Incremental View-dependent Multiresolution Triangulation of Terrain

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

Daniel Cohen-Or
School of Mathematical Sciences
Tel-Aviv University, Ramat-Aviv 69978, Israel

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

November 7, 1997

Abstract

A view-dependent multiresolution triangulation algorithm is presented for a real-time flythrough. The triangulation of the terrain is generated incrementally on-the-fly during the rendering time. We show that since the view changes smoothly only a few incremental modifications are required to update the triangulation to a new view. The resulting triangles form a multiresolution Delaunay triangulation which satisfies a predetermined view-dependent error tolerance. The presented method provides a guaranteed-quality mesh since it has control over the global geometric approximation error of the multiresolution view-dependent triangulation.

1 Introduction

Triangular meshes are currently the most widely-used representation of terrain models in computer graphics. Planar polygons and triangles in particular are standard rendering primitives of common graphics workstations that can rapidly render polygons. Terrain data is usually reconstructed by photogrammetric modeling techniques and other acquisition techniques which produce many more triangles than necessary to visualize the terrain. The literature is rich in algorithms that reduce the number of triangles, by removing redundant vertices such that the simplified mesh satisfies some error tolerance [13].

In applications which require high fidelity visualization of a terrain, millions of triangles are needed. Even with recent progress in graphics hardware performance, it is impossible to achieve real-time rendering rates of such a large number of triangles. Efficient use of the number of triangles that are fed into the geometric pipeline is an important consideration [20]. Clipping out triangles which fall beyond the viewing frustum is necessary, as well as the use of levels-of-detail (LOD) [7, 9, 21]. Low levels approximate the model more coarsely with fewer details, while higher levels contain finer details. The appropriate level is rendered according to the distance of the terrain section from the view point. Since coarse approximations contain fewer triangles, a large terrain area can be rendered faster with no visible penalty.

Usually the levels-of-detail are generated off-line in a preprocessing stage [6, 8, 21, 5, 19]. Since in general the distance of the observer to the terrain is less than the extends of the terrain itself, different levels-of-detail are necessary in different areas of the terrain. Such a multiresolution representation of the terrain should maintain spatial continuity, that is, the combination of different levels of detail should be seamless and leave no gaps or holes. In addition, triangles with sharp angles (slivers) should be avoided in all levels of the hierarchy. Thus, a Delaunay terrain triangulation [6, 10] is considered to be a

guaranteed-quality mesh since it maximizes the minimal angles of its triangles. De Berg and Dobrindt [5] proposed a hierarchy of levels of detail that uses Delaunay triangulation at each level. Their method allows the different levels of detail to be combined in the same scene. Cohen-Or and Levanoni [3] have modified their method to form a tree structure which enables a top-down traversal of the Delaunay hierarchy with a fast culling mechanism.

Hoppe [19] introduced another multiresolution scheme, the *progressive meshes*, which can also be used for terrain modeling (but not for Delaunay terrain triangulations). The basic idea of this approach is to simplify the original meshes by successive edge collapse transformations. 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 neighbouring vertices are necessary. In this a way the whole hierarchy of intermediate levels of detail can be stored in a space-efficient way.

However, a selective refinement cannot be combined directly with the mesh compression described in the paper, since during selective refinement the whole topology information is needed. Furthermore, 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 terrain are known.

Puppo [19] describes a general model for the multiresolution decomposition of planar domains into triangles. His method is based on a collection of fragments of plane triangulations arranged into a partially ordered set. Different levels of detail can be obtained by combining different fragments of the model. A similar approach is used by Cignoni et al. [1]. In both approaches the topology of the hierarchy is stored explicitly, with no data compression. Since typical terrain models are extremely large, data compression is vital to enable storage of the model in the main memory of the workstation.

In the approach proposed by Lindstrom et al. [19], the original mesh is simplified in an on-line process during the rendering stage. In each simplification step a screen-space error resulting from the simplification is computed. Despite the fact that errors may accumulate the authors claim that for empirical data this effect is negligible. Interactive frame rates can be achieved by a compact and efficient quadtree. The simplicity of this data structure does, however, have a drawback. To approximate an arbitrary straight line the quadtree must be subdivided along the line up to the maximum level. For example, if the terrain contains small cliffs, just higher than the allowed error, numerous redundant triangles are generated.

In this paper we show how a multiresolution Delaunay triangulation can be generated on-the-fly by an incremental algorithm. As opposed to the data structures proposed in the above approaches, on-line multiresolution triangulation avoids the storage requirements of the hierarchy and the explicit determination of the number of levels. This paper extends our preliminary work presented in [14, 15]. The terrain triangulation is updated dynamically as a function of the camera position and camera parameters, and is thus called a *view-dependent triangulation*. The proposed method inserts and deletes vertices based on an incremental Delaunay triangulation of points in the plane [16]. In [15] a *bottom up strategy* is used, where similarly to Puppo [19], the computation of the view-dependent triangulation starts with the coarsest triangulation. Then, for each triangle a screen-space error is computed. If this error is larger than one pixel, an additional point of the Delaunay hierarchy is inserted to refine the proximity of the triangle. This process is repeated until the screen-space error of all triangles is small enough and therefore visually negligible. Although in a realistic fly-through there are regions where the triangulation doesn't change, this approach necessarily requires all the triangles to be checked.

