Building multiresolution models for fast interactive visualization

Reinhard Klein, Jörg Krämer
Wilhelm-Schickard-Institut, GRIS, Universität
Tübingen, Germany

Abstract

In this paper we describe a new method for constructing a multi-resolution representation for complex triangle meshes and we describe a new algorithme to refine and to coarsen a simplified mesh at variable resolution with respect to a given error function. Using this algorithm we show how view-dependent adaptive approximations of the triangle nets with guaranteed approximation error can be generated on the fly.

A modified one-sided Hausdorff distance is used as a geometrically meaningful error between the original and the simplified meshes. Since this distance is parameterization-independent, its use as error measure is superior to the use of the L^{∞}-Norm between parameterized surfaces.

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

The visualization of large models of real-world objects, like cars, trains, airplanes, etc. is a major challenge in the context of virtual reality. In most scenarios a variety of such models have to be visualized and animated simultaneously. Examples are the optimization of the cabin of a train or the optimization of a driver's position in a car. In such a setting not only the particular train or car has to be visualized but also other cars, pedestrians, buildings, etc. Furthermore, due to new acquisition techniques and more elaborated modeling and animation techniques the size of the models is still growing. Despite the performance of modern graphics hardware the visualization of such scenarios cannot be realized in real-time without special techniques to reduce the number of triangles that are rendered.

There are three major strategies for the optimization of rendering large triangle meshes for the ultimate goal of real-time performance: texture-based representations, visibility culling, and multiresolution modeling. The common denominator of these three approaches is the notion of output-sensitivity. Output-sensitivity means that the amount of CPU time invested in rendering the image is proportional to the output, no matter what the size of the input is.

Texture-based representations dynamically simplify geometric complexity using textures. The nearby subset of a model is rendered as geometry and the distant but visible subset of a model is represented and rendered as texture [Ali96, SS96].

Visibility culling means culling additional geometry, which might be in the viewing frustum, yet being invisible. For walkthroughs of large architectural scenes this might be quite significant [GKM93, SG96, Gre96].

Multiresolution modeling means maintaining multiple models of the scene geometry, each at a different level of detail or approximation. And it allows to render the model at any given moment with minimal level-of-detail (LOD) sufficient to produce an image of reasonable quality. This approach results in significant savings when rendering models at large distance from the viewer.

In a recent paper Ciampalini, Cignoni, Montani and Scopigno [CCMS96] characterize *multiresolution management* as the most significant point of real multiresolution modeling. This means that once the simplification process has been run, it is possible to *interactively extract* any LOD representation, with the complexity of a single LOD extraction being linear to the LOD output size.

Real multiresolution modeling should support a number of applications, including

1. *Progressive transmission:* When a mesh is transmitted over a communication line, one would like to show progressively ameliorating approximations of the model as data is incrementally received.

2. *Automatic viewing parameter-dependent approximation:* After changes of the viewing parameters the LOD of an object should be automatically adjusted.

3. *Error tolerance editing:* In some applications, like finite element mesh generation, it is useful to interactively define different error tolerances in different areas of the model. In such a way the user is able to get better approximations in the areas of interest of the model.

This paper makes two major contributions. First it analyzes and improves the mesh-simplification algorithm with global error control [KLS96] with respect to its performance and to the quality of the resulting simplified meshes and shows how based on this algorithm a multiresolution model (MRM) is generated.

The other contribution is a new algorithm for the dynamic generation of different levels of detail 'on the

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

fly'. This algorithm is an extension of an algorithm to extract triangulations at variable resolutions from a MRM described by Puppo [Pup96]. To extract a triangulation at variable resolution his algorithm always starts with the coarsest triangulation and traverses the hierarchy until each triangle satisfies a certain approximation condition. The new algorithm allows to dynamically refine or to coarse a triangulation at any given variable resolution. This is essential for real-time visualization of large models. Instead of extracting from frame to frame the whole triangulation from the MRM, the frame-to-frame coherence can be exploited and only those triangles are removed from or inserted into the simplified mesh whose removal or insertion is necessary due to the change of camera parameters.

2 Previous work

In the literature a variety of techniques for the generation of multiresolution models is proposed. To obtain a hierarchy of increasingly coarser approximations of the input object, mesh simplification algorithms are applied. Various techniques were published that aimed to reduce surface complexity. Most of 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 and discuss some general and valuable solutions for mesh simplification. For some surveys see also [CRS96, HG95, Eri96].

- Coplanar facets merging: Coplanar or nearly coplanar data are searched, merged in larger polygons and then retriangulated into fewer simple facets, [HH92, MSS94, KT96].

- 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]. By forcing the simplified mesh to lie between two offset meshes, simplification envelopes [CVM+96] control the global error of this simplification process. In [KLS96] the Hausdorff distance is used as a global error measure between the original and simplified mesh. In addition, all vertices are sorted and removed in the order of the error that would occur if the vertex was removed. This gives the user control of the approximation errors during the simplification process and allows for the managment the multiresolution hierarchy of the simplified meshes in a canonical way, see section 4. In Ciampalini et al. [CCMS96] the vertices are also removed according to a global error that would occur if the vertex was removed. In this paper different local and global approximations of the real error between original and simplified mesh are considered.

