REPRESENTATION AND VISUALIZATION OF TERRAIN SURFACES AT VARIABLE RESOLUTION

P. CIGNONI
Istituto Elaborazione dell'Informazione – Consiglio Nazionale delle Ricerche
Via S. Maria, 46 - 56126 Pisa, ITALY
E-mail: cignoni@iei.pi.cnr.it

E. PUPPO
Istituto per la Matematica Applicata – Consiglio Nazionale delle Ricerche
Via dei Marini, 6 (Torre di Francia) - 16149 Genova, ITALY
E-mail: puppo@ima.ge.cnr.it

and

R. SCOPIGNO
CNUCE – Consiglio Nazionale delle Ricerche
Via S. Maria 36 - 56126 Pisa, ITALY
E-mail: r.scopigno@cnuce.cnr.it

25th February 1997

ABSTRACT

We present a new approach for managing the representation of discrete topographic surfaces at variable resolution, which is based on a unified model encoding a history of either refinement or simplification of a triangulation decomposing a plane domain. An efficient data structure is proposed, which is obtained by interpreting the model containing all triangles of the history as a cell complex embedded in three-dimensional space. A major feature of the model is the ability to provide efficiently a representation of the surface at resolution variable over the domain, according to an application-dependent threshold function. Experimental results on real world data are presented, and applications to flight simulation are discussed.

Keywords: digital terrain modeling, surface simplification, variable resolution representations, terrain visualization.

Address to which proofs should be sent:
R. SCOPIGNO, CNUCE – Consiglio Nazionale delle Ricerche
Via S. Maria 36 - 56126 Pisa, ITALY.

1 Introduction

The search for multiresolution representation schemes of spatial entities has recently become very popular. Multiresolution models offer the possibility to represent and manipulate descriptions of spatial entities at different levels of detail and accuracy, depending on the needs of each specific application. Since the size of a description is somehow proportional to its resolution, the main advantage of a multiresolution scheme is speedup in processing because of data reduction, whenever and wherever a representation at low resolution is adequate to the needs of a given application.

Major applications of multiresolution models involve modeling generic surfaces embedded in 3D space (Von Herzen and Barr, 1987; Dyn et al., 1990; Taylor and Barrett, 1994), topographic surfaces in the context of Geographical Information Systems (De Floriani, 1989; Scarlatos and Pavlidis, 1992; De Floriani et al., 1996; de Berg and Dobrindt, 1995), 3D objects for classical CAD and recognition (Fekete and Davis, 1984; Ponce and Faugeras, 1987; Rossignac and Borrel, 1993), and volume data (Cignoni et al., 1994a; Wilhelms and Van Gelder, 1994; Westermann, 1994; Cignoni et al., 1995a). All models proposed in the literature are based on the general idea that a detailed digital model taken as input can be simplified into an approximated representation: appropriate measures of fidelity to the original model are taken as a quantitative mean to define multiple levels of resolution.

There are two major challenges underlying the construction of multiresolution models (Heckbert and Garland, 1994): (i) to find effective and efficient algorithms for automatically building an approximated representation of reduced size at a predefined level of resolution; (ii) to structure data into a comprehensive framework that allows them to be manipulated at different resolutions according to the needs of a given application or task. Such problems are often interrelated, since the construction of a solution for the second problem can rely on algorithms and principles developed for the first problem.

Most methods for the approximated representation of surfaces use piecewise linear representations based on triangulations, because of their adaptivity. Many practical methods to build approximated triangulated surfaces follow heuristics that try to minimize the amount of data needed to achieve a given resolution, by either discarding less significant points from a detailed model (simplification), or inserting more significant points into a coarse model (refinement). Multiresolution models are usually based either

on sequences of representations of the whole terrain at different resolutions, or on tree-like hierarchies, where each node of a tree is a representation of a portion of terrain at a given resolution. Models from both classes are usually built iteratively through the same techniques used to build approximated representations.

A different approach has been recently proposed by Gross et al. (Gross et al., 1996), which controls the level of approximation of the surface by local spectral estimates determined over a wavelet representation. A hierarchical multiresolution representation (a quadtree built over regular grids) is constructed on the basis of the analysis of the wavelet transform coefficients, and allows the production of level of detail representations of the mesh.

Multiresolution models usually support tasks such as the extraction of a representation at a given resolution, the solution of interference queries (point location, windowing), the navigation through the domain, and across resolutions (browsing). A further, important, yet not much explored operation is rendering at *variable resolution* over different zones of the domain. A typical example is in landscape visualization for either flight simulators, or environmental assessment (Kaneda et al., 1989): the detail of the terrain model presented to the user may be variable, depending on the distance from the point of view. Variable resolution allows a larger number of polygons to be rendered only in the areas where the visual impact is at its most significant, thus speeding up rendering (see, Figure 1). A similar approach has also been outlined in scientific visualization to sharpen resolution only in user-selected focus areas (Cignoni et al., 1994a; Cignoni et al., 1994b). The main problem in providing a representation with variable resolutions is to maintain the continuity of the surface where pieces of surface with different accuracies meet.

In this paper, we present a multiresolution model for triangulated topographic surfaces, called a HyperTriangulation (HyT), which supports variable resolution, and is more compact and flexible than previous models. Our model is based on a structure that can maintain all significant refinement/simplification steps in passing from either a coarse representation to a refined one, or vice-versa. Intermediate representations are maintained implicitly in the model: an efficient data structure allows "on the fly" representations to be retrieved at an arbitrary resolution, either constant or variable over the domain, while guaranteeing the continuity of the resulting surface.

The definition of the model is independent of the refinement/simplification algorithm used for its construction, provided that representations whose resolutions are close to each other can be related through local changes over the domain. Here, we present a construction algorithm based on a refinement technique; however, it is straightforward to build the structure from the output of a simplification algorithm (Ciampalini et al., 1996).

2 Related work

The main idea underlying the construction of approximated terrain models is that a simplified model can be built based on a reduced set of data. Most practical approaches to the construction of approximated models are iterative, and can be classified into *simplification* methods (Schroeder et al., 1992; Turk, 1992; Rossignac and Borrel, 1993; Hoppe et al., 1993; Taylor and Barrett, 1994) - i.e., methods that start from the full resolution, and progressively reduce the dataset on which the model is based, in order to coarsen resolution; and *refinement* methods (Fowler and Little, 1979; Von Herzen and Barr, 1987; Dyn et al., 1990; Scarlatos and Pavlidis, 1992; De Floriani and Puppo, 1995; Cignoni et al., 1994a) - i.e., methods that start from a coarse approximation based on a small dataset, and progressively insert new data, in order to improve resolution. Both approaches rely on the concept of *local update*, and are essentially characterized by criteria to select points that are to be inserted into [deleted from] the model at each iteration. The most common approach is to base point selection on the impact, in terms of error reduction [increase], which is caused by the insertion [deletion] of a point into [from] the dataset.

