Generation of Multiresolution Models from CAD - Data for Real Time Rendering

Reinhard Klein, W. Straßer

Abstract. A mesh refinement and a mesh simplification algorithm are presented. Both algorithms guarantee a user-defined error tolerance and deliver a multiresolution model. After the computation of the multiresolution model triangulation of the surface patches at variable resolutions can be incrementally generated on-the-fly at rendering time. The resulting triangulations form hierarchical Delaunay triangulations in parameter space.

1 Introduction and Previous work

The visualization of large CAD-models, like cars, trains, aero planes, etc. becomes a major challenge in the context of virtual reality. In most experiments a number 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. To examine the panorama in such a place, not only the train or car themselves have to be visualized but also other cars, pedestrians, buildings, etc. Despite the performance of modern graphics hardware this cannot be realized in real time without special simplifications of the objects in the scene.

The most common way in currently available CAD-systems to represent the boundary surfaces of objects is the use of trimmed NURBS. For visualization purposes the surface patches are approximated by triangle meshes, since triangle meshes are the most widely used representation of models in computer graphics. Planar polygons and in particular triangles are standard rendering primitives of common graphics workstations that can rapidly render polygons.

There is a number of algorithms concentrating on real time rendering of trimmed NURBS surfaces, see for example [17, 24, 21, 1]. The main idea of these algorithms is to preprocess the surface patches in an appropriate way, normally generating simpler subpatches. In the second step the subpatches are uniformly tessellated into a grid of rectangles connected by triangles to points evaluated along the trimming curves of the patches. The polygons defined in the (u, v)-parameter space are transformed into facets in object space by evaluating their vertices with the surface functions, resulting in polygonal meshes in three dimensional space. These methods allow fast rendering of CAD models but have disadvantages.

The computations in the preprocessing step and the algorithms to generate triangles near the boundary of the trimmed patches are complicated. Furthermore, each surface patch is sampled with a constant sampling rate depending on the viewing parameters. In many cases this results in oversampling and many unneccessary triangles. Clipping away parts of a patch that do not belong to the view frustrum is not possible either.

Therefore, in this paper we propose a different strategy:

– To compute in a preprocessing step a complete multi resolution representation of the CAD-model.
– To generate a view dependent triangulation of the model on-the-fly during rendering time combining triangles form different levels of detail into one single mesh.

The use of a dynamic triangulation is not a new idea. There is a number of other authors working on that subject [25, 5, 9, 2, 3, 8]. But the main problem of all the resulting algorithms is the additional storage space for additional data structures needed to combine different levels of detail. While this additionally needed storage space does not lead to problems for small models, it becomes a major problem for large models such as a whole car. A sufficient approximation of such a model which corresponds to the finest level of detail in the multiresolution model consists of millions of triangles. For example an approximation of the boundary surface of a car door model consisting of 446 trimmed NURBS Surface patches up to one millimeter of accuracy needs about 350.000 triangles. In most real-life applications such models cannot be loaded entirely into memory. Moreover, the access time to and from the memory may become a bottleneck.

We show in the following how we can trade storage requirements for computation power in order to generate a multilevel triangulation on-the-fly by using an incremental delaunay triangulation algorithm. The on-the-fly multilevel triangulation reduces storage requirements and the explicit determination of the number of levels.

In the next section we give a brief description of the models we are dealing with. In section 3 we describe how to generate multilevel triangulations. Section 4 deals with the on-the-fly computation of view dependent triangulations. In section 4 we show some results and an overview of our current and future work.

2 Parametric Models

We assume a given boundary representation (BREP) of the model. In general, such a boundary consists of several trimmed parametric patches $f : \Omega \longrightarrow \mathbb{R}^3$, where $\Omega \subset \mathbb{R}^2$ is a planar domain. Most of the currently available CAD-systems support as representation of f NURBS-tensorproduct surfaces.

