EUROGRAPHICS '97 / D. Fellner and L. Szirmay-Kalos
(Guest Editors)

Volume 16, (1997), Number 3

Simplification, LOD and Multiresolution Principles and Applications

Enrico Puppo

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

Roberto Scopigno

Istituto CNUCE
Consiglio Nazionale delle Ricerche
Via S. Maria, 36
56126 Pisa, ITALY
Email: r.scopigno@cnuce.cnr.it

Abstract

These tutorial notes provide an introduction, review, and discussion of the state-of-the-art on simplification methods, Level Of Detail, and multiresolution models for surface meshes, and of their applications.
The problem of approximating a surface with a triangular mesh is formally introduced, and major simplification techniques are classified, reviewed, and compared.
A general framework is introduced next, which encompasses all multiresolution surface models based on decomposition, and major multiresolution meshes are classified, reviewed, and compared in the context of such a framework.
Applications of simplification methods, LOD, and multiresolution to computer graphics, virtual reality, geographical information systems, flight simulation, and volume visualization are also reviewed.

1. Introduction

Triangles are the most popular drawing primitive in Computer Graphics (CG). Due to the simplicity of their basic elements and of their connecting structure, triangle meshes are easily manipulated by all graphics libraries and hardware subsystems. In several applications, surfaces are described by very dense triangle meshes. Detailed triangle meshes are obtained by automatically acquiring the shape of physical objects with range scanners, or by extracting isosurfaces from high resolution datasets in volume visualization; huge meshes are needed to describe large terrain areas; displaying CAD surfaces usually involves their conversion into dense triangle meshes; very large meshes may be produced in representing complex scenes in the context of virtual reality applications. Also digital images, especially range images, can be regarded as surfaces, and their reconstruction through triangle meshes can help their processing and understanding.
Storage, manipulation, and rendering of huge meshes may easily require memory and processing resources beyond the power of state-of-the-art computers, especially in a real-time, interactive environment. Therefore, adapting the resolution of a mesh to the needs of each application is a basic issue. For example, using a lower level of detail for small, distant, or background objects may drastically improve performance in data analysis and visualization. Moreover, a highly complex data representation may contain significant geometric redundancy, e.g. when mesh complexity directly depends on the characteristics of the acquisition or fitting process adopted.
This has led to substantial research into devising robust and

efficient techniques for the controlled simplification of surface meshes. Many papers and different approaches have appeared, and potential users are being overwelmed by diffuse, unstable or even contradictory knowledge. Survey papers on surface simplification are still rare [89, 55].

Multiresolution geometric models support the representation and processing of geometric entities at different *Levels Of Detail (LODs)*. Such models are useful in several applied fields to handle geometric data efficiently, depending on specific application needs. The main advantages of a multiresolution model are: data reduction, whenever an approximate representation at low resolution is adequate; fast data access, by exploiting spatial indexing; and hierarchical processing, where speedup is achieved by propagating information across different LODs.

Multiresolution models were proposed in the literature for representing surfaces in computer graphics, virtual reality and GIS, for 3D volumes in CAD and object recognition, for volume data in scientific visualization, and for digital images in computer vision and image processing. Motivations for the use of both simplification and the multiresolution representation of surfaces have been reviewed by Heckbert and Garland [54].
A major approach to multiresolution modeling is based on a decomposition modeling scheme, where resolution is controlled through the level of refinement of the domain decomposition: the underlying idea is that he accuracy in the representation is somehow proportional to the number of cells in the subdivision.
Independently of the decomposition scheme adopted, different architectures have been proposed to handle multiresolution: from a simple collection of versions of an object at different resolutions (see, e.g., the LOD nodes in the OpenInventor toolkit and in VRML [106, 105] and its use in virtual reality applications [39]); to models that maintain relations between consecutive levels of detail (see, e.g., quadtrees, image pyramids, pyramidal and hierarchical triangulations); to more sophisticated models that maintain a compact structure from which many different representations can be extracted, even at variable resolution across domain.

Many other aspects characterize different multiresolution models, such as the shape of cells, the spatial decomposition scheme, the induced hierarchy, etc. Such characteristics make each model more or less suitable to a given application, and no model is best for all needs.

1.0.0.1. Tutorial notes organization. In the following, we focus on two-dimensional models based on the decomposition of a surface into polygonal regions. Extensions to different applications, and higher dimensonal domains are also outlined.

Preliminary formalism and terminology that will be used in the rest of the paper are introduced in Chapter 2.

Chapter 3 presents an introduction to surface mesh simplification methods, and propose a new taxonomy (Section 3.1). The simplification methods are then presented with more details in Section 3.2. Then, an analysis and a comparison of the different approaches adopted to measure the *approximation error* is proposed in Section 3.3, togheter with an empyrical comparison of different simplification codes.

Chapter 4 introduces and characterizes the different multiresolution representation schemes. The *Multi-Triangulation (MT)*, a general framework which allows to interpret all multiresolution models proposed in the literature as special cases of itself, is presented in Section 4.1. Different multiresolution models are then reviewed in Section 4.2. A qualitative evaluation of the models that we reviewed, based on some issues that have relevant impact in surface modeling, processing, and visualization, is given in Section 4.3.

Chapter 5 reviews some applicative domains and the systems or methodologies proposed.

Finally, Chapter 6 briefly list the subjects that, due to space and time limits in preparing these notes, we had to leave out. Some references to literature on such subjects are also given.

Readers are encouraged to send comments or corrections to the authors at: puppo@ima.ge.cnr.it or r.scopigno@cnuce.cnr.it

2. Preliminaries

In this chapter, we give preliminary formalism and terminology that will be used in the rest of the paper. We introduce: surfaces and hypersurfaces, i.e., the objects to be modeled; simplicial meshes, i.e., the modeling tools; and graphs and posets, i.e., combinatorial structures that will be used to organize meshes into multiresolution representations.

2.1. Surfaces and hypersurfaces

2.1.1. Manifold surfaces

A *manifold surface* (or 2-manifold) $\mathcal{S}$ is a subset of Euclidean space $\mathbb{R}^k$, for some $k \geq 3$, such that each point of $\mathcal{S}$ has an open neighborhood homeomorphic to the open disc in $\mathbb{R}^2$.

A *manifold surface with boundary* $\mathcal{S}$ is a subset of Euclidean space $\mathbb{R}^k$, for some $k \geq 2$, such that each point of $\mathcal{S}$ has an open neighborhood homeomorphic to either the open disc, or to the half-disc (which is obtained by intersecting the open disc with the closed half-plane of the positive x coordinates). The *boundary* of a manifold surface with boundary $\mathcal{S}$, denoted $\partial\mathcal{S}$, is the set of all points of $\mathcal{S}$ that do not have a neighborhood homeomorphic to the open disc.

Although most of the methods surveyed in this paper can also deal with non-manifold situations, the manifold condition is commonly assumed, hence we will always conform to it throughout the paper. Hereafter, a surface will be always considered as a 2-manifold, either with or without boundary, embedded in the three-dimensional space $\mathbb{R}^3$.

2.1.2. Parametric surfaces

A *parametric patch* is the image of a continuous function $\psi : \Omega \longrightarrow \mathbb{R}^3$, where Ω is a compact domain in $\mathbb{R}^2$. We denote a parametric patch by $\psi(\Omega)$, with a compact notation that specifies both the function describing the patch and its domain. The space $\mathbb{R}^3$, in which a parametric patch $\psi(\Omega)$ is embedded, is called the *physical space*, while the space $\mathbb{R}^2$ containing the domain Ω is called the *parameter space*. The boundary of domain Ω, denoted $\partial\Omega$, is formed by a finite set of closed curves, called *trimming curves*. We will refer interchangeably to trimming curves in parameter space, and to their images in physical space.

Under suitable conditions on the function ψ, a parametric patch is a manifold surface, possibly with boundary at its trimming curves (i.e., $\partial\psi(\Omega) \subseteq \psi(\partial\Omega)$).