- Mesh optimization and progressive meshes: An energy function is evaluated over the mesh and is minimized either by removing/moving vertices or collapsing/swapping edges [HDD+93]. In [Hop96], Hoppe introduced the *progressive meshes*. He showed that collapsing edges is sufficient for effectively simplifying meshes. The edge collapse transformation unifies two adjacent vertices into a single vertex and the two adjacent faces vanish in the process. The original mesh can be restored by a sequence of vertex split operations (the reverse operation of edge collapse). For a vertex split operation only the vertex to be split and pointers to two of its neighboring vertices are necessary. In this way, the whole hierarchy of intermediate levels of detail can be stored in a space-efficient way. Independently from Hoppe, Ronfard and Rossignac [RR96] proposed a similar algorithm based on edge collapses.

  It is the simplicity of the edge collapse operation and the space-efficient way to store the resulting hierarchy that makes the edge collapse approach so attractive. On the other hand, the edge collapse approach can be considered as being a special case of the mesh decimation approach. This subject is discussed in more detail in section 3.

- 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 neighbors in order 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 parameterization to build a multiresolution representation of the surface from which any approximated representation can be extracted [EDD+95].

The most simple MRMs consist of a hierarchy of 5 to 10 different LODs of a given object. At run-time the perceptual importance of a given object in the scene is used to select its appropriate level of representation from the hierarchy [Cla76, FS93, RH94].

The drawbacks of this approach are, that switching between two different levels of detail causes visual artifacts, and that of course no optimal adaptive approximation is possible with discrete resolution steps.

Simplification models that iteratively remove a vertex or collapse an edge or a triangle of a mesh implicitly define a hierarchy of different levels of detail of the simplified object. Each successive LOD differs from the previous one in the local vicinity of the removed vertex, edge, or triangle only. The advantage of this approach is that the difference between two successive LODs is relatively small. Changing between different levels does only cause small visual artifacts. MRMs that are build on this observation are for example the *progressive meshes* by Hoppe and the MRM proposed by Ciampalini et al. [CCMS96]. In addition, the progressive meshes allows for smooth geomorphing between all different levels of detail, resulting in a smooth transition between different levels of detail. A shortcoming of these approaches is, that they do not efficiently support the co-existence of different levels of detail across different regions of the same object. The selective refinement described by Hoppe [Hop96] is only a first step in this direction. As Hoppe pointed out, a stringent version of the refinement condition for selective refinement can cause the refinement to fail. Therefore, he suggests a less stringent condition, which necessarily introduces additional triangles. Therefore, in the proximity area of these additional triangles no bounds on the approximation errors between the refined triangulation and the original mesh are known.

Xia and Varshney [XV96] propose an algorithm based on a precomputed merge tree which supports several different levels of detail to co-exist across different regions of the object. The different levels of detail in different regions seamlessly merge with one another without introducing any cracks and discontinuities. In the following such a multiresolution model is called *variable multiresolution model*. The drawback of this approach is that the geometric error between simplified mesh and original mesh is only estimated. Every merge tree node stores an Euclidean distance for splitting a vertex to its child as well as the distance at which it will merge to its parents. In many cases this leads to an under-estimation of the actual global error while in other cases these errors accumulate resulting in a poor approximation of the original mesh.

In our previous work [KH96, KHK96a] we showed how an adaptive MRM can be build for parameterized surfaces using a constrained Delaunay triangulation. The MRM is either computed by a mesh refinement-algorithm or by the mesh simplification-algorithm mentioned above [Kle94, KS95, KLS96]. In both cases, a global error between the simplified and original mesh is used to guarantee a sufficient approximation of the original mesh by the adaptive triangulation extracted from the MRM. Through the use of the constrained Delaunay triangulation the topology of the mesh is implicitly given and needs not be stored explicitly. Especially for large models this is a significant advantage [KS96, RKH96]. Unfortunately, this technique does not carry over to non-parameterized surface meshes.

Therefore, for non parameterized surfaces we adopted the multi-triangulation proposed by Puppo [Pup96, FPM96] to build up a MRM during the mesh simplification process [KKL96, KHK96b, Lie96]. In section 3 we briefly review the main ideas of this approach and show how the global approximation error calculated in our simplification algorithm [KLS96] can be used to build an appropriate MRM.

3 The simplification algorithm

The algorithm is a typical mesh simplification algorithm. It starts with the original triangulation Σ_M and successively simplifies it by removing vertices from the actual triangulation and retriangulating the resulting holes. This is done until no further vertices can be removed from the simplified triangulation without exceeding a predefined Hausdorff distance between the original triangulation and the simplified one. After removing m vertices we denote the simplified triangulation by Σ_m.

3.1 The priority queue

The priority queue is one of the central features of the mesh simplification algorithm. For every single vertex v_i of the simplified mesh an error value ϵ_i is computed and stored. This value describes the *potential error*, that is the Hausdorff distance that would occur between the simplified and original mesh if the vertex would be removed. At the beginning of the algorithm this error value is computed for every vertex in the original mesh and the vertices of the original mesh are sorted into a priority queue according to their error-value. Afterwards, we eliminate the vertices with the smallest potential error on top of the priority queue in each simplification step. If a vertex is actually removed from the current simplified triangulation the priority queue is updated. 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 of the vertex is set to infinity.