In the case of a trimmed patch Ω is defined by a set $B = \{b_0, \cdots, b_N\}$ of *boundary polygons*. The set B includes an exterior boundary b_0 and interior boundaries (holes) $b_1, \cdots b_N$. Each boundary b_i consists of a finite number (≥ 3) of oriented curved *domain edges* $e_{i0}, \cdots, e_{in_i}$ and is defined by a set of *boundary nodes* $V \supset V_i = \{v_{i0}, \cdots, v_{in_i}\}$. Every boundary polygon b_i defines a region Ω_i. The interior of these regions $\mathring{\Omega}_i$ is located on the left side of the oriented boundary polygons . In order for Ω to be a well-defined finite domain, the following conditions must hold:

$$\begin{aligned}
&\mathring{\Omega}_i \subset \mathring{\Omega}_0,\ 0 \leq i \leq N,\\
&\mathring{\Omega}_i \cap \mathring{\Omega}_j = \emptyset,\ 1 \leq i, j \leq N,\ i \neq j,\\
&b_i \cap b_j \subset V,\ 0 \leq i, j \leq N,\ i \neq j.
\end{aligned}$$

In most applications the edges are also given by a NURBS representation.

Surface modelers used in the car industry for the design of car hulls often do not provide topological information. Therefore, there is no a-priori knowledge about neighbourhood relations between different surface patches. For visualization purposes this is not a serious problems, since it is sufficient to approximate and visualize each patch independently from the others. The only problem one has to take care of, is that the approximation error along the boundary curves of the patches is always smaller than one pixel in screen space.

In the following we restrict ourselves to cases in which no neighbourhood relationships are known, processing each patch by its own. How to deal with the more complicated case of boundary-conforming approximations is described in [12].

3 Generation of a Multilevel Model

Following, we shortly review a refinement method for the adaptive approximation of trimmed NURBS surfaces [16, 11, 12], that can be used to approximate an initial NURBS surface up to a given error tolerance using a parametric distance between the approximating mesh and the original surface. After that, we show how a simplification method can be used for the same purpose. In this simplification method we use the Hausdorff distance between the approximating mesh and the original surface to control the approximation error. Through this distance measurement higher reduction rates can be achieved. Both methods deliver a multilevel triangulation which can be used to generate adaptive triangulations of CAD-models on-the-fly with controlled approximation errors.

3.1 Pre-sampling

The first step of the refinement and the simplification method is to pre-sample the given patch in such a way that the Delaunay Triangulation of the pre-sampled points $(u_j, v_j) \in \Omega\ j = 0, \cdots, p$ in the parameter-domain delivers a piecewise linear approximation f_S of the surface with a maximum parametric approximation error $d_a(f, f_S) = sup_{(u,v)\in\Omega}||f(u,v) - f_S(u,v)|| < \eta$. In order to build the multilevel model the piecewise linear approximation f_S is used instead of f. f_S has a corresponding Constrained Delaunay triangulation Σ that defines the finest resolution of the multiresolution model. In order to get a unique Delaunay triangulation we assume that no four sample points lie on a circle. Otherwise, the sample points are perturbed slightly.

The main objective of the pre-sampling step is to guarantee a maximum parametric approximation error between f and f_S. If bounds on the second derivative of f are known a regular pre-sampling can be done [7]. In other cases, e.g. for Bézier-TensorproductSurfaces or B-spline-Tensorproduct-Surfaces, the distance between corresponding points of the linear approximation and the control-points can be used, to indicate if the subdivision of a patch is neccessary. This leeds to a quadtree-data structure in the parameter domain.

After the pre-sampling step a Constrained Delaunay triangulation of the sampled points in parameter space is computed. This triangulation is then used to build up a multiresolution model of the surface patch. We denote the triangulation Σ by Σ_M indicating that it contains M vertices.

3.2 Refinement method

The first step of the refinement method is the generation of a coarse piecewise linear approximation Ω_S of the boundary curves of the domain Ω and the computation of an initial Constrained Delaunay triangulation Σ_m of Ω_S containing m vertices on the boundary of Ω_S. This initial static triangulation is refined by an iterative insertion of new points, one at a time. The insertion is based on an incremental insertion of points on the domain Ω. At each iteration, the point p with the maximum approximation error is inserted as a new point and the triangulation is updated accordingly. The refinement process continues until the parametric approximation error between the actual intermediate triangulation Σ_n and the finest triangulation Σ_M is zero.

