. In Figure 3 b) the original triangle is subdivided. Here all subdivided triangles belong to the same case like the one in Figure 2).

- The triangle t of the original triangulation has one or two vertices in common with the simplified triangulation S.

This distinction allows us to reduce all occurring cases to easier-to-handle ones using a simple regular subdivision of the original triangles. If the second case is not treated differently, the subdivision may not converge to one of the simpler cases.

Case 1: We consider the following three subcases, see Figure 7:

- All three vertices are nearest to the same triangle $s \in S$.

- The three vertices are nearest to two triangles $s_1, s_2 \in S$ that share an edge.

- All other cases.

In the first subcase we have

$$d_E(t, S) = max(d(v_1, S), d(v_2, S), d(v_3, S)).$$

The second subcase is a little bit more complicated. We intersect the half-angle plane between the two triangles s_1 and s_2 sharing a common edge with those edges of the triangle t having endpoints that belong to different triangles. We then use the maximum of distances of the vertices of t and the distances of these intersection points to
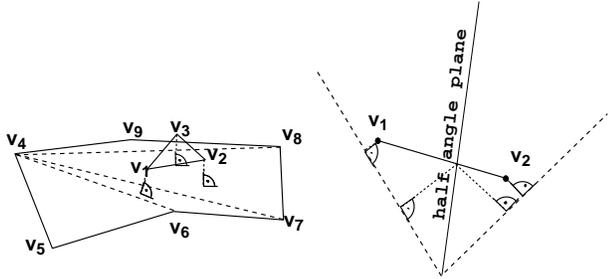
Figure 8: In the left Figure the situation is shown in 3D. Looking in direction of the edge $v_7 v_4$ we get the 2D situation on the right side. To calculate the distance from $\Delta(v_1, v_2, v_3)$ to the triangles $\Delta_1 = \Delta(v_4, v_6, v_7)$ and $\Delta_2 = \Delta(v_4, v_7, v_8)$ the half-angle plane between the two triangles Δ_1 and Δ_2 is intersected with the edge $v_1 v_2$ and the edge $v_1 v_3$. The distance from $\Delta(v_1, v_2, v_3)$ to the simplified triangulation S can then be calculated as the maximum of the distances from the intersection points and vertices v_1, v_2, v_3 to the triangles Δ_1 and Δ_2.
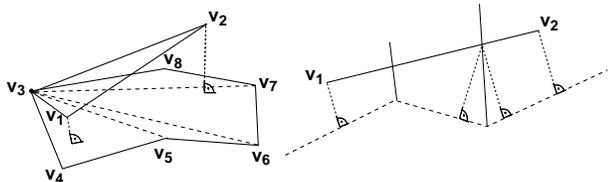

Figure 9: On the left side the 3D situation is shown, on the right side a 2D view. To obtain the distance from $\Delta(v_1, v_2, v_3)$ to the simplified triangulation S the distances to the triangles adjacent to v_3 are calculated using the half-angle planes between adjacent triangles.

the triangles s_1 and s_2 as an upper bound for the error, see Figure 8. In all other cases the original triangle is adaptively subdivided until the subtriangles fulfill either subcase 1 or subcase 2. Or until until the longest edge of a subtriangle is smaller than the predefined error tolerance ϵ in subcase3, see Figure 7-3b). In that subdivision terminating case the maximum distance

$$d(t, S) = max(d(v_1^{sup}, S), d(v_2^{sub}, S), d(v_3^{sub}, S))$$

is used. (To also get a correct upper bound for the approximation error in this case, one should run the algorithm with $\tilde{\epsilon} = \left(1 + \frac{1}{\sqrt{3}}\right)^{-1} \epsilon$ to ensure error ϵ.)

Case 2: In the case that the three vertices of the original triangle belong to triangles in the simplified triangulation S that share a common vertex, an upper bound of the maximum distance is again computed using the half-angle planes between adjacent triangles, see Figure 9. Using adaptive subdivison we reduce the general case 2 either to this case or to case 1 where none of the vertices of the subtriangle belong to the simplified triangulation. During the simplification process it may happen that subtriangles generated in the above cases are no longer needed because the adjacency relationships of the triangle in the simplified mesh change. If the triangles in the simplified mesh grow, more and more vertices of the subdivided triangles are nearest to the same or adjacent triangles in S. In such cases we remove the subdivided triangles.

3.2   Achieving the Hausdorff distance

The simplest way to achieve a sharp upper bound of the Hausdorff distance between the original and the simplified mesh is to measure the distances of the new edges of the retriangulated holes to all triangles containing a vertex that is nearest to one of the potentially removed triangles in each calculation of the potential distance .