A *parametric surface* is a collection of parametric patches $\mathcal{P} = \{\psi_1(\Omega_1), \ldots, \psi_k(\Omega_k)\}$, such that for each pair of patches $\psi_i(\Omega_i), \psi_j(\Omega_j)$, $i \neq j$, we have $\psi_i(\Omega_i) \cap \psi_j(\Omega_j) = \partial\psi_i(\Omega_i) \cap \partial\psi_j(\Omega_j)$ (i.e., patches do not intersect, while thay can touch along their boundaries). The boundary of a parametric surface $\mathcal{P}$ is contained in the collection of boundaries of its patches, namely, it is formed by all portions of boundaries that are not shared between two different patches. We will always assume that a parametric surface is a 2-manifold, either with or without boundary.

2.1.3. Scalar fields

A *scalar field* is a continuous function $\phi : \Omega \to \mathbb{R}$, where Ω is a compact domain in $\mathbb{R}^k$, $k \geq 1$. The image of ϕ embedded in $\mathbb{R}^{k+1}$ space, i.e., $\mathcal{F} = \{(X, \phi(X)) \mid X \in \Omega\} \subset \mathbb{R}^{k+1}$ is called a *hypersurface*.

For $k = 2$, $\mathcal{F}$ is a 2-manifold with boundary (corresponding to the image of $\partial\Omega$), called an *explicit surface* (in the literature, also terms non-parametric surface, height field, $2\frac{1}{2}$D-surface, functional surface, topographic surface are used intechangeably to denote an explicit surface). For $k = 3$, $\mathcal{F}$ is a *volume* with manifold boundary, embedded in $\mathbb{R}^4$. Although we will not consider scalar fields in higher dimensions, the extension of 3D models to a generic dimension is often straightforward.

2.2. Approximating meshes

Complicated surfaces/hypersurfaces can be approximated through meshes made of polygonal/polyhedral cells for digital processing and rendering purposes. Although most of the models and methods reviewed in the following can be easily used on generic polygonal/polyhedral meshes, for the sake of simplicity, throughout the paper we will always deal with simplicial meshes, namely meshes formed of triangles/tetrahedra, in the 2D/3D cases, respectively.

2.2.1. Simplicial meshes

A *k-simplex* (or a *simplex* of *order k*), with $0 \leq k \leq n$ is a subset t of $\mathbb{R}^n$, defined as the locus of points that are convex combinations of $k+1$ affinely independent points; such points are called the *vertices* of t. Examples of simplexes are a point (0-simplex), a straight-line segment (1-simplex), a triangle (2-simplex), a tetrahedron (3-simplex). The *interior* of a simplex t (denoted by $int(t)$) is the locus of points which can be expressed as convex combinations of the vertices of t, with coefficients restricted in $(0,1)$. We say that a q-simplex t' is a *(proper) face* of a k-simplex t (with $q < k$) if the set of vertices of t' is a subset of the set of vertices of t; if t' is a q-simplex, we say that t' is a *q-face* of t.

A finite set T of simplexes in $\mathbb{R}^n$ is a *simplicial mesh* (also called a *regular simplicial complex* when the following conditions hold:

1. for each simplex $t \in T$, all faces of t belong to T;
2. for each pair of simplexes $t_0, t_1 \in T$, either $t_0 \cap t_1 = \emptyset$ or $t_0 \cap t_1$ is a simplex of T;
3. each simplex t is a face of some simplex t' (possibly coincident with t) having maximum order among all simplices of T.

A simplicial mesh T is called a *d-simplicial-mesh* if d is the maximum among the orders of simplexes belonging to T (d is called the *order* of T). The union of all simplexes of T, regarded as point sets, is the *domain* of T and it is denoted by $\Delta(T)$. A d-simplicial complex whose domain is a (polyhedral) set Ω is called a *simplicial decomposition* of Ω.

We call any finite set of d-simplexes in $\mathbb{R}^n$ a *d-set*. A d-simplicial-mesh is completely characterized by the collection of its d-simplexes, i.e., by its *associated d-set*, therefore we will refer to a mesh and to its associated d-set interchangeably. In the following, we will deal with 2- and 3-simplicial-meshes, also called *triangulations* and *tetrahedrizations*, respectively, embedded in either $\mathbb{R}^2$, or $\mathbb{R}^3$, or $\mathbb{R}^4$. We will refer to a d-simplicial-mesh by calling it simply a mesh, whenever no ambiguity arises.

2.2.2. Approximation error

Whenever a simplicial mesh is used to approximate a domain, or a (hyper)surface, it is important to measure the quality of representation by giving a quantitative estimate of the approximation error. We give formal definitions of the approximation error separately for the three cases of manifold surfaces, parametric surfaces, and scalar fields.

2.2.2.1. Error on manifold surfaces. Let $\mathcal{S}$ be a surface, and T be a mesh approximating it. Each triangle t of T approximates a corresponding "triangular" patch of $\mathcal{S}$. We distinguish among three cases:

1. Surface $\mathcal{S}$ is known at every point, and a *representation function* $\rho : \mathcal{S} \to T$ is defined, mapping each point of the surface into the point representing it in the mesh. Therefore, we can define a difference function $\delta_T : \mathcal{S} \to \mathbb{R}$ as $\delta_T(X) = |X - \rho(X)|$, and an error function
$$E(T,\mathcal{S}) = \|\delta_T\|_{\mathcal{S}},$$
where $\|\cdot\|_{\mathcal{S}}$ is some norm on functions defined over $\mathcal{S}$ (e.g., the L_2 norm, or L_∞ norm).
2. Surface $\mathcal{S}$ is known only at a finite set of sample points $\mathcal{D}$, and for each triangle t of T, the subset $\mathcal{D}_t$ of data spanned by the portion of $\mathcal{S}$ approximated by the triangle is known. In this case, for each point $X \in \mathcal{D}$, we can define the difference $\delta_T(X)$ as the Euclidean distance between X and the triangle t related to it (i.e., such that $X \in \mathcal{D}_t$), and the error function as
$$E(T,\mathcal{S}) = \|\delta_T\|_{\mathcal{D}},$$
where $\|\cdot\|_{\mathcal{D}}$ is a discrete norm. In practice, we could consider either the mean square or the maximum of defferences over all data.
3. No direct correspondence between a point of $\mathcal{S}$ and its representative in T is known. In this case, some combination of Hausdorff distance, and usual norms can be adopted. For instance, we can define:
$$\delta_T(X) = \min_{Y\in T}|X,Y|, \quad X \in \mathcal{S},$$
$$\delta_{\mathcal{S}}(Y) = \min_{X\in \mathcal{S}}|X,Y|, \quad Y \in T,$$
$$E(T,\mathcal{S}) = \oplus(\|\delta_T\|_{\mathcal{S}}, \|\delta_{\mathcal{S}}\|_T),$$
where norms are defined as before, and $\oplus$ can be either an average, or a max, or a min, or a projection operator. In the discrete case, i.e., when $\mathcal{S}$ is known only at a finite set of data $\mathcal{D}$, the definition is analogous by using a discrete norm.

2.2.2.2. Error on parametric surfaces. In this case, the approximation occurs at two levels:

1. in the parameter space, the domain Ω and the trimming curves of $\partial\Omega$ of each parametric patch are approximated with a polygonal domain $\tilde{\Omega}$, and polygons of $\partial\tilde{\Omega}$, respectively;
2. the approximated domain $\tilde{\Omega}$ is triangulated, and each triangle of the resulting triangulation $\tilde{T}$ is pojected into physical space through a function $\tilde{\psi}$ to obtain the mesh T approximating $\mathcal{P}$.

Therefore, if we have a function $\tilde{\rho} : \Omega \to \tilde{\Omega}$ mapping each point in parameter space into its approximation, then a representation function $\rho : \mathcal{P} \to T$ relating the surface to its approximation is defined by the commutative diagram

$$\begin{array}{ccc} \Omega & \xrightarrow{\tilde{\rho}} & \tilde{\Omega} \\ \psi\downarrow & & \downarrow\tilde{\psi} \\ \mathcal{S} & \xrightarrow{\rho} & T \end{array}$$

and the definition of error given in the first case of the previous paragraph can be applied. An alternative is to adopt a Hausdorff distance directly in physical space, as in the third case of the previous paragraph.