The priority queue in combination with a global geometric error guarantees that the simplified triangulation Σ_m – obtained after removing the first m vertices from the priority queue – approximates the original triangulation with a global geometric error ϵ_m. This error can also be considered as a *local error*: If the next vertex v_{m+1} is removed also its adjacent triangles are removed and the remaining hole is retriangulated. The global error between the original and simplified mesh changes to ϵ_{m+1}. Since only the triangles in the neighborhood of v_{m+1} are changed, the actual global approximation error either did occur between the new or former triangles of the retriangulated area and the original triangulation.

As already mentioned above the potential errors need to be updated, if a vertex is removed from the actual simplified triangulation for all neighboring vertices $v_1, \cdots, v_n,\ n \in \mathbb{N}$ of v. Furthermore, the vertices have to be removed from and reinserted into the priority queue according to their new potential error. This can be done in $O(\log r)$ time, where r is the number of remaining vertices in the simplified mesh.

3.2 Measuring a modified one-sided Hausdorff distance between simplified and original mesh

As a global error measure between the original and simplified mesh a modified version of the one-sided Hausdorff distance is used. 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 one-sided Hausdorff distance $d_E(X, Y)$ *from* a set X

to a set Y by

$$d_E(X,Y) = \sup_{x\in X} d(x,Y). \tag{1}$$

If the Hausdorff distance $d_E(\Sigma_M, \Sigma_m)$ between the original triangulation Σ_M and the simplified triangulation Σ_m is less than a predefined error tolerance ϵ, then

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

Therefore, this one-sided Hausdorff distance between the original and the simplified triangulation is the one a user would intuitively think of. But in some cases the unsymmetry of the one-sided 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.

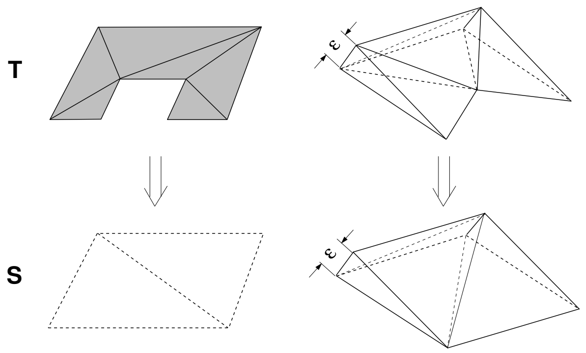

Figure 1: Left: The original nonconvex triangle mesh T is contained within a plane. The simplified mesh S consists of a triangulation of the convex hull of the original mesh and therefore the original mesh is contained in the simplified mesh. Right: The same situation in 3D. $\forall x \in T$ there is 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 $d(x,y) < \epsilon$.

One way to overcome this problem is to use the full *Hausdorff distance* instead of the one-sided distance. It is defined by

$$d_H(X,Y) = \max(d_E(X,Y), d_E(Y,X)). \tag{2}$$

In contrast to the one-sided Hausdorff distance it is symmetric and we have

$$d_H(X,Y) = 0 \Longleftrightarrow X = Y.$$

Unfortunately its computation is too expensive. Therefore, we make a compromise between speed and accuracy. In adddition to the one-sided Hausdorff distance $d_E(\Sigma_M, \Sigma_m)$ between the original and simplified triangulation we compute the one-sided Hausdorff distance between the border of the simplified and the original mesh. In such a way for a given ϵ an accurate approximation of the borders of the original mesh by the border of the simplified mesh up to ϵ can be guaranteed while for every inner point x of the original triangulation there is always a point y of the simplified triangulation with $d(x,y) \leq \epsilon$.

It is worthwhile to mention that for *any* parameterized surface

$$f : \mathbb{R}^2 \supset \Omega :\longrightarrow \mathbb{R}^3$$

that is approximated by a piecewise linear surface

$$f_\Sigma : \mathbb{R}^2 \supset \Omega :\longrightarrow \mathbb{R}^3$$

we always have

$$d_E(f, f_\Sigma) \leq ||f - f_\Sigma||_\infty = \sup_{(u,v)\in\Omega} ||f(u,v) - f_\Sigma(u,v)||.$$

For this reason, using the modified one-sided Hausdorff distance for error measurements results in higher reduction rates for the same error tolerance.

3.2.1 The computation of the Hausdorff distance between the borders of the original and simplified mesh

If the removed vertex v was a border vertex the distance between the new border and the original border must be computed. To simplify the description of the algorithm we concentrate on one border polygon B_M consisting of consecutively numbered vertices $v_1, v_2, \ldots, v_n$. If a vertex v_i is removed the border edges $v_{i-1}v_i, v_iv_{i+1}$ are substituted by the edge $v_{i-1}v_{i+1}$ and so on, resulting in a simplified border polygon $B_m = (v_1 = v_{i_1}, v_{i_2}, \ldots, v_{i_{k-1}}, v_{i_k} = v_n)$. Then, the distance

$$d(B_M, B_m) = \max_{i_l\in\{i_1,\ldots,i_{k-1}\}} \max_{i_l<j<i_{l+1}} d(v_j, v_{i_l}v_{i_{l+1}}) \tag{3}$$

can easily be computed and is an upper bound for the Hausdorff distance $d_H(B_M, B_m)$ between the original and simplified border polygon.