Most multiresolution models proposed in the literature are based on the application of iterative approximation algorithms, and on the organization of local updates in the context of a unified framework. A comprehensive survey on multiresolution models can be found in (De Floriani et al., 1996). The few existing models supporting variable resolution rendering are all very recent, and they were proposed independently, either during the same period in which the results presented here were developed, or later.

Preliminary results on the matter of this paper were given by the authors in (Cignoni et al., 1995b). In the same period, a *hierarchical representation* was proposed in (de Berg and Dobrindt, 1995), which is defined as a classical pyramidal model (i.e., a heap of triangulations at increasingly finer resolution),

whose structure is essentially based on an earlier scheme proposed in (Kirkpatrick, 1983) to support point location. A Delaunay triangulation of the whole dataset is considered, which is simplified iteratively by removing maximal independent sets of vertices of bounded degree. This approach privileges the theoretical efficiency of the resulting structure (e.g., point location in logarithmic time), while in practice the compression ratio for each level of resolution might be worse than those obtained with heuristics proposed in the literature. Indeed, while in this case the only criterion used to select features is to fix a unique set of non-removable vertices, many other methods for approximated representations try to select for each level of resolution those vertices that are likely to be relevant at that level. Furthermore, the levels of the pyramid do not correspond to given accuracies, hence an explicit control of the accuracy is not provided. The hierarchical representation comes together with a simple algorithm, which extracts in time linear in its output size a representation at variable resolution based on a given threshold function. The algorithm is based on a top-down traversal of the pyramid, and on a greedy construction of the result. Unfortunately, the greedy approach, which accepts a triangle in the solution as soon as possible, does not warrant that the desired accuracy is fulfilled everywhere: indeed, because of the configuration of a partial solution, the algorithm can be obliged to accept new triangles whose accuracy is worse than required.

In (De Floriani and Puppo, 1995), a multiresolution model is proposed, which is described by a tree of nested Delaunay triangulations. Tree models are somehow easier to handle because of their strict hierarchical structure, but, on the other hand, the spatial constraints imposed by nesting have drawbacks, both in terms of the number of triangles needed to achieve a given accuracy, and in terms of their shape (slivery triangles often appear near the boundary of each node in the tree).

Two algorithms for variable resolution surface extraction from such a model are proposed. The first algorithm is a simple top-down visit of a tree, which accepts a triangle as soon as its error lies below the threshold. The resulting structure is a subdivision called a generalized triangulation, in which some triangles are added new vertices along their edges. A triangulation of such generalized triangles is performed next to obtain a triangulated surface, and the whole algorithm is completed in time linear in its output size. However, the approximating function is changed by the triangulation of generalized triangles, hence the accuracy of the final structure might be worse than desired. The second algorithm is essentially an adap-

tation of the algorithm proposed here to tree structures, and it was designed later: such an algorithm results more complicated, since it needs special data structures to manage neighbour finding across different nodes of the tree. Experimental results on variable resolution extraction were not presented.

3 Approximated Digital Terrain Representation

A natural terrain is mathematically described by an elevation function $\phi : D \subseteq \mathbb{R}^2 \rightarrow \mathbb{R}$, defined over a connected domain D on the XY plane. The surface described by the image of ϕ is often called a *topographic* or $2\frac{1}{2}D$ surface. In practical applications, function ϕ is sampled at a finite set of points $P = \{p_1, ..., p_n\} \subset D$, known as the set of representative points in the digital terrain. In this case the function ϕ can be defined piecewise over a subdivision Σ of D with vertices in P.

When a triangular subdivision is adopted to partition D, piecewise linear functions are a common choice to compute the elevation of points that are not in P. One such model is called a *Triangulated Irregular Network (TIN)*: TIN models of $2\frac{1}{2}D$ surfaces can be adapted to the characteristics of the surface, they can be built on scattered data, and they are widely used in many different fields, such as Geographical Information Systems, finite element analysis, robotics, computer graphics, and visualization.

Since a TIN is fully characterized by the plane triangulation underlying it, plus the elevation value at each of its vertices, hereafter we will always work on the plane triangulation, by considering triangles that form the actual surface only for the purpose of rendering or error testing.

3.1 Approximation error

As we pointed out in the introduction, the construction of an approximated representation is based on the possibility of selecting a significant subset of data from either a regular or a scattered dataset. The selection is almost always based on some measure of the error in representing a given surface through a simplified model. In the case of TINs, many alternative norms can be adopted to measure the distance between a surface represented by a TIN built over the whole dataset, and the surface corresponding to a reduced model based on a subset of data. A simple and common choice is to measure such errors by the maximum distance between the actual elevation of a datum and its approximated elevation in the reduced representation. The relevance of a given datum p in the current representation is related to the increase

[decrease] in the error as a consequence of the deletion [insertion] of p from [into] the model.

Another critical issue is the preservation of point and lineal features, such as ridges, valleys, peaks, and pits. Features can be identified by sharp discontinuities in the gradients of adjacent facets. Such features may be largely preserved if, while constructing the multiresolution representation, a measure of likelihood is adopted, which tends either to maintain or to insert points belonging to features. For instance, the selection heuristic can take into account the discontinuity on the gradient that is introduced [eliminated] with the insertion [deletion] of a point.

Several algorithms for approximating terrains were analyzed and compared by (Garland and Heckbert, 1995). On the basis of a number of tests they concluded that the best practical solution, in terms of error minimization, and number of triangles in the approximation, is the *greedy insertion* algorithm, which is the same approach we adopted. Our implementation is described in the following subsection. In order to be generic, we assume that at each step a *score* can be computed for each datum that is not a vertex of the model. This score may be dependent on the norm used to measure the approximation error, and on any other parameter involved in point selection, as discussed above. In order to preserve the efficiency of the method, whenever the TIN is updated it must be possible to compute such a score only at the points involved in changes. Moreover, for each such point p, it must be possible to compute its score in constant time, based only on local information (e.g., on the triangle of the current model covering p). We also assume that the approximation error of the TIN is updated while the score for each point is computed, at no extra cost.

In the simplest case, the score is the absolute value of the difference between the approximated and the actual elevation at p, while the current approximation error coincides with the maximum score over the triangulation. In a more sophisticated selection scheme, the score of p may be evaluated by weighting the surface error at p with the difference of the gradients of the facets incident at p after its insertion: the higher is such difference, the more p is likely to characterize the surface.