4   Examples

Two different applications illustrate the superior results of our triangle decimation algorithm. The first application is the approximation of NURBS surfaces by triangle meshes. The NURBS surface is regularly sampled in parameterspace to achieve an error bound to an intermediate triangulation of 10000 vertices. This triangulation is simplified by the new algorithm using the Hausdorff distance and compared to a triangulation simplified using a L^∞ norm on the NURBS-parametrization, [Kle94, Ke95]. This application shows the superiority of the Hausdorff distance to the L^∞-Norm for a *geometric* approximation. The second application applies the decimation algorithm to the isosurface of medical data created using the Marching Cubes algorithm run on a data set of 113 slices with a resolution 512 by 512 pixel. More than 811000 triangles were required to model the bone surface. The results are compared to a result gained by the algorithm of Schroeder, Zarge, Lorensen [SZL92]. Despite of the very small error tolerances of one or one-and-a-half pixel, the reduction rates are even higher than the ones published by Schroeder et al. Further it should be noted that reduction as achieved by this algorithm cannot be achieved in general using a L^∞-Norm. In the third application we use an object containing different features like sharp edges. Due to the ordering of the removed points at the beginning of the algorithm planar regions are reduced first. Removing vertices on sharp edges would lead to illegal approximation errors.

5   Conclusion

We have described an algorithm for solving the mesh simplification problem, that is the problem of approximating an arbitrary mesh by a simplified mesh. The algorithm ensures that for each point in the original mesh there is a point in the simplified mesh with an Euclidean distance smaller than a user-defined error tolerance ϵ. For parameterized surfaces this distance also allows for much better reduction rates and is, in addition, independent of the parameterization.

We have applied our mesh simplification algorithm to different complicated meshes consisting of up to 811.000 vertices. The very impressive reduction rates for Marching Cubes outputs on medical data demonstrate the power of the algorithm even for error tolerances in the range of a voxel.

6   Acknowledgement

References

[CRS96]   P. Cignoni, C. Rochini, and R. Scopigno. Metro: measuring error on simplified surfaces. Technical Report B4-01-01-96, Istituto I.E.I. - C.N.R., Pisa, Italy, January 1996.

[DFP85]   L. DeFloriani, B. Falcidieno, and C. Pienovi. Delaunay-based representation of surfaces defined over arbitrarily shaped domains. *Computer Vision, Graphics and Image Processing*, 32:127–140, 1985.