2.2.2.3. Error on scalar fields. Also in this case, the domain Ω is approximated through a polygonal/polihedral region $\tilde{\Omega}$, where a triangulation $\tilde{T}$ is defined, and the approximating mesh T is defined by the image of a piecewise linear function $\tilde{\phi} : \tilde{\Omega} \to \mathbb{R}$, which is linear over each simplex of $\tilde{T}$. An error function analogous to that of parametric surfaces can be used.

More often, however, surface $\mathcal{F}$ is known only on a finite set of samples $\mathcal{D}$. In this case, domain Ω is assumed to be a polygon/polihedron containing alla data (usually, their convex hull), hence $\tilde{\Omega} \equiv \Omega$. Therefore, the error at each datum X is simply given by $e(X) = |\phi(X) - \tilde{\phi}(X)|$, and the error function is defined by a discrete norm:
$$E(T,\mathcal{F}) = \|e(X)\|_{\mathcal{D}},$$
e.g., the maximum error, or the mean square error at data points.

2.3. Combinatorial structures

The following combinatorial structures will be used in the definition of multiresolution models.

2.3.1. Graphs and trees

A *directed graph* is a pair $G = (N,A)$, where N is a finite set of *nodes*, and $A \subseteq N \times N$ is a set of *arcs*. An arc (n_1,n_2) will be said *outgoing* from n_1, and *incoming* to n_2. A node with no incoming arcs is called a *source* or *root*, a node with no outgoing arcs is called a *drain* or *leaf*. A node that is neither a source nor a drain is said *internal*.

A *path* in G is a sequence $n_1,\dots,n_k$ of nodes such that for each $i = 1,\dots,k-1$ the pair (n_i,n_{i+1}) is an arc of A. The *length* of a path is the number of its nodes. A *cycle* is a path $n_1,\dots,n_k$ with $n_1 = n_k$. We will write $n_1 \rightharpoonup n_2$ to mean that there exists a path from n_1 to n_2, and $n_1 \leftrightarrow n_2$ to mean that either $n_1 \rightharpoonup n_2$, or $n_2 \rightharpoonup n_1$, or both; we will write $n_1 \not\rightharpoonup n_2$, and $n_1 \not\leftrightarrow n_2$ to negate such assertions, respectively.

G is said *connected* if for any two nodes n_1,n_2, there exists a node n_0 (possibly coincident with either n_1 or n_2), such that $n_1 \leftrightarrow n_0$, and $n_0 \leftrightarrow n_2$.

A *Directed Acyclic Graph (DAG)* is a connected directed graph G with no cycles (i.e., such that for any two nodes n_1,n_2, we have $n_1 \rightharpoonup n_2 \;\Rightarrow\; n_2 \not\rightharpoonup n_1$). Given two nodes

n_1, n_2 in a DAG, if (n_1, n_2) is an arc, then we say that n_1 is a *parent* of n_2, and n_2 is a *child* of n_1; if $n1 \rightharpoonup n2$ we say that n_1 is an *ancestor* of n_2, and n_2 is a *descendant* of n_1.

A *rooted DAG* is a DAG with a unique source (root); a *lattice* is a DAG with a unique source, and a unique drain; a *tree* is a rooted DAG such that for any (non-root) node n_i there exists exactly one path from the root to n_i. The *level* of a node n_i in a tree [in a rooted DAG] is the [minimum] length of a path from the root to n_i.

A *subgraph* $G' = (N', A')$ of a graph $G = (N, A)$ is a graph such that $N' \subseteq N$, $A' \subseteq A$, and each arc of A' joins nodes of N'. A *consistent subgraph* of a rooted DAG (or a tree) G is a subgraph G' of G containing its root, and such that for each node $n \in N'$ also all its ancestors belong to N', and all arcs of A joining nodes of N' belong to A'. Given a consistent subgraph G' of a DAG (or a tree) G, the set $A^* \subset A$ of arcs connecting the leaves of G' to nodes of $N \setminus N'$ is called a *cut* in G generating G'.

2.3.2. Posets

Let $\mathcal{C}$ be a finite set. A *partial order* on $\mathcal{C}$ is a reflexive, antisymmetric and transitive relation $\leq$ on its elements. For every $c, c' \in \mathcal{C}$, we will write $c < c'$ to mean that $c \leq c'$ and $c \neq c'$; we will write $c \prec c'$ to mean that $c < c'$ and there does not exist any other c'' such that $c < c'' < c'$. Note that relation $\leq$ is the reflexive closure of relation $<$, which is in turn the transitive closure of relation $\prec$. A pair $(\mathcal{C}, \leq)$ is called a *poset*.

An element $c \in \mathcal{C}$ is said *minimal* if there does not exist any $c' \in \mathcal{C}$ such that $c' < c$; $c \in \mathcal{C}$ is the *least* element of $\mathcal{C}$ if $c \leq c'$ for any $c' \in \mathcal{C}$.

A subset $\mathcal{C}' \subseteq \mathcal{C}$ is called a *lower set* if $\forall c' \in \mathcal{C}', \forall c \leq c'$ then $c \in \mathcal{C}'$. For any $c \in \mathcal{C}$, the set $\mathcal{C}_c = \{c' \in \mathcal{C} \mid c' \leq c\}$ is the smallest lower set containing c, and it is called the *down-closure* of c. We also define the *sub-closure* of c as $\mathcal{C}_c^- = \{c' \in \mathcal{C} \mid c' < c\} = \mathcal{C}_c \setminus \{c\}$.

Given a lower set $\mathcal{C}' \subseteq \mathcal{C}$, a *consistent order* on $\mathcal{C}'$ is any total order $\leq_{\mathcal{C}'}$ on its elements such that $\forall c, c' \in \mathcal{C}', c \leq c' \Rightarrow c \leq_{\mathcal{C}'} c'$.

A poset can be represented by a DAG having $\mathcal{C}$ as set of nodes and an arc from c to c' whenever $c \prec c'$. We will often refer to $(\mathcal{C}, \leq)$ and to the DAG representing it interchangeably. Therefore, we will use interchangeably the following terminology: poset - DAG; element - node; least element - root; lower set - consistent subgraph. Note that a cut in a DAG always defines a consistent subgraph, hence a lower set in the corresponding poset.

3. Mesh Simplification

In this chapter we initially give a brief introduction to surface mesh simplification methods, and propose a new taxonomy (Section 3.1). The scope of our analysis is limited to simplification methods for manifold or non-manifold surfaces immersed in 3D space. The many other approaches proposed for other types of data (images, height fields, range maps or triangulated terrains) [66, 55] are not considered here. The simplification methods are then presented with more details in Section 3.2. Then, an analysis and a comparison of the different approaches adopted to measure the *approximation error* is proposed in the last section .

3.1. Simplification techniques

Substantial results have been reported in the last few years on surface simplification. The data domain of the solutions proposed generally covers all types of triangular meshes (e.g. laser range data, terrains, synthetic surfaces). Different error criteria have been devised to measure the fitness of the approximated surfaces. Any level of reduction can be obtained with most of the approaches listed below, on the condition that a sufficiently coarse approximation threshold is set. The following are some of the existing methods:

- *coplanar facets merging*:
  coplanar or nearly coplanar facets are searched for in the mesh, merged into larger polygons, and then retriangulated into fewer facets than those originally required [47, 62, 56]; face merging is driven by a co-planarity test.
  The *superfaces* method [61] extends this approach by providing bounded approximations and more robust retriangulations of the merged faces;
- *controlled vertex/edge/face decimation*:
  these methods work by the iterative elimination of components (vertices, edges, triangles), chosen upon local geometric optimality criteria. All decimation methods are restricted to manifold surfaces, and generally preserve mesh topology.
  - the original *mesh decimation* approach [97] applies multiple passes over the triangle mesh and progressively removes those vertices that pass a distance or angle criterion (based on local geometry and topology checks). The resulting holes are then patched using a local retriangulation process. The candidate vertex selection criterion is based on a *local error* evaluation;
  - a decimation approach can also be adopted to simplify a mesh by iteratively collapsing edges in a single vertex [45, 88, 1], or by collapsing faces [48];
  - extensions to the decimation method which support *global error* control have been proposed (*Global error* is defined here in opposition to *local error*, i.e. whether the approximation error introduced by the elimination of the current vertex is operated by comparing the resulting new mesh patch with the initial mesh M^0 or with the intermediate, partially simplified mesh M^i; see Section 3.3.1 on error evaluation for a more precise definition). In particular, the *simplification envelopes*

method [21] supports bounded error control by forcing the simplified mesh to lie between two offset surfaces (but it works only on orientable manifold surfaces). Some other methods adopt heuristics for the evaluation of the *global* error introduced by each vertex removal and re-triangulation step, and work under an incremental simplification framework [99, 11, 3, 63, 88, 45];
  - controlled local modifications of re-triangulated patches, based on edge flipping, have been proposed to improve approximation accuracy in mesh decimation [11, 3];
  - the decimation approach has also been generalized to the simplification of 3D simplicial decompositions (tetrahedral sets) [87, 14, 49];

