Hierarchical Triangulation for Multiresolution Surface Description

Leila De Floriani
Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova
Viale Benedetto XV, 3 - 16132 Genova - ITALY
Email: deflo@disi.unige.it

Enrico Puppo
Istituto per la Matematica Applicata
Consiglio Nazionale delle Ricerche
Via De Marini, 6 (Torre di Francia) - 16149 Genova - ITALY
Email: puppo@ima.ge.cnr.it

Abstract

A new hierarchical triangle-based model for representing surfaces over sampled data is proposed, which is based on the subdivision of the surface domain into nested triangulations, called a *Hierarchical Triangulation (HT)*. The model allows compression of spatial data and representation of a surface at successively finer degrees of resolution. An HT is a collection of triangulations organized in a tree, where each node, except for the root, is a triangulation refining a face belonging to its parent in the hierarchy. We present a topological model for representing an HT, and algorithms for its construction and for the extraction of a triangulation at a given degree of resolution. The surface model, called a *Hierarchical Triangulated Surface (HTS)*, is obtained by associating data values with the vertices of triangles, and defining suitable functions that describe the surface over each triangular patch. We consider an application of a piecewise-linear version of the HTS to interpolate topographic data, and we describe a specialized version of the construction algorithm that builds an HTS for a terrain starting from a high resolution rectangular grid of sampled data. Finally, we present an algorithm for extracting representations of terrain at variable resolution over the domain.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling - *Curve, surface, solid, and object representations; Object hierarchies.*

General Terms: Algorithms, Design.

Additional Key Words and Phrases: Triangulation, Hierarchical subdivision, Multiresolution surface model, Terrain model.

1 Introduction

The representation of surfaces defined by bivariate functions plays an important role in different application fields, like computer aided geometric design, geographical information systems, finite element analysis, robotics, computer vision, and computer graphics. Surfaces can be represented by means of a digital model: a digital surface model is characterized by a domain partition, and by a space of functions used on such partition. Digital surface models have been studied by many researchers, and much attention has been paid to spaces of functions defined over either regular or irregular subdivisions made of triangles or quadrilaterals. In this paper, we focus on issues related with the domain partition, while remaining as general as possible about the functions used to represent the surface within each region of the subdivision.

Subdivisions based on quadrilaterals have been preferred in the past, and are commonly used in commercial surface modelers, mostly because polynomial surfaces of any order can be easily defined through tensor products over rectangular patches, and the continuity across different patches can thus be easily controlled. Lately, triangulated surface models have become more and more popular, because of their adaptivity: recent studies have shown the effectiveness of smooth triangle-based representations of surfaces with intricate geometries (see, e.g., [Fon92]).

Digital models are often intended as approximations of surfaces describing natural objects or phenomena, which are either simulated or measured through some sampling process. A better precision in the approximation is generally achieved by a finer partition of the domain, yielding higher storage cost and computation time. On the other hand, not all tasks in the framework of a given application necessarily require the same accuracy, and even a single task may need different degrees of resolution in different areas of the domain. For instance, in landscape visualization and in flight simulation, the terrain surface and the objects on the terrain must be rendered at different degrees of resolution, depending on their distance from the viewpoint.

Multiresolution surface models offer the possibility of representing and analyzing a surface at different degrees of resolution: thus, for a given application, a coarse representation can be used over certain areas, while high resolution can be focused on specific areas of interest. A multiresolution surface model is effective if its storage cost does not introduce a serious overhead with respect to a simple surface model at the maximum precision, and if its access and manipulation algorithms are maintained efficient.

A straightforward approach to multiresolution is to build a *layered model* consisting of a collection of independent surface models at different fixed degrees of resolutions. The proper resolution is selected among the available ones depending on the needs of each specific task or application. Such an approach yields serious limitations in supporting complex operations for manipulating and visualizing the surface, like efficient geometric query processing, browsing over the surface and across different resolutions, distributed storage of large datasets, variable resolution rendering, etc. The addition of interference links between layers, as in the *pyramidal model* proposed in [DeF89], can provide means to overcome some such limitations but, on the other hand, it involves a non-negligible storage overhead.

The above limitations are not the only drawback of layered models. In principle, one might desire a model that spans a continuous range of resolutions, rather than just a few fixed degrees of resolution. A discrete approximation of such an ideal model could be obtained by a layered model in which a high number of different resolutions, very close to one another, are maintained.

Unfortunately, this approach can involve a tremendous amount of storage. The main reason behind this inefficiency is that each degree of resolution yields a representation of the whole surface, though only some parts of it might have changed from the representation at a resolution immediately coarser or finer. This fact is especially clear when the degrees of resolution are very close to one another. For example, let us consider a topographic surface model, and let us assume the resolution of the model is measured, as it often happens, by the maximum error in approximating the elevation at each point on terrain. Given a representation based on a domain subdivision Σ, whose resolution corresponds to an error ε, let $\varepsilon' < \varepsilon$ be the error of the next finer resolution. A representation at error ε' will be based on a subdivision different from Σ. However, if $\delta = \varepsilon - \varepsilon'$ is relatively small, such subdivision might coincide with Σ over some regions, since the precision obtained with the old representation over such regions was already below error ε'. The smaller δ the higher the number of regions that "survive" across consecutive levels. Taking into account this fact, we can argue that efficiency can be gained over layered models by building a model that keeps track only of changes between different resolutions, while avoiding duplications of regions that remain unchanged across different levels.

In this paper, we introduce a formal model, called the *Hierarchical Triangulation (HT)*, that consists of a subdivision of a plane domain into nested triangulations, and whose hierarchical structure is described by a tree. We discuss some properties of hierarchical triangulations, and we describe a data structure for its encoding, and some basic algorithms for its construction and analysis. Moreover, we propose a multiresolution surface model based on a hierarchical triangulation, and we show its application in representing topographic surfaces in the context of a geographic application, and in visualizing landscapes at variable resolution.

Hierarchical triangulations allow representing a virtually continuous range of resolutions, while still achieving a good compression ratio; moreover, their tree-like structure makes them suitable for local processing and distributed storage. Endowed with their access and manipulation algorithms, such structures provide efficient means to support complex operations: geometric queries such as point location, and segment and region intersecion; navigation of the structure (browsing) both through a representation at fixed resolution (pan) and across different resolutions (zoom); variable resolution rendering.

The remainder of the paper is organized as follows. In Section 2, we briefly review some existing hierarchical models for multiresolution surface representation. In Sections 3.1, the hierarchical triangulation is introduced and its properties are discussed, while in Section 3.2, a data structure for encoding an HT is described. In Section 3.3, techniques for navigating an HT are described. In Section 3.4, a general algorithm for building an HT is proposed, which is parametric on conditions and criteria that might be typical of any specific application. In Section 3.5, an algorithm for expanding a subhierarchy of an HT is described and analyzed. In Section 4, we discuss, through some examples, important characteristics of hierarchical triangulations. In Section 5, we formally introduce a general multiresolution surface model based on a hierarchical triangulation, while in Section 6, a specific multiresolution model that we have developed for terrain representation is described, together with a construction algorithm that is based on the general technique described in Section 3.4. In Section 7, we show techniques for extracting terrain representations at variable resolution over the domain. Finally, in Section 8, we present some results on the multiresolution representation of a natural terrain by using our model, while in Section 9, we present some concluding remarks, and we outline briefly our current and future research on this subject.

2 Previous Work

Hierarchical surface models proposed in the literature have been developed mainly for representing terrains in the context of geographical information systems. In our brief review, we will focus on domain subdivisions, by assuming that the elevation of the surface is known at all vertices of a subdivision, and for all triangular and quadrilateral subdivisions linear and bilinear interpolation are used, respectively. Higher order approximating functions could be used in the context of any hierarchical model, without changing the main principles that characterize its hierarchical structure.

Quadtree-based models require regularly spaced data and provide a subdivision of the domain into rectangles of different sizes. Such models are based on the recursive partition of a rectangle enclosing the projections of all data points into a set of nested rectangles, called quadrants, having vertices at such projections. Whenever the approximation of surface within a quadrant is not sufficiently precise, such quadrant is recursively split into four subrectangles by joining its center to the midpoints of its edges. Subquadrants are linked to their parent quadrant, giving rise to a quadtree. In [Che86], different interpolation techniques for approximating a surface defined by a quadtree are evaluated in terms of precision, computational speed and storage cost. The problem with such interpolants is the difficulty in preserving the continuity of the surface approximation along the sides of the subdivision. Von Herzen and Barr propose a method for avoiding discontinuities [Von87], called the *restricted quadtree*. Such model is based on a quadtree where all adjacent leaves cannot differ by more than one level in the hierarchy; each quadrant in the quadtree is then subdivided into triangles to achieve continuity at the borders of adjacent regions and to allow linear interpolation within each patch.

Quaternary triangulations are conceptually similar to quadtree-based models: the surface is approximated first by a plane interpolating data at the vertices of an initial triangle covering the whole domain; then, the domain is recursively partitioned into four subtriangles, by joining the midpoints of its edges. Like in quadtree-based models, the splitting rule is geometrically-fixed, regularly spaced data are needed, and the hierarchy can be represented by a quaternary tree. Quaternary triangulations suffer from the same discontinuity problems at the boundaries of adjacent triangles as quadtree-based models. Quaternary triangulations of plane domains have been widely used as domain partitions for surface representation [Gom79, Bar84], and for the finite element method [Ban86]; generalizations of such a partition scheme to spherical domains have been used for various applications [Fek84, Dyn90, Pot90, Goo92].

The *ternary triangulation* was proposed in [DeF84] as a hierarchical model for terrain description. In such a model, an initial triangle is recursively split into three subtriangles by joining its vertices with the internal data point at which the surface it approximated worst. Thus, a hierarchy described by a ternary tree is built. The ternary triangulation does not require regularly spaced data, and it provides a continuous representation of the surface by using linear interpolation, but it has a main disadvantage in producing a subdivision made of triangles with elongated shape that leads to inaccuracies in numerical interpolation. This undesirable feature was partially overcome by a *heuristic hierarchical triangulation* proposed in [Sca92], where the splitting rule permits introducing new points either inside and/or on the border of a triangle.

The *Delaunay pyramid* described in [DeF89] is a multiresolution triangle-based model whose structure is completely different from all models described above. The Delaunay pyramid is composed of an ordered sequence of Delaunay triangulations of the whole domain, each of which contains an increasing number of vertices. The Delaunay pyramid introduced for the first time the idea of explicit multiresolution through a decreasing sequence of tolerance thresholds given a priori: each level in the pyramid corresponds to a decreasing threshold, bounding the error in approximating the surface. Any two consecutive levels in the pyramid are connected by a set of links, joining the triangles modified between the two levels. The use of a Delaunay triangulation over the whole domain permits to control the shape of triangles, without requiring regularly distributed data. On the other hand, the hierarchical structure of a Delaunay pyramid cannot be described by a tree, since a triangle belonging to the triangulation at a given level can intersect a portion of the domain covered by several triangles in the next level. This fact introduces an important drawback in terms of storage cost, due do the high number of links between consecutive levels. Moreover, since such a model is not based on a strict spatial hierarchy, operations such as independent storage and processing of different areas, focusing on areas of interest, and extraction of representations at different resolutions over different areas, are either difficult or prevented a priori.

The quaternary, the ternary, and the heuristic hierarchical triangulations will be discussed more extensively in Section 4 in the framework of the general model presented in the next Section.

3 Hierarchical Triangulation

In this Section, we introduce the concept of hierarchical triangulation of a set of points in the plane, we describe a data structure for encoding it, and we give algorithms for its construction, traversal, and expansion at a given degree of resolution.

3.1 Basic definitions

Let $V = \{v_1, \ldots, v_n\}$ be a finite set of points in the Euclidean plane $\mathbb{E}^2$. A *triangulation* of set V is a maximal plane straight-line graph $G = (V, E)$, where E is a set of non-crossing line segments with endpoints in V [Pre85]. A triangulation of V can be expressed as a triple $\tau = (V, E, T)$, where T is the set of triangular faces induced by G on the plane. The faces of T cover the interior of the convex hull of V; such region will be called the *domain* of triangulation τ, and denoted $D(\tau)$. We will also call any edge e of τ, which lies on the boundary of $D(T)$, a *boundary edge* of τ.

A hierarchical triangulation is defined by applying the concept of recursive refinement to a triangulation. Intuitively, given a triangulation τ, any of its triangles can be seen as an individual entity. A triangle t of τ can be refined at a higher degree of resolution into a triangulation τ', whose domain covers t, by adding new vertices either inside t or on its sides. Each triangle of a given triangulation is refined independently of the others. The recursive application of the refinement process results in a hierarchy of triangulations $\mathcal{H}$ that satisfies the following *hierarchy rule*:

> each triangulation in $\mathcal{H}$ is formed of more than one triangle, and, with the exception of the root, it is the refinement of a triangle belonging to some other triangulation in $\mathcal{H}$.

Formally, let $\mathcal{T} = \{\tau_0, \ldots, \tau_h\}$ be a collection of triangulations such that $\forall j = 0, \ldots, h$, $\tau_j = (V_j, E_j, T_j)$, with $\#T_j > 1$, and $\forall j > 0$ there exists a unique triangle $t_j \in T_i$ for some $i < j$, such that $t_j \equiv D(\tau_j)$. A *hierarchical triangulation (HT)* is a triple $\mathcal{H} = (\mathcal{T}, \mathcal{E}, \ell)$, where

$$\mathcal{E} = \{(\tau_i, \tau_j) \mid \tau_i, \tau_j \in \mathcal{T},\ \exists t_j \in T_i,\ t_j \equiv D(\tau_j)\}, \text{ and}$$
$$\ell : \mathcal{E} \longrightarrow \cup_{i=0}^{h} T_i \text{ is defined } \ \ell(\tau_i, \tau_j) = t_j \ \textbf{if} \ \ t_j \in T_i \ \text{ and } \ t_j \equiv D(\tau_j).$$

If the elements of $\mathcal{T}$ are considered as atomic entities, then the triple $(\mathcal{T}, \mathcal{E}, \ell)$ can be regarded as a tree having $\mathcal{T}$ as its set of nodes, and labeled arcs in $\mathcal{E}$, which will be called the *tree describing* $\mathcal{H}$. For each $(\tau_i, \tau_j) \in \mathcal{E}$, $\ell(\tau_i, \tau_j)$ is called the *label* of (τ_i, τ_j), and any triangle in $\ell(\mathcal{E})$ is called a *macrotriangle*, to denote that its internal structure is specified further in the hierarchical triangulation; conversely, a triangle not in $\ell(\mathcal{E})$ is called a *simple* triangle. We introduce the following notation on $\mathcal{H}$:

- $\forall j > 0$:
  - $Par(\tau_j) = \tau_i \in \mathcal{T} \mid (\tau_i, \tau_j) \in \mathcal{E}$ is called the *parent* of τ_j;
  - $DAbs(\tau_j) = \ell(Par(\tau_j), \tau_j)$ is called the *direct abstraction* of τ_j;
  - $Anc(\tau_j) = \{\tau_i \in \mathcal{T} \mid$ there exists a simple path in $\mathcal{H}$ from τ_i to $\tau_j\}$ is called the set of *ancestors* of τ_j;
- $\forall \tau_i$ internal node of $\mathcal{H}$:
  - $Chi(\tau_i) = \{\tau_j \in \mathcal{T} \mid (\tau_i, \tau_j) \in \mathcal{E}\}$ is called the set of *children* of τ_i;
  - $Des(\tau_i) = \{\tau_j \in \mathcal{T} \mid$ there exists a simple path in $\mathcal{H}$ from τ_i to $\tau_j\}$ is called the set of *descendants* of τ_i;
- $\forall t_j \in \ell(\mathcal{E})$:
  - $DRef(t_j) = (\tau_j \in \mathcal{T} \mid t_j = DAbs(\tau_j))$ is called the *direct refinement* of t_j.

Note that the refinement of a macrotriangle can cause splitting its edges into chains of edges through insertion of new vertices. Since any non-boundary edge e is shared by two adjacent triangles, we investigate next what is the effect on $\mathcal{H}$ of refining e. To this aim, we introduce the following definitions: two triangulations $\tau_i, \tau_j \in \mathcal{T}$ are said to be *adjacent* along a straight-line segment l if their domains intersect only along l; τ_i and τ_j are said to be *matching* along l if they are adjacent along l and $l \cap V_i \equiv l \cap V_j$, i.e., their boundary edges along l are pairwise coincident; τ_i and τ_j are said to be *weakly matching* along l if they are adjacent along l and either $l \cap V_i \subseteq l \cap V_j$, or $l \cap V_j \subseteq l \cap V_i$, i.e., the boundary edges along l of one of them form a subdivision of the boundary edges along l of the other one.

Let us consider two macrotriangles t_j and t_k of a triangulation $\tau_i \in \mathcal{T}$, such that t_j and t_k are adjacent along an edge e. In $DRef(t_j)$ and $DRef(t_k)$, some new vertices might be inserted on e; however, the rules mentioned above do not guarantee that *the same* vertices are inserted on e in both refinements. The two direct refinements of t_j and t_k are triangulations adjacent along e. If e is split differently in the two cases, such triangulations are not matching, and their union is not a triangulation, though it is still a plane subdivision composed of triangles (see Figure 1 a-b). Even when e is split consistently in refining t_j and t_k, the matching could be lost in the

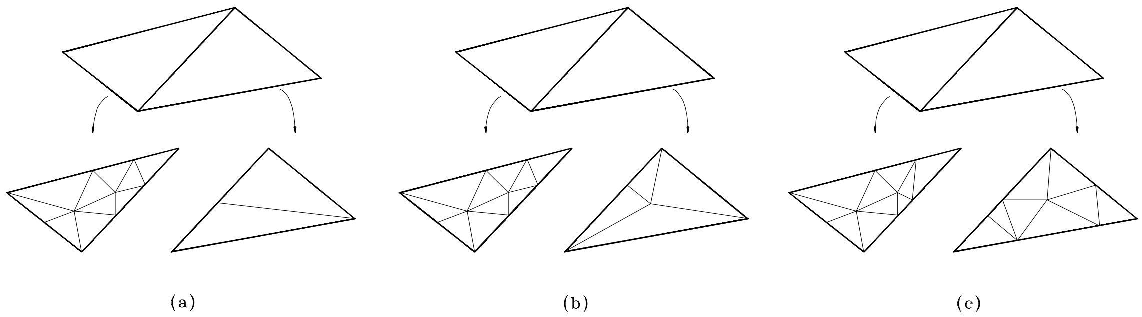


Figure 1: Two adjacent triangles and three possible refinements: not matching (a), weakly matching (b), matching (c).