Figure 11: Approximation of a parameterized NURBS surface using the new decimation algorithm with Hausdorff distance on the left side and using a L^∞-Norm on the right side. For each case the same maximum distance is used. Due to the L^∞-Norm the reduction rate in the upper Figure is less than in the lower Figure.
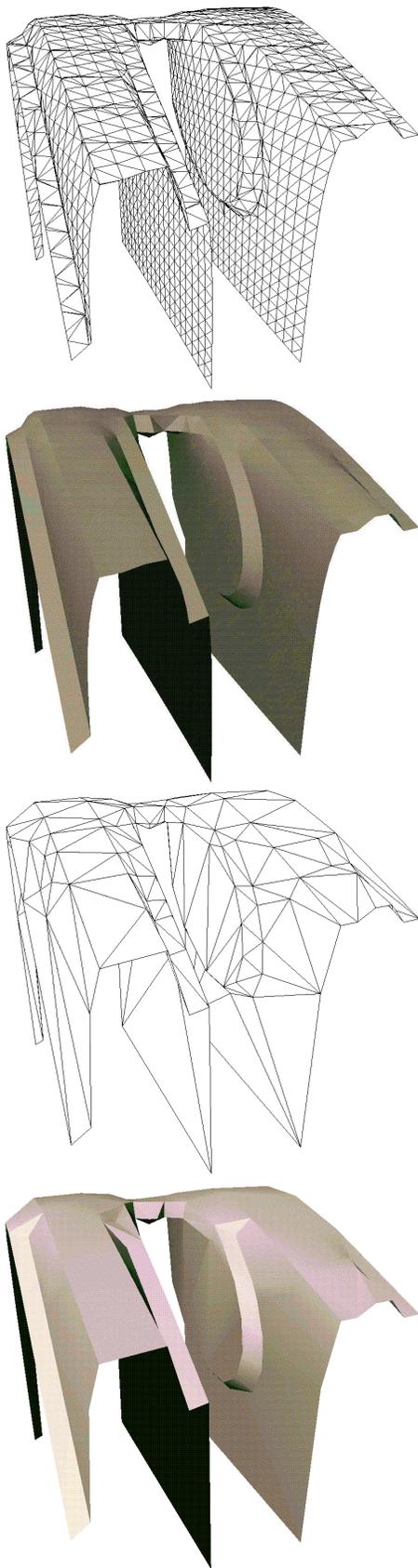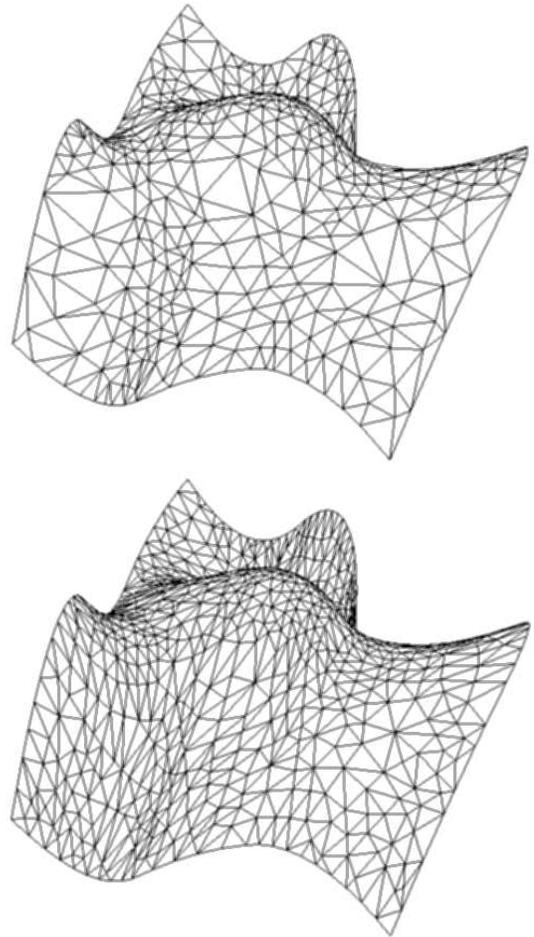
Figure 10: Simplification of an object consisting of 1341 vertices and 2449 triangles. The size of its bounding box is $20 \times 20 \times 20$ units. The approximation error is 0.25 units. The simplified mesh contains 124 vertices and 208 triangles. This is a reduction rate of 91%. Note the preservation of the sharp edges.
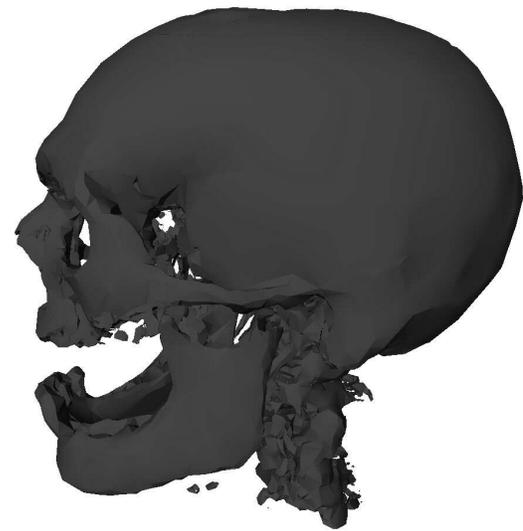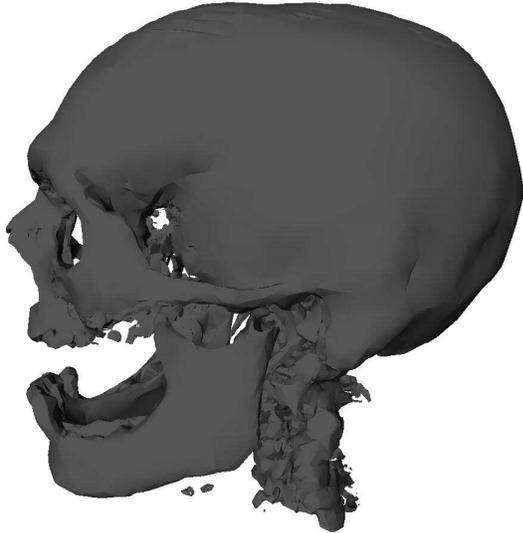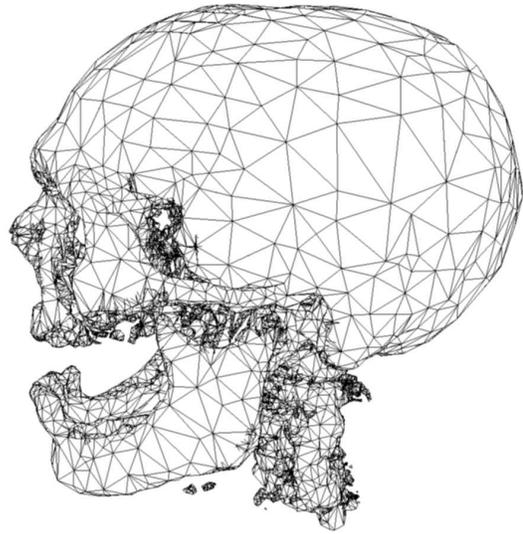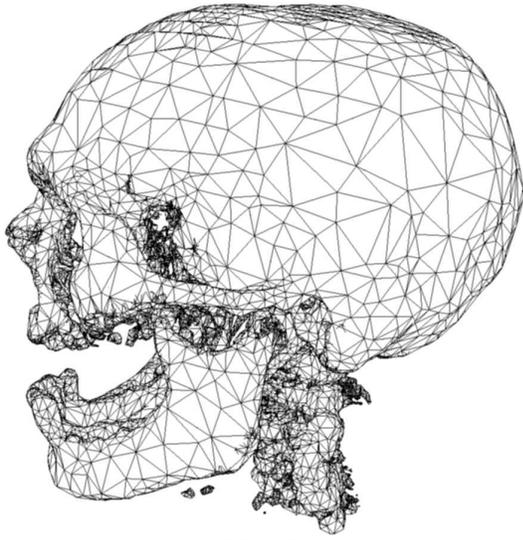
Figure 12: Approximation of the original mesh produced by a Marching Cubes algorithm up to the size of one pixel. The original dataset is reduced from 811k vertices (1622k triangles) to 30.5k vertices (62.7k triangles). This is a reduction rate of 96.2 %.