Dynamic triangulation is not a new concept. Independently to our work [14, 15], others [7, 19, 19, 18] have used a view-dependent triangulation. However, they do not guarantee a geometric approximation error, and their approximation to the optimal triangulation is crude.

2 A View-dependent Error Function

A terrain is described mathematically by a bivariate elevation function $h : D \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ defined over a rectangular domain D in the XY plane. This function is sampled at a finite set of points $P = \{p_1, \ldots, p_n\} \subset D$, forming the vertices of a triangulation approximation. This piecewise linear representation is known as a *triangulated irregular network* (TIN). A simple and common measure of the approximation error between the elevation function h and the approximating TIN is the maximum vertical distance between the two representations. In the following this error is called *geometric approximation error*. The maximum geometric approximation error that cannot be perceived by an observer is called *allowable error*. This error is view-dependent and a function of the camera position and camera parameters. For its computation not only the distance between camera position and a location in the XY-plane has to be considered but also the height of the terrain at that position. In a number of previous approaches the terrain topology was ignored. The allowable error allows a geometric approximation error ϵ such that it does not project onto the screen to a size larger than one pixel (see also [19, 14]). Denote the pixel size on the viewing plane by τ, the relation between ϵ and τ is estimated by:

$$\epsilon \leq \frac{cos(\alpha)\ (f + d_v)\ \tau}{sin(\beta - \alpha)\ f} \qquad (1)$$

where α denotes the angle between the viewing direction d and v (as in Figure 1), the pitch angle β, and on f the focal distance and d_v the distance between a terrain point p and the viewing plane. (see Figure 1).

This expression defines the allowable approximation error for each sample point $p_i \in P$ as a function of its height $h(p_i)$ and the camera position and camera parameters.

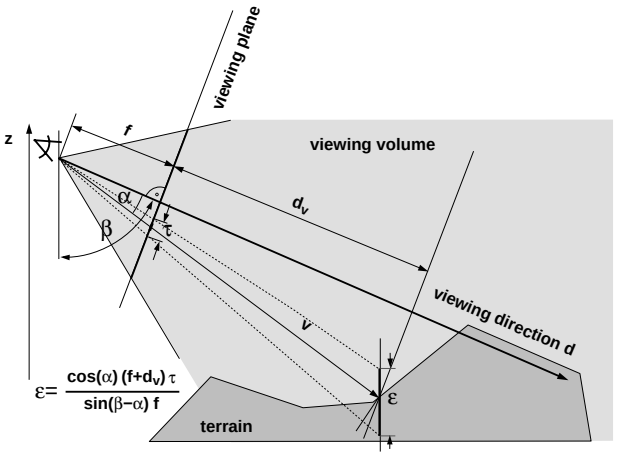

Figure 1: The maximum allowed error tolerance ϵ for the approximation of the topographic surface depends on the distance between the surface and the observer, on the angle α between the viewing direction d and the vector v between surface point and observer, on the pitch angle β, and on the pixel size τ in the viewing plane.

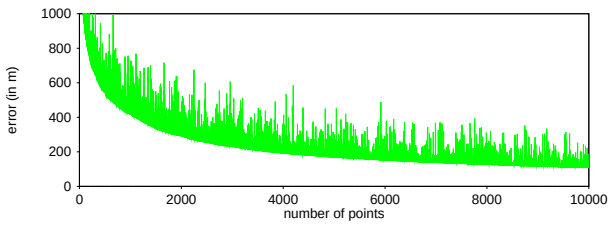

Figure 2: The insertion of a single point during the insertion process does not necessarily causes a decrease in the global geometric approximation error.

3 The algorithm

In a preprocessing step a view independent approximation of the terrain is computed, yielding a Delaunay triangulation of the entire set of points. This approach was first described by [11]. After that, the "history" of the triangulation is used by the on-line algorithm to compute the view-dependent triangulations.

3.1 The off-line view independent triangulation

Let $\epsilon \geq 0$ be a global tolerance value for the whole data set, P be a finite set of points in the XY-plane, and h be the terrain elevation function. We start with an initial

constrained Delaunay triangulation, denoted by Σ_0. This is a regular triangulation of an arbitrary initial set of vertices, which remains static throughout the on-line process. This initial set of vertices contains at least the four corner points of the rectangular array. It may also contain vertices and edges that should be contained in all the levels of details. For example, ridges of mountains, coasts of rivers or the borders of highways. The initial static triangulation is refined by an iterative insertion of new points, one at a time. The insertion is based on an incremental Delaunay of points on the XY domain [11, 16]. At each iteration, the point p which causes the maximum global geometric approximation error is inserted as a new point and the triangulation is updated accordingly. After the insertion of m points we call the corresponding triangulation Σ_m, indicating that it contains m additional vertices. The refinement process continues until the global geometric approximation error is less than the predefined ϵ. The corresponding final triangulation is called Σ_N. As illustrated in Figure 2, 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 improves after some other vertices have been inserted.

The N inserted points transforming Σ_0 into Σ_N are stored in a sorted list L ordered by the insertion time. In addition, for each point p_n in the list the maximum geometric approximation error of the reduced triangulation Σ_n is recorded at the insertion time. Thus, each point p_n corresponds to a global geometric approximation error ϵ_n. In other words, if all vertices of the list L up to a certain point p_n are inserted into the initial triangulation, then the global geometric approximation error of Σ_n is ϵ_n. We denote by $G_\epsilon(p_n)$ the function which maps a given point p_n to its associated global geometric approximation error ϵ_n.