3.2 A refinement algorithm

The method we adopted in this work builds an approximated TIN through a refinement technique based on an on-line Delaunay triangulation of points on the XY domain, which is derived from an early method

proposed in (Fowler and Little, 1979). This approach is called the *Delaunay Selector*, and an efficient implementation is described in (De Floriani et al., 1996). Here, we give only a summary of the method.

Let $\varepsilon \geq 0$ be a tolerance value, let P be a finite set of points in $\mathbb{R}^2$, and let ϕ be the elevation function known at the points of P. An initial triangulation Σ is built first, whose vertex set is composed of all extreme points of the convex hull of P: such a triangulation covers the whole domain of the sampled data.

The triangulation is refined through the iterative insertion of new vertices, one at a time: at each iteration, the point of P with the highest score is inserted as a new vertex, and Σ is updated accordingly. The refinement process continues until the error of Σ goes below ε. A pseudo code description of the Delaunay Selector algorithm is shown in Figure 2.

Note that, as with most refinement techniques, the insertion of a single point during the Delaunay selector does not necessarily cause a decrease in the approximation error (simplification techniques have a symmetric behaviour). However, the convergence of the method guarantees that the approximation will improve after a number (expectedly small) of vertices have been inserted. Henceforth, we will call a *refinement step* a minimal sequence of consecutive point insertions such that the error of the resulting approximation is smaller than the error in the previous step. The area of the domain involved in a refinement step is always a polygonal region (which may be unconnected and/or multiply connected), which we will call the *refinement region*. The refinement region at a given refinement step is always bounded by edges that belong to both the triangulation before refinement, and the triangulation after refinement.

4 HyperTriangulation

Let us suppose that a Delaunay Selector is being run with an error tolerance $\varepsilon = 0$: the final structure generated by the algorithm will be a model at full resolution. If we consider all models built at intermediate refinement steps, we have a whole sequence of triangulations $\{\Sigma_0, \ldots, \Sigma_n\}$, where Σ_0 is the initial triangulation, $\Sigma = \Sigma_n$ is the full resolution model, and $\forall i = 0, \ldots, n$, the TIN associated with triangulation Σ_i approximates the full resolution with an error ε_i. The sequence of error tolerances monotonically decreases: $\varepsilon_0 > \varepsilon_1 > \ldots > \varepsilon_n = 0$.

If all such triangulations were piled up into a layered model, such as the Delaunay pyramid (De Floriani, 1989), a high number of layers would be obtained, and many triangles would appear several times in different layers. Our alternative approach is to store a sort of history of the incremental refinement process into a unique structure, which avoids replicating triangles that belong to more than one layer, while encoding adjacencies also between triangles that would appear in different layers. In fact, the concept of layer is not present in our structure: it makes sense only in the comparison with pyramidal models.

Our model maintains a history that is independent of the construction algorithm (indeed, the same model can be built through a simplification technique that iteratively demolishes a triangulation), and is simplified with respect to the previous ones. Moreover, being not based on a strict hierarchy, our model is superior to tree-like models, such as the one proposed in (De Floriani and Puppo, 1995), since it avoids drawbacks caused by spatial constraints (see Sect. 2).

Let us consider the refinement region that is re-triangulated in passing from Σ_{i-1} to Σ_i. Triangles of Σ_{i-1} and Σ_i can be classified as follows:

- *living triangles*: the triangles that are not changed during refinement (i.e., triangles outside the refinement region, that belong both to Σ_{i-1} and Σ_i);
- *dead triangles*: old triangles destroyed while updating the triangulation (i.e., the triangles of Σ_{i-1} that belong to the refinement region);
- *newborn triangles*: new triangles created while updating the triangulation (i.e., triangles of Σ_i inserted into the refinement region).

Note that usually most triangles are *living*, because the incremental insertion process acts only locally. By definition, the set of dead triangles and the set of newborn triangles respectively form two triangulations of the refinement region. Such triangulations share the edges that bound this region. Hence, instead of simply replacing the triangulation inside the refinement region, as in the standard Delaunay selector, we can "sew" along such boundary edges the patch formed by the newborn triangles over the triangulation formed by the dead triangles, while saving the dead triangles below the newborn ones. The refinement proceeds by iteratively sewing a patch at each refinement step.

In order to make the whole structure understandable, we embed it in 3D space: the triangulation Σ_0 lies on the XY plane, while at refinement step i the new vertices inserted are raised up along the Z axis at elevation i, and the new patch is welded onto the old triangulation at the boundary of the influence region: this can be visualized as a "bubble" popping up from the triangulation (see Figure 3). The resulting structure is a 2D simplicial complex embedded in 3D space, such that at each step the current triangulation is formed by the triangles of the upper surface of the complex. Note that the elevation of vertices in the HyperTriangulation has no relation with their elevation on the terrain surface. This structure is called a *HyperTriangulation (HyT)*: it maintains both the topological information collected during the refinement process, and information on the error of each triangle, which is useful for extracting representations at arbitrary resolutions.

Note that now different triangulations of the sequence $\Sigma_0, \ldots \Sigma_n$ are not stored explicitly and independently, but they are interconnected in order to store only once any portion that is common to different triangulations. This fact makes the model quite compact. Each intermediate triangulation is encoded implicitly in *HyT*. In order to show this, let us define the following two attributes for each triangle t in *HyT*:

- ε_b: *birth error* the global error reached by the triangulation just before triangle t was created (some value larger than ε_0 if the triangle belongs to the initial triangulation);
- ε_d: *death error* the global error of the triangulation just before t was destroyed (zero if the triangle belongs to the complete triangulation).

The birth and death errors allow to detect those triangles in *HyT* that were contained in the triangulation Σ_i, produced as an intermediate result of the refinement process of the Delaunay Selector, which satisfied approximation error ε_i. Consider a triangle t in *HyT*, which satisfies the following inequality:

$$t.\varepsilon_d \leq \varepsilon_i < t.\varepsilon_b, \tag{1}$$

where $t.\varepsilon_b$ and $t.\varepsilon_d$ are the birth and death errors of t, respectively: t is called an ε_i-alive triangle. From the definition above and from (1) it follows that all ε_i-alive triangles must belong to Σ_i. We show that Σ_i is in fact formed only by such triangles.