3.2.2 The computation of the one-sided Hausdorff distance $d_E(\Sigma_M, \Sigma_m)$.

To compute the potential error of a vertex v first of all the one-sided Hausdorff distances

$$d_E(\Sigma_M, \Sigma_m)(v) = max_{\Delta\in\Sigma_M} d_E(\Delta, \Sigma_m(v))$$

have to be computed. The computation of the one-sided Hausdorff distance in this general form is computationally too expensive. Therefore, a correspondence map between the vertices and triangles of the original triangulation and the triangles of the simplified triangulation is introduced into the algorithm: Every vertex of the original triangulation is projected onto the simplified triangulation and the triangle in the simplified triangulation with minimum distance is stored and vice versa every triangle of the simplified triangulation stores the vertices projected on it. If the vertices of an original triangle are projected to triangles which do not have at least one vertex in common, e.i. that are not adjacent, the original triangle is recursively subdivided into four triangles, until the projections of the vertices of a subtriangle are adjacent or the subtriangles are smaller than a predefined error tolerance. In such a way, the computation of the distance $d_E(\Delta, \Sigma_m)$ between a triangle Δ of the

	reduction time	number of retriangulation calc.	number of distance calc
bunny error 0.01	34,841 vertices reduced to 91 vertices		99,7%
with optimization	147,5 sec	194,583	6,081,303
no optimization	235,9 sec	244,650	14,410,289
mandible error 1.0	38,924 vertices reduced to 204 vertices		99,5%
with optimization	169,9 sec	223,595	6,304,963
no optimization	240,0 sec	262,558	12,834,570
face error 1.0	12530 vertices reduced to 589 vertices		95,2%
	reduction time	number of retriangulation calc.	number of distance calc
with optimization	38,0 sec	73,198	1,034,095
no optimization	49,1 sec	81,382	1,895,480

Table 1: The number of distance calculations and the number of retriangulations that can be saved using the optimization techniques.

original triangulation and the simplified triangulation is easy to compute.

3.3 Acceleration of the simplification algorithm

The computation costs of a simplification algorithm can be estimated as follows: If we remove the $m-$th vertex from the triangulation the potential error of its neighbor vertices has to be updated. If we consider a closed triangle mesh without border the relation between the number of triangles f and the number of vertices v is given by

$$f = 2v - 4. \tag{4}$$

To estimate the cost of this operation for the removal of one vertex we have to retriangulate the remaining hole and to measure the distance between the vertex and the retriangulated area. In addition, the distances between all vertices corresponding to triangles of the retriangulated area and the new triangles of the retriangulated area have to be computed. According to Eq. 4 these are in average

$$\frac{m-1}{f} = \frac{m-1}{2v-4} \tag{5}$$

vertices per triangle. The distance between these vertices and in average 5 triangles in the retriangulated area have to be computed. In total, in each simplification step we have to compute in average

$$6 \times 6\left(\frac{m-1}{2(v-m)-4} + 1\right) \times 5 \tag{6}$$

distances between vertices and triangles. Therefore, the total number of distance computations necessary to remove n vertices is in average

$$\begin{aligned} & 180 \sum_{m=1}^{n} \left(\frac{m-1}{2(v-m)-4} + 1\right) \\ = \; & 180\left(n + \sum_{m=1}^{n} \frac{m-1}{2(v-m)-4}\right). \end{aligned} \tag{7}$$

To reduce this amount of distance computation we exploit the following observations:

- Consider all neighbor vertices of the vertex and let ϵ be the maximum of their potential error. As soon as we find one vertex with a distance greater than ϵ, the distance computation can be stopped, since in this case the neighboring vertex will be removed first and a further potential error has to be computed for the vertex. If the distance computations are stopped the potential error is set to infinity.

- For the vertex itself, instead of computing its distance to all new triangles of the retriangulated hole, the distance of the border vertices of the remaining hole to an average plane passing through the vertex itself is measured. If all vertices are on the same side of the average plane, this distance is always smaller than the minimum distance of the vertex to the triangles. Therefore, if it exceeds the ϵ defined in the previous item, the distance computations can be stopped as well.

Table 1 shows the number of distance calculations and the number of retriangulations that can be saved using these techniques.

The topology tests mentioned above ensure that the border polygons of the remaining holes can be projected onto plane, non-selfintersecting polygons. These polygons are not necessarily convex, but we found that in average about 75% of the projected polygons are. Therefore, to avoid visibility tests for the convex polygons during the retriangulation, first of all, the convexity of the polygon is checked and only then the remaining hole is retriangulated. Furthermore, we restricted the order of a vertex to be removed to 8. In most cases this does not influence the quality of the resulting triangulations but it accelerates the retriangulations of the remaining holes and improves the depth of the resulting MRM.

3.4 Vertex removement or edge collapse?

At a first glance, the approach of removing vertices and retriangulating the remaining hole and the edge collapse approach seem to be different, but on closer examination, the edge collapse approach can be considered as a special case of removing vertices and retriangulating the remaining hole.