Given the initial Delaunay triangulation Σ_0 and the sequence of points $(p_1, p_2, \ldots, p_N)$, all the unique Delaunay triangulations $\Sigma_1, \ldots, \Sigma_N$ can be reconstructed via incremental point insertions. Therefore, the multiresolution representation of a terrain requires only an initial Delaunay triangulation Σ_0 of the domain in the XY-plane and the sequence of points transforming Σ_0 into the model Σ_N at full resolution. The topology of the triangulation is given implicitly by the use of the Delaunay triangulation and does not have to be stored explicitly. This leads to a massive reduction in the storage costs of the multiresolution model. It is important to point out that with respect to the storage requirements the storage cost is superior to all other techniques described above, including the progressive meshes. These techniques may considerably reduce the storage requirements for the representation of the topology, but they still have to store it explicitly.

During the on-line process the geometric error of each point needs to be translated to its allowable view-dependent error by Eq. 1. One way to achieve this goal is to compute the allowable error for each triangle in the current triangulation Σ_i and to refine the triangulation locally in the neighbourhood of the triangle. This approach is used in [15] and in [19]. However, this necessarily requires that for every frame, even for small camera movements, all the triangles must be considered. This cause many redundant checks as for small camera changes the allowable error of the vast majority of the triangles does not change.

To reduce the number of unnecessary checks and to accelerate the computation of the allowable error the points $p_1, \ldots, p_n$ are partitioned into cells. The minimum and maximum elevation values of all points in a cell define a bounding box which is used to compute an allowable error ϵ for the whole cell, (see Figure 3). As can be seen in Eq. 1 the allowable error at a position p in the direction of the height z depends on the given focal size f, on the angle β, the difference between the angles α and β, and the distance to the viewing plane. For $\alpha \equiv \beta$, the allowable error is unbounded. Increasing α while keeping constant the distance d_v to the viewing plane, decreases the allowable error. Therefore, to find an estimate for the representative error of the bounding box, the distance d_v of the cell's bounding box to the viewing plane and the maximum angle $\alpha(p_i) - \beta$ of all corner points p_i of the bounding box are needed. The distance d_v and the maximum angle $\alpha - \beta$ are computed by projecting the corner points p_i of the bounding box to the viewing plane and taking the minimum of the distances between the original points p_i and the projected ones and the maximum of the angle $\alpha(p_i) - \beta$, respectively. Note that since all bounding boxes have the same orientation, the same corner of the bounding boxes has the minimum distance to the viewing plane. As a byproduct, partitioning into cells also accelerates the access to the points and offers a fast culling mechanism. The preprocessing stage can be summarized by the following steps:

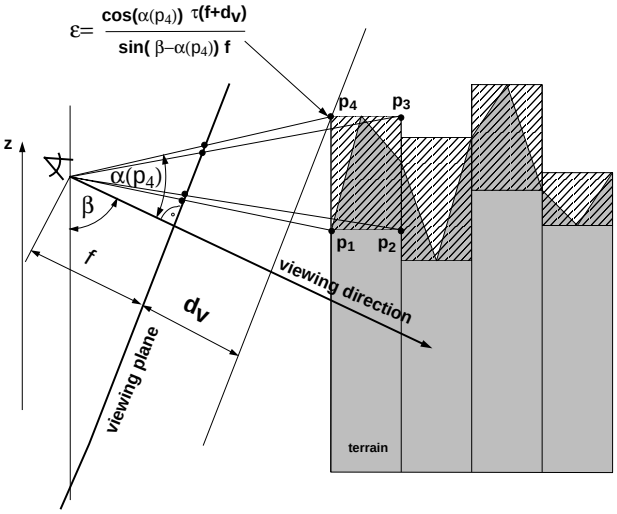

Figure 3: The elevation values define bounding box for each cell. Using the smallest distance d_v to the viewing plane and the maximum angle $\beta - \alpha$ the allowable error ϵ for the cell is given by $\frac{cos(\alpha(p_4))\ (f+d_v)\ \tau}{sin(\beta-\alpha(p_4))\ f}$.

1. Define an initial triangulation Σ_0.
2. Start with Σ_0 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 approximation error of the reduced triangulation is zero, or another prescribed user-defined error.
3. Choose an appropriate regular grid which bounds all the sample points $p \in P$. Distribute all the points $p \in P$ into the grid cells according to their XY coordinates.
4. Sort the points in each grid cell according to their corresponding global geometric approximation error and store them in local lists.

3.2 The on-line view-dependent triangulation

The computation of the view-dependent triangulation is based on the ordered list L, the view-independent global geometric approximation errors and the view-independent global geometric approximation errors of the grid cells. For a given frame, first, the view-independent global geometric approximation errors for the grid cell are computed. Then in the second step, the points of each grid cell which have a corresponding global geometric approximation error that exceeds the view-independent error of the cell itself are inserted into the current constrained Delaunay triangulation. This can be done very efficiently since the points in each grid cell are sorted according to their global geometric approximation error. In practice, in most of the cells, no further points are inserted into the triangulation.