Let p be a point in the domain D of the HyperTriangulation *HyT*. For the sake of simplicity, let us assume that p does not lie on the projection of any edge of *HyT* on the XY plane (points that lie on projected edges can be treated exactly the same way, but the proof is more technical). We define the set of triangles that cover p as:

$$T_p = \{\ t \in \mathit{HyT} : p \in \hat{t}\ \} \tag{2}$$

where $\hat{t}$ is the projection of t on the XY plane. For each T_p there exists an ordering $t_1, t_2, ..., t_n$ on the set of its elements such that:

$$\forall i: \quad t_i.\varepsilon_b > t_i.\varepsilon_d = t_{i+1}\varepsilon_b > t_{i+1}.\varepsilon_d \quad \text{where } 1 \leq i < n; \tag{3}$$

Indeed, whenever a newborn triangle containing p is generated during construction, the triangle containing p in the current triangulation must die, and the birth error and death error of the newborn and dead triangle, respectively, must coincide. More informally, for each point p of D there must exist only one triangle in *HyT* whose projections in the XY plane contain p and which is ε_i-alive. Hence, the set of ε_i-alive triangles cover the whole domain, and thus there cannot be other triangles in Σ_i.

Since the birth and death error of each triangle in *HyT* will be used to efficiently extract terrain representations from *HyT* (see Section 5), they will be encoded explicitly in the model.

5 Encoding and traversing HyperTriangulations

In this section we describe how to encode and move through HyperTriangulations. In the following description, we mimic the traversal functions provided with the *facet-edge*, a data structure for representing cell complexes in three dimensions (Dobkin and Laszlo, 1989). Actually, we are only interested in the 2-skeleton of a three dimensional complex, i.e., the 2-simplicial complex formed by all triangles sewn onto the HyT during refinement.

In the facet-edge data structure, an atomic entity is associated with each pair that is identified by a face and one of its edges: the so-called *facet-edge*. This structure is equipped with traversal functions that permit to visit the complex. These functions are used to move from a facet-edge to an adjacent one, either by changing edge or by changing face (note that in 3D space more than two faces (triangles) may be incident at each edge).

Let t be a face (triangle) of a cell complex C, and let e be one of the edges of t. The facet-edge te denotes two rings in C: the *edge-ring* is formed by all the edges of the boundary of t; the *facet-ring* is formed by all the faces incident at e (see Figure 4). The traversal functions **enext** and **fnext** permit to move from one facet-edge to the next along the edge-ring and the facet-ring, respectively.

In order to make the data structure more suitable to our needs, we modify it by adding more traversal functions. We add two descriptive functions to each facet-edge, **birtherr** and **deatherr**, that report the birth and death error of the triangle to which it belongs, respectively. We also note that the triangles incident at a given edge e of the HyT can be subdivided into two groups, namely, those formed by faces whose projection on the XY plane lie to the left and to the right of the projection of e, respectively. Therefore, e can be considered as formed of two half-edges, where each half-edge corresponds to the group of facet-edges incident into it from one of its sides. Similarly, the *facet-ring* is actually subdivided into two half-facet-rings, each corresponding to a half-edge. Henceforth, a half-edge and its corresponding half-facet-ring will be referred to interchangeably.

Let he be a half-edge, let fe_l and fe_h be the facet-edges at the lowest and highest level in its corresponding half-facet-ring, respectively: the *life* of he is defined as the interval containing all values ε such that there exists some facet-edge in its corresponding ring that is ε-alive, i.e., interval $[fe_h.\varepsilon_d, fe_l.\varepsilon_b]$. We add two other descriptive functions, **hebirth** and **hedeath**, which report, for a given facet-edge, the birth and death of the half-edge at which it is incident.

When traversing the HyT, we may need to move through two different domains: the *spatial* domain D and the *error* domain $[0, \varepsilon_0]$. In the former case, we may need to cross an edge e by moving from one of its half-edges to the other. More precisely, we are interested to move from one triangle incident into a half-edge, to another which is incident into the other half-edge, and which has a compatible accuracy (i.e., an accuracy for which both triangles are alive). In the latter case, we may need to adjust the accuracy by moving to the facet that either precedes or follows the current one in the half-facet-ring.

Each half-facet-ring is encoded as a bidirectional chain, which is identified with the corresponding half-edge. In order to maintain elements of these two chains connected across the edge, we add another traversal function, **fother**, which connects a facet-edge te with the facet-edge on the opposite side of e

that was either created together with te, or had minimal error when te was created.

In summary, for each facet-edge we define the following traversal functions:

- **enext**: next facet-edge in the edge-ring (on the same face);
- **fnext**: next facet-edge in the half-facet-ring (with lower error);
- **fprev**: previous facet-edge in the half-facet-ring (with higher error);
- **fother**: the compatible facet-edge on the other half-edge.

Figure 5 shows a side view of the facet-edges in Figure 4. The two arrows represent the two parts of the edge-ring. Figure 6 shows how the facet-edges are connected through the function **fother**; the numbers represent the values of death error for the facet-edges. Note that the **fother** function does not induce a symmetric relation between facet-edges (i.e., e.fother.fother does not necessarily coincides with e).

6 Extracting triangulations from HyT

If a HyperTriangulation is encoded following the guidelines given in the previous Section, it is possible to efficiently extract triangulations defining TINs such that:

1. the approximation error is either constant over the domain D, **or** variable according to a function $E()$ defined on D, where $E(p)$ is the error tolerance accepted at each point p, **and**
2. the continuity of the surface extracted is guaranteed everywhere.

Algorithms for the extraction at constant and variable resolution, respectively, are described in the following subsections.

6.1 Extraction at constant approximation

Let ε_0 be the error corresponding to the bottom of the HyT, and let ε be an arbitrary value such that $0 \leq \varepsilon \leq \varepsilon_0$. As we have seen in Section 4, a TIN at constant accuracy ε will be formed by all triangles of the HyT that are ε-alive, i.e., such that $t.\varepsilon_d \leq \varepsilon \leq t.\varepsilon_b$. Such a triangulation can be extracted from the HyT

through a topological visit, starting from a triangle t that is ε-alive, and moving to adjacent triangles that satisfy the same relation.

A pseudo-code of the algorithm is presented in Figure 7. The algorithm traverses the HyT, moving from triangle to triangle through the traversal function **fother**. A list `ActiveStack` keeps the facet-edges that must be visited: the algorithm stops when the list becomes empty, i.e., when the whole domain has been traversed. The `ActiveStack` is not handled exactly like a stack, since elements can be either pushed onto its head or appended to its tail, while they are always popped from the head. The rule for inserting elements in the `ActiveStack` is the following: if a facet-edge is ε-alive it is pushed onto the head, otherwise it is appended to the tail. This mechanism ensures that all the elements of a given bubble are visited before the algorithm moves to a different bubble.