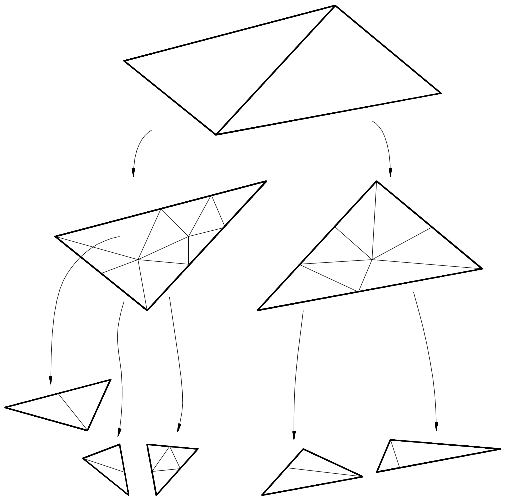


Figure 2: A tree describing a hierarchical matching triangulation.

next levels of the hierarchy, when refining triangles in descendants of τ_i that are adjacent along e. This fact is undesirable, because the continuity of adjacent patches could not be guaranteed when a hierarchical triangulation is used for surface description (see Section 5).

Our aim is to have a model in which two adjacent triangles are always refined by inserting the same vertices on their common edge, and which maintains this property throughout the whole hierarchy (see Figure 1c). Thus, we impose the following *matching rule*:

1. for each pair of triangulations $\tau_i, \tau_j \in \mathcal{T}$ adjacent along a straight-line segment l, one of the following conditions must hold:
   (a) τ_i and τ_j are matching along l;
   (b) $l \cap V_i \subset l \cap V_j$ and there exists $\tau_h \in Des(\tau_i)$ that is matching with τ_j along l;
   (c) $l \cap V_i \supset l \cap V_j$ and there exists $\tau_h \in Des(\tau_j)$ that is matching with τ_i along l;
2. for every simple triangle t_j of some $\tau_i \in \mathcal{T}$, the edges of t_j are never refined further in $\mathcal{H}$.

A hierarchical triangulation satisfying the matching rule is called a *hierarchical matching triangulation (HMT)* (see Figure 2). Throughout the paper, we will consider especially matching

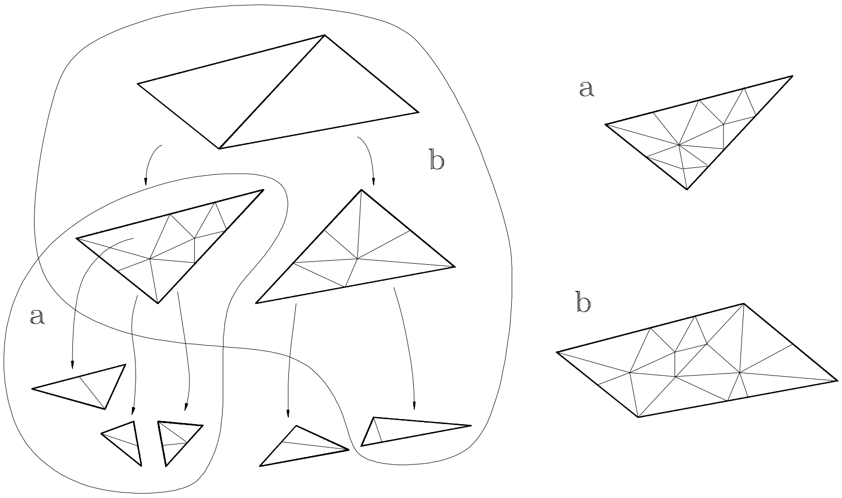


Figure 3: Consistent substructures of an HMT and their expanded triangulations.

hierarchies. However, when dealing with non-matching ones, we will always assume that a hierarchical triangulation is at least *weakly matching*, i.e., that situations like the one depicted in Figure 1a never occur. In Section 4.2, we will show a widely used class of hierarchical triangulations that are not necessarily matching, but are weakly matching.

A hierarchical triangulation $\mathcal{H}$ can be seen as a compact representation of a whole family of triangulations, namely all triangulations that are composed from triangles of $\mathcal{H}$.

Let us consider the tree describing a hierarchical triangulation $\mathcal{H}$, and let $\mathcal{H}'$ be a hierarchical triangulation described by a connected subgraph of such a tree, rooted at a triangulation $\tau_i \in \mathcal{T}$. We can apply a recursive expansion process that replaces each macrotriangle in $\mathcal{H}'$ with the triangulation refining it. It is easy to see that if $\mathcal{H}'$ is an HMT, then the result of the recursive expansion will be a triangulation $\mathcal{H}'_E$ covering $D(\tau_i)$, and composed of all triangles that are simple in $\mathcal{H}'$. In this case, $\mathcal{H}'$ will be called a *consistent substructure* of $\mathcal{H}$ (see Figure 3).

However, not all substructures of an HMT are consistent. In general, the expansion of a substructure $\mathcal{H}'$ will be a tessellation of $D(\tau_i)$, such that each of its faces is a *generalized triangle* whose domain corresponds to a simple triangle of $\mathcal{H}'$, according to the following definition:

> A *generalized triangle* t is a simple polygon defined by a sequence of vertices $V_t = (v_0, \ldots, v_i)$ such that there exist $v_{j_1}, v_{j_2}, v_{j_3} \in V_t$ whose convex hull coincides with the region covered by t.

In other words, a generalized triangle is a triangle in which some vertices are added to subdivide its edges into chains. A plane tessellation whose faces are all generalized triangles is called a *generalized triangulation* (see Figure 4).

We will always call $\mathcal{H}'_E$ the *expanded triangulation* associated with $\mathcal{H}'$, although in some cases it can be a generalized triangulation. In Section 3.5 we present an algorithm for computing the expanded triangulation associated with a substructure of a hierarchical triangulation.

We show next that the structure of a hierarchical triangulation $\mathcal{H}$ does not introduce a

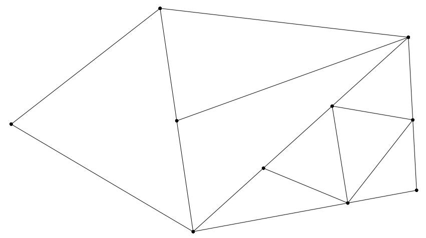


Figure 4: A generalized triangulation.

serious overhead with respect to its expanded triangulation $\mathcal{H}_E$. Let us call *size* of a hierarchical triangulation $\mathcal{H}$ the total number N_t of triangles in the triangulations of $\mathcal{T}$ (also the total number of vertices and edges in $\mathcal{H}$ is linear in N_t). In the following, we prove that N_t is smaller than twice the number N_s of simple triangles in $\mathcal{H}$ (i.e., the number of triangles in the expanded triangulation $\mathcal{H}_E$).

The result is proven by induction on the height of the tree describing $\mathcal{H}$, and it is based on the fact that each macrotriangle is refined into a non-trivial triangulation, i.e., a triangulation composed of at least two triangles. For each k smaller than the height of the tree describing $\mathcal{H}$, let $\mathcal{H}^{(k)}$ be the substructure corresponding to the first k levels of such tree, and let $N_t^{(k)}$ and $N_s^{(k)}$ be the total number of triangles and the number of simple triangles in $\mathcal{H}^{(k)}$, respectively. We have trivially $N_t^{(0)} = N_s^{(0)}$. Let us assume that $N_t^{(k)} < 2N_s^{(k)}$. $\mathcal{H}^{(k+1)}$ is obtained from $\mathcal{H}^{(k)}$ by refining $m^{(k+1)} < N_s^{(k)}$ simple triangles of $\mathcal{H}^{(k)}$ into $m^{(k+1)}$ new triangulations, yielding a total number $n^{(k+1)}$ of new triangles. Notice that the refined triangles will not be simple triangles in $\mathcal{H}^{(k+1)}$. Thus, we have the following:

(1) $N_t^{(k+1)} = N_t^{(k)} + n^{(k+1)}$;

(2) $N_s^{(k+1)} = N_s^{(k)} + n^{(k+1)} - m^{(k+1)}$;

(3) $n^{(k+1)} \geq 2m^{(k+1)}$.

By substituting (3) into (2) we obtain: $2N_s^{(k+1)} \geq 2N_s^{(k)} + n^{(k+1)} > N_t^{(k)} + n^{(k+1)}$, and the thesis follows from (1).

It is easy to see that the overhead factor depends of the minimum number of triangles present in a triangulation of $\mathcal{T}$. The definition given in Section 3.1 imposes that this number must be at least two. In general, if a hierarchical triangulation $\mathcal{H}$ is such that $\forall i = 0 \dots h \ \#T_i \geq m$, for some $m \geq 2$, then we have $N_t \leq \frac{m}{m-1} N_s$.

3.2 Encoding a hierarchical triangulation

A hierarchical triangulation $\mathcal{H} = (\mathcal{T}, \mathcal{E}, \ell)$ is a combination of topological structures - the triangulations of $\mathcal{T} = \{\tau_0, \dots, \tau_h\}$ - that are organized in a tree, whose edges are specified by $\mathcal{E}$ and labeled by ℓ. Here, we propose a relational representation of an HT, called the *hierarchical incidence graph.* Such data structure encodes both local and hierarchical relations, and permits navigation of the structure by supporting efficient retrieval of adjacencies between triangulations sharing common boundaries.

We first describe a graph structure that represents each single component τ_i of $\mathcal{T}$ by directly encoding its entities and the incidence relations among them. Then, we extend such representation with boundary information that relate τ_i with the exterior of its domain. Finally, we organize all local structures into a global structure that represents the hierarchical triangulation $\mathcal{H}$.

In [Ede87] a directed graph, called the *incidence graph*, is introduced to represent planar subdivisions: an incidence graph is a graph having one node per vertex, edge, and face of the subdivision; arcs emanating from vertices connect them to edges incident upon them; arcs emanating from edges connect them to their endpoints, and to faces incident upon them; arcs emanating from faces connect them to their bounding edges. An incidence graph describes the *symmetric* data structure proposed in [Woo85] for representing the subdivision of the boundary of a solid object. Such structure has been proven to be optimal, i.e., all topological relations that are not explicitly encoded can be retrieved in a time that is linear in their output size.

In the special case of a triangulation, the number of edges of each internal face in the subdivision is bounded by a constant (namely, three). This fact can be exploited to design a simpler, yet optimal, relational structure, that we call the *partial incidence graph*. In the following, we formally define the partial incidence graph for the representation of a triangulation of $\mathcal{T}$.

Let $\tau_i = (V_i, E_i, T_i)$ be a triangulation in $\mathcal{T}$, and let N_{V_i}, N_{E_i}, and N_{T_i} be three sets of *nodes* corresponding, through bijections η_V, η_E, and η_T, to the elements of V_i, E_i, and T_i, respectively. Let us define the following sets of directed *incidences*:

- $ET_i = \{(n_e, n_t) \in N_{E_i} \times N_{T_i} \mid \eta_E(n_e) \text{ is an edge of } \eta_T(n_t)\}$;
- $TE_i = \{(n_t, n_e) \in N_{T_i} \times N_{E_i} \mid \eta_E(n_e) \text{ is an edge of } \eta_T(n_t)\}$;
- $EV_i = \{(n_e, n_v) \in N_{E_i} \times N_{V_i} \mid \eta_V(n_v) \text{ is a vertex of } \eta_E(n_e)\}$;
- $VE_i = \{(n_v, n_e) \in N_{V_i} \times N_{E_i} \mid \eta_V(n_v) \text{ is a vertex of } \eta_E(n_e)\}$.

Finally, let us consider a subset $VE_i^* \subset VE_i$ obtained by taking only two incidences[1] per node of N_{V_i}. The *partial incidence graph* of τ_i is a tripartite directed graph $\mathcal{IG}_i = (N_i, I_i)$, where

- $N_i = (N_{V_i}, N_{E_i}, N_{T_i})$
- $I_i = (ET_i, TE_i, EV_i, VE_i^*)$.

Unlike the general incidence graph (where the complete VE_i relation is encoded), the number of arcs emanating from any node of a partial incidence graph $\mathcal{IG}$ is bounded by a constant, while the number of incident arcs can be arbitrarily numerous for nodes in N_{V_i} (in generalized triangulations, also for nodes in N_{T_i}). More precisely, if we consider the partition of N_{E_i} into two disjoint subsets N_{BE_i} and N_{IE_i}, corresponding to the edges on the boundary and in the interior of τ_i, respectively, we have: for any node of N_{V_i}, two outgoing arcs in VE_i^*; for any node

[1] In fact, a single incidence would be sufficient in order to store a triangulation. We use two incidences to allow the same structure to encode also generalized triangulations, and to maintain boundary sequences described in the following.

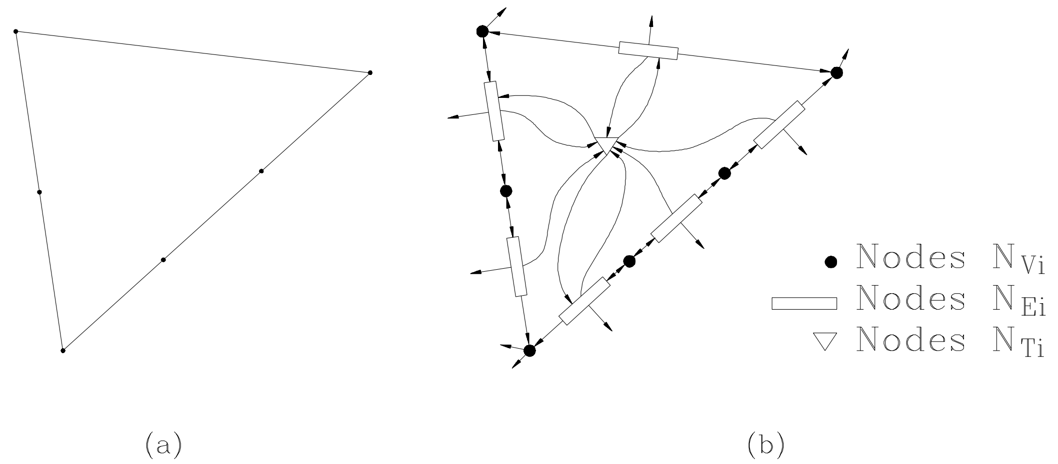


Figure 5: A generalized triangle (a) and its corresponding partial incidence graph (b): dangling arcs outgoing from nodes N_{E_i} are pointing to nodes N_{T_i} corresponding to adjacent triangles; dangling arcs outgoing from nodes N_{V_i} are pointing to nodes N_{E_i} corresponding to other incident edges.

of N_{E_i}, two outgoing arcs in EV_i; for any node of N_{IE_i}, two outgoing arcs in ET_i; for any node of N_{BE_i}, one outgoing arc in ET_i; for any node of N_{T_i}, three outgoing arcs in TE_i.

This structure can be also used to encode a generalized triangulation (see Figure 5). The following observations are taken into account:

1. since a generalized triangle can have more than three edges, relation TE_i encode as above is not total: for each node $n_t \in N_{T_i}$, its three outgoing arcs are connected with nodes corresponding to the first edges that are met when moving in clockwise order around $\eta_T(n_t)$ from the its three corners, respectively;

2. for each vertex v, at most one generalized triangle t may exist such that v is a vertex, but not a corner, of t: for each node $n_v \in N_{V_i}$ corresponding to one such vertex its two outgoing edges are connected with the nodes corresponding to edges of t incident at v.

Therefore, for each generalized triangle, three ordered and doubly connected sequences of interlaced vertices and edges are maintained, which specify the boundary of the generalized triangle. Each such sequence can be accessed from the generalized triangle itself, and all relations that are not explicitly encoded can be retrieved efficiently.

In order to relate a (generalized) triangulation τ_i to the exterior of its domain, it is also useful to maintain an "interface" by encoding information about the boundary of τ_i. To this aim, consider the boundary edges around the border of $D(\tau_i)$: such edges form a circular list of edges B_i. More precisely, since all triangulations of $\mathcal{T}$ have a triangular domain[2], we partition B_i into three sublists as follows. Let e_{i_0}, e_{i_1}, and e_{i_2} denote the three sides of $D(\tau_i)$ respectively (for $i > 0$, e_{i_0}, e_{i_1}, and e_{i_2} are the edges of $DAbs(\tau_i)$ in $\mathcal{H}$): we define $B_i = (s_{i_0}, s_{i_1}, s_{i_2})$, where for $j = 0, 1, 2$, $s_{i_j} = (e_{i_{j_0}}, e_{i_{j_1}}, \ldots)$ is the ordered sequence of boundary edges of τ_i along side e_{i_j},

[2]For simplicity, we assume that also τ_0 has a triangular domain. The definition can be easily extended to any polygonal domain.