Unfortunately, the resulting constrained Delaunay triangulation $\mathcal{T}$ may contain triangles that do not belong to any of the triangulations $\Sigma_0, \ldots, \Sigma_m$. Since in the preproccessing step only for triangles of the intermediate triangulations $\Sigma_0, \ldots, \Sigma_m$ a global geometric approximation error was computed, the geometric approximation in the area of these new triangles is unknown. Note that in the worst case the amount of this error is in the same order of magnitude as the height of the terrain. Therefore, in a following *correction step* further points of the list L are inserted into the triangulation until all triangles of the resulting triangulation belong to one of the triangulations $\Sigma_0, \ldots, \Sigma_N$.

This further insertion step is based on the following observation: Let $\Delta(p_i, p_j, p_k)$ be a triangle of the triangulation $\mathcal{T}$. If the circumcircle of $\Delta(p_i, p_j, p_k)$ does not contain any points with corresponding insertion index less than $m = \max(i, j, k)$, then the triangle is contained in the intermediate triangulation Σ_m and approximates the terrain surface up to an approximation error ϵ_m, (see Figure 4).

Proof: Since none of the points already contained in the constrained Delaunay triangulation $\mathcal{T}$ with corresponding insertion index greater than $m = \max(i, j, k)$ belongs to the circumcircle of triangle $\Delta(p_i, p_j, p_k)$, their removal from $\mathcal{T}$ will not influence the triangle itself. Furthermore, due to the assumption that none of the points with corresponding insertion index less than m belongs to the circumcircle of triangle $\Delta(p_i, p_j, p_k)$, inserting all these points with corresponding insertion index less than m will also not influence the triangle. Therefore, this triangle must belong to the unique constrained Delaunay triangulation containing all points up to p_m, i.e., to Σ_m. Due

to the first observation above this triangulation approximates the initial triangulation with an approximation error less than or equal to ϵ_m.

The above implies that we have to check the circumcircle of each new triangle to insert more points if needed. It also shows that a new triangle can cause points to be inserted in its proximity only, and thus the correction operation is local.

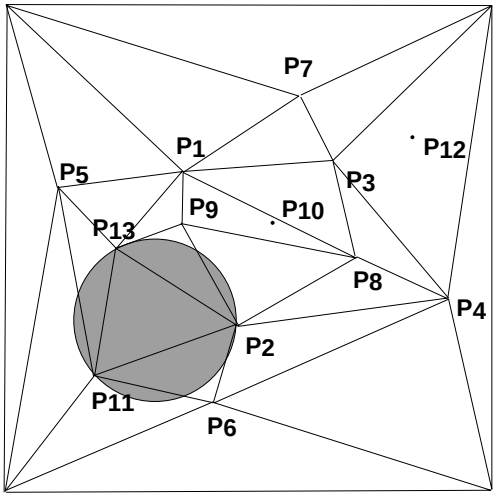

Figure 4: Since the circumcircle of the triangle $\Delta(p_{13}, p_{11}, p_2)$ does not contain any other point with corresponding insertion index less than 13, the triangle is contained in Σ_{13} and it approximates the terrain surface up to a approximation error ϵ_{13}.

The on-line view-dependent triangulation is based on the above and can be stated as follows:

1. For each grid cell C_j compute the maximum allowed approximation error ϵ_j by Equation 1. This defines a threshold for each point.
2. Remove from the current constrained Delaunay triangulation all points $p_k \in C_j$ with $G_\epsilon(p_k) < \epsilon_j$.
3. Insert into the constrained Delaunay triangulation all points $p_k \in C_j$ with $G_\epsilon(p_k) > \epsilon_j$.
4. Start with the point p_m which has the greatest index m among all corner points of the new triangles generated during the second and third step. Check if the triangles adjacent to p_m contain points in their circumcircle that have an insertion index less than m. If such points are found, insert all points p_l into the current triangulation that are adjacent to p_m with respect to the triangulation Σ_m. This guarantees that all the triangles adjacent to p_m will belong to the triangulation Σ_m.

Note that the global geometric approximation error is independent to the view direction. Furthermore, this global geometric approximation error is also a local error: Before inserting the point p_n into the triangulation the maximum global geometric approximation error ϵ_{n-1} was caused by a point in the interior of the influenced area of p_n. The influenced area of p_n consists of the set of triangles which are deleted and replaced by a new set of triangles during the insertion of p_n.

Although this approach accelerates the update process between two consecutive frames compared to the bottom up strategy described in [15], it may happen that by this algorithm a small number of points are inserted into the resulting triangulation that are not necessary to guarantee the view-dependent allowable error. This is because the insertion of a single point during the insertion process in the processing stage does not necessarily causes a decrease in the global geometric approximation error. Using a bottom up strategy [15, 19], the insertion of points stops regardless of whether further insertion of points would increase or decrease the geometric approximation error. In this approach in each grid cell all the points whose geometric approximation error exceeds the geometric approximation error of the cell itself are inserted (see Figure 5).

The algorithm is advantageous for real-time rendering due to the frame-to-frame coherence in the temporal domain. Only a very small number of vertices has to be inserted and removed from frame-to-frame, and the triangulation can be updated incrementally on-the-fly (see section 5).