- *re–tiling*:
  new vertices are inserted at random on the original surface mesh, and then moved on the surface to be displaced over maximal curvature locations; the original vertices are then iteratively removed and a re–tiled mesh, built on the new vertices, is given in output [103];
- *energy function optimization*:
  the *mesh optimization* approach, originally proposed in [59], defines an *energy function* which measures the "quality" of each reduced mesh. Mesh reduction is iteratively obtained by performing legal moves on mesh edges: collapsing, swapping or splitting (in the latter case, a new vertex is inserted in the edge and two new edges connect it to the front-most vertices). Legal moves selection is driven by an optimization process of the energy function. At each step, the element whose elimination causes the lowest increase in the energy function is deleted.
  An enhanced version, *progressive meshes*, provides multiresolution management, mesh compression, selective refinements and enhanced computational efficiency [57, 80], and is based only on edge collapsing actions;
- *vertex clustering*:
  based on geometric proximity, this approach groups vertices into clusters, and for each cluster it computes a new representative vertex [90]. The method is efficient, but neither topology nor small-scale shape details are preserved. The visual and geometric quality of the meshes simplified with a clustering approach have been improved in [72].
  Another extension to the clustering approach was proposed to cope with the perceptual effects of degradation [86]. Here, clustering is driven by bounding box decomposition into subvolumes; couples of edges internal to each subvolume are merged on a test based on curvature and size.
  A very recent approach [40] applies an efficient error evaluation, based on *quadric error matrices*, to a less general clustering approach which performs only vertex pair contractions (a vertex pair is eligible for contraction if either a connecting edge exists or the vertices satisfy a proximity criterion). The solution is characterized by its high computational effciency and the capability to simplify disconnected or non-manifold meshes;
- *wavelet-based approaches*:
  the wavelet decomposition approach seems very promising for surface simplification (and, moreover, multiresolution comes for free). Conversely, a regular, hierarchical decomposition is required to support wavelet decomposition, and computational efficiency is not at the best. Wavelet approaches have been proposed to manage regularly gridded meshes [42, 53] or more generic meshes [36, 8].
  In particular, the *multiresolution analysis* approach is based on a three-phase process (re–meshing, re-sampling and wavelet parametrization) to build a multiresolution representation of the surface, from which any approximated representation can be extracted [36]. An extension to this approach has recently been proposed to manage the approximation of both geometry and surface color [8];
- *simplification via intermediate hierarchical representation*:
  an intermediate octree representation [2] may be adopted to automatically produce simplified representations, because the octree may be purged at various levels and then converted into a (simplified) boundary representation;
  alternatively, an intermediate voxel-based hierarchical representation (built using signal-processing techniques for the controlled elimination of high-frequency details), together with adaptive surface fitting, was proposed in [50, 51].

Several approaches have also been proposed for particular occurrences of the simplification problem (but, because of no general applicability, they will not be discussed in the rest of the tutorial). Among these are: the simplification of *range maps*, i.e. regular meshes automatically acquired by a range scanner (see for example the *adaptive subdivision* approach by De Haemer and Zyda [47]); or the simplification of the isosurfaces fitted on high resolution volume datasets. In the latter case, some less general techniques – peculiar to volume rendering applications – have been proposed, and may be subdivided in the two following main streams:

- *adaptive fitting* approaches: ad hoc data traversing and fitting are devised to reduce output data complexity: (a) by using approximated fitting [22], (b) by bending the mesh [75], (c) by adapting the poligonalization criterion to the shape of the surface [6, 76], (d) by subdividing bi-cubic patches until they are sufficiently close to the underlying samples [95];
- *datasets simplification* approaches: first, the volume datasets is organized into a hierarchical [108, 50, 22] or a multiresolution [12, 14] representation; then, isosurfaces are fitted onto the simplified datasets.

3.1.1. A characterization

Simplification approaches may be classified firstly by characterizing the input and output domains.

All of the methods briefly reviewed above accept in *input* simplicial meshes, but only a few of them can manage non–manifold meshes (e.g. vertex clustering and intermediate hierarchical representation).

If we take into account the *output*, and given the scope of this tutorial (to review simplification *and* multiresolution), an simplification methods characterization can be based on the capability to produce a *multiresolution output*. Only a few approaches support directly this feature (multiresolution analysis, progressive meshes, multiresolution decimation), but most other incremental methods can be simply extended to support it. The adoption of an *incremental* approach is thus another important characteristics. A simplification method is characterized as an incremental one when simplification proceeds through a sequence of local mesh updates which, at each step, reduce the mesh size and monotonically decrease the approximation precision (e.g. mesh decimation, progressive meshes, clustering via quadric matrices).
If we consider the topology of the meshes produced in output, most simplification approaches produce manifold simplicial meshes (e.g. decimation, energy function optimization, re-tiling), while others may produce not 2-manifold geometries (e.g. vertex clustering may produce dangling faces, edges, or points). Moreover, simplification methods may be characterized by highlighting two main orthogonal classes [96]:

- approaches which *preserve mesh topology* (e.g. mesh decimation, mesh optimization), and those which don't (e.g. vertex clustering, intermediate hierarchical representation);
- approaches based on *vertex subset selection* (e.g. coplanar facets merging, mesh decimation) or *re-sampling* (e.g. mesh optimization, re–tiling, multiresolution analysis, intermediate hierarchical representation).

The importance of *preserving mesh topology* depends directly on the application domain. It is not mandatory if the goal is to speedup rendering, at least for the lower resolution representation of a LOD model (and topology simplification is generally a must to produce highly simplified models out of topology–rich objects). On the other hand, topology has to be preserved if the simplification goal is to produce a representation which might be nearly indistinguishable from the original, or which preserves shape features (e.g. medical application requirements).
The choice between using a *subset* of the original vertices or using *re-sampled* vertices again depends on the application and this usually affects approximation precision. There are, in fact, many applications where re-sampling is not allowed or feasible, e.g. in the case of datasets where the sampling of a scalar/vectorial field is associated with the mesh vertices and we cannot safely recompute the field value in the re-sampled locations. On the other hand, better approximation accuracy is obtained when vertices are resampled, e.g. by moving the vertices on the lines of maximal curvature.

Another possible classification may be based on the simplification goal [21]:

- *Min-#*: when, given some error bound ε, the objective is to build the approximated mesh of a minimal size which satisfies precision ε (size is generally measured in number of vertices);
- *Min-varepsilon*: when, given an expected size for the approximated mesh, the objective is to minimize the error, or difference, between the original and the resulting mesh.

Other important characteristics are:

- the adoption of a *local* or a *global* approach; in the first case, mesh modifications are operated upon a local optimization criterion (e.g. simplification envelopes and other decimation approaches); in the second one, a global optimization process is applied to the whole mesh (e.g. energy optimization approaches, re-tiling, multiresolution decimation, and multiresolution analysis);
- the measurability, and preservation under tight bounds of the *approximation error* introduced (e.g. simplification envelopes and some other decimation approaches);
- the preservation of *geometric* or *attribute discontinuities* of the mesh, for example feature edges and color or pictorial information (e.g. mesh decimation, progressive meshes).