For the refinement method a parametric distance is used. This simplifies the computation of the point p with maximum approximation error.

The insertion of a single point during the insertion process does not necessarily cause a decrease in the approximation error. However, the convergence of the method guarantees that the approximation will improve after some additional vertices have been inserted. This scheme for adaptive approximation was proposed by several authors (see e.g. Floriani, Falcidieno, and Pienovi [4], Lee and Schachter [18], and Rippa [20]) in other contexts.

The inserted points are stored in a list L ordered by the insertion time. In addition, for each point in the list the maximum approximation error of the reduced triangulation is recorded at the insertion time. Therefore, for each point we can assign an approximation error. Since this approximation error is independent from the view direction, it is called, hereafter, the *geometric error*. We denote by $G_\epsilon(p)$ the function which maps a given point p to its associated global geometric error.

The refinement algorithm can be outlined as follows:

1. Compute the boundary of the surface including trimming curves in parameter space.
2. Generate an initial Constrained Delaunay triangulation Σ_M of the parameter domain of the faces containing the vertices on the boundary-curves and on the trimming-curves so that the corresponding function f_S approximates f up to a predefined error tolerance η.
3. Compute a coarse approximation Ω_S of Ω and a Constrained Delaunay triangulation Σ_m of Ω_S.
4. Start with Σ_m and insert, one at a time, the point causing the maximum geometric error. The point and its associated error are stored in the list L. This continues until the maximum geometric error of the reduced triangulation is zero, or up to another prescribed user-defined error.

A detailed description of this algorithm can be found in [16, 12]. Implementation aspects of the algorithm are described in [10].

3.3 The simplification method

In the following we briefly review our simplification algorithm for arbitrary triangle meshes in 3D [15] and show how this algorithm can be modified to also produce a multiresolution model.

The algorithm One of the main ideas of the simplification method is the use of the Hausdorff distance:
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

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

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 $d_E(X, Y) \neq d_E(Y, X)$. The *Hausdorff distance* 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

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

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 the simplified triangulation is the one a user would intuitively think of.

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

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

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

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

A main 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 the vertex would be removed. In each step we actually eliminate one of the vertices with 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 [22], 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 [23]. In both cases the potential error is set to infinity.

If we remove a vertex v from the triangulation its adjacent triangles are removed and the remaining hole is retriangulated. In addition, for all neighbouring vertices $v_1, \cdots, v_n,\ n \in \mathbb{N}N$ of v the potential errors need to be updated. 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.

Modificaton of the algorithm In the original algorithm [15] for the retriangulation of the remaining hole the adjacent vertices are projected into a plane similar to the algorithm of Turk [23]. If the corresponding polygon in the plane does not self-intersect, the polygon is triangulated using a Constrained Delaunay triangulation.

For parametric surface approximation this complicated procedure is not neccessary. Each triangulation in space has a unique corresponding Constrained Delaunay triangulation in the parameter domain Ω. If a vertex of a triangulation in 3D-space has to be removed, its corresponding point in the domain Ω is removed from the Constrained Delaunay triangulation in parameter space and the 3D-triangulation is updated accordingly instead of retriangulating the remaining hole in 3D-space. In such a way it is guaranteed for each step of the algorithm that the triangulation of the domain is a Constrained Delaunay triangulation. Like in the case of the refinement algorithm the removed points in parameter space are stored in a list L in a descending order by the removal time. In addition, for each point in the list the maximum approximation error of the reduced triangulation is recorded at the removal time.