4 Real-time performance

4.1 Temporal continuity

Real-time rendering of levels of detail causes a noticeable temporal aliasing when the transition between different

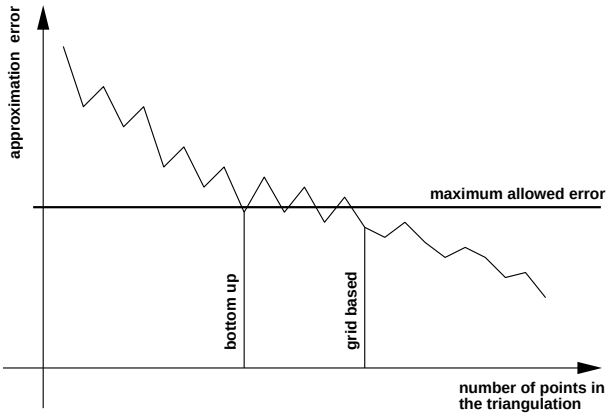

Figure 5: In a bottom up strategy the insertion process stops regardless of whether further insertion of points would increase or decrease the geometric approximation error. Thus, it achieves the minimum number of required points. In our approach all points of a grid cell whose geometric approximation error exceeds the allowable error computed for the cell itself are inserted.

levels is sharp. Therefore, the transition between different triangulations must be handled in a way that appears to be smooth, so that the temporal alias effect will not be noticed [2]. Hoppe [19] coined the term *geomorph* for the continuous transition between different geometries. In his method the transition between two triangulations is a split of a vertex and its continuous transformation to an edge, or an edge collapse into a vertex. This elegant solution cannot be adopted for the incremental insertion of a vertex into a Delaunay triangulation. In [3] a method based on object blending visually softens the transition between two levels of Delaunay triangulation. The transition between two Delaunay triangulations consists of a series of edge collapse transformations. The method can also handle the insertion and removal of multiple vertices simultaneously.

4.2 Minimum frame rate

In visual applications there is always a need to balance the imaging quality and the frame rate. In interactive on-line systems one is required to maintain a user-specified minimal frame rate. In [12, 17] algorithms were proposed to adjust the image quality adaptively by choosing the level-of-detail and rendering algorithm according to its estimated rendering cost.

Since our multiresolution triangulation is incremental, in most cases the mesh generation time is insignificant in comparison to the rendering time. If the available rendering hardware is not fast enough to render a given triangulation in real-time, it is easy to adjust the error tolerance and trade the mesh accuracy for a coarser and "lighter" triangulation.

However, there is a need to guarantee that the new frame requires only a small incremental update of the mesh. For most practical flying trajectories the image footprint does not change much, unless there is a sharp rotational transition of the viewing direction (the yaw angle). A fast rotation of the yaw angle causes the image footprint to cover large areas which need to be refined for the new frame. The new areas can be displayed coarsely for a few frames until the mesh is updated. Instead we prefer to maintain a wider footprint around the viewer location which can "absorb" fast rotational changes of the viewing direction [4]. Although the footprint is wider, not all its triangles are fed into the graphics pipeline, but only those which are in the viewing frustum. Maintaining a wider footprint increases the storage cost of the dynamic mesh, but this size is usually insignificant.

4.3 Storage and traffic

A real-time flythrough must also deal with other important issues, i.e, storage space and traffic overhead. In most real-life applications the terrain size is very large and cannot be loaded entirely in the main memory. Moreover, the access time to and from the memory may become a bottleneck unless some culling mechanism is employed which avoids redundant data movement.

In the off-line stage the terrain data is converted to a linear list of vertices and error values. As mentioned above, it avoids the explicit storage of the mesh topology, and the data storage is thus compact. Assuming the list is stored on the disk and the dynamic mesh in main memory, the traffic between the disk and the memory is slight. Note that the access to the vertices in the list is fast due to the regular partition of the domain into cells. Since each cell contains a relatively short list the search is fast. More-

over, since the list is sorted by the error values, adjacent accesses in time are likely to be close along the list. Thus, for each list we maintain a pointer to the most recently accessed vertex to further reduce the search.

5 Results

As a first test dataset we used the elevation data of the Grand Canyon area. The original data set contains 1.440K vertices. We started with an initial triangulation of four corner vertices of the original rectangular data and inserted the rest of the sampled points as described above up to a geometric error of 5 meters. This results in an initial triangulation of about 490K vertices, and about 950K triangles.

For the flythrough we assumed a velocity of about 700 km/h. At this velocity it would take about 13 minutes to fly from one corner to the opposite corner of the terrain area. To realize a frame rate of 25 frames/sec at this velocity, the triangulation must be updated every 7.6 meters. The focal size of the camera is 50 mm, which is equivalent to a camera with standard objective. The height of the flight over the terrain was between 700m and 5000m. Different levels of approximation are shown in Figure 6.

The two images in Figure 7 shows the camera over the adaptive TIN of the Grand Canyon area. The eyepoint is visualized with a sphere, the two bars mark the boundaries of the viewing volume. Note that due to the dependencies in the hierarchy between different levels of detail in the multiresolution triangulation, there is still an excess of some triangles behind the camera.

The two images in Figure 8 show the Grand Canyon area with a simple texture calculated based on the height field from the same positions as in Figures 6 and 7.

As a second test data set we used data of an area in southern Germany. The data was provided by the 'Landesvermessungsamt, Baden-Württemberg, Germany'. The original data set contains about 30K vertices and 60K triangles. Figure 9 shows the path of the fly. It is 15.7 km long and the flight time with a velocity of 700 km/h is about 1 min 18 sec. Figure 10 shows the time needed to adapt the triangulation from one frame to the next one.