The retriangulation of the remaining hole determines the quality of the resulting simplified triangulations, as well as the reduction rates that are achievable

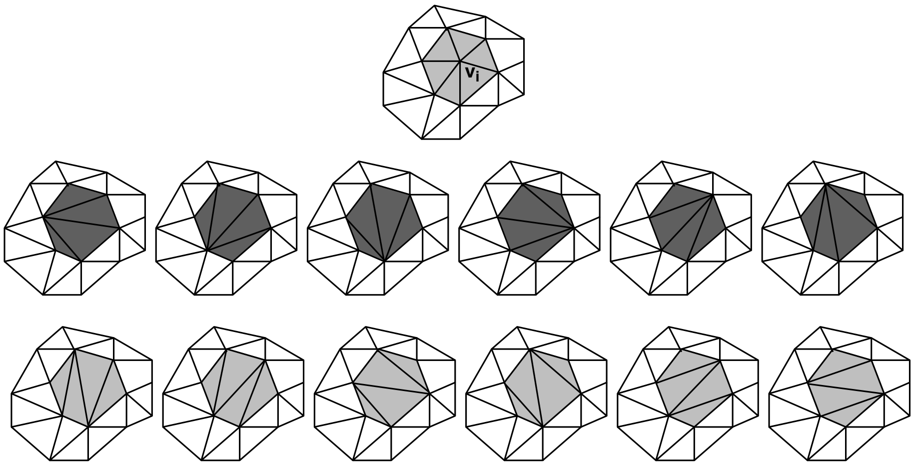

Figure 2: Edge collaps as a special case of retriangulation: The six possible edge collapse operations collapsing vertex v_i to one of its neighbor vertices v_j can be interpreted as a vertex remove operation of vertex v_i using special retriangulations of the hole (middle row). More retriangulations are possible that cannot be realized using an edge collapse operation (lower row).

method	reduced to	reduction rate	angle$< 30^o$	angle$> 150^o$
bunny error 0.001 34,841 vertices				
Delaunay	815 vertices	97.7%	9.4%	0.1%
Optimized	646 vertices	98.1%	19.3%	1.5%
Edge-Collapsing	621 vertices	98.2%	21.0%	1.8%
mandible error 0.5 38,924 vertices				
Delaunay	556 vertices	98.5%	10.3%	0.1%
Optimized	439 vertices	98.6%	23.9%	1.3%
Edge-Collapsing	443 vertices	98.6%	26.1%	2.3%
face error 0.5 12530 vertices				
Delaunay	1638 vertices	86.9%	13.8%	0.2%
Optimized	1378 vertices	89.0%	23.0%	1.7%
Edge-Collapsing	1370 vertices	89.0%	25.0%	2.1%

Table 2: Reduction rates and mesh quality using different retriangulation techniques.

by the algorithm. We implemented different strategies for the triangulations of the remaining hole:

1. **Lookup-table** For convex polygons, precomputed triangulations are stored in a lookup-table. For non-convex polygons, any triangulation is used.

2. **Delaunay** Constrained Delaunay triangulation

3. **Optimized** Geometry-dependent triangulation. This triangulation is computed by a sequence of edge flips. An edge of the triangulation is flipped if the distance of the removed vertex to the flipped triangles is less than to the considered triangles. If no further edge can be flipped, the flipping process stops.

4. **Edge collapse** An edge collapse operation which collapses a vertex v_i onto a vertex v_j can be considered as a vertex-remove operation in which the resulting hole is retriangulated by connecting all vertices of the remaining hole to vertex v_j, see Figure 2. To get all possible edge collapses we consider all retriangulations of the remaining hole of this kind.

The quality of the simplified triangulations using the first type of retriangulation is not very good. This influences the reduction rates that can be achieved while guaranteeing a certain error. In general they are not as high as the reduction rates that can be achieved by the other strategies. The constrained Delaunay triangulation optimizes the shape of the triangles in the simplified triangulations. For certain objects like the 'coke can' in Figure 3 this is not always the thing we want to have. In this case much better reduction rates can be achieved using the optimization technique or the edge collapse technique. Nevertheless, there are cases like the 'mandible' in figure 6 where we want to do finite-element computations. Here the results using a Delaunay triangulation are superior to the other techniques, see Table 2.

The optimization technique delivers a little bit higher reduction rates than the edge collapses but in general the results are similar. This is due to the fact that using the edge collapse technique several different retriangulation possibilities are checked and the one with minimum resulting approximation error is taken. For polygons with 4 or 5 vertices the edge collapse approach checks all possible triangulations. For polygons with more vertices only a few triangulations out of all possible ones are checked: In general, for a