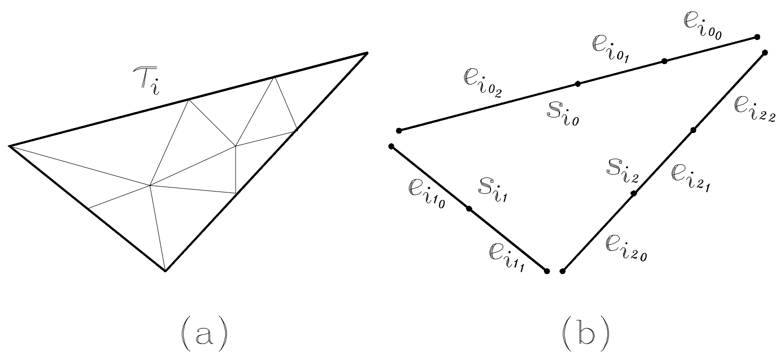


Figure 6: A triangulation (a) and its boundary sequences (b).

in counterclockwise order around $D(\tau_i)$ (see Figure 6). The three boundary sequences forming B_i are encoded easily in the partial incidence graph: for each node of N_{V_i} corresponding to a boundary vertex of τ_i, its two outgoing arcs are connected to nodes corresponding to the two boundary edges incident at the vertex, so as to form sequences of interlaced vertices and edges analogous to the ones maintained for generalized triangles.

We complete our structure by encoding hierarchy relations according to the arcs of $\mathcal{E}$ labeled by ℓ. Let $\mathcal{G} = (\mathcal{IGB}_0, \ldots, \mathcal{IGB}_h)$ be the collection of incidence graphs corresponding to the triangulations $(\tau_0, \ldots, \tau_h)$ of $\mathcal{T}$. The *hierarchical incidence graph* representing $\mathcal{H}$ is a labeled tree $\mathcal{HIG} = (\mathcal{G}, \mathcal{E}_g, \ell_g)$, where

$$\mathcal{E}_g = \{(\mathcal{IGB}_j, \mathcal{IGB}_i) \mid (\tau_i, \tau_j) \in \mathcal{E}\};$$
$$\ell_g : \mathcal{E}_g \longrightarrow \cup_{i=0}^{h} N_{T_i} \text{ is defined as } \ell_g(\mathcal{IGB}_j, \mathcal{IGB}_i) = \eta_T^{-1}(\ell(\tau_i, \tau_j)).$$

Hence, $\mathcal{HIG}$ is a labeled hierarchy of graphs, where each label associates a node corresponding to a macrotriangle with the graph describing its internal structure. Note that the arcs of $\mathcal{E}_g$ are directed upwards in the hierarchy, thus encoding a child-parent relation, in order to have only one outgoing arc per node of $\mathcal{G}$.

Encoding a hierarchical incidence graph is straightforward: one record per node of $\mathcal{HIG}$ is stored, which contains one pointer corresponding to its outgoing arc; each pointer is set to the address of the other end node of the arc. The label function ℓ_g is encoded by setting a pointer from each graph $\mathcal{IGB}_j$ $(j > 0)$ to the node corresponding to the direct abstraction of τ_j, and a reverse pointer from such a node to $\mathcal{IGB}_j$.

3.3 Neighbor finding

The data structure just described supports efficient implementation of many access operations on a hierarchical triangulation. In particular, information encoded in each partial incidence graph permits to navigate through a single node of the hierarchy, while hierarchy links permit to move across different levels of the tree.

A less immediate operation is navigation through triangles that are adjacent in space, but belong to different triangulations of the hierarchy. Such an operation is called *neighbor finding*, and it is formalized as follows.

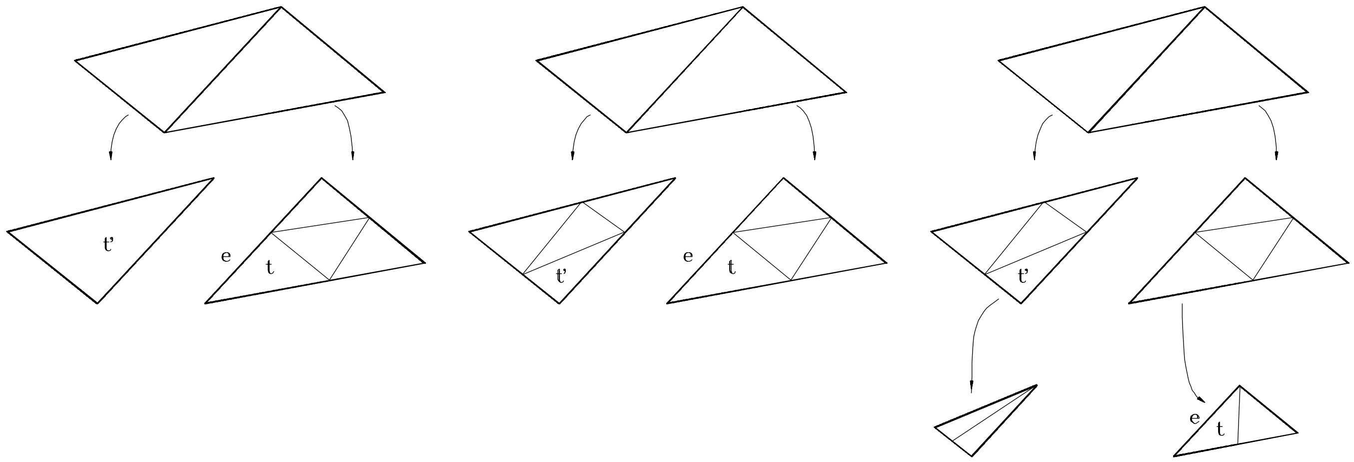


Figure 7: The three possible situations in neighbor finding.

> Let t be a boundary triangle of a triangulation $\tau_i \in \mathcal{T}$, and let e be a boundary edge of t: find a triangle t' of $\mathcal{H}$ that is adjacent to t on the other side of e.

Triangle t' will belong to some other triangulation τ_j in the hierarchy. Three situations may occur:

(a) in case the hierarchy is not an HMT, there might exist no triangle that is matching with t along e, i.e., all triangles on the other side of e might be "coarser" than t: in this case, t' will be selected as the finest triangle in the hierarchy that is adjacent to t along e and covers e with one of its edges (see Figure 7a);

(b) there might exist exactly one triangle t' that is matching with t along e (see figure 7b);

(c) in case edge e "survives" across different levels of the hierarchy without being refined, there might exist more than one triangle that is matching with t along e: in this case, t' is selected as the one among all possible candidates, which lies at the hieghest level in the hierarchy: it is straightforward to trace all other candidates through the data structure (see figure 7c).

Neighbor finding can be implemented on the basis of the data structure just described, through suitable algorithms, similar to the well known algorithms for neighbor finding on quadtrees [Sam90]. The main principle adopted by such techniques is the following. The hierarchy is traversed bottom-up starting at t, until an ancestor of t is found, having the corresponding ancestor of e as an internal edge. Then, such an edge is crossed, and the hierarchy is traversed top-down on the other side of it until t' is found. In order to do that, a stack containing ancestors of e is maintained, with respect to the boundary lists of triangulations traversed when moving bottom-up. Such a stack permits to trace back the path in the corresponding sequences when moving top-down.

However, we prefer here a more direct and faster implementation, which is based on an extension of the data structure described in Section 3.2 that can be obtained at virtually no cost.

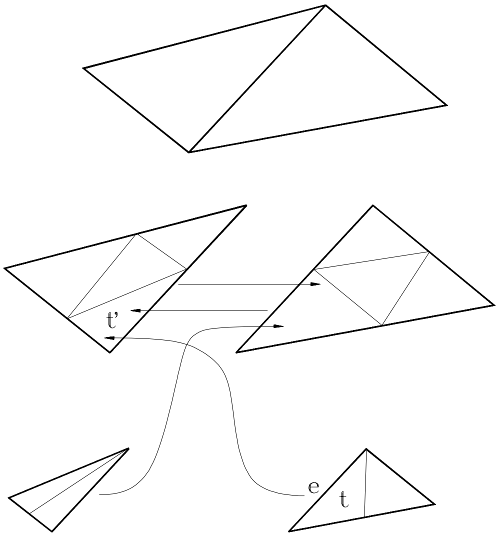


Figure 8: Ropes between neighbors along edge e in the HMT of Figure 7c (hierarchy links are omitted).

Let us consider the partial incidence graph just described. We can observe a lack of symmetry between nodes corresponding to internal edges, and nodes corresponding to boundary edges: indeed, while nodes of type N_{IE_i} have two outgoing arcs in ET_i, nodes of type N_{BE_i} have only one such outgoing arc. We can use the "free" entry on a node of type N_{BE_i}, corresponding to a boundary edge e of a triangle t of τ_i, to set an arc from it to a node of type N_{T_j} corresponding to the neighbor t' of t along e. Such links are called *ropes*, for analogy with similar data structures proposed for quadtrees [Sam90]. According to the above discussion, in case there exist more than one admissible neighbor for triangle t, the rope will connect e to the triangle t' among candidate neighbors, which lies at the highest level in the hierarchy. All other admissible neighbors of t belong to direct descendants of t', and the boundary sequence corresponding to e in each such descendant is formed by only one edge (coinciding with e). All such admissible neighbors can be accessed easily through hierarchy links. Note that the rope from the edge of t' coinciding with e will not necessarily be connected to t (see Figure 8).

3.4 Building a hierarchical triangulation

Let $\Omega \subset \mathbb{E}^2$ be a convex polygonal region in the plane, and let $\mathcal{S} \subseteq \Omega$ a set of points of Ω, not necessarily finite or discrete, called the *data set*. For simplicity, we assume that $\mathcal{S}$ contains at least all vertices of the convex hull of Ω. We are interested in building a hierarchical triangulation $\mathcal{H}$ with vertices in $\mathcal{S}$ that covers Ω. As we require $D(\tau_0) \equiv \Omega$, at least all vertices of the convex hull of Ω will be vertices of τ_0. We avoid making here further assumptions on which points of $\mathcal{S}$ will be used as vertices of the triangulations of $\mathcal{H}$, since such choice is strictly dependent of the specific application.

On the other hand, in order to be not too abstract, we will keep in mind the following concept. We assume that some information is "attached" to the points of Ω and that such information is known (or can be computed/retrieved) for all points of $\mathcal{S}$. Given a triangulation $\tau = (V, E, T)$, whose domain is contained in Ω, we assume that for every $t \in T$, information about all points covered by t will be approximated in a more synthetic form by some other information attached to t (e.g., a function). We can expect that the amount of information stored in the model will increase as the domain subdivision is refined (i.e., as more and more triangles are used to cover the domain), and that the quality of the approximation will improve as a consequence.

As we want to focus on the construction of the domain subdivision, while remaining generic about the information represented, we assume that a set of Boolean *conditions* is given: $C_A = \{c_\alpha \mid \alpha \in A\}$, where A is a totally ordered set, not necessarily finite or countable, and C_A is such that $\forall \beta > \alpha,\ c_\beta \Rightarrow c_\alpha$. We denote by $c_\alpha(\tau)$ the Boolean value of condition c_α evaluated in triangulation τ, and we say that τ is at *degree of resolution* α if and only if $c_\alpha(\tau)$ is **true**. We naturally extend conditions to triangles as follows: for any triangle t, $c_\alpha(t)$ has the value of c_α evaluated in a triangulation containing only t. We assume the following consistency rule to hold: let τ be a triangulation formed by triangles $\{t_0, \ldots, t_k\}$, then $\wedge_{i=0}^{k} c_\alpha(t_i) \Rightarrow c_\alpha(\tau)$, i.e., a triangulation is at least at the same degree of resolution of the highest degree satisfied by all its triangles. Some examples presented in Section 4 show that the reverse is not always true. We also assume that the conditions in C are all *feasible* for data set $\mathcal{S}$, i.e., given a condition $c_\alpha \in C$ and a triangle t with vertices in $\mathcal{S}$, there always exists a finite triangulation refining t with vertices in $\mathcal{S}$ and satisfying c_α.

In the following we give first a generic algorithm for building a hierarchical triangulation which is based on successive refinements controlled by a sequence of conditions of the type C_I, where $I = \{0, 1, \ldots, d_{max}\}$ is a finite set of natural numbers. Next, we discuss a generalization of such an algorithm to a virtual sequence of the type C_A, where A is a continuous interval.

We define a hierarchical triangulation $\mathcal{H}$ based on C_I by induction: let $\mathcal{H}_0$ be a hierarchical triangulation formed by a single triangulation τ_0 satisfying c_0. For $i < d_{max}$, given hierarchical triangulation $\mathcal{H}_i$, the hierarchical triangulation $\mathcal{H}_{i+1}$ is obtained by refining all simple triangles of $\mathcal{H}_i$ that do not satisfy c_{i+1} into triangulations that do satisfy it. Finally, we define $\mathcal{H} \equiv \mathcal{H}_{d_{max}}$.

Note the difference between the definition above and the definition of $\mathcal{H}^{(i)}$ given in Section 3.1: while $\mathcal{H}^{(i)}$ is the subgraph corresponding to the first i levels of the tree describing $\mathcal{H}$, the hierarchical triangulation $\mathcal{H}_i$ is defined here only on the basis of its degree of resolution. Indeed, there is only a very weak relation between $\mathcal{H}_i$ and the first i levels of $\mathcal{H}$: $\mathcal{H}_i$ will have no more than i levels, but it does not necessarily contain components of $\mathcal{H}$ at level i. Indeed, any component of $\mathcal{H}$ at a level l can have a degree of resolution higher than l. This is due to the fact that a simple triangle can pass several conditions before being refined: if a triangle t belongs to a triangulation satisfying condition c_i, and t satisfies itself condition $c_{i'}$, for some $i' > i$, then t can be a simple triangle also in all hierarchies $\mathcal{H}_j$, for $i \leq j \leq i'$. This is the main reason why an HT performs much better in terms of data compression than a layered model. In Section 4 we will see some special HTs for which the level of a triangulation and its degree of resolution will necessarily coincide. Conversely, in Section 8 we will see that, in the model we propose for surface modeling, a very large number of degrees of resolution can be obtained with a tree having only few levels.

The construction algorithm sketched in the following is parametrized over the strategy for refining a single triangulation. A specialized algorithm for any given application can be derived from such a framework by instancing the suitable conditions and by deciding a refinement procedure to satisfy them (see Sections 4 and 6).

$\mathcal{H}$ is built according to a top-down approach that directly implements the inductive definition:

τ_0 is built first; then, for all triangles of τ_0 that do not satisfy condition c_1, their direct refinements are computed in such a way that they satisfy c_1, and so on, until the maximum degree d_{max} is reached. Let REFINE($\mathcal{S}_t, t, c$) be a function that, given a triangle t, a data set $\mathcal{S}_t$ (a restriction of data set $\mathcal{S}$ to triangle t), and a condition c, returns a triangulation with vertices in $\mathcal{S}_t$, whose domain coincides with t, and that satisfies c. The following general algorithm builds $\mathcal{H}$ based on such function:

Algorithm BUILD_HT($\mathcal{S}, C,$**out** $\mathcal{H}$)
begin
  build an initial triangulation τ_0 satisfying c_0;
  $\mathcal{T} \leftarrow \{\tau_0\}$;
  $\mathcal{E} \leftarrow \emptyset$;
  for $d = 1$ **to** d_{max} **do**
    for every $\tau \in \mathcal{T}$ containing simple triangles **do**
      for every simple triangle t of τ such that $c_d(t) ==$ **false do**
        $\tau' \leftarrow$ REFINE($\mathcal{S} \cap D(t), t, c_d$);
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau'\}$;
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\tau, \tau')\}$;
        $\ell(\tau, \tau') \leftarrow t$;
      end for
    end for
  end for
end

Procedure REFINE is the core of the algorithm that must select a finite subset of data $V_t \subseteq \mathcal{S}_t = \mathcal{S} \cap D(t)$, such that the triangulation τ' of V_t satisfies c_d. The technique adopted for selecting vertices is application dependent. Different selection and refinement techniques can yield different complexities for the algorithm BUILD_HT. In Sections 4 and 6, we will outline conditions and techniques applied for building some meaningful examples of hierarchical triangulations we have considered.

Let us consider now a set of conditions C_A, where A is an interval in $\mathbb{R}$, and let us assume to have a hierarchy $\mathcal{H}_\alpha$ whose simple triangles satisfy condition c_α, for some $\alpha \in A$. The algorithm above can be adapted to refine $\mathcal{H}_\alpha$ further. In this case, procedure REFINE will be asked to refine each simple triangle that does not satisfy any condition $c_{\alpha'}$, for $\alpha' > \alpha$, into a triangulation satisfying a condition $c_{\alpha+d\alpha}$, for some $d\alpha$ arbitrarily small. In fact, each refinement will not cause an infinitesimal improvement, but a finite one. Let $\overline{\alpha} > \alpha$ be the maximum value of A such that all simple triangles of the new hierarchy all satisfy $c_{\overline{\alpha}}$. Such number always exists since there are finitely many simple triangles refined at each step, and finitely many triangles in their direct refinements. Then, $\overline{\alpha}$ is the next degree of resolution achieved. In practice, this mechanism allows us to approximate a continuous range of resolutions through a discrete, but arbitrarily numerous, set of degrees of resolution, which is not given a priori, but is determined while refinement occurs.

Note that, while all triangles are refined independently, some vertices introduced when refining a triangle t could lie on its edges. If we want to obtain a hierarchical matching triangulation, we must impose conditions and use a strategy that allows us to satisfy the matching rule during refinement. In other words, we must guarantee that, for any pair of triangles t_i and t_j adjacent along an edge e at degree $(d-1)$, which are refined at degree d, their corresponding direct refinements at degree d will be matching triangulations along e; also, if only one of the two triangles is refined at degree d, its direct refinement will not have any new vertex on e. This means that REFINE must select points on e independently of t_i and t_j.

3.5 Expanding a hierarchical triangulation

Let $\mathcal{H}$ be a hierarchical triangulation and let $\mathcal{H}'$ be a substructure of $\mathcal{H}$. As mentioned in Section 3.1, the expansion of $\mathcal{H}'$ is a generalized triangulation $\mathcal{H}'_E$ composed of all simple triangles in $\mathcal{H}'$. In the following, we present an algorithm for expanding $\mathcal{H}'$ into $\mathcal{H}'_E$. The algorithm, besides extracting the set of geometric entities forming $\mathcal{H}'_E$, constructs the topological structure of the subdivision, i.e., the incidence and adjacency relations among its entities and its boundary. Note that if $\mathcal{H}'$ is a consistent substructure of $\mathcal{H}$, then $\mathcal{H}'_E$ is a triangulation. The resulting triangulation (either generalized or not) is encoded through the partial incidence graph described in Section 3.2.