An attempt to give an overall characterization of different simplification algorithms is presented in Table 1. Columns 2-4 characterize the strategy adopted to manage mesh approximation: the goal which drives the simplification process (*Min-#*, *Min-*ε, or *both*); if the approach simplifies the mesh *incrementally*; and the *topologic entity* taken into account during simplification (*v*: vertices, *e*: edges, *f*: faces, *v-pair*: vertex pairs). Columns 5-8 characterize the approximation error management policy. The ε_{loc} column is marked if for each simplification step the L_∞ *error* introduced is evaluated by a *local* shape comparison between the modified patch and the corresponding patch just before the current step; ε_{glob} is marked if a *global* shape comparison with the starting input mesh is performed (using an L_∞ norm again); or column *other* is marked if another policy is adopted, e.g. energy function optimization (which adopts an L_2 norm), or clustering evaluation. Moreover, we mark those methods which guarantee *bounded* accuracy on the whole mesh in column 8. The multiresolution column highlights those methods which produce in a single run a real *multiresolution output*, encoded with an *ad hoc* representation. *Preserving mesh characteristics* is evaluated in columns 10-12 in terms of: the preservation of global mesh topology (column *meshTop*); possible relocation of the vertices of the simplified mesh (column *vertLoc*), with value *unchanged* or *relocated*; the preservation of feature/solid edges or angles (column *featEdg*). The estimated simplification *speed* reported in column 13 (measured in *KTr/sec*, i.e. thousands of triangles simplified for CPU second) has been taken directly from the results presented in the original papers. Since these results were obtained on different meshes and on different machines, they only give a rough and imprecise estimate of the efficiency of the algorithms, but are presented in the table to give the order of magnitude of simplification times (and also to emphasize proposals which did not report any evaluation of running times, indicated in the table with the "*??*" tag). Finally, column 14 lists whether the code is available in the *public domain*, as part of a *commercial product*, or is *not available* at all.

The capability to preserve discontinuities of vertices/faces attributes is a very important feature, but it has been not included in Table 1. This is because although this feature is only supported by a few proposals [86, 57, 8, 98], most other approaches could simply be extended to support it (e.g. by providing an enhanced classification of vertices for the vertex decimation approach).

An overall comparison of simplification approaches is not easy, because simplification accuracy largely depends on the geometric and topological structure of the input mesh and on the required results. For example, the presence of sharp edges or solid angles is managed better by *coplanar facet merging* and *decimation* approach, while on smooth surfaces *mesh optimization* and *re–tiling* give better results. On the other hand, the good results in the precision and conciseness of the output mesh given by *mesh optimization* and *re–tiling* techniques are counterbalanced by substantial processing times. Although no time comparisons between different methods have been reported in the literature, an informed guess would be that the *mesh decimation* and the recent *quadric matrices clustering* approaches are the most efficient methods.

3.2. Review of simplification methods

3.2.1. Coplanar facets merging

All of the following approaches are based on the idea to detect and merge set of nearly-coplanar faces, appeared first in [47, 62].

3.2.1.1. Geometric Optimization (Hinkler, Hansen). The Geometric Optimization method [56] detects in a single pass coplanar or nearly coplanar triangular patches, merge these triangles in larger polygons and then re–triangulates them in fewer triangles. The methods is based on: (a) quick grouping facets in nearly coplanar sets; (b) a fast approach for merging coplanar sets; and (c) simple and robust re–triangulation.

Coplanar face sets are grouped by sorting faces on their respective normals (normals are immersed in a discrete 3D space). The error criterion is based on the angular difference between normals: faces are merged if their global normals spans in a $(-\varepsilon, +\varepsilon)$ interval.

Evaluation
Because only one grouping of the nearly coplanar facets is applied, the criterion may be classified as *global* but, because it estimates only differences among normals, the evaluation of the error is highly *approximated* and inaccurate (due to the surface invariance of the criterion, the same difference on normals may results in different errors for couple of small or large facets).
The simplified mesh do not assure a bounded precision.
Significant sharp edges are maintained by definition.
The new mesh has vertices which are a subset of the original ones.
Times: authors state the method reduces from 0.7 to 2.7 K Tr/sec (workstation model not declared).

3.2.1.2. Superfaces (Kalvin, Taylor). The *Superfaces* method [61] merges nearly-planar faces as well, but it supports bounded approximation and more roboust re-triangulation.
To simplify a surface mesh *S*, the faces of *S* are grouped into a set of surface patches (called *superfaces*) and each surface patch is approximated with a new triangulation which satisfies a given tolerance (no vertices of the initial mesh is at a distance larger than a user-specified tolerance from the re-triangulated superfaces).
Superfaces are created by growing from seeds (a single original face, randomly selected). The criterion used to merge an adjacent face into the current growing superface is twofold: its vertices have to be within an $\varepsilon/2$ distance from the plane which approximates the superface, and face orientations must be similar to those of the faces already inserted in the set. Once detected, superfaces borders are simplified (different strategy for border straightening are provided, based on co-linear vertices removal). Re-triangulation is operated by dividing superfaces into star-shaped polygons and then decomposing them.

Evaluation
It is an efficient approach, which provides bounded precision, preserves topology and maintains significant sharp edges.
The criterion used to detect nearly co-planar faces is more precise than Hinkler and Hansen's one [56].
The new mesh has vertices which are a subset of the original ones.
Times: authors state their method is more efficient than the Geometric Optimization one. It reduces from 0.3 to 0.8 KTr/sec, on a mesh with 350K triangles extracted from a volume dataset (human skull), on a IBM R6000/550.

3.2.2. Controlled vertex/edge/face decimation

Methods based on the iterative elimination of components (vertices, edges, triangles) and driven by euristics on local

	Method Char.			**Approximation Error**				**Multi-res**	**Preserve Mesh Charact.**			**Speed**	**Availability**
	optim. goal	incre-mental	top. entity	ε_{loc}	ε_{glob}	other crit.	bound.	output	mesh topol.	vert. locat.	feature edges	KTr/sec.	
Coplanar Facet Merging Approaches													
Geom. Opt. [56]	Min-#		f			x	no		yes	unch.	yes	0.7-2.7	not avail.
Superfaces [61]	Min-#		f		x		yes		yes	unch.	yes	0.3-0.8	not avail.
Decimation Approaches													
Mesh Decimat. [97]	Min-ε	x	v	x			no		yes	unch.	yes	2.-2.5	publ.dom.
Triangle Remov. [48]	Min-#	x	f			x	no		yes	unch.	yes	??	not avail.
Hierarch.Triang.[99]	Min-#	x	v		x		yes		yes	unch.	yes	??	comm.prod.
Err.Bound.TMR [3]	Min-#	x	v		x		yes		yes	unch.	yes	??	not avail.
Multires.Dec. [11]	both	x	v		x		yes	x	yes	unch.	yes	0.15-0.2	publ.dom.
Hausd.Distance [63]	Min-#	x	v		x		yes		yes	unch.	yes	??	not avail.
Simpl.Envelop. [21]	Min-#		v		x		yes		yes	unch.	yes	0.07-0.09	publ.dom.
Toler.Volumes [45]	Min-#	x	e		x		yes		yes	reloc.	yes	0.08-0.1	not avail.
Full-range Appr. [88]	both	x	e		x		yes		no	unch.	yes	??	not avail.
Mesh Simpl. [1]	Min-#		e	x			no		yes	reloc.	yes	0.2	not avail.
Energy Optimization Approaches													
Mesh Opt. [59]	Min-#	x	v+e			x	no		yes	reloc.	≈	0.008	publ.dom.
Progr.Meshes [57]	Min-#	x	e			x	no	x	yes	reloc.	yes	0.04	not avail.
Clustering Approaches													
Vert.Clust. [90]	Min-#		v+e+f			x	yes		no	reloc.	no	??	comm.prod.
Percept.Clust. [86]	Min-#		e			x	yes		no	unch.	yes	0.1-0.05	not avail.
Quadric Err.Matr. [40]	both?	x	v-pairs		x		no	x	no	reloc.	no	4.5	not avail.
Intermediate Hierarchical Representation Approaches													
Octree-based [2]	Min-#		–			x	yes		no	reloc.	no	??	not avail.
Voxel-based [51]	Min-#		–			x	yes		no	reloc.	no	??	not avail.
Other Approaches													
Re-Tiling [103]	Min-ε		v			x	no		yes	reloc.	no	??	not avail.
Multires.Anal. [36]	Min-#		–		x		yes	x	yes	reloc.	no	0.04	not avail.