Figure 13: Approximation up to the size of one and a half pixel. The original dataset is reduced from 811k vertices (=1622k triangles) to 22.7k vertices (47.1k triangles). This is a reduction rate of 97.2 %.
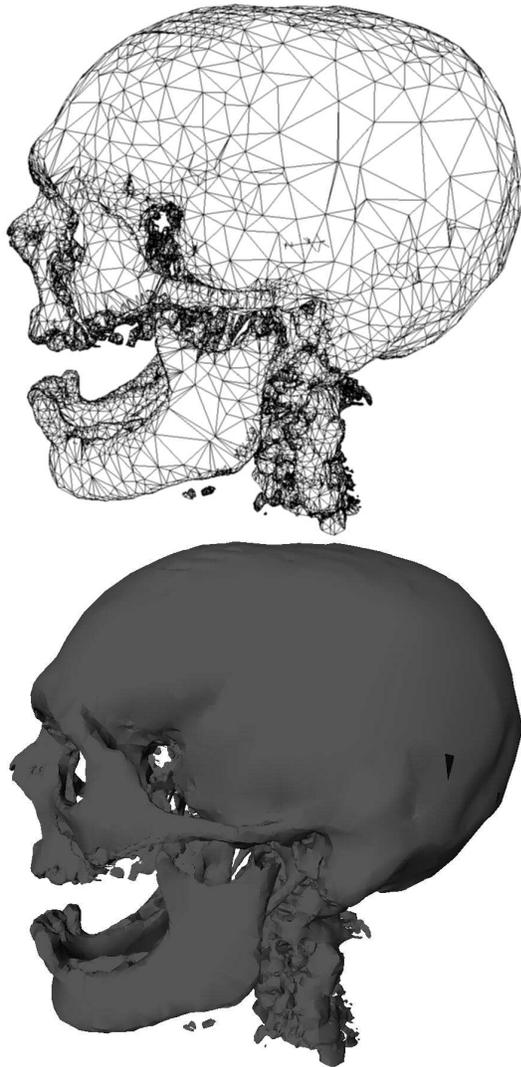
Figure 14: The same dataset reduced by the algorithm of Schroeder et al. down to 74.2k vertices. This is a reduction rate of 91%.

[EDD+95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 173–182. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 19–26, August 1993.

[HG95] P.S. Heckbert and M. Garland. Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1995.

[HH92] Charles Hansen and Paul Hinker. Isosurface extraction SIMD architectures. In *Visualization'92*, pages 1–21, oct 1992.

[Ke95] R. Klein and W. Straßer. Mesh generation from boundary models. In C. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 431–440. ACM Press, May 1995.

[Kle94] Reinhard Klein. Linear approximation of trimmed surfaces. In R.R. Martin, editor, *The Mathematics Of Surfaces VI*, 1994.

[MSS94] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In R. D. Bergeron and A. E. Kaufman, editors, *Visualization '94 Proceedings*, pages 281–287. IEEE Computer Society, IEEE Computer Society Press, 1994.

[RB93] J. Rossignac and P. Borrel. Multi-resolution 3d approximation for rendering complex scences. In B. Falcidieno and T. L. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465. Springer Verlag, 1993.

[Sb94] F. Schröder and P. Roßbach. Managing the complexity of digital terrain models. *Comput. & Graphics*, 18(6):775–783, December 1994.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

[Tur92] Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.