During traversal, we mark a half-edge each time one of its corresponding facet-edges is visited: in this case, all facet-edges of the corresponding group are considered marked. Facet-edges incident into a half-edge are packed in the data structure: therefore, each half-edge can be marked, and each facet-edge can be tested for mark in constant time.

The first triangle t to visit is extracted through a suitable procedure. In the simplest approach, a sorted list containing one triangle for each of the n levels of accuracy of the HyT can be maintained. The list is scanned to find the initial triangle in a time linear in the number of levels preceding the level of the extracted triangulation. A more efficient approach is to maintain a balanced search tree rather than a list, thus achieving logarithmic time. In both cases, the worst case time complexity is bounded by the size of the triangulation extracted: indeed, the construction mechanism of the HyT warrants that a triangulation will have at least as many triangles as the value of its level in the HyT.

The `ActiveStack` is initialized with the neighbours of t that are reached with function **fother**. At each cycle, a facet-edge is popped from the `ActiveStack`, it is visited, and its accuracy is tested against threshold ε (denoted `eps` in the pseudo-code). If the facet-edge is not ε-alive, it means that the border of a bubble has been processed, thus function **fnext** is used iteratively to climb bubbles incident into the same half-edge until a facet-edge that is ε-alive is found. The corresponding triangle is added to the output, and its three half-edges are marked as visited, while all its neighbours through function **fother** that were not

visited yet are added to the `ActiveStack`. The algorithm stops when the `ActiveStack` is empty, which means that the whole domain has been traversed.

The time complexity of this algorithm depends on the number of triangles of the extracted triangulation, and on the number of facet-edges visited by the climbing loop. Indeed, let e be the current facet-edge popped from the active stack. If e is either marked or is ε-alive, then it is processed in constant time, otherwise the **fnext** function is used iteratively to climb to the desired level of the same half-edge. Let e' be the facet-edge found by the climbing loop. The neighbours of e' that belongs to the same bubble are directly addressed by the **fother** function. Therefore, the mechanism for adding elements to the `ActiveStack` warrants that the whole bubble to which e' belongs will be visited before the climbing loop is called again. This means that the climbing loop is called at most once for each bubble that is traversed by the algorithm. Moreover, at each cycle of the climbing loop a bubble is discarded, which is never visited again. Therefore, the total number of cycles of the climbing loop is at most equal to the number of bubbles traversed by the algorithm. Since such bubbles are either at the same level of the triangulation extracted or below it, their number is bounded by the number of triangles in the triangulation extracted.

We conclude that the worst case time complexities of both the location of the initial triangle, and of the traversal phase, are at most linear in its output size. Thus, the worst case complexity of the whole algorithm is linear in its output size, and therefore optimal.

6.2 Extraction at variable approximation

The extraction of a surface with a variable approximation error over the domain can be performed through a similar traversing strategy. We describe an algorithm that is suitable whenever the approximation error follows a function $E : D \to \mathbb{R}$ of the class:

$$E(p) = f_e(d(v_p, p)) \tag{4}$$

where $f_e : \mathbb{R} \to \mathbb{R}$ is a monotonically increasing function, $d()$ is the standard Euclidean distance in $\mathbb{R}^2$, and v_p is a fixed point called the *viewpoint*. This error function is adequate for applications such as flight simulators: the farther the viewpoint, the larger the tolerance error that can be accepted for terrain

representation.

In this case, the algorithm visits the HyT starting from the triangle that is closest to (or, possibly, contains) the viewpoint v_p, and satisfies the error function $E(v_p)$. Then, it proceeds as before by visiting the HyT through adjacencies, while increasing the current error according to distance from the viewpoint. The delicate point in this algorithm is to warrant that each time a triangle is added to the solution this is a correct choice, i.e., that such a triangulation can be completed to cover the whole domain. In the previous algorithm, this fact was ensured because the solution was known to be formed by all ε-alive triangles. Since such fact does not hold now, we will have to use a different criterion.

Let us suppose that we are at an intermediate cycle of the algorithm described in the previous section, and let us call the *current triangulation* the partial solution built so far by the algorithm. Such a triangulation covers a connected region that is bounded by a chain of edges. For each such edge e, there exists a facet-edge in the current `ActiveStack` that is incident at e. The next triangle t that we insert will belong to one of the half-edges corresponding to such facet-edges, and it will be a valid one provided that the region external to the current triangulation can be covered by a triangulation containing t, and whose error lies below the threshold function.

Let us consider the minimum birth error B_{min} and the maximum death error D_{max} among all half-edges corresponding to the facet-edges currently contained in the `ActiveStack`, i.e.,

$$B_{min} = \min\{fe.hebirth : e \in ActiveStack\} \tag{5}$$

$$D_{max} = \max\{fe.hedeath : e \in ActiveStack\} \tag{6}$$

If the new triangle that we add is ε-alive for some $\varepsilon \in [D_{max}, B_{min}]$, we will be certain that, in the worst case, the external region can be triangulated by using all triangles that are ε-alive. Indeed, the triangulation at constant error ε must necessarily contain all edges that bound the current triangulation. Moreover, if ε is lower than the minimum E_{min} of the threshold function over the border of the current triangulation, then we will be also sure that such completion of the triangulation will satisfy the threshold everywhere, since the threshold outside the current triangulation can become only larger. Since all the edges bounding the current triangulations are E_{min}-alive, we are sure that $E_{min} \geq D_{max}$. Therefore, the correctness is warranted if we select at each cycle $\varepsilon = \min(E_{min}, B_{min})$.

In order to favour a rapid increase of the threshold, the algorithm selects at each cycle the active facet-edge that correspond to the value B_{min}, and searches its half-facet-ring to get a valid triangle. In summary, the main steps of the algorithm at each cycle are:

1. extract from the set of active facet-edges the facet-edge fe whose corresponding half-edge has minimum birth error;
2. compute the new tolerance ε;
3. find the correct facet-edge fe' in the chain containing fe;
4. store unvisited facet-edges of the edge-ring of fe' into the set of active facet-edges.

In order to obtain efficiently the current facet-edge fe, as well as values B_{min} and E_{min}, we maintain all active facet-edges in two heaps MBH (Minimum Birth Heap), and MEH (Minimum Error Heap): the minimum element in MBH is the facet-edge visited at each cycle, while the minimum element in MEH is the one corresponding to the minimum of function $E()$ on the border of the current triangulation. At each step, the next facet-edge to be visited is selected by extracting the minimum element from the MBH, and removing it from MEH, while the minimum of MEH is read without removing it from the heaps. The tolerance ε (`eps` in the pseudo-code) is computed on the basis of such two minima, then the triangle to be added to the solution is found by searching the half-facet-ring containing the current facet-edge.