3.4 The multiresolution model

From the coarsest Constrained Delaunay triangulation Σ_m and the sequence of inserted or removed points $(p_{m+1}, p_{m+2}, \ldots, p_M)$ transforming Σ_m into the triangulation Σ_M at full resolution all intermediate unique Delaunay triangulations Σ_m, $\Sigma_{m+1}, \ldots, \Sigma_M$ can be recomputed through simple point insertion. Therefore, the multiresolution representation of a surface patch only requires a coarse Delaunay triangulation Σ_m of the domain in the XY- plane and the sequence of points transforming Σ_m into the triangulation Σ_M at full resolution. The topology of the triangulation is implicitly given by the use of the Delaunay triangulation and does not have to be stored explicitly. This leads to a massive reduction of the storage costs for the multiresolution model. Since for each point p_n of the sequence the approximation error ϵ_n between the original triangulation Σ_M and the corresponding triangulation Σ_n is additionally stored in the multiresolution

model, we also have knowledge about the corresponding approximation error which is neccessary to guarantee the correctness of the visualized surface object.

The storage cost for the initial triangulation can be considered constant. Thus, the total storage cost for the multiresolution model is $6 * (M - m) + B$, where $6 * (M - m)$ is the number of floating point values of the model (2 cartesian coordinates in parameter space, 3 cartesian coordinates in 3D-space and one error-value) and $B = B(m)$ is the total storage cost for the initial triangulation.

4 Extracting triangulations from different levels of detail

In this section we show how triangulations from different levels of detail can be computed using the multiresolution model. We further show how the simplest triangulation can be computed from triangles of the multiresolution model that satisfies a user-defined resolution function.

4.1 Extracting a triangulation with constant error tolerance

To extract a triangulation that guarantees a user defined error tolerance ϵ we start with the coarsest triangulation Σ_m and stepwise insert points into the corresponding Constrained Delaunay triangulation in parameter space until the geometric error $G(p)$ of the inserted point p is less than ϵ. If we have already extracted a triangulation that guarantees a smaller error tolerance than ϵ it is not neccessary to start again with the coarsest triangulation. Instead vertices are removed from the triangulation, beginning with the vertex with the smallest index (the one which was inserted last).

Since the size of one patch is small compared to the visible part of a complex CAD-model, for many practical visualization purposes it is sufficient to extract each patch with its own constant approximation error. This approximation error depends on the viewing parameters and the bounding box of the patch in 3D-space and can easily be updated after a change of the camera position. We found that for real-world applications, due to the smooth changes of the camera, only a few vertices have to be inserted or removed from the triangulation. Using fast algorithms for inserting and removing points into and from a Constrained Delaunay Triangulation real time performance can be achieved [13, 10].

4.2 Extracting a triangulation at variable resolution

The following notation is adopted from Puppo [19]. We assume that for each triangle in the multiresolution model a Boolean condition $c()$ is defined. $c(t)$ is true if and only if the resolution of t is acceptable. Similarly, the notation $c(T)$ means that all triangles of a triangulation satisfy $c()$.

We consider the following problem:

> *Given a multiresolution model as described above, extract the smallest triangulation T consisting of triangles from possibly different levels of detail, so that $c(T)$ is true.*

Based on the ordered list L such a triangulation can also be rapidly computed without storing additional information, taking advantage of the following observations:

1. If all points p_i of the list L with corresponding maximum geometric error $\epsilon_i > \epsilon$ are inserted into a triangulation, the global geometric error is guaranteed to be less than ϵ.
2. Suppose that a triangle $\Delta(p_i, p_j, p_k)$ of a reduced Delaunay triangulation T at variable resolution in the domain is given. If the circumcircle of $\Delta(p_i, p_j, p_k)$ does not contain any points with corresponding insertion index less than $l = \max(i, j, k)$, the triangle $\Delta(p_i, p_j, p_k)$ belongs to the intermediate triangulation Σ_l and approximates the surface up to an approximation error $G(p_l)$, see Fig. 1. A proof of this observation can be found in [13]. Since such a triangle belongs to an intermediate triangulation it is called *valid*. An intermediate triangulation is called *valid* if all its triangles are valid.
3. A valid triangle $\Delta(p_i, p_j, p_k)$ of a reduced Delaunay triangulation remains in the triangulation until the point with lowest index r greater than $l = \max(i, j, k)$ contained in the circumcircle of $\Delta(p_i, p_j, p_k)$ is inserted into the triangulation.