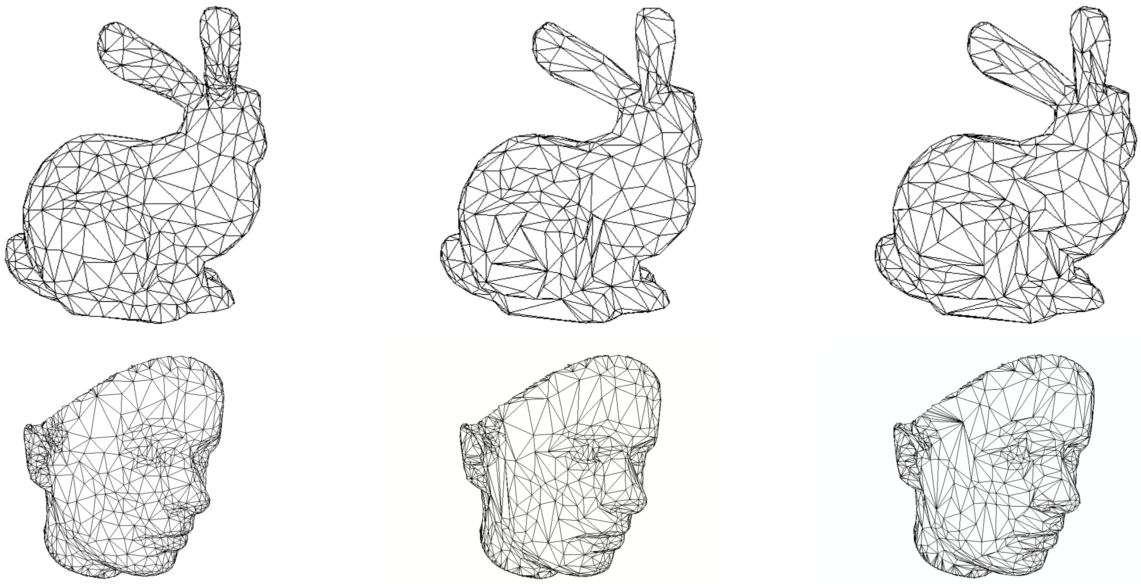

Figure 4: Three simplified meshes of a bunny and a face approximating the initial mesh with the same guaranteed approximation error. In left figure a constrained Delaunay triangulation, in the middle the geometric-dependent triangulation and on the right edge collapsing was used to retriangulate the remaining holes. Using a constrained Delaunay triangulation for the retriangulation of the holes delivers triangles with better aspect ratio than the other techniques. Less than 10% of the triangles contain angles with less than 30°, while using the other techniques about 25% of the triangles contain such triangles. Especially for finite element meshing this is an important criteria.

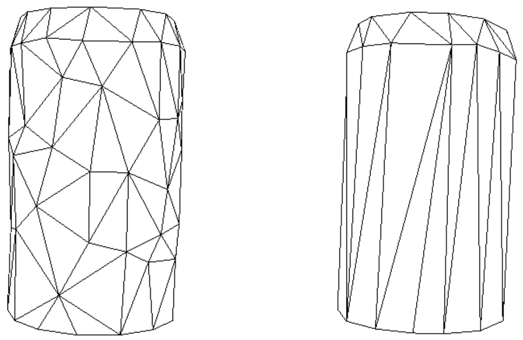

Figure 3: Two simplified meshes of a coke-can approximating the initial mesh with the same guaranteed approximation error. In left figure a constrained Delaunay triangulation was used, to retriangulate the hole. In the right figure the geometry dependent triangulation was used for the remaining hole. The left model contains 120 triangles, the right one only 50.

convex polygon with n vertices there exist

$$\sum_{i=1}^{n-2} c_{i+1} c_{n-i}, \; c_2 := 1, \; n \geq 3 \tag{8}$$

triangulations. For $n = 6$ this are 12 different possible triangulations of which 6 are not considered using the edge collapse technique. For higher n this number is higher (n=7, 36/29, n=8, 112/104). Since for concave polygons not all possible triangulation are valid, these numbers may be less, but nevertheless the general situation remains the same.

4 Dynamic change between different levels of detail

Despite the acceleration technique discussed above, the computation of a MRM with guaranteed approximation error for each LOD is relatively time-consuming and cannot be achieved in real-time. But after the MRM is build up, the extraction of variable levels of detail from the MRM with guaranteed approximation error in real-time is feasible. Therefore, we propose a two-stage approach towards generating view-dependent triangulation meshes:
In the first step a view-independent MRM is computed. This computation is expensive and therefore not appropriate for real time interaction. In the second step a view dependent triangulation is extracted from the precomputed MRM. This step is inexpensive and can therefore be easily achieved in real time.

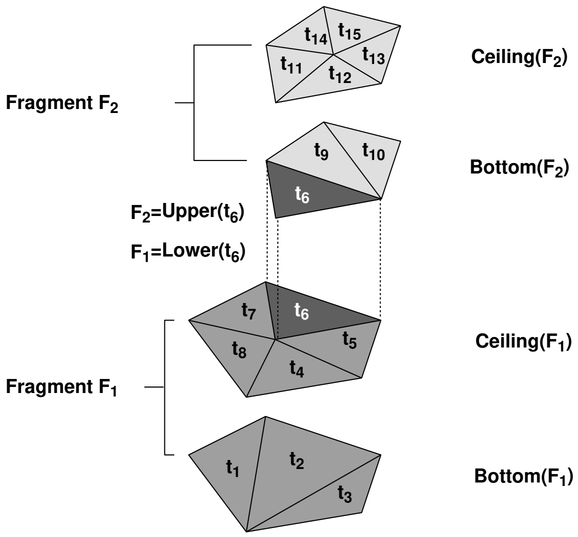

Figure 5: A fragment contains indices the triangles of the bottom and of the ceiling. On the other hand each triangle stores two pointer of the fragment containing it.