Table 1: *Characterization of different simplification algorithms.*

geometrical optimality, which evaluate the (local/global) approximation error generated by each new elimination.

3.2.2.1. Decimation of Triangle Meshes (Schroeder, Zarge, Lorensen). The Mesh Decimation method [97] reduces triangle mesh complexity by applying multiple passes over the mesh and using local geometry and topology to remove vertices that pass a distance or angle criterion. For each removed vertex, the resulting hole is patched using a local triangulation process.
The Decimation method first classifies mesh vertices in the six classes presented in Figure 1. Only vertices of class *simple*, *boundary* and *interior edge* are candidate for removal.
The decimation criterion is based on *local error* (*Local error* as opposed to *global error*, i.e. whether the error evaluation is operated on the current partially simplified mesh M_i or on the original mesh M_0. See Section 3.3.1). evaluation. For each vertex candidate to elimination the algorithm computes: the distance of the vertex from the *average plane*, in the case of *simple* vertices; the distance of the vertex from the new *boundary edge*, in the case of *boundary* vertices, or from the new interior edge, in the case of *interior edge vertices*. If distances are lower than a user-specified threshold, then the vertex is removed and the resulting surrounding triangle patch is re-triangulated (a recursive loop splitting algorithm is adopted for re-triangulation).

Evaluation
It is a fast method, but it does not assure bounded approximation. It computes an *approximated* estimate of the *local error*, which does not take into account the relation between the new patch and the original mesh (and error therefore accumulates).
On the other hand, significant sharp edges are maintained by adopting a *threshold* for the maximal *solid angle* (used for interior edges or corner vertices classification), which prevents vertex elimination on the sharp edges. The new mesh has vertices which are subset of the original ones.
An implementation of the Mesh Decimation algorithm is available on the web, part of the Visualization Toolkit (VTK) by Bill Lorensen, Ken Martin and William Schroeder, at http://www.cs.rpi.edu/∼martink/.
Times: no info in the original paper, but timings have been reported in [68]: 2.-2.5 KTr/sec on a SGI Onix and very complex meshes (hundreds of thousands of faces extracted from the Visible Human dataset). For other results see our tests in Subsection 3.3.3.

3.2.2.2. Triangle Removal (Hamann). The Triangle Removal method [48] reduces mesh complexity by removing triangles. The criterion to weight candidate triangles is based on a local estimate of the surface curvature (each triangle is assigned a weight proportional to the sum of the absolute curvature at its vertices) and of triangle equiangularity. Then, iteratively, the triangle with the lowest weight is removed, replaced by a new point that lies on a local surface

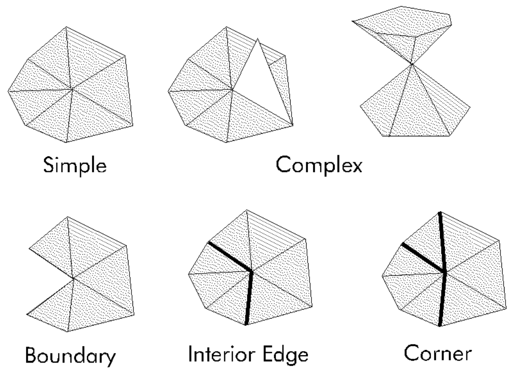


Figure 1: *Mesh Decimation: vertices classification.*

approximant point, and the affected region is re-triangulated and re-weighted.

Evaluation
The evaluation of the approximation error introduced is not supported, unless surfaces obtained by the graph of a known bivariate function are used. The criterion proposed to detect removable triangles seems not very computationally efficient
The new mesh will contain new vertices which are relocated on the removed surface patch.
Times: no info is reported.

3.2.2.3. Hierarchical Triangulation Modeling (Soucy Laurendeau). The Hierarchical Triangulation Modeling method [99] extends the Schroeder et al. approach [97] by providing a global error management. In particular, for each face of the intermediate simplified mesh it stores all of the removed vertices which project on the face. A *global error approximation* is evaluated by computing for each face of a new patch the distance between these removed vertices and the face. The global error evaluated is therefore an approximation (a slight underestimation) of the real error. To determine if a vertex can be decimated, it has to be temporarily removed, the patch has to be re-triangulated and errors for all the faces of the new patch are evaluated. Only if these errors are lower then the selected error threshold, then the vertex is decimated.

Patch re-triangulation is fulfilled via 2D constrained Delauny triangulation. Moreover, the paper discuss: conditions for retriangulation, curvature optimization, discontinuity preservation, and special handling of contour vertices.

An interesting extension of this method has been proposed to manage the problem of both compressing shape and preserving color detail through texture mapping [98]. A mapping is mantained during simplification between the original mesh vertices and the simplified mesh faces. Once the mesh has been simplified, a texture is built for each face using the color of the associated removed vertices.

Evaluation
The approach computes an approximated estimate of the gloal error (under-estimate based on *vertex – triangle* distance computation).
The new mesh has vertices which are a subset of the original ones.
This proposal is the base of a commercial simplification package (IMEdit/IMCompress 2.0 by Innovmetric, http://www.innovmetric.com./).
Times: no info is reported.

3.2.2.4. Error-bounded Triangle Mesh Reduction (Bajaj and Schikore). The Error-bounded Triangle Mesh Reduction method [3] follows the vertex decimation strategy to produce a reduced model in which errors in both the geometric representation and any number of scalar variables defined at the nodes of the surface mesh are bounded by a user-specified level.
It proposes a novel method for global *error propagation* (originally devised for 2D meshes with scalar data [94]), which bounds the errors incurred from all steps of mesh simplification. For each decimation step, a mapping is defined between the removed patch and the new patch resulting from the deletion of the candidate vertex. This mapping allows to quantify errors in both the geometry and the data defined on the mesh, because the mutual projection of the two patches segments the triangulations into pieces within which the variables and the geometry all vary linearly. Thus, it suffices to compute errors at the intersections of the projected edges in order to compute the error bound for each new triangulated patch.
Edge flipping is also proposed to find, in a greedy manner and for each decimation step, the new patch which gives the lowest error.

Evaluation
The simplified mesh is defined on a subset of the original vertices.
A global error criterion is proposed (based on *mesh – mesh* comparison).
Edge flipping is proposed to improve new patch approximation.
Times: no times produced; experiments are run on isosurface from volume data, no standard dataset used.

3.2.2.5. Multiresolution Decimation (Ciampalini et al.). The Multiresolution Decimation method [11] extends the Mesh Decimation approach by providing both increased approximation precision, based on global error management, and multiresolution output. Error evaluation is based on a mixed *local error propagation* and *direct global error* evaluation. In the first case, for each decimation step an estimate of the *local error* is computed by computing a *mesh–to–mesh* distance between the removed and the re-triangulated patches; local errors are then propagated to give a global bound. In the second case, *global error* is directly estimated by maintaining for each simplified face the set of decimated vertices which map on the face, and by computing *vertex–to–face* distances. Then, to reduce error under-estimation which is possible with both approaches, the larger of the two is set at each step the estimated *global error*. Candidate vertices are incrementally decimated in a number of passes, and at each pass the global error threshold ε_i value is increased, until ε_i becomes higher than the user selected target precision ε. The explicit ordering in vertex removal has been adopted to improve simplified mesh quality and to increase the number of different level of detail representations produced in the multiresolution output. The quality of the simplified mesh is also enhanced by performing flipping on each simplified patch (the edge flipping test is driven by heuristics which evaluates mesh accuracy).

The multiresolution output (the list of all of the faces generated during the simplification process, with for each face its approximation accuracy interval) allows the efficient extraction of whichever level of detail representation in the approximation accuracy interval $(0.,\varepsilon)$ in interactive times.

Evaluation
The new mesh has vertices which are a subset of the original ones.
The adoption of a global error criterion allows to simply derive from the simplification process a *multi–resolution* mesh representation.
The code is available at http://miles.cnuce.cnr.it/cg/jade.html.
Times: it simplifies from 0.15 – 0.2 K Tr/sec, on a SGI Indigo2 workstation. For other results see our tests in Section 3.3.3.