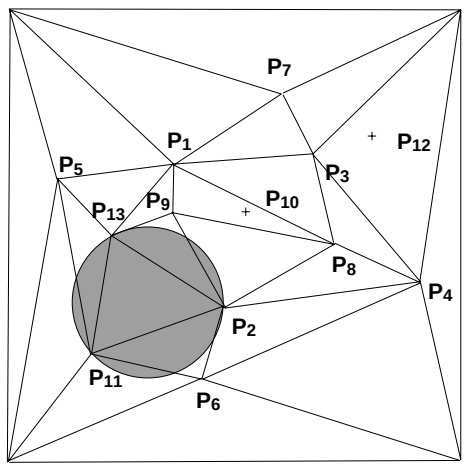

Fig. 1. Since the circumcircle of the triangle $\Delta(p_{13}, p_{11}, p_2)$ does not contain any other point with corresponding insertion index less than $13 = \max(i, j, k)$ the insertion of the points p_{10} and p_{12} will not change the triangle. Therefore, the elevation surface is approximated over the triangle $\Delta(p_{13}, p_{11}, p_2)$ up to a maximum approximation error ϵ_{13}.

This leeds to the following algorithm to compute a triangulation at variable resolution:

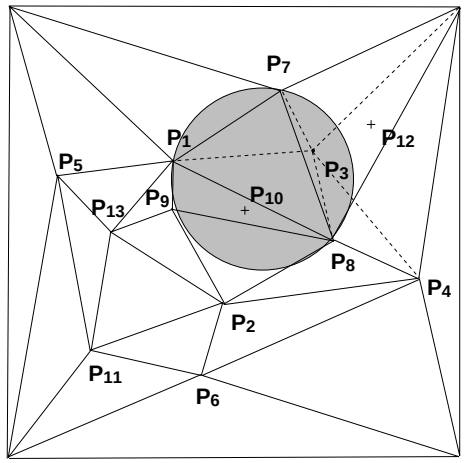

Fig. 2. Since the circumcircle of the triangle $\Delta(p_1, p_8, p_7)$ does contain point p_3 this triangle does not belong to an intermediate triangulation of the multiresolution model. Therefore, in the correction step vertex p_3 is inserted into the triangulation. After the insertion of p_3 the whole triangulation is valid.

1. Start with the coarsest Constrained Delaunay triangulation Σ_m.
2. Put each triangle $\Delta \in \Sigma_m$ onto a Stack U.
3. While U is not empty
   (a) Fetch the first unmarked triangle Δ from the top of U.
   (b) If $c(\Delta)$ is not true
   determine $r := \max(p_i, p_j, p_k)$, where p_i, p_j, p_k are the vertices of Δ and find the point p_r with smallest index s, so that $s > r$ and p_s is contained in the circumcircle of Δ.
   Insert p_s into the Constrained Delaunay triangulation and correct the resulting triangulation.
   Sort all triangles that are generated during the correction step into a List V in descending order. The sorting criteria is the maximum index number of the vertices of the triangle. Mark all triangles that are deleted during the insertion step.

Note that in the average case the insertion of a point into a Constrained Delaunay triangulation can be done in constant time.

The correction step The crucial part of the above algorithm is the correction step. In general, the triangles in the list V created during the insertion of the point p_r into the Constrained Delaunay triangulation do not belong to an intermediate Constrained Delaunay triangulation $\Sigma_m, \Sigma_{m+1}, \ldots, \Sigma_M$. Therefore, no error bounds can be guaranteed for these triangles. Further points have to be inserted until the whole triangu-

lation only consists of triangles belonging to intermediate triangulations $\Sigma_{r_1}, \ldots, \Sigma_{r_n}$ contained in the multiresolution model.

The following algorithm is based on the observations above:

1. While V is not empty fetch the first triangle not marked from the list V and determine the vertex p of the triangle with the highest index. Let us call this index l.
2. For each triangle incident to p containing vertices in its circumcircle with indices less than l, insert that vertex into the triangulation that would be a neighbour of p in a triangulation containing all points with indices less than the index of p, see Fig. 2. As shown in [6, 14] for practical applications this step can be performed in constant time using a uniform grid.
3. Insert all triangles created during the inserting step into list V and mark all triangles deleted during the insertion step.
4. Go to step 1.