The general structure of the algorithm is a top-down traversal of $\mathcal{H}'$, based on recursive procedure EXPAND_SUBTREE, which takes as input a subtree rooted at a triangulation τ and returns a generalized triangulation τ_{exp} that expands such a subtree. Procedure EXPAND_SUBTREE performs the following three steps:

1. the set of triangles forming τ_{exp} is extracted, which is composed of all simple triangles of τ plus all triangulations expanding its macrotriangles;
2. the topological structure of τ_{exp} is constructed, based on the topological structure of τ and on the boundary sequences of the triangulations recursively extracted at step 1: triangles that do not match with their neighbors are replaced with generalized triangles by adding vertices on their boundary edges;
3. the three sequences of edges forming the boundary of τ_{exp} are extracted.

The expanded triangulation $\mathcal{H}'_E$ is then computed by applying EXPAND_SUBTREE to the root τ_r of $\mathcal{H}'$. We prefer not to specify $\mathcal{H}'$ explicitly as a procedure parameter. We will rather use the whole $\mathcal{H}$ as parameter, while identifying $\mathcal{H}'$ implicitly as follows:

- the root triangulation τ_r of $\mathcal{H}'$ is given;
- a Boolean function $\text{MACRO}_{\mathcal{H}'}(t)$ is given that, for each triangle t of $\mathcal{H}$, returns **true** if and only if t is a macrotriangle in $\mathcal{H}'$.

This is not merely a peculiar definition to make things more complicated; it is rather a way of allowing the algorithm to dynamically select a substructure $\mathcal{H}'$, unknown a priori, while the expansion occurs: in practice, the structure of $\mathcal{H}'$ is not necessarily specified by the user, but it is decided on the basis of a rule incorporated in $\text{MACRO}_{\mathcal{H}'}$.

An example of major importance is the following. Let us consider a hierarchical matching triangulation $\mathcal{H}$ built on a sequence of conditions $C_I = \{c_0, \ldots, c_{d_{max}}\}$ as in the previous Section. Given $i \leq d_{max}$, we want to expand $\mathcal{H}_i$ defined as in the previous Section. In this case, $\text{MACRO}_{\mathcal{H}_i}$ is defined as follows:

$\text{MACRO}_{\mathcal{H}_i}(t) :== \textbf{not } c_i(t)$.

In the following, we give a pseudo-code description of procedure EXPAND_SUBTREE that makes use of procedures EDGE_LINKING and BOUNDARY_RECONSTRUCTION implementing steps 2 and 3 described above. Within the pseudo-code, a list L_τ is used to store the expansions of macrotriangles; we assume that direct access to such list is possible through the triangles indexing its elements. List L_τ is used afterwards as a parameter for procedure EDGE_LINKING, which produces in turn a second list L_e of edge sequences, corresponding to the expansions of the boundary edges of the current triangulation. List L_e is passed as a parameter to procedure BOUNDARY_RECONSTRUCTION. Simple procedures acting on lists are used in the pseudo-code: procedure ADD_LIST adds one element to a list, and function EMPTY tells whether a list is empty or not.

```
procedure EXPAND_SUBTREE(H,MACRO_H',τ,out τ_exp)
begin
   for every triangle t of τ do
         if MACRO_H'(t) then
               EXPAND_SUBTREE(H,MACRO_H', DRef(t), τ_t);
               ADD_LIST(L_τ, τ_t);
         end if
   end for ;
   if EMPTY(L_τ) then
      τ_exp ← τ
   else
      EDGE_LINKING(τ, L_τ, τ_exp, L_e);
      BOUNDARY_RECONSTRUCTION(τ, L_e, τ_exp);
   end if ;
end
```

The pseudo-codes of procedures EDGE_LINKING and BOUNDARY_RECONSTRUCTION are omitted here for brevity, while their structure is detailed in the following. Procedure EDGE_LINKING, besides inserting all expanded triangles into τ_{exp}, replaces each non-matching triangle with a suitable generalized triangle, and it reconstructs the topological structure of τ_{exp} by identifying shared edges extracted while expanding adjacent triangles of τ. A list L_e of edge sequences is also produced as a side effect: each sequence in L_e is indexed by a boundary edge of τ, and corresponds to its expansion in τ_{exp}. In procedure EDGE_LINKING, edges of the current triangulation τ are considered in sequence; let e be one of such edges, and let (t_0, t_1) be its two adjacent triangles in τ. The following situations can occur:

1. Neither t_0 nor t_1 were expanded (see Figure 9a). Thus, e is maintained in τ_{exp}; relations involving only e, t_0, and t_1 are already present in τ, and they can be copied into τ_{exp}.
2. Only t_0 was expanded into a triangulation that matches with t_1 along e (see Figure 9b). In this case, the boundary sequence corresponding to e in τ_{t_0} will be formed by a single edge e^*, coincident with e. Thus, e and e^* are identified, and relations involving t_1, e, and the triangle of τ_{t_0} that is bounded by e^* are set accordingly. The situation is fully symmetric if only t_1 was expanded.
3. Only t_0 was expanded into a triangulation that does not match with t_1 along e (see Figure 9c). In this case, t_1 is transformed into a generalized triangle by substituting to e the

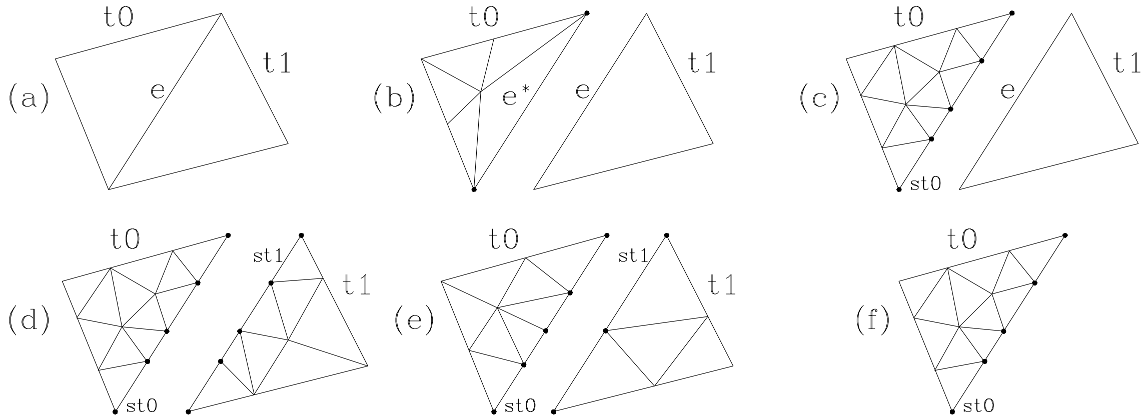


Figure 9: The six cases of edge linking.

boundary sequence corresponding to e in τ_{t_0}. The situation is fully symmetric if only t_1 was expanded.

4. Both t_0 and t_1 were expanded into matching triangulations (see Figure 9d). Thus, there exists two ordered sequences of boundary edges, s_{t_0} in τ_{t_0}, and s_{t_1} in τ_{t_1}, that correspond to e, and such that the elements of s_{t_0} and s_{t_1} are pairwise coincident. In this case, the two sequences are scanned in parallel, their edges are pairwise identified, and relations are set accordingly between such edges and their adjacent triangles in τ_{t_0} and τ_{t_1}.

5. Both t_0 and t_1 were expanded into triangulations that do not match (see Figure 9e), but triangulations τ_{t_0} and τ_{t_1} are always weakly matching, by definition. Without loss of generality, let us assume that τ_{t_0} is the one that refines e into more pieces. The two ordered sequences of boundary edges, s_{t_0} and s_{t_1} are scanned in parallel as in the previous case. For each element of s_{t_1}, if there exists a corresponding matching element in s_{t_1}, then identification occurs as in the previous case, otherwise there exists a subsequence of elements of s_{t_0} that correspond to the current element of s_{t_1}, and the corresponding triangle in τ_{t_1} is transformed into a generalized triangle as in case 3.

6. e is a boundary edge of τ (see Figure 9f). In this case only t_0 is defined. If t_0 has been expanded, then there is a boundary sequence of edges of τ_{t_0} corresponding to e: such a sequence is appended to L_e.

Note that if $\mathcal{H}'$ is a consistent substructure, then cases 3 and 5 never occur. The boundary of the expanded triangulation τ_{exp} is finally computed by procedure BOUNDARY_RECONSTRUCTION, on the basis of the boundary of the current triangulation τ, and of the list of expanded edges L_e computed by EDGE_LINKING. The sequences of edges (s_0, s_1, s_2) forming the boundary of τ are scanned sequentially, and the three corresponding lists forming the boundary of τ_{exp} are generated respectively. For each list s_i $(i = 0, 1, 2)$, edges of s_i are considered in sequence. For every edge $e \in s_i$, if e was expanded, then list l_e expanding e is taken from L_e and it is appended to s_{exp_i}, else e itself is appended to s_{exp_i}.

Procedure EXPAND_SUBTREE has a linear time complexity in the size of $\mathcal{H}'_E$, provided that function $\text{MACRO}_{\mathcal{H}'}(t)$ can be evaluated in constant time. We first observe that procedures EMPTY and ADD_LIST can both be implemented with constant time complexity (linked lists with pointers to both ends are used). We can evaluate the time complexity by globally counting the operations executed during the three steps of the algorithm: triangle extraction, edge linking, and boundary reconstruction. Recall that the number of elements composing $\mathcal{H}'$ is of the same order of the size of $\mathcal{H}'$.

1. During triangle extraction and edge linking, a triangle of $\mathcal{H}'$ is visited only twice, and the operations for each triangle are executed in constant time.
2. During edge linking, an edge e is visited at most twice: once in the main loop, when the appropriated sequences of edges must be identified and substituted for e in the expanded triangulation, and once as a part of one such sequence. It is important to note that, during a single call to procedure EDGE_LINKING, a sequence of edges expanding an edge e of the current triangulation is scanned only if e is an internal edge; otherwise the sequence is appended to L_e without visiting its elements. Also, each triangle of $\mathcal{H}'$ is visited at most three times during edge linking, one for each of its edges. Thus, a constant number of operations are executed for each visit of an edge or a triangle.
3. During boundary reconstruction, each edge of $\mathcal{H}'$ is visited only once, and a constant number of operations is executed each time: indeed, when substituting an edge e with the sequence s_e expanding it, s_e need not be scanned (linked lists with pointers to both ends are used to encode edge sequences).

As proven in Section 3.1, the size of $\mathcal{H}'$ is linear in the size of the expanded triangulation $\mathcal{H}'_E$. Thus, $\mathcal{H}'_E$ can be extracted from $\mathcal{H}'$ in time linear in the size of $\mathcal{H}'_E$.

4 Examples of Hierarchical Triangulations

As outlined in the introduction, the main purpose of a hierarchical triangulation is to serve as a multiresolution domain discretization for applications like, e.g., Geographical Information Systems (GISs), and Computer Aided Geometric Design (CAGD).

Several triangle-based structures proposed in the literature that we have reviewed in Section 2 can be formalized as hierarchical triangulations. In this Section, we give the formalization of such structures in the framework described in Section 3, through the conditions that define them, and the refinement strategies to build them. We also discuss, through such examples, some critical issues about alternative approaches that can be followed, and the need for tradeoffs between different requirements that are often hardly compatible. We consider the following main issues:

- *matching:* as we outlined in Section 3.1, hierarchical triangulations that fulfill the matching rule are desirable in order to preserve continuity;
- *adaptivity:* in several applications, especially when the data points are not very uniformly distributed, it is desirable to focus the refinement in areas of interest, while maintaining a coarse discretization in other areas;
- *shape:* it is widely recognized that a triangular domain discretization should be composed of triangles whose shapes are maintained as much regular as possible. In other words, long and thin (slivery) triangles should be avoided;

- *precision vs size:* the most natural way to conceive the degrees of resolution in a hierarchical triangulation is to associate a higher degree with a higher precision in representing data (explicit multiresolution). On the other hand, it is sometimes desirable to set conditions where a constant bound is imposed on the number of data introduced at each degree of resolution. In this case, the multiresolution of the model is only implicit, since an intermediate precision does not necessarily correspond to a given degree of resolution. As the total number of data used to achieve a certain precision cannot be bounded by a constant, implicit models, which set a constant bound on the size of components, might require an arbitrary number of degrees of resolution to achieve a given precision.

- *geometry-driven vs error-driven refinement:* the strategy used to refine triangles can be based on the use of a given geometric or topological pattern, or can be based on the attempt of achieving a better precision in approximating data. In general, the two approaches cannot coexist.

In Section 6, we will describe a model for multiresolution surface description based on a hierarchical Delaunay triangulation [DeF92]. The model we propose is matching, adaptive, and error-driven, it supports explicit multiresolution, and admits sparse data, while the shape of triangles is controlled by imposing the Delaunay criterion within each component of the hierarchical structure.

In the following Subsections, we use the following notations: given a triangulation τ, the number of triangles of τ is denoted $\#\tau$; given a triangle t, the portion of data set covered by t is denoted $\mathcal{S}_t$; whenever applicable, we assume that the error of a model in approximating the information attached to data can be measured by a real-valued function E, and we denote by $E(t)$ the error made in approximating the data points of $\mathcal{S}_t$ with triangle t, and by $E(t,p)$ the error made in approximating $p \in \mathcal{S}_t$ with t.

4.1 The ternary triangulation

A simple approach to hierarchical triangulation, called the *ternary triangulation*, was proposed in [DeF84] for the multiresolution representation of terrain in a GIS. In a ternary triangulation, a triangle t is refined by inserting a point inside it, and subdividing t into three new triangles (see Figure 10). Triangles that do not achieve a given precision, corresponding to an error tolerance ε, are refined recursively, until all simple triangles in the hierarchical structure satisfy $E(t) \leq \varepsilon$.

A ternary triangulation is defined by a sequence C_I of Boolean conditions that are all coincident:

$$\forall i,\ c_i(\tau) :== (\#\tau = 3) \vee (\forall t \in \tau,\ E(t) \leq \varepsilon).$$

In a ternary triangulation, the degree of resolution of a triangulation τ coincides with the level of τ in the tree describing the hierarchical structure, but it is not necessarily related with the approximation error of τ. Possible alternatives to the definition above are to stop refinement after the tree has reached a predefined number of levels, or when all data have been inserted as vertices.

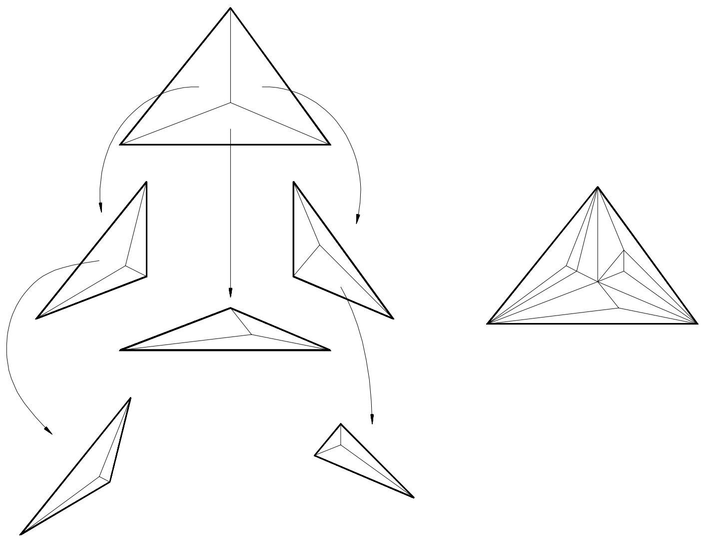


Figure 10: A ternary triangulation.

The refinement strategy for a triangle t consists in picking a point p inside t with maximum approximation error (i.e., the argmax of $E(t,p)$). A possible alternative is to pick the datum p that lies closest to the centroid of t.

The ternary triangulation is a matching hierarchical triangulation, since edges are never split during refinement. Note that this fact makes any subhierarchy of a triangulation consistent: therefore, any expanded subhierarchy of a ternary triangulation is a triangulation. The ternary triangulation is adaptive, since it does not impose a geometrically fixed rule for splitting. On the other hand, the refinement strategy is topology-driven, the precision is not guaranteed to improve at each refinement step, and multiresolution supported is only implicit. The main disadvantage of a ternary triangulation is that many slivery triangles are generated: after l levels of refinement, the smallest angle in the hierarchical triangulation will be not larger than $\frac{\pi 2^{-l}}{3}$. This fact is essentially due to the impossibility of splitting edges of existing triangles.

4.2 The k^2 triangulations

The *quaternary triangulation*, opposite to the ternary triangulation described above, is a hierarchical structure based on a geometrically fixed splitting rule, whose primary purpose is to preserve the regular shape of triangles. In a quaternary triangulation, a triangle t is refined into four new triangles by connecting the midpoints of the edges of t (see Figure 11). A generalization of this concept consists in refining t into k^2 new triangles (for $k \geq 2$), obtained by splitting each edge into k equal parts, then connecting pairs of splitting points through straight-line segments parallel to the edges of t, and adding all intersections of such lines as new vertices. The refinement of t is a triangulation made of k^2 new triangles. In one such hierarchical structure, all descendants of a triangle t have the same shape of t; thus, in order to ensure the quality of the resulting hierarchical triangulation, it is sufficient to build a good triangulation at the root.