The frame-to-frame update rate for the TIN depends on the velocity of the camera, the distance between the camera and the terrain, the focal size and the resolution of the image. In Table 1 the average data for the frame-to frame update time and the number of inserted and removed vertices between two frames are listed for a small part of the flight at a height of about 1900 meters over the terrain. The update times were measured on an Indigo with a 150MHz R4400 Processor. At an image resolution of 200x200, an update rate of 25 updates/sec (without rendering) can be achieved. We want to emphasize that at this resolution only about 6000 triangles are necessary to represent the terrain. Assuming a frame rate of 25 frames/sec only 150000 triangles/sec should be rendered. Such a rendering performance is already available on small workstations or even on low cost PCs.

6 Conclusions

We have presented an algorithm for incremental view-dependent triangulation of terrain surfaces. We showed that except for the sample points only a set of approximation errors is necessary to compute the view-dependent triangulation on-the-fly. The dynamic triangulation is view-dependent and satisfies the error defined by eq 1.

The proposed technique offers a multiresolution representation without explicitly storing any hierarchy. Furthermore, it shows that it is worth to tradeoff storage costs and storage access time for computing power for huge terrain data sets.

7 Acknowledgment

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

References

[1] 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.

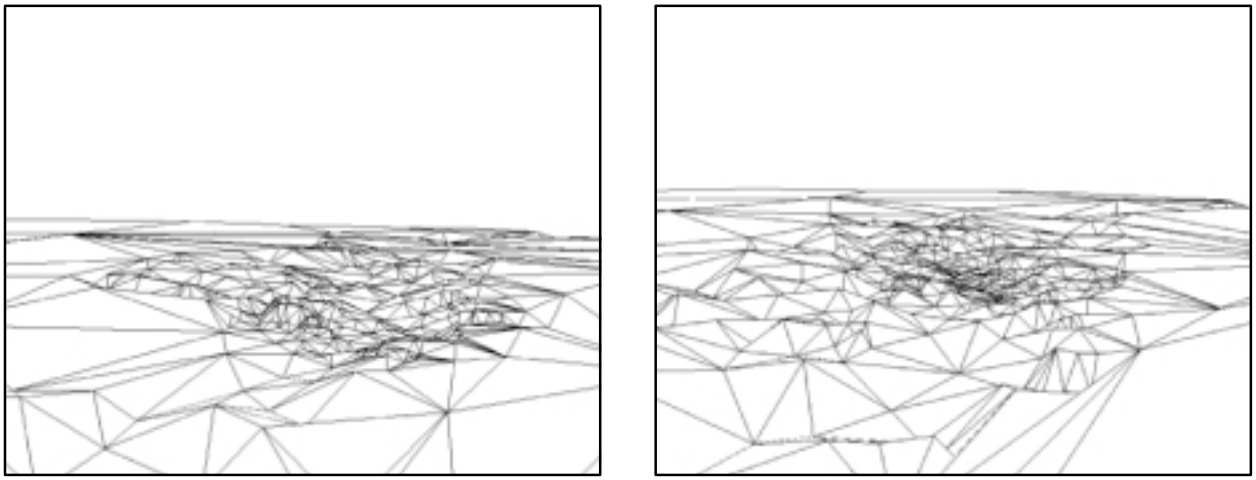

Figure 6: Camera dependent TINs of the Grand Canyon area. In the hand right picture the viewpoint has moved in the direction of the canyon.

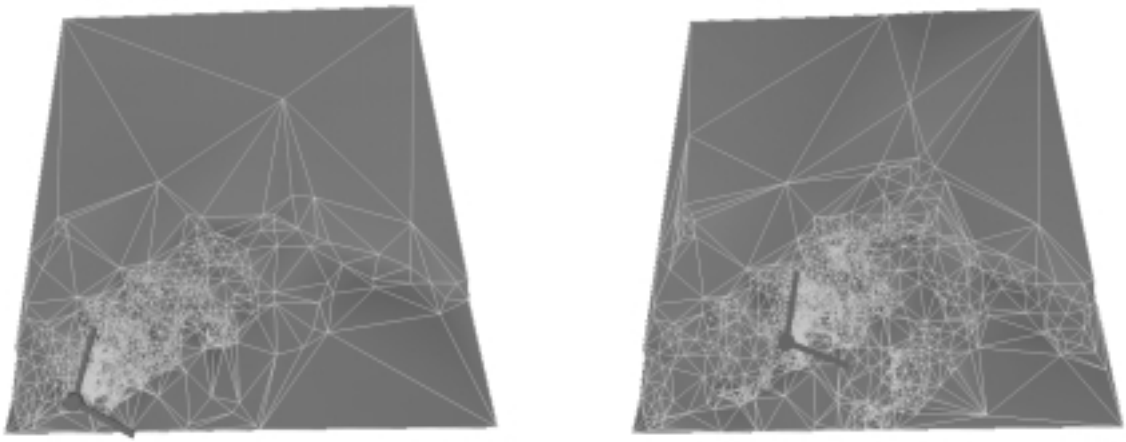

Figure 7: Camera dependent TINs with different camera positions.