The extraction starts by detecting the triangle in HyT that contains v_p and has error $E(v_p)$. A hierarchical point location query can be efficiently solved on the HyT structure by exploiting the HyT hierarchical structure (which represent the history of the on-line construction of the triangulation), with a method similar to the one proposed in (Boissonnat and Teillaud, 1993). However, if we look to a particular class of applications, such as flight simulators, the knowledge of the problem allows us to avoid to cope with a generic point location query. In this case, the location of the viewpoint at a time $t + \Delta t$ is generally near the location of the viewpoint at time t. Therefore, the triangle containing the current viewpoint can be simply found by traversing the HyT through adjacencies, starting at the triangle containing the previous viewpoint.

Figure 8 shows a pseudo-code of the algorithm. Note that instead of using two separate heaps we adopt a double heap DH which maintains both the B_{min} and E_{min} of active facet-edges. The operations performed on the double heap are: extracting the facet-edge fe corresponding to B_{min}, by removing it from both heaps; reading the value E_{min}; inserting a facet-edge. The value of B_{min} is obtained as $fe.hebirth$. All such operations can be performed in logarithmic time in the size of the heap, which is bounded from above by the output size n_t.

Therefore, the time complexity of this algorithm is increased at each cycle for a factor $\log n_t$ with respect to that of the previous algorithm. Hence, the overall complexity of the extraction of a surface with variable approximation error is $O(n_t \log n_t)$.

7 Results

A number of tests were performed on public domain datasets, to evaluate the simplification rates, and the times required both to build the multiresolution representation and to extract level of detail representations out of it. All the original datasets used are regular DEMs[1]. The Bangor, S.Bernardino and Lake Charles models are given on a 128×128 grid, while the Devil Peak model is defined on a larger 231×165 grid. The size of the resulting Delaunay TINs are indicated in Table 1. Note that only a subset of the input samples were needed to build the triangulations at maximal precision: for example, 11,762 samples out of the original 16,384 samples have been sufficient to construct the *zero-error* representation of the Bangor dataset. This happens because all elevation data are integers, and some of them can be interpolated exactly by triangular patches having vertices at different data points.

The tests were performed on an SGI Indigo2 workstation (R4400 200MHz cpu, 16KB primary cache, 1MB secondary cache, 32 MB RAM, IRIX 5.3 OS).

The results are presented numerically through the following tables.

Table 1 presents the number of triangles contained in the HyT, the number of refinement steps performed to build the HyT and the number of triangles in the representation of the terrain at maximal resolution. Note that the number of triangles at maximal resolution are roughly twice the number of sites (i.e. it

[1]The datasets are publically available from the U.S. Geological Survey - National Mapping Information - EROS Data Center at the following URL: `ftp::/edcftp.cr.usgs.gov/pub/data/DEM/250`.

corresponds roughly to split each square cell in the DEM in two triangles), while the total size of the HyT is roughly twice the number of triangles at maximal resolution.

Then, we have a table pair for each of the four datasets. For each dataset we report data on constant approximation extractions on the leftmost table (on the basis of a number of accuracies, measured in terms of a percentage over the height field), and data on variable approximation extractions on the rightmost table. For each accuracy (i.e. each table row) we list: the number of triangles in the representation that was extracted at the given accuracy (*no.*), and the percentage with respect to the number of facets in the original terrain (*%*); the time required for the extraction (in seconds) and the number of triangles extracted per second.

In the case of the rightmost tables, the meshes extracted satisfy maximal accuracy in the proximity of one map corner (error equal to zero) and error equal to the selected ε value in the opposite map corner; error increases linearly with the distance from the maximal accuracy corner, and in the tables we report on each row the extent of the error interval ($0 \rightarrow \varepsilon$).

Obviously the terrain extracted at constant error ε is more compact than the terrain extracted at variable accuracy ($0 \rightarrow \varepsilon$), but the visual quality of the second mesh is impressively higher. The linear function which drives the increase of error in the variable accuracy extractions has been designed to allow the extraction of terrain meshes where the distribution of facets, measured per unit of viewing space area, remains almost constant on the entire image plane. In fact, the projection of the approximation error onto the viewplane remains constant.

A visual representation of our results on the Devil Peak terrain is in Figures 9 - 13.

A top view of two terrains, extracted at constant accuracy (2.1 % error, 8613 triangles) on the left, and at variable accuracy ($0 \rightarrow 2.7\%$ error, 8340 triangles) on the right, is shown in Figure 9. For the variable extraction, the error range was conveniently selected to get in output a mesh with a size close to that returned by the constant extraction on the left. The location of the point with maximal accuracy is in the proximity of the midpoint of the lower edge. In the following pictures (Figures 10 and 11), the two terrains are shown by selecting as viewpoint the point with maximal accuracy, chosen for the variable approximation extraction. The difference in visual quality, given the same size of the two meshes above,

is evident.

A couple of terrains, extracted at full and variable accuracy, are shown in the last two figures (wire frame and flat shaded). In this case, the variable resolution terrain is only 15% of the size of the leftmost representation but, once shaded, the two meshes look very similar.

Datasets		**Hypertriangulation**		
	no. of sites	no. trian. in HyT	no. refinem. steps	no. trian. max res.
Bangor	11,762	52,606	3,896	23,226
Lake Charles	11,939	52,352	2,601	23,569
San Bernardino	15,828	70,390	3,697	31,153
Devil Peak	27,432	112,633	1,657	54,340

Table 1: Resolution and hyper triangulation complexity on a number of test datasets (no. trian. max prec.: number of triangles in the maximal accuracy map).

Error	Extracted triangles			Time
%	no.	%	trian./sec.	sec.
0	23,226	100	42,229	0.55
1	16,340	70.35	41,897	0.39
2	11,972	51.54	41,282	0.29
3	9,388	40.42	42,672	0.22
5	5,558	23.93	42,753	0.13
7	3,794	16.33	42,155	0.09
10	2,288	9.85	45,760	0.05
15	1,136	4.89	37,866	0.03

Error	Extracted triangles			Time
%	no.	%	trian./sec.	sec.
0	23.226	100		
$0 \rightarrow 1$	16,191	69.71	10,378	1.56
$0 \rightarrow 2$	14,367	61.85	10,884	1.32
$0 \rightarrow 3$	12,069	51.96	10,775	1.12
$0 \rightarrow 5$	8,928	38.43	11,022	0.81
$0 \rightarrow 7$	6,681	28.76	11,135	0.60
$0 \rightarrow 10$	5,026	21.63	11,168	0.45
$0 \rightarrow 15$	3,155	13.58	11,685	0.27

Table 2: Bangor dataset: extraction of terrains at constant error, on the left, and at variable error, on the right.