In the following we briefly describe our data-structure of multi-triangulation, see also [Pup96].The data structure of the multiresolution model consists the following three data sets.:

1. List of all vertices,
2. List of all triangles,
3. Set of all fragments, see figure 5.

The set of all fragments represents the whole multi-triangulation. A fragment represents a remaining hole in the simplification process. The *floor* of the fragment contains the triangles of the retriangulation and the *ceiling* contains the triangles of the hole before the retriangulation. Each fragment stores the indices of the triangles contained in its floor and its ceiling and the global error between the simplified and the original triangle that occurred before removing the vertex. Vice versa each triangle stores two pointer to the two fragments containing it: The *upper fragment* containing the triangle in its floor, the *lower fragment* containing the triangle in its ceiling.

In [Pup96] Puppo gives an algorithm for extraction of a LOD at variable resolution in linear time. This algorithm always starts from the fragments of the coarsest LOD.

Then the fragments are iteratively replaced by upper fragments in a breadth-first order until all triangles T of the current fragment fulfill a user-specified Boolean condition $c(T)$ or until the finest LOD is reached. Since this algorithm always starts from the coarsest triangulation the frame to frame coherence cannot be exploited by the algorithm.

We implemented a new algorithm that is able to refine a given triangulation in a certain area and to coarse it inn another area. Using this algorithm the frame to frame coherence of the meshes can be exploited. In the algorithm a function $c(\Delta)$ is needed which determines whether the approximation error of a triangle stored in its upper fragment is sufficient or not, e.i. whether the triangle approximates a neighborhood of the original triangulation with sufficient exactness. During the refinement step fragments are iteratively replaced by upper fragments in a breadth-first order until all triangles T of the current fragment fulfill a user specified Boolean condition $c(T)$ or the finest LOD is reached.

In practice it is necessary to coarsen the mesh a certain locations and refine it at other locations. An example is the change of a viewpoint during the visualization process. It is trivial to modify the algorithm given by Puppo in such a way, that the given mesh can be further refined.

The main idea that enables us to coarsen the mesh in certain regions is to keep track of all fragments for that all triangles of their ceiling are contained in the current triangulation. The triangles in the ceiling of that fragments are the only ones, that can be removed from the current mesh. Therefore, the algorithm that coarsens the triangulation checks if the condition $c(\Delta)$ is still true for all triangles in the floor of such a fragment. If this is the case the triangles in the ceiling are removed from the current mesh and the queue containing these fragments is updated accordingly.

The C++-code of the algorithms to refine and coarsen the mesh can be downloaded from *http://www.gris.uni-tuebingen.de/people/staff/reinhard/publications.html*.

4.1 Interactive error tolerance editing

In many applications it is desirable to refine or to coarse the mesh interactively in some areas, see for example figure 6. In this example for the purpose of finite element computations the original mesh is refined around a single tooth where the maximum deformations during the computation are expected. To realize interactive editing the user can mark an area of triangles of the mesh trough picking and assign smaller error tolerances on the triangles of the marked area. The algorithm then automatically refines the mesh until all triangles of the area fulfill the new error tolerances.

4.2 Progressive transmission

The proposed MRM-model built upon the mesh simplification algorithm also supports progressive transmission. During the simplification process vertices are successively removed causing coarser and coarser meshes. The removal of vertices results in an increasing approximation error, although the removal of a single vertex in some cases may produce even an intermediate result with smaller approximation error. For progressive transmission first of all the coarsest mesh, building the bottom of the MRM, is transmitted. Then, the vertices with their accompanying fragments are transmitted in the reverse order of their removement. Once the bottom of the mesh is transmitted the receiving side can immediately start to refine the bottom mesh getting better and better approximations. The main difference to the progressive mesh algorithm is, that aside from the new vertex and two anchor vertices also the triangulation the accompanying fragment have to be transmitted.

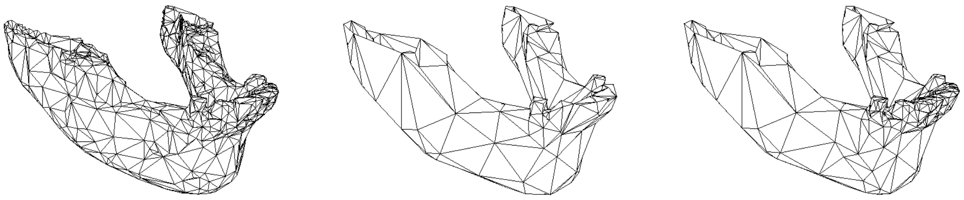

Figure 6: On the left side the mesh of the mandible produced by the marching cube algorithm. In the middle a simplified mesh. On the right side in the area of interest around the left tooth the mesh of the mandible is interactively refined.