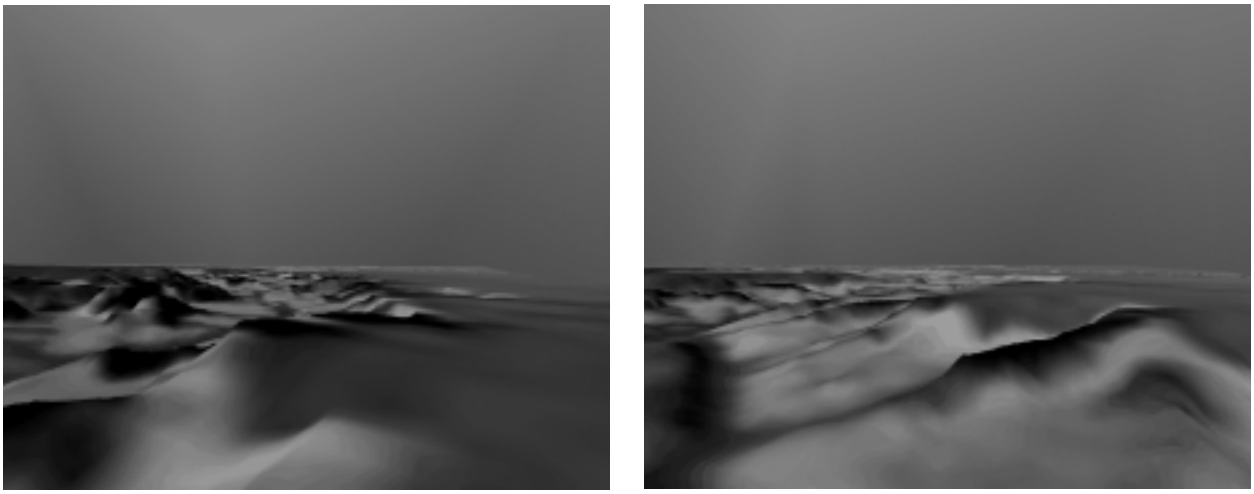

Figure 8: Camera dependent TINs with different camera positions.

image resolution (pixel)	200 x 200	400 x 400	720 x 576
vertices inserted per frame	2.5	3.5	10
vertices removed per frame	4	5	12
vertices inserted(correction step) per frame	1	1.5	2.5
triangles	6200	20500	42500
insertion (msec)	17	22	65
removal (msec)	20	30	50
correction step (msec)	6	14	40

Table 1: Update statistics for a part of the flight over the Grand Canyon area with about 700 km/h.

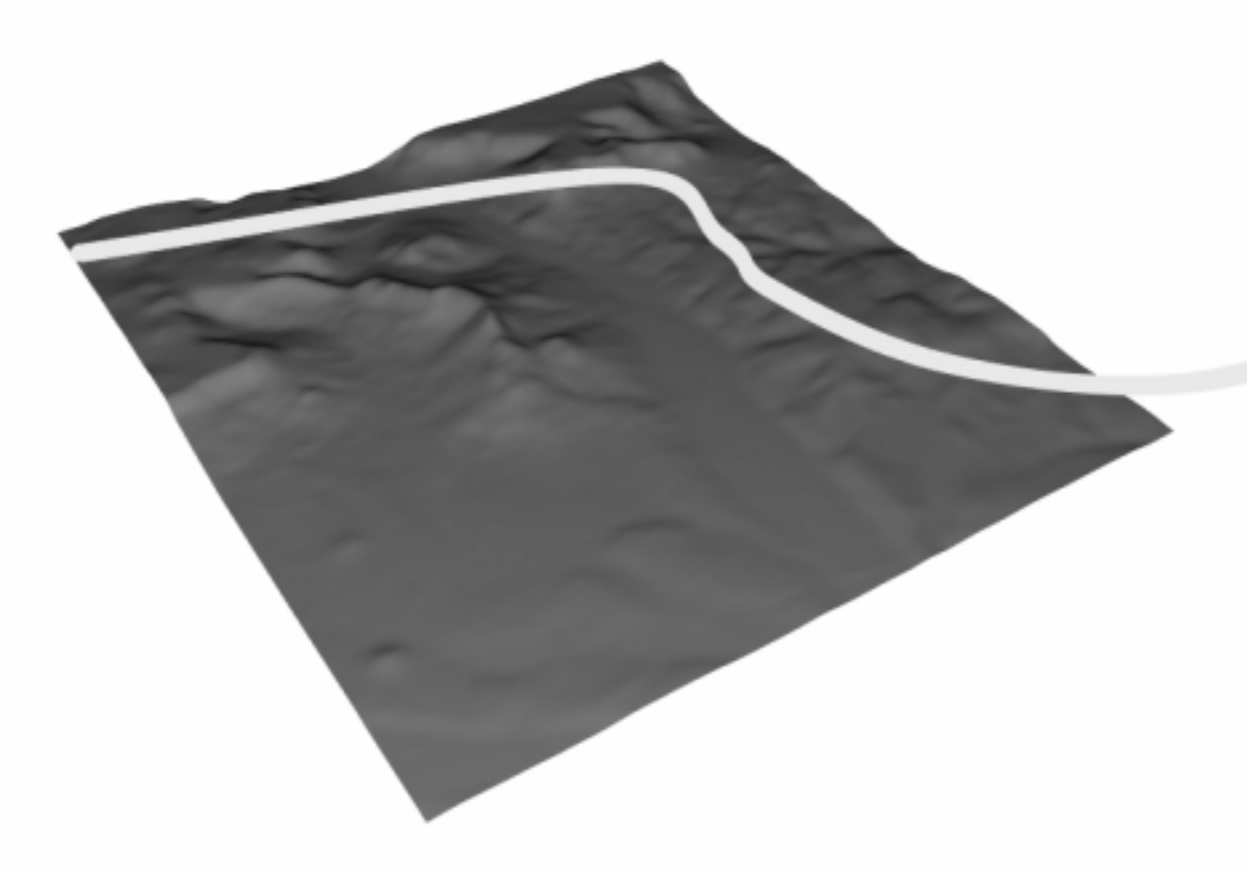

Figure 9: The flight path over the Teck-area in southern Germany.