3.2.2.6. Hausdorff–distance Enhanced Decimation (Klein, Leibich, Strasser). The Hausdorff–distance Enhanced Decimation method [63] extends the Mesh Decimation approach by replacing its approximated local error estimate with the Hausdorff distance between the original and the simplified mesh (a global error criterion). This *mesh–to–mesh* distance is evaluated by maintaining trace, during simplification, of all the original faces that map to simplified faces, and then computing *face–to–mesh* distances. For all the original faces which are contained into a simplified face, the computation of *vertices–to–face* distances is enough. Conversely, for all original faces which map into more than one simplified face, their distance has to be computed by evaluating distances of mutual edge intersections or by adaptively decomposing the face.
Using the Hausdorff distance allows a more precise and symmetric evaluation of the error. Moreover, because this distance is always less than the non symmetric distance induced by the L^{∞} Norm, a Hausdorff–based decimator provides reduction rates higher than those of the canonical algorithm.

Evaluation
The new mesh has vertices which are subset of the original ones.
It adopts a global error criterion (but it seems of faily complex evaluation).
Times: no times produced; experiments on CAD and medical data, no standard dataset used.

3.2.2.7. Simplification Envelopes (Cohen, Varshney et al.). The Simplification Envelopes method [21] extends Decimation methods with global error control. The simplified mesh is forced to lie within a user specifiable distance ε from the original mesh M. It manages manifold bordered meshes or of meshes with holes and guarantees preservation of topology.
Given the requested precision ε, the method builds two *simplification envelopes* $M^{+\varepsilon}$ and $M^{-\varepsilon}$, defined as poligonal surfaces which approximate the offset surfaces built at distance $+\varepsilon/-\varepsilon$ from M. An important difference between envelopes and offset surfaces is that envelopes do not self-intersect by construction.
Once the envelopes have been constructed, two different algorithms are proposed to simplify the mesh. The first one is a *local* approach, similar to Mesh Decimation (vertex selection, patch removal and re-triangulation); re-triangulation is driven by a greedy strategy, which looks for equiangular aspect ratio (but it does not guarantee optimality). The second one, a *global* approach, has the advantage of performing multiple vertices removal in a single step, thus preventing to be trapped in "local minima" during the simplification process, but requires quadratic times and is therefore unefficient on large models.

Simplification of border edges (edges adjacent to a single face) may not be managed with the standard surface envelopes. An auxiliary data structure, the *border tubes* (pseudo-cylindrical surfaces built on the border edges), is constructed and used to manage border edge simplification: each simplified edge is accepted only if it does not intersect the border tube surfaces.

Adaptive approximation is proposed, because during the construction of the envelopes it is possible to smootly control the level of approximation (for example, let the ε-value varies proportionally to the distance from the observer). But the definition of an analitical rule to drive smooth and continuos variations of the ε-value on the mesh surface is not simple.

Evaluation
It manages only manifold triangle meshes, with no degeneracies (e.g. no coincident triangles or T-vertices).
The new mesh has vertices which are a subset of the original ones.
Adoption of a roboust global error criterion, which also preserves sharp-edges, topology and prevents self-intersection of the simplified mesh (the latter is an exclusive feature).
Times: experiments on CAD and range scanned data; 0.09 Ktriang/sec simplified on the bunny dataset (69K faces) and 0.07 Ktriang/sec on the phone dataset (165K faces) on a HP 735/125 ws. For other results see our tests in Section 3.3.3.

3.2.2.8. Simplification Inside a Tolerance Volume (Gueziec). The Simplification Inside a Tolerance Volume method [45, 44] adopts an edge collapse (Figure 2) approach to decimate meshes. Candidate edges are selected upon their length, shorter first, and four tests are performed to check for edge decimability: to prevent changes in mesh topology, to avoid the generation of sliver faces (the paper defines a measure for the *compactness* of a triangle, measured in terms of its area and edges length: $c = \frac{4\sqrt{3}a}{l_0^2+l_1^2+l_2^2}$), to reduce surface normals variation and to maintain a bounded error. The latter test is the most original part of the paper. It constrains the reduced mesh to stay within a tubular neighborhood of known width, defined around the simpified surface, called *tolerance volume*. With respect to the Simplification Envelopes method, the tolerance volume is not built in a preprocessing step, but conversely it is incrementally computed during the simplification. The tolerance volume is defined as a union of balls whose radii are the current error values hold by the simplified surface vertices, smootly interpolated in between surface vertices (approximation errors are computed on the vertices). For each simplification step, approximation errors are recomputed for all vertices adjacent to the collapsed edge, the tolerance volume is updated and the method verifies if its size remains under the global tolerance value set by the user.

Evaluation
The edges are removed by collapsing edges into single vertices (vertices are thus relocated).
It adopts an approximated global error criterion, but this results in a over-estimation of the error, i.e. the bound given by the error volume may be much larger than the effective approximation error of the simplified mesh.
Times: times produced on a IBM Risc6000; no standard dataset used, experiments on: femur mesh (medical data): 180K triangles, 1:20 reduction in 25 min.; arteries mesh (medical data): 57K triangles, 1:5 reduction in 8 min.

3.2.2.9. Full-range Approximation (Ronfard and Rossignac). The Full-range Approximation method [88] adopts an edge collapse approach. Edges collapsing cost is evaluated in a preprocessing phase and stored in a priority queue. Then, it iterates on edge collapsing and relaxation (i.e. the cost of all edges adjacent to the collapsed one are re-computed and the priority queue is updated) until required compression ratio is obtained. Edge collapsing is implemented by moving the first vertex in the second one (i.e. no vertex relocation).
The approach is incremental, and is driven by a peculiar global evaluation of the approximation error introduced by each edge collapsing step. Two errors are evaluated on each candidate edge: the *local geometric error* (LGE) and the *lo-*

cal tessellation error (LTE).
LGE gives an approximated measure of the mesh variation for each edge collapse. At the initialization step, for each vertex v_i we built the set of planes P_{v_i} where the faces incident in v lie. For each candidate edge (v_1, v_2), LGE is computed as the maximum distance from the planes in P_{v_1} and its collapsed position v_2. Edges are stored in the priority queue by increasing LGE errors. For each edge collapse, the set of planes associated to the two vertices are unified and associated to the merged vertex v_2.
Conversely, LTE measures the amount of rotation undergone by the normal vectors of all the faces involved in an edge collapse. LTE is therefore used as a penalty function in order to take into account smoothness and validity of the simplified mesh: an edge collapse is operated *iff* LTE is below a selected threshold.
The heuristic proposed gives therefore an approximation of the global error, because the LGE distance is not a precise measure of the distance between the original surface and the simplified mesh.

Evaluation
Incremental approach, based on edge collapsing (vertices are not relocated).
Topology is not preserved.
It uses an approximated global error criterion, which might produce an over-estimation of the error.
Times: authors report times in the range few seconds – several minutes (no details on the dataset and the workstation used).

3.2.2.10. Mesh Simplification (Algorri and Schmitt). The Mesh Simplification method [1] adopts a decimation approach based on vertex characterization and edge collapsing. Vertices are characterized depending on a nearly-planarity test (following a user-selected planarity threshold), and then the edges connecting adjacent nearly-planar vertices are collapsed in the center of the edge.
Because no evaluation of the approximation error is provided, the simplification process may be driven only by selecting different planarity and collinearity threshold values.

Evaluation
Approach based on edge collapsing (vertices are relocated).
Topology is preserved.
The method support a local error criterion (based on planarity or collinearity of vertices). Approximation error is not bounded.
Times: roughly 0.2 KTr/sec (medical dataset, SUN SparkCenter 80MHz CPU, no precise timings).

3.2.2.11. Generalized Unstructured Decimation (Renze and Oliver). The Generalized Unstructured Decimation method [87] extends the vertex decimation strategy to *volume tessellations*, e.g. unstructured tetrahedrizations. Once a candidate vertex is selected, the 2D- or 3D-hole is retriangulated using a general unconstrained Delaunay tessellation algorithm, and the tessellation obtained is then checked for consistency.