5 Conclusion

The mesh-simplification algorithm with guaranteed approximation error is well suited for constructing multi-resolution representations of complex triangle meshes. Using the acceleration techniques described in this paper the algorithm is fast enough to process even very complex meshes in reasonable time. The method to compute a global approximation error also applies to the edge-collapse approach proposed by different authors. Nevertheless, the approach of removing a vertex and retriangulating the remaining hole is more flexible than only collapsing edges. Removing vertices allows the user to choose an appropriate technique for retriangulating the remaining hole. This results in more adequate triangulations and better reduction rates of the simplified meshes.

The proposed multiresolution model supports several different levels of detail to co-exist across different regions of the object. The different levels of detail in different regions seamlessly merge with one another without introducing any cracks and discontinuities. The algorithms to refine and to coarsen a simplified mesh allow to exploit the frame to frame coherence between successive frames during the visualization process. Therefore, variable levels of detail can be extracted from the MRM in real-time, even for very complex meshes.

6 Acknowledgement

We would like to thank A. Schilling for many fruitful discussions.

References

[Ali96] Daniel G. Aliaga. Visualization of complex models using dynamic texture-based simplification. In *IEEE Visualization '96*. IEEE, October 1996. ISBN 0-89791-864-9.

[CCMS96] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. Technical Report CNUCE: C96-021, Istituto per l'Elaborazione dell'Informazione - Condsiglio Nazionale delle Richere, Pisa, ITALY, July 1996.

[Cla76] J. H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, October 1976.

[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.

[CVM+96] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick P. Brooks, Jr., and William Wright. Simplification envelopes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 119–128. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[EDD+95] M. Eck, T. DeRose, T. D., H. Hoppe, M. Lounsbery, and W. 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.

[Eri96] Carl Erikson. Polygonal simplification: An overview. Technical Report TR96-016, Department of Computer Science, University of North Carolina - Chapel Hill, February 16 1996. Wed, 26 Jun 1996 18:07:48 GMT.

[FPM96] L. De Floriani, E. Puppo, and P. Magillo. A formal approach to multiresolution modeling. In W. Straßer, R. Klein, and R. Rau, editors, *Theory and Practice of Geometric Modeling)*. Springer-Verlag, 1996.

[FS93] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247–254, August 1993.

[GKM93] N. Greene, M. Kass, and G. Miller. Hierarchical Z-buffer visibility. *Computer Graphics*, 27(Annual Conference Series):231–238, 1993.

[Gre96] N. Greene. Hierarchical polygon tiling with coverage masks. *Computer Graphics*, 30(Annual Conference Series):65–74, 1996.

[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.

[Hop96] H. Hoppe. Progressive meshes. In *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)*, pages 99–108, 1996.

[KH96] R. Klein and T. Hüttner. Simple camera-dependent approximation of terrain surfaces for fast visualization and animation. In R. Yagel, editor, *Visualization 96*. ACM, November 1996.

[KHK96a] R. Klein, , T. Hüttner, and J. Krämer. Viewing parameter dependent approximation of nurbs-models for fast visualization and animation using a discrete multiresolution representation. In B. Girod, editor, *Herbsttagung '96 3D Bildanalyse und -synthese*, 1996.

[KHK96b] Reinhard Klein, T. Hüttner, and J. Krämer. Meshreduction tools, sep 1996. Demonstration at CeBit 97, Hannover.

[KKL96] Reinhard Klein, J. Krämer, and G. Liebich. Meshreduction tools, sep 1996. Demonstration at the conference Visualization in Biomedical Computing, Hamburg.

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

[KLS96] R. Klein, G. Liebich, and W. Straßer. Mesh reduction with error control. In R. Yagel, editor, *Visualization 96*. ACM, November 1996.

[KS95] 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.

[KS96] R. Klein and W. Straßer. Generation of multiresolution models from cad-data for real time rendering. In W. Straßer, R. Klein, and R. Rau, editors, *Theory and Practice of Geometric Modeling)*. Springer-Verlag, 1996.

[KT96] Alan D. Kalvin and Russel H. Taylor. Superfaces:polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Appl.*, 16(3), May 1996.

[Lie96] G.A.G. Liebich. Erzeugung von oberflächen- und finite element modellen aus tomographiedaten. Master's thesis, WSI/GRIS, Universität Tübingen, Dez 1996.

[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.

[Pup96] E. Puppo. Variable reolution of terrain surfaces. In *Proceedings Eight Canadian Conference on Computational Geometry*, August 1996.

[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.

[RH94] John Rohlf and James Helman. IRIS performer: A high performance multiprocessing toolkit for real–Time 3D graphics. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 381–395. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[RKH96] Daniel Cohen-Or Reinhard Klein and Tobias Hüttner. Incremental view-dependent multiresolution triangulation of terrain, 1996. submitted to Pacific Graphics.

[RR96] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3):C67–C76, C462, September 1996.

[SG96] O. Sudarsky and C. Gotsman. Output-sensitive visibility algorithms for dynamic scenes with applications to virtual reality. *Computer Graphics Forum*, 15(3):C249–C258, September 1996.

[SS96] G. Schaufler and W. Sturzlinger. A three-dimensional image cache for virtual reality. *Computer Graphics Forum*, 15(3):C227–C235, C471–C472, September 1996.

[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.

[XV96] J. C. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In Holly Rushmeier, editor, *Computer Graphics (SIGGRAPH '96 Proceedings)*, volume 30(4), August 1996.