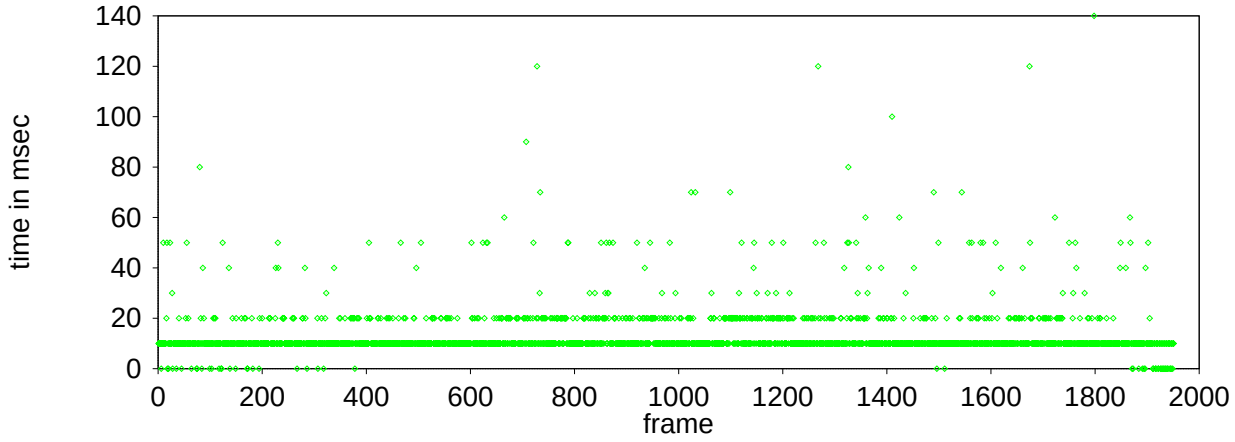

Figure 10: Time needed to adapt the triangulation from one frame to the next one during the flight over the Teck-area. The flight path is shown in Figure 9.

[2] D. Cohen-Or. Exact antialiasing of textured terrain models. *The Visual Computer*, 13(4):184–198, June 1997.

[3] Daniel Cohen-Or and Yishay Levanoni. Temporal continuity of levels of detail in Delaunay triangulated teuain. In Roni Yagel and Gregory M. Nielson, editors, *Proceedings of the Conference on Visualization*, pages 37–42, October 27–November 1.

[4] Daniel Cohen-Or, Eran Rich, Uri Lerner, and Victor Shenkar. A real-time photo-realistic visual flythrough. *IEEE Transaction on Visualization and Computer Graphics*, 2(3):255–264, September 1996.

[5] Mark de Berg and Katrin Dobrindt. On levels of detail in terrains. In *Proc. 11th Annual ACM Symp. on Computational Geometry*, Vancouver, B.C., June 1995.

[6] Leila De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Applications*, 9(2):67–78, March 1989.

[7] Michael J. DeHaemer, Jr. and Michael J. Zyda. Simplification of objects rendered by polygonal approximations. *Computers and Graphics*, 15(2):175–184, 1991.

[8] John S. Falby, Michael J. Zyda, David R. Pratt, and Randy L. Mackey. NPSNET: Hierarchical data structures for real-time three-dimensional visual simulation. *Computers and Graphics*, 17(1):65–69, January–February 1993.

[9] R. L. Ferguson, R. Economy, W. A. Kelley, and P. P. Ramos. Continuous terrain level of detail for visual simulation. In *Proceedings of the 1990 Image V Conference*, pages 145–151. Image Society, Tempe, AZ, June 1990.

[10] L. De Floriani, P. Marzano, and E. Puppo. Hierarchical terrain models: survey and formalization. In *Proceedings SAC'94*, pages 323–327, Phoenix (AR), March 1994.

[11] R. J. Fowler and J. J. Little. Automatic extraction of irregular network digital terrain models. *Computer Graphics*, 13(3):199–207, August 1979.

[12] T. A. Funkhouser and C. H. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Computer Graphics*, 27(Annual Conference Series):247–254, 1993.

[13] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report, CS Dept., Carnegie Mellon U., to appear.

[14] 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.

[15] 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.

[16] Dani Lischinski. Incremental Delaunay triangulations. In Paul S. Heckbert, editor, *Graphics Gems IV*, pages 47–59. Academic Press, 1994.

[17] Paulo W. C. Maciel and Peter Shirley. Visual navigation of large environments using textured clusters. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 95–102. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7.

[18] B. Rabinovich and C. Gotsman. Visualization of large terrains in resource-limited computing environments. In *IEEE Visualization '97*, pages 95–102, October 1997.

[19] J. Rohlf and J. Helman. IRIS Performer: A high performance multiprocessing toolkit for real-time 3D graphics. *Computer Graphics*, 28(Annual Conference Series):381–394, 1994.

[20] Lori Scarlatos and Theo Pavlidis. Hierarchical triangulation using cartographic coherence. *Computer Vision, Graphics, and Image Processing. Graphical Models and Image Processing*, 54(2):147–161, March 1992.

[21] 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.