8 Conclusions and future work

The HyperTriangulation supports the efficient extraction of continuous surfaces at variable resolution, and is more compact than other models described in the literature.

The efficient manipulation of surface information achieved by the model allows us to obtain a dynamic visualization of landscapes with high quality images. The extraction rate provided, 10k triangle per second at variable resolution, is sufficient to produce in real time representations of the terrain which vary dynamically. In fact, in many applications (e.g. flight simulators) we do not need to produce a different model for each frame, because the position of the viewpoint remains in the proximity of a given loca-

Error %	Extracted triangles			Time sec.
	no.	%	trian./sec.	
0	23.569	100	42.852	0,55
1	14.718	62,44	42.051	0,35
2	9.514	40,36	41.365	0,23
3	6.642	28,18	41.512	0,16
5	4.257	18,06	42.570	0,10
7	2.919	12,38	41.700	0,07
10	1.760	7,46	44.000	0,04
15	915	3,88	45.750	0,02

Error %	Extracted triangles			Time sec.
	no.	%	trian./sec.	
0	23.569	100		
$0 \rightarrow 1$	16.550	70,21	10.541	1,57
$0 \rightarrow 2$	11.533	48,93	10.297	1,12
$0 \rightarrow 3$	8.530	36,19	11.077	0,77
$0 \rightarrow 5$	5.992	25,42	11.096	0,54
$0 \rightarrow 7$	4.882	20,71	11.353	0,43
$0 \rightarrow 10$	3.660	15,52	11.806	0,31
$0 \rightarrow 15$	2.489	10,56	11.852	0,21

Table 3: Lake Charles dataset: extraction of terrains at constant error, on the left, and at variable error, on the right.

Error %	Extracted triangles			Time sec.
	no.	%	trian./sec.	
0	31.153	100	42.098	0,74
1	12.059	38,70	41.582	0,29
2	7.307	23,45	42.982	0,17
3	4.906	15,74	40.883	0,12
4	3.447	11,06	43.087	0,08
5	2.507	8,04	35.814	0,07
6	1.941	6,23	38.820	0,05
7	1.522	4,88	50.733	0,03
8	1.161	3,72	58.050	0,02

Error %	Extracted triangles			Time sec.
	no.	%	trian./sec.	
0	31.153	100		
$0 \rightarrow 1$	12.046	38,66	10.566	1,14
$0 \rightarrow 2$	9.927	31,86	10.790	0,92
$0 \rightarrow 3$	7.331	23,52	10.323	0,71
$0 \rightarrow 4$	5.569	17,87	10.709	0,52
$0 \rightarrow 5$	4.845	15,55	10.766	0,45
$0 \rightarrow 6$	4.212	13,52	11.383	0,37
$0 \rightarrow 7$	3.833	12,30	11.273	0,34
$0 \rightarrow 8$	3.115	9,99	11.125	0,28

Table 4: SanBernardino dataset: extraction of terrains at constant error, on the left, and at variable error, on the right.

tion for a number of consecutive frames, accordingly to the aircraft speed and direction. For this reason, the same variable resolution model may be used for a number of frames, hence reducing the throughput required in the extraction.

Although we have given a description of the model based on a refinement construction technique, the same structure can be obtained through any technique that either refines or simplifies a surface defined on the basis of a discrete dataset. A simplification algorithm, based on the decimation approach, has been recently proposed (Ciampalini et al., 1996) which returns multiresolution results in the same format than the refinement construction technique described here.

We are now extending this approach to generalized surfaces in 3D space. An interactive system is in an advanced implementation phase which supports, starting from the multiresolution data produced by a mesh simplification code, the interactive selective refinement/simplification of the mesh (Cignoni et al.,

Error %	Extracted triangles			Time sec.
	no.	%	trian./sec.	
0	54,340	100	35,058	1.55
1	21,918	40.33	28,464	0.77
2	9,207	16.94	30,690	0.30
3	5,113	9.40	31,956	0.16
5	2,480	4.56	31,002	0.08
7	1,425	2.62	28,500	0.05
10	736	1.13	26,285	0.03

Error %	Extracted triangles			Time sec.
	no.	%	trian./sec.	
0	54,340	100		
$0 \rightarrow 1$	27,914	51.36	9,462	2.95
$0 \rightarrow 2$	14,469	26.62	10,291	1.40
$0 \rightarrow 3$	9,549	17.57	10,267	0.93
$0 \rightarrow 5$	5,287	9.72	10,574	0.50
$0 \rightarrow 7$	3,663	6.74	10,773	0.34
$0 \rightarrow 10$	2,292	4.21	11,460	0.20

Table 5: Devil Peak dataset: extraction of terrains at constant error, on the left, and at variable error, on the right.

1997). The domain of this new system is *multiresolution modeling* or, using a metaphor, *geometric painting*. Given an input surface and the multiresolution results of its simplification, the idea is to allow the user to choose first a constant approximation level and then, interactively, to modify the mesh by increasing or reducing the accuracy in selected areas, using a logical interface extremely similar to those provided by image painting systems. The error function in this case is not dependent on the current view parameters, but it is completely user–driven and depends on the definition of a *region of interest* on the mesh. User selects the refinement focus point and the size of the area to be refined, and dispose of a graphic editing window to draw interactively the error function, which will be used to reduce/increase resolution in the selected region of interest.

Another straightforward extension of the model to handle volume data can be defined through an analogous structure built on tetrahedra embedded in 4D space. A 3D generalization of the Delaunay selector has also been used for multiresolution volume modeling and visualization (Cignoni et al., 1994a), and can be easily adapted to build a 3D HyT.

After this work was submitted for publication, more results on variable resolution surface modeling appeared in the literature. It is worth mentioning models proposed in (Klein and Straßer, 1996; Hoppe, 1996), which are based on linear sequences of updates on an initial mesh: such models achieve high conciseness in the data structure, but require involved techniques with high computational costs to extract a representation at variable resolution.

For the special case of data points on a regular grid, an algorithm for the real-time construction of variable resolution maps made of right triangles was proposed in (Lindstrom et al., 1996), which exploits an im-

plicit quadtree-like decomposition of the domain. The screen space threshold adopted in this work bounds the maximum approximation error to the projected image space, ensuring controlled image quality and no popping effects. The same strategy can also be used in our approach: in variable resolution extraction, error can be evaluated by taking into account the screen space magnitude (computed on-the-fly) of the error associated to each facet.