Note that due to the ordering of the list V triangles that are generated during the correction step and are incident to the vertex with highest index will remain in the triangulation during the whole correction step, since these triangles do not contain less vertices than the highest index in their circumcircle. Together with observation 2 this proves the validity of the resulting triangulation.

Since we can assume that the insertion of points into the Constrained Delaunay triangulation can be done in linear time, the time complexity of the algorithm is linear in the size of inserted points, and therefore in the size of resulting triangles.

5 Results and future work

We have presented two algorithms for building multiresoltution models for CAD-data. We showed that in addition to the sample points only a set of approximation errors is necessary to compute the camera-dependent approximations in real time using the multiresolution model. We showed furthermore, that for huge data sets, e.g. the model of a whole car the trade of storage costs and storage access time for computing power is worth to be considered.

Our future work focuses onto two parts: First, we think that an easy parallelization of the proposed algorithm can easily be done. Second, we consider how perceptional errors rather than geometric errors can be incorporated into our extraction algorithm.

Acknowledgement

We would like to thank A. Schilling and T. Hüttner for many fruitful discussions. Furthermore, I would like to thank J. Krämer for the implementation of the fast incremental Delaunay triangulation algorithm, which encouraged us in developing the presented algorithm.

References

1. S. S. Abi-Ezzi and S. Subramaniam. Fast dynamic tessellation of trimmed NURBS surfaces. In *Computer Graphics Forum*, volume 13, pages 107–126. Eurographics, Basil Blackwell Ltd, 1994. Eurographics '94 Conference issue.
2. P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution. In R. Scatenied, editor, *Scientific Visualization 95 (Int. Symp. Proc.)*, pages 50–68. World Scientific, 1995.
3. Leila de Floriani and Enrico Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14(4):363–411, October 1995.
4. 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.
5. 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.
6. Tsung-Pao Fang and Les A. Piegl. Delaunay triangulation using a uniform grid. *IEEE Computer Graphics & Applications*, pages 36–47, may 1993.
7. D. Filip, R. Magedson, and R. Markot. Surface algorithms using bounds on derivatives. *Computer Aided Geometric Design*, 3(4):295–311, 1986.
8. 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.
9. H. Hoppe. Progressive meshes. In *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)*, pages 99–108, 1996.
10. 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.
11. R. Klein. *Netzgenerierung impliziter und parametrisierter Flächen in einem objektorientierten System*. PhD thesis, WSI/GRIS, 1995.
12. 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.
13. 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.
14. R. Klein and J. Krämer. Fast algorithms for constructing the 2D-Delaunay-triangulation. Technical Report WSI-1994-16, Wilhelm-Schickard-Institut, Graphisch Interaktive Systeme, Universität Tübingen, 1994.
15. R. Klein, G. Liebich, and W. Straßer. Mesh reduction with error control. In R. Yagel, editor, *Visualization 96*. ACM, November 1996.
16. Reinhard Klein. Linear approximation of trimmed surfaces. In R.R. Martin, editor, *The Mathematics Of Surfaces VI*, 1994.
17. J. Lane and R. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Trans. Pattern Analysis Machine Intell.*, 2(1):35–46, 1980.
18. D. T. Lee and B. J. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal Computer and Information Sciences*, 9(3):219, 1980.
19. E. Puppo. Variable reolution of terrain surfaces. In *Proceedings Eight Canadian Conference on Computational Geometry*, August 1996.

20. S. Rippa. Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scatterd data. *SIAM Journal Sci. Stat. Comput.*, 13(5):1123–1141, September 1992.
21. Alyn Rockwood, Kurt Heaton, and Tom Davis. Real-time rendering of trimmed surfaces. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 107–116, July 1989.
22. 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.
23. Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.
24. Brian Von Herzen and Alan H. Barr. Accurate triangulations of deformed, intersecting surfaces. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 103–110, July 1987.
25. 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.