As in the case of a ternary triangulation, the refinement can stop when a given precision is met. In this case, a similar sequence of conditions is obtained:

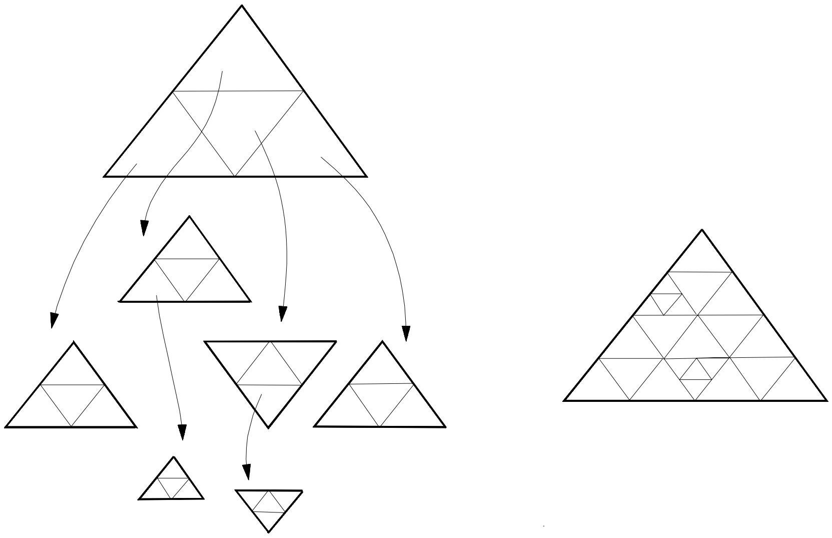


Figure 11: A quaternary triangulation.

$$\forall i,\ c_i(\tau) :== (\#\tau = k^2) \vee (\forall t \in \tau,\ E(t) \leq \varepsilon).$$

Possible alternatives are stopping refinement after a predefined number of levels, or when all data have been inserted. Again, the degree of resolution of a triangulation coincides with its level in the tree, but it is not related with its precision. In analogy with the case of quadtrees, the resolution is more directly related with the area of triangles: mode precisely, a triangle at level l of the hierarchy will have an area of $\frac{1}{4^l}$ times the area of its ancestor in the root triangulation. Note that, in this case, an algorithm for expanding the hierarchy at a given resolution is ruled by a procedure

$$\mathrm{MACRO}_{\mathcal{H}'}(t) :== \Delta(t) > \delta,$$

where Δ denotes the area of a triangle, while δ is a selected threshold area.

A k^2 triangulation is adaptive, and it is formed by nearly equilateral triangles, provided that triangles in the root are nearly equilateral. The k^2 triangulations follow a geometry-driven decomposition requiring uniformly distribuited data, and they support only implicit multiresolution. The main drawback of such structures is matching: as all edges of a triangle t are split when refining it, in order to fulfill the matching rule, a triangle can be refined if and only if all triangles of all triangulations at the same level of the tree are refined. Thus, either local refinement is not permitted, and the tree describing a hierarchical triangulation is full with k^2 children per internal node, or the resulting structure will not be matching. Similarly, only an expanded hierarchy in which all triangles are expanded at the same level of resolution will be a triangulation.

4.3 Heuristic hierarchical triangulation

The ternary and quaternary triangulations are two extreme examples of opposite approaches in driving the refinement, which can be summarized as follows: if new vertices are inserted

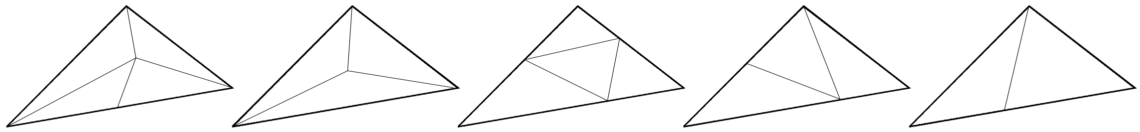


Figure 12: Splitting rules for the heuristic triangulation.

on all edges while refining a triangle t, even all neighbors of t must be refined to satisfy the matching rule, and local refinement is thus prevented; conversely, if t is refined by inserting only internal points, slivery triangles will be generated. In order to preserve the possibility for local refinements, heuristic techniques have been proposed that permit the refinement of some edges while leaving some other edges unchanged.

In [Sca92] an error-driven splitting rule has been proposed, which implements a hybrid strategy: a triangle t is refined by inserting points corresponding to the maximum approximation errors along its edges and/or in its interior. Five (out of seven) possible topological combinations are considered for splitting, yielding a subdivision of the triangle into two to four new triangles (see Figure 12). Splitting of triangles is performed iteratively within a given triangulation τ until the precision required for the given degree of resolution is met.

The heuristic hierarchical triangulation proposed in [Sca92] is built based on a decreasing sequence of tolerance values $\varepsilon_0 > \varepsilon_1 > \ldots > \varepsilon_{d_{max}}$, like the Delaunay pyramid described in Section 2. The heuristic hierarchical triangulation is thus characterized by the following conditions:

$$\forall i = 0, \ldots, d_{max},\ c_i(\tau) :== (\forall t \in \tau,\ E(t) \leq \varepsilon_i).$$

The procedure for refining a triangle t at a given degree of resolution i starts from a triangulation τ_t containing only t, and iteratively splits its triangles until $c_i(\tau_t)$ is satisfied. The heuristic hierarchical triangulation is matching, since the precision along the edges is tested independently of the precision inside triangles, and edges are split according to a fixed deterministic rule. This structure is also adaptive, hence permitting local refinement and sparse data, it is error-driven, and it supports explicit multiresolution. Being based on heuristics, there is no guarantee that the shape of triangles is well maintained.

Note that the set of conditions given above for the heuristic hierarchical triangulation is valid in general for any hierarchical triangulation supporting explicit multiresolution. Of course, the error function E is application dependent, and even for a given application, different approximation estimates can be obtained by using different functions (see Section 5). For a given error function E, the structure of the HT is thus completely dependent on the refinement procedure. A "good" refinement procedure should refine a triangle into a triangulation that contains as few new vertices as possible, and is composed of triangles with a good shape. Also, new vertices should be admitted both inside the triangle and on its edges, but vertices on edges should be not too numerous, in order to maintain the possibility of local refinement, while preserving matching.

5 Surface Representation Based on a Hierarchical Triangulation

A *nonparametric surface* in 3D (also called a $2\frac{1}{2}$D surface) is a compact and connected 2-manifold with boundary, described by the graph of a real-valued function of two variables defined over a compact and connected domain in the real plane. Formally, given a domain $\Omega \subseteq \mathbb{R}^2$, and a function $f : \Omega \to \mathbb{R}$ of class $C^k(\Omega)$, the surface (of class C^k) corresponding to f over Ω is $\psi(\Omega, f) = \{(x, y, f(x, y)) \mid (x, y) \in \Omega\}$. We call the pair $\mathcal{M} = (\Omega, f)$ a *mathematical surface model*.

In practical applications, it is hardly possible that a surface can be described through a single analytic function f over the whole domain Ω. A finite description can be given by tessellating Ω into regions, such that surface ψ can be described by a function over each region. Let Γ be a partition of the domain Ω into connected regions $\{\gamma_1, \ldots, \gamma_m\}$, and let $F = \{f_1, \ldots, f_m\}$ be a family of functions such that

- $\forall i = 1, \ldots, m, \quad f_i : \gamma_i \to \mathbb{R}$ is a function of class $C^k(\gamma_i)$;
- the function
$$\hat{f} = \sum_{i=1}^{m} f_i \chi_i$$
(where χ_i is the characteristic function of region γ_i) is of class $C^k(\Omega)$, i.e., all f_i's match on the borders between adjacent regions up to the k-th order of derivatives.

The pair $\mathcal{D} = (\Gamma, F)$ is called a *digital surface model.* Partition Γ is usually formed either by quadrilaterals or by triangles. If the partition is a triangulation τ, the pair $\mathcal{D} = (\tau, F)$ is called a *triangulated surface model.*

A *piecewise-linear triangulated surface model (PLTS)* is a digital surface model $\mathcal{D} = (\tau, F)$ of class C^0 (i.e., continuous), where τ is a triangulation and F is a family of linear functions. For a PLTS, the family F is characterized completely by the set of values $\hat{f}(V)$ corresponding to the set of vertices V of τ: a function $f_i \in F$ corresponding to a triangle t_i of τ is the linear function interpolating $\hat{f}$ at the vertices of t_i.

Though smooth surface models are desirable for many applications in CAGD and computer graphics, piecewise-linear triangulated surface models are widely used both in finite element methods to represent scalar fields [Sil90], and in GIS to represent terrains [Lee91]. Moreover, piecewise-linear models can be used as control nets to define surfaces of higher class [Far86, Sei92].

In case a digital model is used to approximate a surface, its effectiveness depends on the precision of the approximation. The approximation error is application dependent, and it is often measured through norms. Let $\|\cdot\|_\Omega$ be a norm on real functions defined over a domain Ω, and let $f, \hat{f} : \Omega \to \mathbb{R}$ be two such functions. We define the *error* made in approximating f with $\hat{f}$ over Ω as follows:
$$E(f, \hat{f}) = \frac{\|f - \hat{f}\|_\Omega}{\|\mathbf{1}\|_\Omega},$$

where $\|\mathbf{1}\|_\Omega$ is the norm of the unit function used as a normalization factor. Thus, given a mathematical surface model $\mathcal{M}$, a digital surface model $\mathcal{D}$, defined as above, and a tolerance value $\varepsilon \geq 0$, we say that $\mathcal{D}$ is an approximation of $\mathcal{M}$ with precision ε if and only if $E(f, \hat{f}) \leq \varepsilon$.

Examples of norms used in the literature to measure the approximation error are the L_n norm (for n positive integer) defined

$$\|f\|_{\Omega,n} = \sqrt[n]{\int_\Omega \mid f(x,y) \mid^n dxdy},$$

the L_∞ norm defined

$$\|f\|_{\Omega,\infty} = \sup_\Omega \mid f \mid,$$

and the Sobolev norm defined

$$\|f\|_{\Omega,\mathrm{S}} = \sqrt{\|f(x,y)\|^2 + \|\nabla f(x,y)\|^2}.$$

The L_∞ norm is normally adopted in GIS applications, while in finite element analysis the L_2 and the Sobolev norms are used more often.

It is often useful to measure the precision of an approximation only over a subset of the domain. Note that all the norms defined above can be redefined for any subdomain $\mathcal{S} \subset \Omega$ (provided that the area of $\mathcal{S}$ is not zero). Note also that E can be regarded as an average error over the domain (because of the normalization factor): thus, measures of approximation over different subdomains can be considered somehow consistent among them. We denote the measure of precision over $\mathcal{S}$ by

$$E_\mathcal{S}(f, \hat{f}) = \frac{\|f - \hat{f}\|_\mathcal{S}}{\|\mathbf{1}\|_\mathcal{S}}.$$

Whenever models $\mathcal{M}$ and $\mathcal{D}$ (and thus functions f and $\hat{f}$) are implicitly fixed, we will simply write $E(\mathcal{S})$; also, if Γ is the underlying partition of $\mathcal{D}$, we will denote by $E(\Gamma)$ the error of approximation over the whole domain Ω.

When the function f describing a natural surface ψ over a domain Ω is known only at a finite set of data points $\mathcal{P} = \{p_1, \ldots, p_n\} \subset \Omega$, $\psi(\mathcal{P}, f(\mathcal{P}))$ is called a *sampled surface*. In this case, the precision of a digital model $\mathcal{D}$ approximating ψ over Ω can be only estimated. Since the area of $\mathcal{P}$ is zero, the estimate E can be expressed in a discrete form, by redefining the norms through sums instead of integrals. An alternative is to build a continuous model $\mathcal{M}$ that interpolates ψ at all points of $\mathcal{P}$, then taking $\mathcal{M}$ as *reference model*, and measuring the error in approximating $\mathcal{M}$ with $\mathcal{D}$.

A hierarchical triangulated surface is defined by combining the concepts of hierarchical triangulation and of triangulated surface model. Let $\mathcal{M} = (\Omega, f)$ be a mathematical surface model, and let $\mathcal{H} = (\mathcal{T}, \mathcal{E}, \ell)$ be a hierarchical triangulation on Ω (where $\mathcal{T} = \{\tau_i,\ i = 0, \ldots, h\}$). For $i = 0, \ldots, h$, let $\mathcal{D}_i = (\tau_i, F_i)$ be a triangulated surface model approximating $\mathcal{M}$ over $D(\tau_i)$, and let $\mathcal{F} = \{F_i,\ i = 0 \ldots, h\}$. Then, the 4-tuple $\mathcal{HD} = (\mathcal{T}, \mathcal{E}, \ell, \mathcal{F})$ is a *Hierarchical Triangulated Surface (HTS)* approximating $\mathcal{M}$ over Ω. We say that $\mathcal{HD}$ is *based on* hierarchical triangulation $\mathcal{H}$.

Let $\mathbf{T} = \cup_{\tau_i \in \mathcal{T}} T_i$ be the set of all triangles of $\mathcal{H}$ (where T_i is the set of triangles of τ_i), let $\mathbf{F} = \cup_{F_i \in \mathcal{F}} F_i$ be the set of all functions of $\mathcal{F}$, and for each $t_i \in \mathbf{T}$, let $f_i \in \mathbf{F}$ be its corresponding function. Given any triangulation τ formed of triangles of $\mathbf{T}$, let $\mathcal{F}|_\tau$ be the subset of $\mathbf{F}$ formed by all functions corresponding to the triangles of τ. We call $\mathcal{HD}$ a *consistent HTS of class* k if and only if

- $\mathcal{H}$ is a matching triangulation;
- $\forall \mathcal{H}'$ consistent substructure of $\mathcal{H}$, if $\mathcal{H}'_E$ is the expansion of $\mathcal{H}'$, then the pair $(\mathcal{H}'_E, \mathcal{F}|_{\mathcal{H}'_E})$ is a triangulated surface model of class k.

Notice that each component of a consistent HTS of class k must necessarily be a digital surface model of class k, but this condition is not sufficient, since also the continuity across different components must be guaranteed (up to the k-th order of derivatives). However, this subject is not developed in general in this paper: in the following Sections, we consider the case of a piecewise-linear HTS of class C^0, that we have developed for the representation of topographic surfaces in the context of a geographical application [DeF92].

A piecewise-linear hierarchical triangulated surface $\mathcal{HD}$ is an HTS whose components are PLTSs. In this case, given the set $\mathbf{V}$ of all vertices in $\mathcal{HD}$, and the corresponding set of values $f(\mathbf{V})$, the families of functions of $\mathcal{F}$ are uniquely defined through $(\mathbf{V}, f(\mathbf{V}))$. It is easy to see that $\mathcal{HT}$ is a consistent HTS of class 0 if and only if $\mathcal{HT}$ is based on a matching triangulation.

6 Multiresolution Representation of a Terrain

In this Section, we consider the surface of a terrain whose elevation values are sampled at the nodes of a fine rectangular grid. Such a sampling scheme is widely used in practice: large areas of territory have been sampled and encoded through rectangular grids, and are made available by the geographic agencies in many countries. A reference surface model of one such sampled surface could be easily obtained by using piecewise-bilinear interpolation within each mesh of the fine grid. On the other hand, the amount of data in one such model is usually very large, and the need has been pointed out for models that are able to compress information using a smaller data set [Lee91].

Here, we are interested in building a multiresolution approximation of such a sampled surface through a piecewise-linear HTS, having its vertices either at the nodes or on edges of the grid. For the sake of simplicity, we use an $m \times m$ square grid, scaled on the domain $\Omega = [0, m-1] \times [0, m-1]$. Thus, our sampled surface $\psi(\mathcal{P}, f(\mathcal{P}))$ is defined from a set of samples $\mathcal{P} = \{(i,j) \mid i,j = 0, \ldots, m-1\}$, and an associated set of elevation values $f(\mathcal{P})$. Let $\mathcal{S}$ be the lattice obtained by joining each node $(i,j) \in \mathcal{P}$ with its four neighbors through straight-line edges: $\mathcal{S} = \{(i,y),(x,j) \mid i,j = 0, \ldots, m-1, \;\; x,y \in [0, m-1]\}$. Function f is extended to $\mathcal{S}$ through linear interpolation along each edge joining adjacent nodes of $\mathcal{P}$, thus obtaining a wire-frame surface representation (see Figure 13).

We use $\mathcal{S}$ as a data set to build a hierarchical triangulation $\mathcal{H} = (\mathcal{T}, \mathcal{E}, \ell)$, and we approximate the terrain surface over Ω by a piecewise-linear hierarchical triangulated surface model $\mathcal{HD} = (\mathcal{T}, \mathcal{E}, \ell, \mathcal{F})$, interpolating f at its vertices through a function $\hat{f}$, defined as in Section 5. The resulting model $\mathcal{HD}$ has the following characteristics:

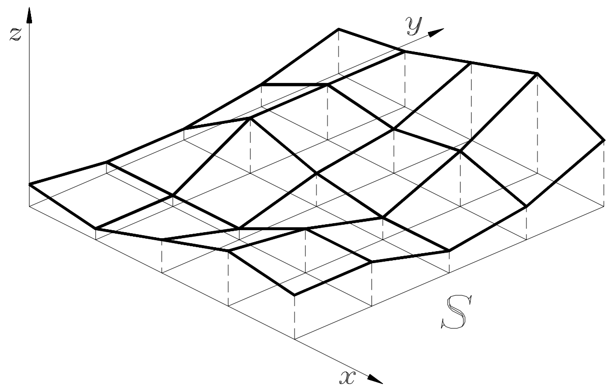


Figure 13: The wire-frame surface defined by lattice $\mathcal{S}$.