Finally, (Puppo, 1996) gives a comprehensive multiresolution model, which includes all models mentioned above, and the one described in this paper, as special cases: an algorithm has been also proposed, which extracts from such a model a representation of minimal size for a given threshold, variable over the domain, in optimal time, i.e., linear in the output size. Some experiments using such a model are presented in (De Floriani et al., 1997).

9 Acknowledgements

The work described in this paper has been partially funded by the Progetto Coordinato "*Modelli multirisoluzione per la visualizzazione di campi scalari multidimensionali*" of the Italian National Research Council (CNR) and the Progetto Coordinato "Modelli e Sistemi per il Trattameno di Dati Ambientali e Territoriali" of the CNR.

References

Boissonnat, J. and Teillaud, M. (1993). On the randomized construction of the delaunay tree. *Theoretical Computer Science*, 112:339–354.

Ciampalini, A., Cignoni, P., Montani, C., and Scopigno, R. (1996). Multiresolution decimation based on global error. Technical Report C96-021, CNUCE – C.N.R., Pisa, Italy, (to appear on The Visual Computer).

Cignoni, P., Floriani, L. D., Montani, C., Puppo, E., and Scopigno, R. (1994a). Multiresolution Modeling and Rendering of Volume Data based on Simplicial Complexes. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 19–26. ACM Press.

Cignoni, P., Montani, C., Puppo, E., and Scopigno, R. (1995a). Multiresolution Modeling and Visualization of Volume Data. Technical Report C95-22, Istituto CNUCE – C.N.R., Pisa, Italy.

Cignoni, P., Montani, C., Rocchini, C., and Scopigno, R. (1997). Resolution modeling. Technical Report C97-02, CNUCE – C.N.R., Pisa, Italy.

Cignoni, P., Montani, C., and Scopigno, R. (1994b). MagicSphere: an insight tool for 3D data visualization. *Computer Graphics Forum*, 13(3):317–328. (Eurographics '94 Conf. Proc.).

Cignoni, P., Puppo, E., and Scopigno, R. (1995b). Representation and visualization of terrain surfaces at variable resolution. In Scateni, R., editor, *Scientific Visualization '95 (Proc. Inter. Symposium on)*, pages 50–68. World Scientific.

de Berg, M. and Dobrindt, K. (1995). On levels of detail in terrains. In *11th ACM Computational Geometry Conf. Proc. (Vancouver, Canada)*, pages C26–C27. ACM Press.

De Floriani, L. (1989). A pyramidal data structure for triangle-based surface description. *IEEE Comp. Graph. & Appl.*, 9(2):67–78.

De Floriani, L., Magillo, P., and Puppo, E. (1997). Efficient encoding and retrieval of triangle meshes at variable resolution. Technical Report PDISI-97-01, Department of Computer and Information Sciences, University of Gemova, Genova, Italy.

De Floriani, L., Marzano, P., and Puppo, E. (1996). Multiresolution models for topographic surface description. *The Visual Computer*, 12(7):317–345.

De Floriani, L. and Puppo, E. (1995). Hierarchical triangulation for multiresolution surface description. *A.C.M. Trans. on Graphics*, 14(4):363–411.

Dobkin, D. and Laszlo, M. (1989). Primitives for the manipulation of three–dimensional subdivisions. *Algorithmica*, 4:3–32.

Dyn, N., Levin, D., and Gregory, J. (1990). A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169.

Fekete, G. and Davis, L. (1984). Property spheres: a new representation for 3-d object recognition. In *Proceedings Workshop on Computer Vision: Representation and Control*, pages 192–201, Los Alamitos, CA. CS Press.

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

Garland, M. and Heckbert, P. (1995). Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, School of Computer Sciences, Carnegie Mellon University, Pittsburgh, PA, USA.

Gross, M., Staadt, O., and Gatti, R. (1996). Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Trans. on Visual. and Comp. Graph.*, 2(2):130–144.

Heckbert, P. and Garland, M. (1994). Multiresolution Modeling for Fast Rendering. In *Graphics Interface '94 Proceedings*, pages 43–50.

Hoppe, H. (1996). Progressive meshes. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '96)*, pages 99–108.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1993). Mesh optimization. In *Proceedings of SIGGRAPH '93(Anaheim, CA, August 1-6). In Computer Graphics Proceedings, Annual Conference series, ACM SIGGRAPH*, pages 19–26.

Kaneda, K., Kato, F., Nakamae, E., Nishita, T., Tanaka, H., and Noguchi, T. (1989). Three dimensional terrain modeling and display for environment assessment. *ACM Computer Graphics*, 23(3):207–214.

Kirkpatrick, D. (1983). Optimal search in planar subdivisions. *SIAM Journal of Computing*, 12:28–35.

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

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L., Faust, N., and Turner, G. (1996). Real-time, continuous level of detail rendering of height fields. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '96), ACM Press*, pages 109–118.

Ponce, J. and Faugeras, O. (1987). An object centered hierarchical representation for 3d objects: the prism tree. *Computer Vision, Graphics and Image Processing*, 38(1):1–28.

Puppo, E. (1996). Variable resolution terrain surfaces. In *Proceedings Eight Canadian Conference on Computational Geometry, Ottawa, Canada*, pages 202–210.

Rossignac, J. and Borrel, P. (1993). Multi-resolution 3d approximations for rendering complex scenes. In B. Falcidieno, T. K., editor, *Modeling in Computer Graphics*, pages 455–465. Springer-Verlag.

Scarlatos, L. and Pavlidis, T. (1992). Hierarchical Triangulation Using Cartographic Coherence. *CVGIP: Graphical Models and Image Processing*, 34(2):147–161.

Schroeder, W., Zarge, J., and Lorensen, W. (1992). Decimation of triangle mesh. *ACM Computer Graphics*, 26(2):65–70.

Taylor, D. and Barrett, W. (1994). An Algorithm for Continuous Resolution Polygonalization of a Discrete Surface. In *Graphics Interface '94 Proceedings*, pages 33–42.

Turk, G. (1992). Re-Tiling Polygonal Surfaces. *ACM Computer Graphics*, 26(2):55–64.

Von Herzen, B. and Barr, A. (1987). Accurate triangulations of deformed, intersecting surfaces. *Computer Graphics*, 21(4):103–110.

Westermann, R. (1994). A Multiresolution Framework for Volume Rendering. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 51–58. ACM Press.

Wilhelms, J. and Van Gelder, A. (1994). Multi-dimensional Trees for Controlled Volume Rendering and Compression. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 27–34. ACM Press.