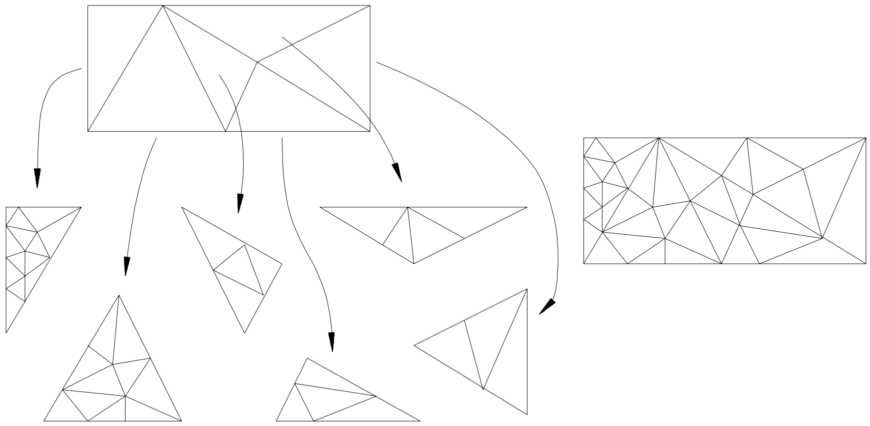


Figure 14: An example of hierarchical Delaunay triangulation and its expansion.

- each triangulation $\tau_i \in \mathcal{T}$ is a Delaunay triangulation[3], having its internal vertices in $\mathcal{P}$, and its boundary vertices in $\mathcal{S}$ (an example of hierarchical Delaunay triangulation is shown in Figure 14);
- the precision of the approximation is measured with respect to $\psi(\mathcal{P}, f(\mathcal{P}))$ by an error function based on the L_∞ norm, defined as follows: for $A \subset \Omega$, $E(A) = \max_{A \cap \mathcal{P}} \mid f - \hat{f} \mid$;
- the refinement is error-driven, on the basis of a given sequence of tolerance values

  $$\varepsilon_0 > \varepsilon_1 > \ldots > \varepsilon_{d_{max}},$$

  yielding the conditions:

  $$\forall i = 0, \ldots, d_{max},\ c_i(\tau) :== (\forall t \in \tau,\ E(t) \leq \varepsilon_i).$$

Considering the definitions given in Section 3.4, and the fact that each triangulation $\tau_i \in \mathcal{T}$ is a Delaunay triangulation, $\mathcal{HD}$ is specified completely by defining the refinement strategy adopted to select its vertices.

[3]The Delaunay triangulation of a set of points V is a triangulation $\tau = (V, E, T)$ such that for each triangle $t \in T$, the open circumcircle of t does not contain any point of V in its interior [Pre85].

The problem of approximating a sampled surface at a given precision $\varepsilon > 0$ through a PLTS has been studied by several authors in the literature (see [Lee91] for a survey). Different heuristics have been proposed to extract a "small" number of vertices (and triangles) that are sufficient to achieve precision ε. A recent breakthrough on this subject is a negative result given in [Aga94]: it is NP-hard to decide if n sampled points can be approximated by a PLTS of k triangles within the L_∞ error tolerance of a specified $\varepsilon > 0$. In [Aga94] an algorithm is also given for constructing one such PLTS with $O(k_0 \log k_0)$ triangles, where k_0 is the optimal number. Unfortunately, such an algorithm has a worst case time complexity of $O(n^7)$, and it appears very hard to implement, hence having small practical impact.

Here, we adopt a simple algorithm proposed in [Fow79], which refines an existing Delaunay-based PLTS by inserting one vertex at a time, and updating the Delaunay triangulation until the required precision is met. The vertex inserted at each cycle is the point that causes the maximum error in the approximation. Given a sampled surface $\psi(\mathcal{P}, f(\mathcal{P}))$, a Delaunay-based PLTS $(\tau(V), f(V))$, where V is a finite subset of $\mathcal{P}$, and a condition $c_\varepsilon(\tau) :== (E(\tau) \leq \varepsilon)$, for some $\varepsilon \geq 0$, the *Delaunay selector* is described by the following pseudo-code:

procedure DELAUNAY_SELECTOR($\mathcal{P}, \tau, c_\varepsilon$);
    while not $c_\varepsilon(\tau)$ **do**
        let v be the point of $\mathcal{P}$ that maximizes E;
        $\tau \leftarrow$ ADD_VERTEX(τ, v);
    end while ;
    return (τ)
end ;

where ADD_VERTEX(τ, v) is a procedure that updates a given Delaunay triangulation τ by inserting one new vertex v in the set of vertices of τ. The convergence of the Delaunay selector is guaranteed for every value of ε, since the number of points in $\mathcal{P}$ is finite.

Let $N = m^2$ be the number of points in $\mathcal{P}$, and let n be the total number of vertices of the resulting triangulation τ. The time complexity of the Delaunay selector is computed in terms of such numbers, by counting the cost of selecting a point v at each iteration, plus the global cost of procedure ADD_VERTEX. We maintain a data structure that associates with each triangle t in the current triangulation the list of data points contained inside t: the datum corresponding to the maximum error within t is placed at the head of such a list. Moreover, triangles of the current triangulation are maintained in a balanced search tree, organized according to the approximation error corresponding to each triangle. By using such data structure, point v can be selected in time $O(\log n)$ at each iteration, and thus the total contribution of point selection is $O(n \log n)$. Procedure ADD_VERTEX must take care of two tasks:

1. update the triangulation;
2. update the links between triangles and data, and the tree storing triangles.

We implemented the triangulation update through an algorithm based on local edge flipping[4] proposed in [Gui85], and we added the operations for updating links after each flip operation. The algorithm first inserts point v by joining it with the vertices of the triangle t containing

[4]This is a well-known technique that considers a pair of adjacent triangles forming a convex quadrilateral, and swaps the common edge with the opposite diagonal if the two triangles do not satisfy the Delaunay criterion.

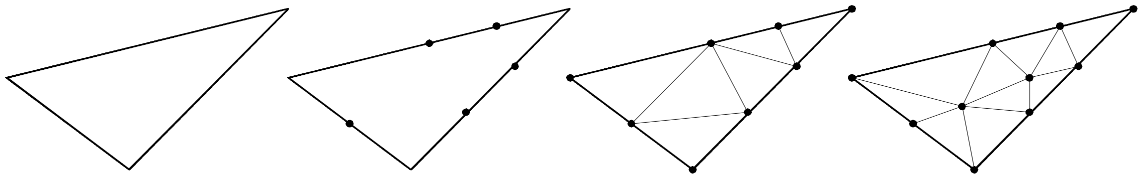


Figure 15: Steps of the refinement algorithm.

v (t is known from point selection), then performs a sequence of edge flips, until the resulting triangulation is a Delaunay triangulation. It is known from [Gui85] that the number of flips at a generic refinement step i is at most $O(n_i)$, where n_i is the current number of vertices in the triangulation, yielding a total cost of $O(n^2)$ in the worst case for computing the whole triangulation. Moreover, new triangles generated at each step are radially arranged around the new point v, and a sequence of such triangles sorted radially can be obtained from the updating procedure at no extra cost. Since the area covered by the new triangles could contain at most $O(N)$ data points, the links between points and triangles can be updated in $O(N \log n_i)$ at step i, through binary search on the sorted sequence, hence yielding a total cost of $O(Nn \log n)$ for this task. Finally, the tree of triangles can be updated in time $O(n_i \log n_i)$ at step i, yielding a total cost of $O(n^2 \log n)$. Since we always have $n \leq N$, we can conclude that the total complexity of the algorithm in the worst case is $O(Nn \log n)$. We wish to point out that such worst-case complexity can possibly be achieved only through *ad hoc* ill-conditioned data, while the practical performance of the Delaunay selector on real world data shows a slightly superlinear behaviour.

The root triangulation τ_0 of a hierarchical triangulated surface $\mathcal{HD}$ is built by applying the Delaunay selector algorithm with condition c_0 over the set of samples $\mathcal{P}$. The basic idea to build the rest of the structure is to use the same approach for generic procedure REFINE defined in Section 3.4. However, some attention must be paid to the refinement of edges, both to ensure matching, and to obtain triangles with a good shape. Note that, given a triangular region $\mathcal{D}_t \subset \Omega$, there is small probability that some point of $\mathcal{P}$ lie exactly on the border of $\mathcal{D}_t$. Thus, if only points of $\mathcal{P}$ are used as data, edges would be seldom refined, and the resulting triangulations would contain slivery triangles along their boundary. In order to avoid this problem, we adopted the following internal structure for procedure REFINE (see Figure 15):

(i) **edge refinement**: for each edge e of triangle t that must be refined, we consider a profile of the terrain along e as follows:

- if e lies on an edge of S (either horizontal or vertical), we take the chain formed by the endpoints u, v of e plus all points of $\mathcal{P}$ covered by e, $P_e = \{u\} \cup (\mathcal{P} \cap e) \cup \{v\}$, ordered along e (see Figure 16a);
- otherwise, we consider the chain formed by the endpoints u', v' of e plus the points where e intersects the lattice $\mathcal{S}$, $P_e = \{u'\} \cup (\mathcal{S} \cap e) \cup \{v'\}$, ordered along e; for each $w \in P_e \setminus \mathcal{P}$, the value of $f(w)$ is estimated by linear interpolation between two points of $\mathcal{P}$ (see Figure 16b).

We take as a reference function the piecewise linear function f_e that interpolates f at the vertices of P_e, and we solve the one-dimensional problem of approximating f_e with another piecewise linear function defined by a chain $\mathcal{C}_e$ whose vertices are a subset of vertices of P_e. Initially, $\mathcal{C}_e$ is coincident with e (i.e., the linear function interpolating f at the endpoints of e is taken). Then, $\mathcal{C}_e$ is refined by inserting other points of P_e on-line, until the required

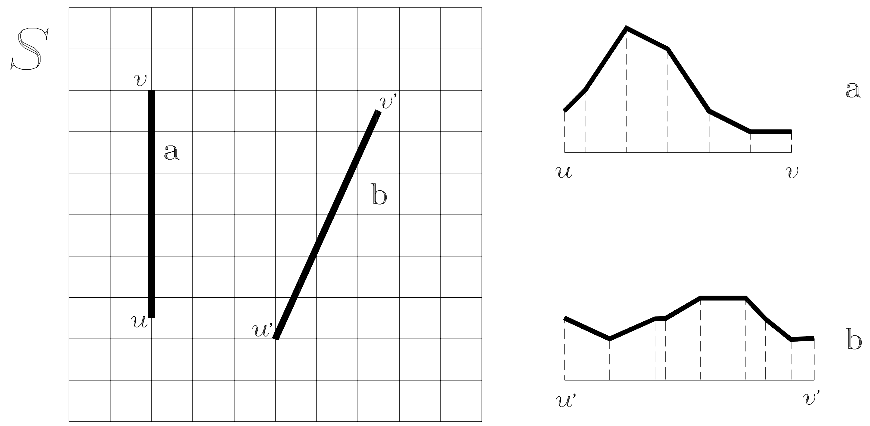


Figure 16: Terrain profile along a vertical edge (a) and a slanted edge (b).

precision ε is achieved. This approach can be regarded as the one-dimensional version of the Delaunay selector, and is known in the literature as *line simplification* algorithm [Dou73]. The procedure is detailed by the following pseudo-code, where we denote with $E(\mathcal{C}_e)$ the error of approximating f_e with the piecewise-linear function interpolating f_e at the vertices of chain $\mathcal{C}_e$, while procedure SPLIT_CHAIN is self-explanatory:

procedure LINE_SIMPLIFICATION(P_e,$\mathcal{C}_e$,ε);
  while not $E(\mathcal{C}_e) \leq \varepsilon$ **do**
    let v the point of $\mathcal{P}_e$ that maximizes E;
    $\mathcal{C}_e \leftarrow$ SPLIT_CHAIN($\mathcal{C}_e, v$);
  end while
end ;

The above procedure can be implemented with a worst-case time complexity equal to $O(N_e \log N_e)$, where N_e is the number of points in P_e [Her92]. Since P_e is obtained by intersecting e with a grid of $2\sqrt{N}$ lines, where N denotes the total number of points in $\mathcal{P}$, the time complexity of the procedure is maximized by $O(\sqrt{N} \log N)$. After applying this procedure to all edges of t, the edges whose error exceeded ε have been split into chains.

(ii) **triangulation refinement**: the set V_t composed by the vertices of triangle t, plus all vertices inserted onto its edges at the previous step is considered, and the Delaunay triangulation $\tau(V_t)$ is computed (through the on-line algorithm proposed in [Gui85]). Finally, procedure DELAUNAY_SELECTOR($\mathcal{P}, \tau(V_t), c_\varepsilon$) is called, to complete the refinement. The time complexity of this procedure is $O(N_t n_t \log n_t)$, where n_t is the number of points inserted at that step of refinement, and N_t is the number of points of $\mathcal{P}$ that lie within t.

The time complexity of the refinement of a triangle t is thus given by the sum of the time complexities of the two steps described above, i.e., $O(N_t n_t \log n_t + \sqrt{N} \log N)$. If we consider all refinement steps performed to obtain the hierarchical structure $\mathcal{HD}$, and we denote with n the total number of vertices inserted, it is easily seen that the total time complexity is maximized by $O(Nn \log n)$, i.e., it is not higher than the time complexity for obtaining an approximating surface at the maximum degree of resolution $\varepsilon_{d_{max}}$ using the Delaunay selector.

Similarly to the general case described in Section 3.4, it is easy to adapt the algorithm just described in order to build a hierarchy that approximates a multiresolution model spanning all resolutions in error range $[\varepsilon_{min}\varepsilon_{MAX}]$. The sequence of degrees of resolution of such model is unknown a priori: at each iteration, the algorithm stops refinement as soon as the error gets smaller than the current one, and the next degree of resolution is set as the maximum error among all simple triangles after refinement. Therefore, at each iteration, only triangles whose error is exactly the same as the current error are refined: in this way, many degrees of resolution are built, each of which involves the refinement of few triangles.

7 Extracting a surface at variable resolution

As we already outlined, an important application of multiresolution terrain representations is landscape visualization, in the framework of systems such as flight simulators. In this context, it is important to warrant high resolution over areas close to the observer, while resolution can progressively decrease with distance from the viewpoint. Such an approach allows the rendering system to visualize a reduced number of primitives, while maintaining a high grade of visual realism.

The most difficult issue in this respect is to produce a representation of the terrain surface that adapts to a variable resolution over the domain while remaining continuous. For instance, a straightforward multiresolution approach based on raster models can be obtained by subsampling over far areas. A similar result is obtained from quadtree models through a more structured procedure that selects quadrants from appropriate levels of the tree depending on their distance from the viewpoint. All such approaches suffer from discontinuities of the rendered surface at the edges where different resolutions meet.

Here, we describe two different approaches to the extraction of a continuous model at variable resolution from a hierarchical triangulated surface. The first algorithm that we present is a straightforward extension of the algorithm for expanding an HT, while the second algorithm is based on a more sophisticated technique that uses navigation of the hierarchy. The two algorithms are currently under implementation.

Let $f_\varepsilon : \Omega \to \mathbb{R}_+$ be a positive-valued function called the *resolution function*. Given a hierarchical triangulated surface $\mathcal{H}$, we wish to extract from it a continuous surface such that for every point $p \in \Omega$, the elevation of terrain at p is represented within an error $f_\varepsilon(p)$. With abuse of notation, for any triangle t of $\mathcal{H}$, $f_\varepsilon(t)$ will denote the minimum of f_ε over the area spanned by t. Therefore, the desired triangulation τ must be such that $E(t) \leq f_\varepsilon(t)$, for every t triangle of τ.

Let us consider any HTS in which the representation error $E(t)$ at each triangle t is known. We can apply the expansion procedure described in Section 3.5, based on the following macro function:

$$\mathrm{MACRO}_{\mathcal{H}}(t) :== E(t) > f_\varepsilon(t).$$

Such an algorithm will produce a generalized triangulation $\mathcal{H}_E$ that satisfies the precision requirement at each point over the domain, but cannot warrant matching, hence continuity. A continuous model can be obtained by considering each generalized triangle in $\mathcal{H}_E$, and triangulating it. Since a generalized triangle is a convex polygon, a Delaunay triangulation can be

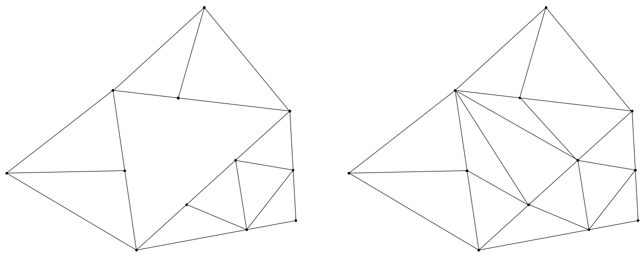


Figure 17: A generalized trianglulation (a) and the easy triangulation of a generalized triangle during edge linking (b).

computed in linear time through an algorithm described in [Agg89].

If an arbitrary triangulation of each generalized triangle is sufficient, the final triangulation can be easily obtained in a single step as follows. Consider step 3 of procedure EDGE_LINKING described in Section 3.5. Instead of changing triangle t_1 into a corresponding generalized triangle obtained by replacing edge e with the refinement of e in $\mathcal{T}_{t_0}$, we can change t_1 directly into a triangulation obtained by connecting each vertex of the refinement of e to the vertex of t_1 opposite to e (see Figure 17). The corresponding modification of step 5 of the same procedure is completely similar. The result of such modified algorithm is always a triangulation, thus yielding a continuous surface model. Note that, for any generalized triangle that had more than one edge refined because of the expansions of its neighbors, its triangulation is not uniquely defined: it depends on the order in which its edges are considered during edge linking.

In any case, the final triangulation can be obtained in a time linear in the number of its vertices, i.e., in a time linear in the number of output primitives. Moreover, note that the hierarchical nature of the model allows the algorithm to discard from the early levels of the hierarchy all areas that are beyond the field of view, hence speeding up the search through large areas.

The above approach can be applied successfully in practice, but it has a theoretical drawback: the resolution requirements might be violated in triangulating generalized triangles. Indeed, let us consider a generalized triangle t obtained through procedure EXPAND_SUBTREE described in Section 3.5. A triangulation τ of t does not necessarily satisfy the same precision of t: we could have $E(\tau) > f_\varepsilon(t)$ while $E(t) \leq f_\varepsilon(t)$. In principle, it is possible to refine τ further to obtain the desired precision, but this would involve a time complexity that might be prohibitive for real time rendering. Note that the problem does not arise if the resolution desired is dependent on the area of triangles (i.e., the farther the triangle, the larger its area), rather than on their precision: such an alternative is often suitable in landscape visualization with models like the k^2 triangulations.

The second approach that we present overcomes the problem discussed above, since it extracts only triangles that were already in the hierarchy $\mathcal{H}$. This method modifies a technique developed recently for variable resolution rendering of surfaces represented through pyramidal models [Cig95]. The method requires a matching hierarchy, and it is limited to the case of resolution functions of the type

$$f_\varepsilon(p) = g(d(p, v_p)),$$

where $g : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is a monotonically non-decreasing function, d is the standard Euclidean distance over $\mathbb{R}^2$, and v_p is a fixed point, called the *viewpoint*. Most meaningful applications are in this class. This is indeed the case of flight simulators, in which the error is directly proportional to distance from the viewpoint. however, one might have different situations in which multiple focus areas are defined over the domain, where high resolution is required, while resolution progressively decreases when moving far from focus areas.

The key idea here is the possibility to weld representations at different resolutions through edges that "survive" across different levels of the hierarchy. Consider, for example, triangle t_0 in Figure 18a: both triangles t_1 and t_2 are admissible neighbors of t_0, since they both match with t_0 at e. This means that, in principle, it might be possible to extract either a triangulation containing t_0 and t_1, or a triangulation containing t_0 and t_2. We will call e a *bypass edge*. Intuitively, we can exploit bypass edges to move across different resolutions while saving the continuity of the surface. Note that this approach will be effective only if the hierarchy contains plenty of bypass edges, like in the model approximating a continuous range of errors outlined in Section 6.

The algorithm follows an incremental technique that starts from the triangle closest to the viewpoint, at the proper resolution, and builds the triangulation by traversing the hierarchy through the domain, while iteratively pasting new triangles that are compatible to the current triangulation. While doing this, the algorithm coarsens the resolution of the surface according to function f_ε. At each step, a boundary edge e of the current triangulation is considered, and a triangle t is selected from the hierarchy, such that $E(t) \leq f_\varepsilon(t)$. Then, t is pasted to the current triangulation at e. The algorithm stops when all the domain has been covered, i.e., when all boundary edges of the current triangulation are on the boundary of the domain of $\mathcal{H}$.

In practice, each time edge e currently considered is a bypass edge, the algorithm tries to select t as the coarsest triangle that satisfies the precision requirement, among all condidates that match at e. Note that, in order to guarantee the correctness of the algorithm, we must ensure that one such triangle t always exists, and that the current triangulation can always be completed to a triangulation that covers the whole domain. This is the most delicate point of the algorithm. Indeed, such requirements cannot be satisfied on a local basis, i.e., by considering only a portion of triangulation close to e: we must take into account global information.

In order to understand problems that might arise if t were extracted only on the basis of its error, let us consider the HT in Figure 18a, and let us assume the triangulation depicted in solid lines in Figure 18b is the current triangulation. Let us assume also the algorithm is currently extending such a triangulation across edge e': although triangle t might satisfy relation $E(t) \leq f_\varepsilon(t)$, it cannot be selected, otherwise the shaded area could not be covered by any triangle while maintaining matching. Indeed, triangle t'' would match with t, but not with the rest of the triagulation, while its direct refinement would not match with t. Hence, the triangulation must be necessarily extended at e' with triangle t'.

The proper selection of a triangle at each step is made possible by maintaining global information about the boundary of the current triangulation. Let us assume that $\mathcal{H}$ has been built through the algorithm described in Section 6. We tag each edge e of $\mathcal{H}$ with two numbers: the *birth error* $e.\varepsilon_b$, which corresponds to the current error when e appeared first during construction, and the *death error* $e.\varepsilon_d$, which corresponds to the error just before e was refined into a sequence of edges. In other words, we know that e belongs to all expanded models at fixed

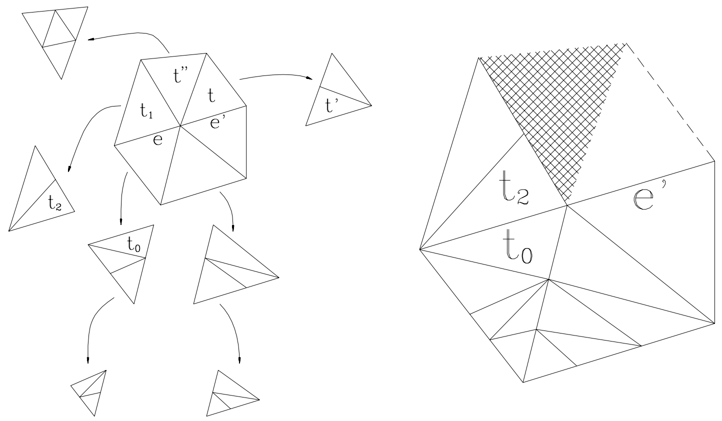


Figure 18: A hierarchical triangulation (a), and (in solid lines) the current triangulation at an intermediate step of extraction (b): if the current triangulation is extended at e' with t, neither t'', nor its direct refinement can be used to fill the shaded area while preserving matching: triangle t' is the correct choice at e'.

precision ε_i for all ε_i such that $e.\varepsilon_d \leq \varepsilon_i \leq e.\varepsilon_b$. Hence, we call interval $[e.\varepsilon_d, e.\varepsilon_b]$ the *life* of e, and for each ε_i such that $\varepsilon_i \in [e.\varepsilon_d, e.\varepsilon_b]$ we will say that e is ε_i*-alive*. We define the life of each triangle in a completely similar way. Note that the life of each edge contains the lives of all triangles incident at it.

The three edges of the initial triangle have lives that intersect in a non-empty interval. Let us assume inductively that at an intermediate step all boundary edges of the current triangulation have lives that intersect in a non-empty interval. This means that there exists at least one degree of resolution ε_j such that all such edges belong to the expanded triangulation at that level of resolution. Hence, in the worst case, it will be always possible to complete the current triangulation consistently by using all triangles that are ε_j-alive and lie outside the domain of the current triangulation. Note that all such triangles will also satisfy f_ε, since its values out of the current domain cannot be smaller than ε_j.

In particular, let ε_{Bmin} be the minimum of all birth errors of boundary edges of the current triangulation: then, each such edge will be ε_{Bmin}-alive. Let $\overline{e}$ be exactly the boundary edge with birth error ε_{Bmin}. If we paste to $\overline{e}$ a new triangle whose life contains ε_{Bmin}, we will ensure that the lives of the boundary edges of the updated triangulation will intersect in a non-empty interval. Since $\overline{e}$ will not be a boundary edge any more, it is possible that the minimum birth error in the updated triangulation will be larger than ε_{Bmin}. Therefore, the selection of the new triangle will depend both on the resolution function and on the current value of ε_{Bmin}. Both such values will become larger and larger as the boundary of the current triangulation moves farther and farther from the viewpoint.

In order to implement the algorithm we need to maintain the boundary edges of the current triangulation ordered according to their birth errors. In fact, we will maintain a dictionary Q_b, implemented with a binary balanced tree, such that the following operations can be performed in optimal logarithmic time: INSERT, adds an element to the dictionary; DELETE, removes a element from the dictionary; MIN, returns the minimum element of the dictionary; MEMBER, tells whether an element belongs to the dictionary or not; EMPTY, tells whether the dictionary is empty or not. Each element of the dictionary will be a pair formed by a boundary edge together with its incident triangle in the current triangulation. The sorting key in the dictionary is the birth error of each edge.

The algorithm uses of a procedure FIND_FIRST_TRIANGLE($\mathcal{H}, v_p, f_\varepsilon$) which finds the triangle t of $\mathcal{H}$ that lies closest to the viewpoint and such that $E(t) \leq f_\varepsilon(t)$. Such a procedure is implemented efficiently as a hierarchical point location [DeF94]: a candidate triangle is located by searching the root triangulation and, if its error is not sufficiently small, the procedure is activated recursively on its direct refinement. Moreover, we use two procedures for neighbor finding, which are easily implemented with constant time complexity through the roped data structure described in Section 3.3. Given a triangle t and one of its edges e, procedure NEIGHBOR(t, e) returns the triangle t' matching with t on the other side of e, which lies at the highest possible level in the hierarchy. Given t' and e defined as above, procedure REFINED_TRIANGLE(t', e) returns the triangle of $DRef(t')$ having e as edge (this makes sense provided that e survives in $DRef(t')$). Function IS_BOUNDARY returns a boolean value indicating whether an edge lies on the boundary of the domain of $\mathcal{H}$; function LIFE returns the interval corresponding to the life of either an edge or a triangle. The pseudo-code of the algorithm follows:

```
Algorithm EXTRACT_SURFACE(H, v_p, f_ε, out τ);
begin
  Q_b ← ∅;
  t ← FIND_FIRST_TRIANGLE(H, v_p, f_ε);
  τ ← {t};
  for every e edge of t do
    INSERT(Q_b, (e, t));
  end for ;
  while not EMPTY(Q_b) do
    (e, t) ← MIN(Q_b);
    t' ← NEIGHBOR(t, e);
    while E(t') > f_ε or e.ε_b ∉ LIFE(t') do
      t' ← REFINED_TRIANGLE(t', e);
    end while ;
    for every e edge of t' do
      if MEMBER(Q_b, (e, *)) then
        DELETE(Q_b, (e, *))
      else if not IS_BOUNDARY(H, e) then
        INSERT(Q_b, (e, t'))
      end if
    end for ;
    τ ← τ ∪ {t'}
  end while
end
```

Let n_τ be the number of triangles of the triangulation τ produced by the algorithm. Note that all triangles visited belong to a subhierarchy of $\mathcal{H}$ having triangles of τ as simple triangles. Thus, it follows from the result proven in Section 3.1 that the total number of visited triangles is linear in n_τ. The number of edges in Q_b at any time is also at most linear in n_τ; hence, the cost of each primitive operation on Q_b is $O(\log n_\tau)$. Based on such remarks, it is easy to conclude that the total time complexity of the above procedure is $O(n_\tau \log n_\tau)$ plus the cost of locating the first triangle. Such an operation could add a cost of $O(n)$, where n is the size of $\mathcal{H}$, in the worst case. In practice, the hierarchical approach to search makes the actual cost almost negligible. Note also that in a dynamic context such as flight simulation, in which the

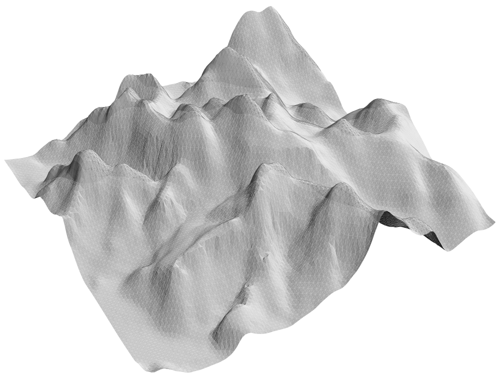


Figure 19: Perspective plot of the original dataset (32,258 triangles)

algorithm above is applied iteratively as the viewpoint moves through space, the location of the first triangle becomes even simpler: once the system is initialized, the first triangle for the next iteration can be located efficiently by navigating the hierarchy through adjacencies, starting from the current initial triangle.

8 Experimental Results

In this Section, we present some experimental results about piecewise-linear HTSs for terrain representation, which are built upon real data from the United States Geological Survey. Input data are originally encoded in a regular grid of $128^2 = 16,384$ points: the grid contains elevation data sampled over an area of roughly $\frac{1}{10} \times \frac{1}{10}$ degrees of the Earth's surface, and representing a varying topographic surface. Elevations are encoded by integer values, and range within 599 and 1,591 meters. A perspective plot of the original data is shown in Figure 19.

From such a dataset, we have built three different surface models. Model 1 is simply a (non-hierarchical) piecewise-linear triangulated surface obtained by applying algorithm DELAUNAY_SELECTOR described in Section 6 with a tolerance error of 10 meters, corresponding to approximately 1% of elevation range. The triangulation is formed by 4,522 triangles, having their vertices at 2,309 points, approximately corresponding 14% of data size.

Model 2 has been obtained through algorithm BUILD_HT described in Section 3.4, and implemented according to the description given in Section 6. This model contains three degrees of resolution a priori defined, namely at tolerances of 200, 40, and 10 meters, corresponding to approximately 20%, 4%, and 1% of elevation range, respectively. The hierarchical model contains a total of $N_t = 6,293$ triangles, having their vertices at 2,775 points, corresponding to approximately 15% of data size. The number of triangles in the expanded triangulation at maximum resolution (error 1%) is $N_s = 5,442$. The storage overhead of the hierarchical model is measured by the ratio $N_t/N_s = 1.16$, which is significantly smaller than the theoretical bound $N_t/N_s \leq 2$ that we computed in Section 3.1.

Perspective and wire-frame plots (top view) of expanded surfaces extracted from model 2 at

Model	ε_{Max}	ε_{min}	degrees	levels	points	%	N_t	N_s	N_t/N_s
1	1%	1%	1	1	2,309	14	4,522	4,522	1.00
2	20%	1%	3	3	2,775	15	6,293	5,442	1.16
3	20%	1%	1,969	11	3,659	22	12,774	7,213	1.77

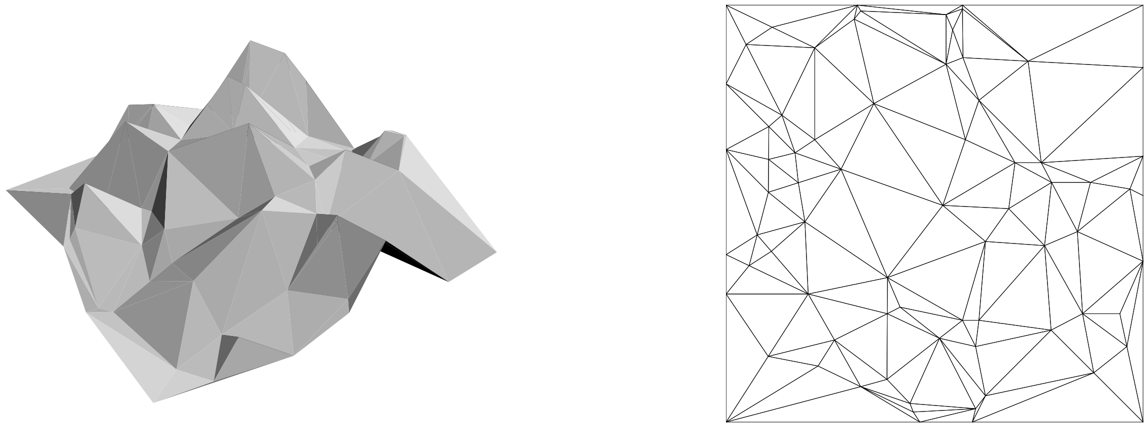
Table 1: Comparisons of models 1, 2, and 3 (data size: 16,384 points).

resolutions 200, 40, and 10 are depicted in Figure 20.

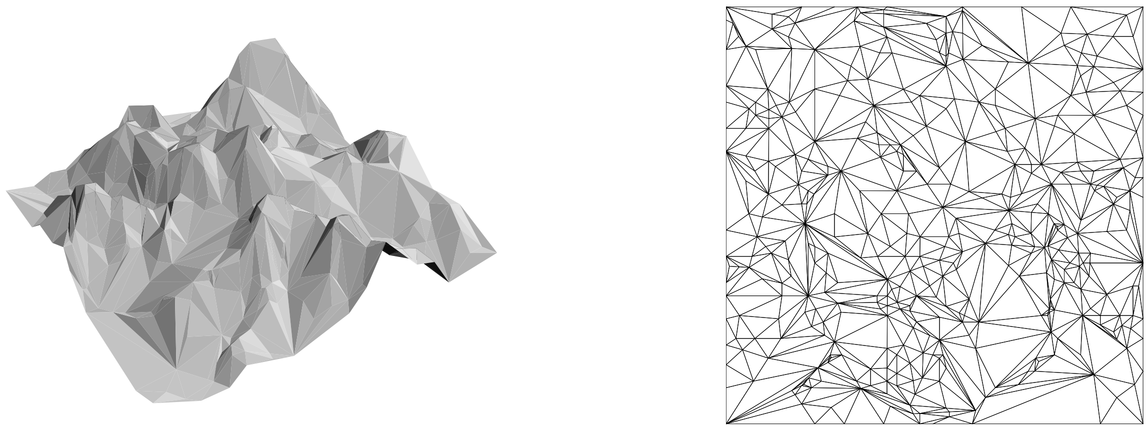
Model 3 has been obtained through the same algorithm, modified to approximate a continuous range of resolutions, as described in Section 3.4. This models spans all resolutions between 200 meters and 10 meters, as above, through 1,969 intermediate degrees of resolutions obtained automatically by the algorithm. The hierarchical model contains a total of 12,774 triangles, having their vertices at 3,659 points, corresponding to approximately 22% of data size. The number of triangles in the expanded triangulation at maximum resolution (error 1%) is 7,213. In this case, the overhead ratio is 1.77. In spite of the high number of degrees of resolution, the actual height of the tree encoding the hierarchy is of only 11 levels.

In Table 1, we compare the results obtained in the three cases. It is not surprising that the number of data points needed to achieve the same resolution is increasing with the number of intermediate degrees of resolution considered. Indeed, in the hierarchical case, points inserted at each degree of resolution must obey to spatial constraints imposed by the edges of the previous degree, while in the non-hierarchical model points are selected freely over the domain. The ratio between the number of points used in model 3 and in model 1 is 1.58: corresponding ratios for all possible models spanning the same range of tolerances with a number of degrees of resolution between two and 1,969 are progressively larger, ranging between 1.05 and 1.58 (ratio for model 2 is 1.2).

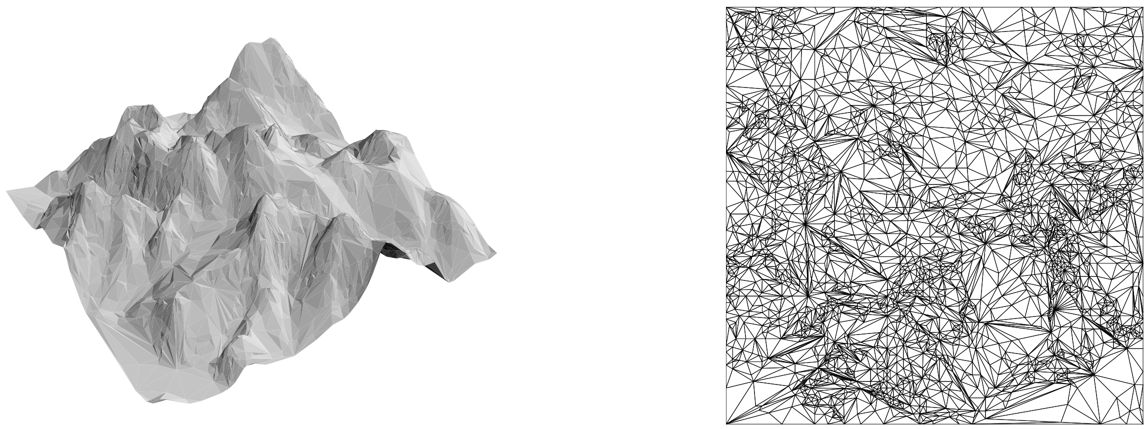
In terms of storage cost, our model can be compared to a multi-layered model obtained as a collection of independent layers, each built through algorithm DELAUNAY_SELECTOR. The storage costs of both our model and a multi-layered one can be easily measured by counting the total number of triangles encoded, since the size of the data structures is proportional to such number (hierarchy links in our model give a minor contribution to storage cost). When the number of layers is small, and either few or no triangles survive across different levels, the multi-layered model can be slightly cheaper. For instance, model 2 contains a total of 6,293 triangles, while a model with three layers at the same resolutions of model 2 contains a total of 5,379 triangles, thus yielding a ratio 1.17 between the storage costs. The reason for the higher cost is that, in this case, our model needs more points than the multi-layered model to achieve the same resolutions. On the contrary, when the number of degrees of resolution gets larger, our model becomes tremendously cheaper than a multi-layer one: a multi-layered model comparable to model 3 would contain millions of triangles. This gain in storage cost is actually proportional to the number of triangles that survive across different resolutions, hence it is due to the fact that consecutive resolutions are very close to one another, as in the extremal case of model 3.



(a)



(b)



(c)

Figure 20: Perspective and wire-frame plots extracted from model 2: error 200, triangles 130 (a); error 40, triangles 774 (b); error 10, triangles 5,442 (c).

9 Concluding Remarks

We have introduced a formal model for hierarchical decomposition of planar domains into triangles, and we have described its application to the multiresolution description of surfaces defined by bivariate functions. The model can be encoded and manipulated efficiently: it does not introduce a serious overhead with respect to triangulated models that do not support multiresolution, while offering a support to important complex geometric operations like browsing, point location, interference queries, and variable resolution rendering.

We are currently developing a prototype system for the multiresolution representation, analysis, and visualization of topographic surfaces in the framework of a geographical information system, by using the piecewise-linear model described in Section 6. Our prototype system already integrates the following modules:

- A *kernel* module containing the data structure, its access primitives, the construction algorithm described in Section 6, and the expansion algorithm described in Section 3.5.
- A *visualization* module that permits to visualize the surfaces through different rendering modalities: perspective views, wire frame triangulations, and contour plots. The algorithm for extracting contour plots from the HTS is based on a technique similar to the expansion algorithm presented in Section 3.5, and has been described in [DeF93].
- An *analysis* module incorporating the following functionalities:
  - interference queries, such as point location, segment and region intersection, as described in [DeF94];
  - visibility algorithms, such as point-to-point visibility and computation of the horizon from a given viewpoint, as described in [DeF95a].

We are currently implementing in such a framework the algorithms for extracting a surface at variable resolution described in Section 7. We are also developing algorithms for multiresolution map overlay [Ber95b], and algorithms for computing the viewshed from any viewpoint at different degrees of resolution. All functionalities mentioned above are of interest in the context of GISs.

Moreover, we plan to develop algorithms that exploit the hierarchical model for giving efficient solutions to path planning of autonomous vehicles that navigate a terrain surface by orienting themselves on the basis of horizon matching. This is a topic of interest in robotics.

Hierarchical triangulations, the data structures for encoding them, and the algorithms for their construction and analysis are very general, and can be applied to build different hierarchical models for specific applications. The Delaunay triangulation, which we adopted in our model, has some desirable properties (e.g., minimal roughness [Rip90]) that make it suitable for many applications. However, some researchers recommend using data dependent triangulations for surface representation, since they are more adaptive to sampled data [Rip92]: a model based on such triangulations can be obtained by simply modifying the update procedure in the construction algorithm described in Section 6, according to the methods proposed in [Rip92] or [Sch93].

It has been observed in the experiments that, with the current refinement strategy, the shape of triangles in the model tend to become more and more elongated as the number of degrees of resolution become larger and larger. Although the shape of triangles is overall better than in other hierarchical matching models (e.g., ternary triangulations and adaptive hierarchical triangulations), it might be important for some applications to avoid slivery triangles at all. To this purpose, we are investigating the possibility of applying refinement techniques based on the *conforming Delaunay triangulation* [Ede92]: such an approach would permit to maintain constraint edges necessary to the hierarchical structure, while satisfying the Delaunay rule for the expanded triangulation at each degree of resolution.

Smooth surfaces over triangulations can be obtained by using one among the numerous methods developed in the literature (see, e.g., [Ren84, Far86, Sei92]). Since adjacent portions of the domain are refined independently in the hierarchical triangulation, it is our intent to investigate further the relation between the matching property of hierarchical triangulations and the smoothness of the surface across adjacent patches. Our model of surface is not necessarily interpolant, as in the example we described: other approximating surfaces, whose precision can be measured through a suitable norm, can be defined by using the hierarchical triangulation as a hierarchical control grid.

Other issues involving two-dimensional hierarchical triangulations, which we plan to tackle in the future, are: the application of hierarchical triangulations to the representation of parametric surfaces, via hierarchical triangulation in parameter space; the multiresolution reconstruction of solid objects through hierarchical triangulations of points sampled on their boundary. These are topics of interest in Computer Aided Geometric Design, and in machine vision.

Hierarchical triangulations have been extended to arbitrary higher dimensions into a model called *hierarchical simplicial complex*. In particular, we have implemented a system for the multiresolution representation, visualization, and analysis of volume data through three-dimensional hierarchical Delaunay tetrahedralizations [Ber94].

10 Later work

Recently, after this paper had been submitted for publication, other work on related subjects appeared, mainly in the framework of multi-layered models. In [Cig94], a three-dimensional extension of the Delaunay pyramid was used for the multiresolution modeling and visualization of volume data. In [Ber95a], pyramidal models were extended to multidimensional models, called *pyramidal simplicial complexes*; a simple yet efficient data structure was also proposed, which avoids duplications of simplices across different layers, thus making pyramidal models competitive with hierarchical models in terms of storage cost. Another competitive and more sophisticated data structure for encoding pyramidal triangulations was also proposed in [Cig95], together with an efficient algorithm for extracting surfaces at resolution variable over the domain. The concept of model approximating a continuous range of resolutions, as well as the ideas underying the second algorithm for variable resolution described in Section 7 of this paper (which were added during revision) were inspired by the work in [Cig95].

Independently of the above work, other algorithms and data structures for the Delaunay pyramid were proposed in [deB95]. In such work, emphasis was put on the possibility of performing point location in logarithmic time, and of extracting a Delaunay triangulation that represents a surface at variable resolution in linear time. The pyramid is built bottom-up through a simplification technique essentially based on the method proposed in [Kir83]. Better efficiency is paid in terms of accuracy: the model does not support explicit multiresolution. The structure of the different layers is constrained by the construction process, and it cannot be built to fulfill a predefined sequence of tolerances; also, surfaces extracted at variable resolution follow only approximatively an error function defined on the domain, while the result is not guaranteed to fulfill exactly the required accuracy over all points of the domain.

Recently, an effort has been undertaken towards the formalization of a broader class of multiresolution models that can incorporate both multi-layered and hierarchical simplicial complexes as special cases [Ber95c]. More work on this subject is being carried out. The objective is to compare pyramidal versus hierarchical complexes, in terms of storage cost and of performances on the basic operations a multidimensional surface model should support (e.g., extraction of hypersurface at variable resolution, answering geometric queries at different resolution). Such a comparative analysis is intended to highlight common aspects and structural differences, as well as advantages and drawbacks of different models, in order to understand which models are best for various application requirements.

Acknowledgments

We wish to thank Silvia Bussi, Daniela Mirra, and Giacomo Gattorna, former MS. students at the Department of Computer and Information Sciences of the University of Genova, for their precious contribution in implementing the algorithms described in this paper, as well as algorithms for contour line extraction and for interference queries that were not discussed here. We also wish to thank Annita Pavan that engineered all such algorithms in the framework of an interactive prototype system with graphical interface.

References

[Aga94] Agarwal, P.K., Suri, S., "Surface approximation and geometric partitions", *Proceedings 5th ACM-SIAM Symposium On Discrete Algorithms*, pp.24-33, 1994.

[Agg89] Aggarwal, A., Guibas, L.J., Saxe, J., Shor, P., "A linear-time algorithm for computing the Voronoi diagram of a convex polygon", *Discrete and Computational Geometry*, 4, pp.591-604, 1989.

[Ban86] Bank, R.E., "A posteriori error estimate. Adaptive local mesh refinement and multigrid iteration", *Multigrid Methods II*, Lecture Notes in Mathematics, N.1228, Springer-Verlag, 1986, pp.7-22.

[Bar84] Barrera, R., Vazquez, A.M., "A hierarchical method for representing relief", *Proceedings Pecora IX Symposium on Spatial Information Technologies for Remote Sensing Today end Tomorrow*, Sioux Falls, South Dakota, Oct. 1984, pp.87-92.

[Ber94] Bertolotto, M., De Floriani, L., Puppo, E., Hyerarchical Hypersurface Modeling, in *IGIS'94: Geographic Information Systems*, J. Nievergelt, T. Roos, H. Schek, P. Widmayer (Editors), Springer-Verlag, *Lecture Notes in Computer Science*, N.884, 1994, pp.88-97.

[Ber95a] Bertolotto, M., De Floriani, L., Marzano, P., "Pyramidal simplicial complexes", *Third ACM Symposium on Solid Modeling and Applications*, Salt Lake City, Utah, May 17-19, 1995.

[Ber95b] Bertolotto, M., Magillo, P., De Floriani, L., "A hierarchical approach to the overlay problem", *Technical Report* DISI-95-06, Dipartimento di Informatica e Scienze dell'Informzione, Università di Genova, 1995.

[Ber95c] Bertolotto, M., De Floriani, L., Marzano, P., "A unifying framework for multilevel description of spatial data", *Proceedings International Conference on Spatial Information Theory 95*, Semmering (Austria), September , 1995.

[Che86] Chen, Z.T., Tobler, W.R., "Quadtree representation of digital terrain", *Proceedings Autocarto*, London, 1986, pp. 475-484.

[Cig94] Cignoni, P., De Floriani, L., Montani, C., Puppo, E., Scopigno, R., "Multiresolution modeling and visualization of volume data based on simplicial complexes", in *Proceedings 1994 ACM Symposium on Volume Visualization*, Washington, DC, October 17-18, 1994, pp.19-26.

[Cig95] Cignoni, P., Puppo, E., Scopigno, R., "Representation and visualization of terrain surfaces at variable resolution", *Proceedings International Symposium on Scientific Visualization*, Cagliari (Italy), September 1995.

[deB95] de Berg, M., Dobrindt, K.T.G., "On the levels of detail in terrains", *11th ACM Symposium on COmputational Geometry*, Vancouver, BC (Canada), June 5-7, 1995.

[DeF84] De Floriani, L., Falcidieno, B., Pienovi, C., Nagy, G., "A hierarchical data structure for surface approximation", *Computers and Graphics*, Vol.8, No.2, 1984, pp. 475-484.

[DeF89] De Floriani, L., "A pyramidal data structure for triangle-based surface description", *IEEE Computer Graphics and Applications*, March 1989, pp.67-78.

[DeF92] De Floriani, L., Puppo, E., "A hierarchical triangle-based model for terrain description", in *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, A.U. Frank, I. Campari, U. Formentini (Editors), Springer-Verlag, *Lecture Notes in Computer Science*, N.639, 1992, pp.236-251.

[DeF93] De Floriani, L., Mirra, D., Puppo, E., "Extracting contour lines from a hierarchical surface model", *Computer Graphics Forum*, 12, 3 (EUROGRAPHICS Conference Issue), September 1993, pp.249-260.

[DeF94] De Floriani, L., Gattorna, G., Marzano, P., Puppo, E., "Spatial queries on a hierarchical terrain model", *Proceedings 6th International Symposium on Spatial Data Handling*, Edimburgh (United Kingdom), September 5-9, 1994, pp.819-834.

[DeF95a] De Floriani, L., Magillo, P., "Horizon computation on a hierarchical triangulated terrain model", *The Visual Computer*, Vol.11, 1995, pp.134-149.

[Dou73] Douglas, D.H., Peucker, T.K., "Algorithms for the reduction of the number of points required to represent a line or its caricature", *The Canadian Cartographer*, 10, 2, 1973, pp.112-122.

[Dyn90] Dyn, N., Levin, D., Gregory, J.A., "A butterfly subdivision scheme for surface interpolation with tension control", *ACM Transactions on Graphics*, Vol.9, N.2, April 1990, pp.160-169.

[Ede87] Edelsbrunner, H., *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.

[Ede92] Edelsbrunner, H., Tan, T.S., "An Upper Bound for Conforming Delaunay Triangulations", *Proceedings 8th ACM Symposium on Computational Geometry*, 1992, pp.53-62.

[Far86] Farin, G., "Triangular Bernstein-Bézier patches", *Computer Aided Geometric Design*, 3(2), 1986, pp.83-128.

[Fek84] Fekete, G., Davis, L.S., "Property spheres: a new representation for 3-D object recognition", *Proceedings Workshop on Computer Vision: Representation and Control*, CS Press, Los Alamitos, CA, May 1984, pp.192-201.

[Fon92] Fong, P., Seidel, H.-P., "An implementation of triangular B-spline surfaces over arbitrary triangulations", *Computer Aided Geometric Design*, 10, 1993, pp.267-275.

[Fow79] Fowler, R.J., Little, J.J., "Automatic extraction of irregular network digital terrain models", *Computer Graphics* 13-3, Aug. 1979, pp.199-207.

[Gom79] Gomez, D., Guzman, A., "Digital model for three-dimensional surface representation", *Geo-Processing*, Vol.1, 1979, pp.53-70.

[Goo92] Goodchild, M.F., Shiren, Y., "A hierarchical spatial data structure for global geographic information systems", *CVGIP - Graphical Models and Image Processing*, Vol.54, No.1, January 1992, pp.31-44.

[Gui85] Guibas, L.J., Stolfi, J., "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams", *ACM Transactions on Graphics*, Vol.4, 1985, pp.74-123.

[Her92] Hershberger, J., Snoeyink, J., "Speeding up the Douglas-Peucker line simplification algorithm", *Proceedings 5th International Symposium on Spatial Data Handling*, Charleston, SC, U.S.A., August, 3-7, 1992, pp.134-143.

[Kir83] Kirkpatrick, D.G., "Optimal search in planar subdivisions", *SIAM Journal of Computing*, 12, 1983, pp.28-35.

[Lee91] Lee, J., "Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models", *International Journal of Geographical Information Systems*, 5,3, 1991, pp.267-285.

[Pot90] Pottmann, H., Eck, M., "Modified multiquadric methods for scattered data interpolation over a sphere", *Computer Aided Geometric Design*, 7, 1990, pp.313-321.

[Pre85] Preparata, F.P., Shamos, M.I., *Computational Geometry: An Introduction*, Springer-Verlag, Berlin, 1985.

[Ren84] Renka, R.J., Cline, A.K., "A triangle-based C^1 interpolation method", *Rocky Mountain Journal of Mathematics*, Vol.14, N.1, 1984, pp.223-237.

[Rip90] Rippa, S., "Minimal roughness property of Delaunay triangulation", *Computer Aided Geometric Design*, 7, 1990, pp.489-497.

[Rip92] Rippa, S., "Adaptive approximations by piecewise linear polynomials on triangulations of subsets of scattered data", *SIAM Journal on Scientific and Statistic Computing*, Vol.13, N.1, 1992, pp.1123-1141.

[Sam90] Samet, H., *Applications of spatial data structures*, Addison Wesley, Reading, MA, 1990.

[Sca92] Scarlatos, L.L., Pavlidis, T., "Hierarchical triangulation using cartographic coherence", *CVGIP: Graphical Models and Image Processing*, 54, 2, 1992, pp.147-161.

[Sch93] Schumaker, L.L., "Computing optimal triangulations using simulated annealing", *Computer Aided Geometric Design*, 10, 1993, pp.329-345.

[Sei92] Seidel, H.P., "Polar forms and triangular B-spline surfaces", *Computing in Euclidean Geometry*, edited by D.-Z. Du, F. Hwang, World Scientific, 1992.

[Sil90] Silvester, P.P., Ferrari, R.L., *Finite Elements for Electrical Engineers (second edition)*, Cambridge University Press, 1990.

[Von87] Von Herzen, B., Barr, A.H., "Accurate triangulations of deformed, intersecting surfaces", *Computer Graphics*, 21, 4, July 1987, pp.103-110.

[Woo85] Woo, T.C., "A combinatorial analysis of boundary data structure schemata", *IEEE Computer Graphics and Applications*, 5, 3, 1985, pp. 19-27.