Evaluation
The simplified mesh is defined on a subset of the original vertices.
Vertex classification and selection is not discussed, and neither is error evaluation.
Times: times measured on a SGI R4400 150MHz, 0.8K vert/sec removed on surfaces and 0.08K vert/sec on **volume data** (3D tessellations).

3.2.3. Re-Tiling Polygonal Surfaces (Turk)

The Re-Tiling Polygonal Surfaces method [103] distributes randomly a user-specified number of new vertices over the input mesh and connect them to create a re–tiling of the surface that is faithful to both the geometry and topology of the original surface. Then, an iterative relaxation procedure is applied to move each point away from all nearby points (following a *repelling* force which falls off linearly with distance). Once the points have been placed, the next task is to connect them to form a triangular mesh that reflect the topology of the original mesh. The problem is solved in steps. First, a *mutual tessellation*, i.e. containing old and new vertices, is built. Second, (part of) the old vertices are removed one at the time and the surface is re-triangulated locally. Topology consistency checks prevent old vertex removal in the case that their removal will modify the topology.
Point relaxation may be uniform on the mesh, or the density of vertices may be increased in regions of high curvature (to more accurately represent the object's features).
Main points of the method are: (1) a robust re–tiling method, (2) the use of an estimate of surface curvature to distribute more new vertices at regions of higher curvature and (3) a method for the smooth interpolation between models that represent the same mesh at different levels of detail. The latter provides smooth interpolation, with no jumps or discontinuities, but not a real "multiresolution" management because no knowledge on the precision of the obtained interpolated mesh is provided. Moreover, the approach proposed for the interpolation between any couple M^i, M^{i+1} of level of detail produces an intermediate mesh whose complexity is equal to that of M^{i+1}, the mesh at higher resolution. This approach therefore privileges interpolation smoothness rather than effective mesh simplification.

Evaluation
The method best suits models that represent curved surfaces, with no sharp edges (which may be smooted or removed).
The process of re–tiling does not take into account the error between the new and the old mesh, apart from the increased density of vertices in the areas at higher curvature (that is done following a *global* relaxation criteria). No error metric

is provided. The user has no means to set (to drive the simplification) or to be acquainted (after simplification) with the error of the simplified mesh.
The new mesh has vertices which are not a subset of the original ones.
Times: no info is reported.

3.2.4. Energy Function Optimization

These methods perform mesh reduction incrementally (by the elimination of vertices, edges, faces) and are driven by the optimization of the *energy function*, which measures the "quality" of each reduced complexity mesh.

3.2.4.1. Mesh Optimization (Hoppe, De Rose, et al.). The Mesh Optimization method [59], given a set of points V_0 and an initial triangular mesh M_0 built on V_0, produces iteratively a mesh M_j with the same topological type that fits the data well and has a smaller number of vertices than M_0. To solve the problem authors design and minimize an *energy function* that captures the competing desires of thight geometric fit and compact representation. The process is based on a non-linear optimization process, where the number, position and connectivity of the vertices vary to reduce the *energy function* value.
The *energy function* is defined over a simplicial complex M_i and its geometrical realization V_i (i.e. the M_i vertex coordinates in R^3) in terms of three components: $E(M_i,V_i) = E_{dist}(M_i,V_i) + E_{rep}(M_i,V_i) + E_{spring}(M_i,V_i)$, where E_{dist} is the sum of squared distances of the points in V_0 from the current mesh (L_2 norm), E_{rep} is proportional to the number of vertices in (M_i,V_i), and E_{spring} is the sum of the edge lenghts in (M_i,V_i).
The method does not guarantee the detection of the *energy function* global minimum, but it is able to produce good results. The optimization process is modulated by selecting adequate values for the three parameters introduced in the energy function components.
The *energy function* minimization is obtained through iteration on two nested subproblems: an inner minimization (optimization of vertex positions) and an outer minimization (evaluation of legal modifications of the mesh, e.g. edge collapse, split or swap). The first is a *continuous* optimization problem, the second a *discrete* one. The outer step has a similarity with the Mesh Decimation [97] method: in both cases the mesh is iteratively simplified, but here the simplification steps (or *legal moves*) operated on the mesh are *edge–based*.

Evaluation
An advantage of this approach is the high quality of the approximated mesh, but it is counterbalaced by the long processing times required.
The method adopts a peculiar error metric, based on the optimization of the energy function. By definition, the evaluation of the energy function takes into consideration all of the vertices of the initial mesh, and therefore the method applies a *global* error approach. On the other hand, the method is driven by a set of tuning parameters, and a measure on the effective error of the reduced mesh is not given to the user (apart from the energy function value, which is hard to be interpreted by a common user). The only parameter which can be simply interpreted is the number of vertices of the reduced mesh.
The method preserves topology. Feature elements (sharp or boundary edges, corner vertices) may be recovered by the selection of ad hoc values of the energy function parameters. The ouput mesh will contain new vertices, because new positions are selected for the vertices in the inner phase optimization.
Times: 40-50 minutes on a Dec Alpha to simplify meshes with 4K - 9K vertices (reduction factor: 1:15). For other results see our tests in Section 3.3.3.

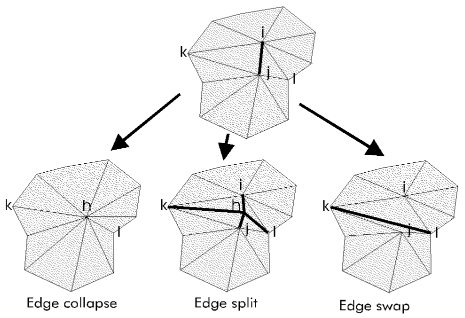


Figure 2: *Mesh Optimization: edge-based legal moves.*

3.2.4.2. Progressive Meshes (Hoppe). The Progressive Meshes method [57, 80] encompasses both a revised mesh simplification approach based on edge collapsing and energy function optimization, and a new multiresolution representation.
The adoption of a single and invertible legal move, *edge collapsing*, allows to store the whole simplification process in a multiresolution data structure (called *Progressive Meshes (PM)* representation). The PM scheme is composed of a coarse low resolution mesh M_0, togheter with the sequence of refinement records obtained by inverting the simplification steps operated to build M_k from the input mesh M_0. These refinement records allow the incremental refinement of M_k into a mesh M_i at whatever precision, with interactive times (see Figure 3).
During simplification, the selection of the edges to be collapsed is done via a priority queue, ordered by the energy cost improvement estimated for each edge collapse.
Moreover, the paper addresses many collateral problems, such as: preserving both scalar attributes defined at the mesh vertices and discontinuities (e.g. mesh color); geometric morphing between meshes; progressive transmission on low bandwith channels or networks; mesh compression; and res-

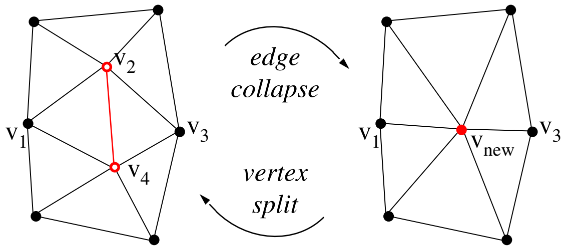


Figure 3: *The invertible edge collapse/refine action.*

olution modeling via selective refinements.

Evaluation
Analogously to the previous method, the error metric based on the optimization of the energy function gives a *global* evaluation of the mesh quality, but the effective approximation error is not produced.
Feature elements (sharp or boundary edges, corner vertices) are recovered. The ouput mesh will contain new vertices, because new positions are selected for each edge collapsing action.
The PM representation allows to confine the lenghty simplification process to a pre-processing phase, and gives the possibility to reconstruct whatever level of detail interactively.
Times: the Fandisk mesh is simplified in 19 min. on a SGI Indigo2 150MHz, i.e. 0.01 KTr/sec (times presented in the paper); the current code version has been higly optimized and simplifies, on the same dataset, 0.03 KTr/sec. For other results see our tests in Section 3.3.3.

3.2.5. Clustering Methods

3.2.5.1. Vertex Clustering (Rossignac and Borrel). The Vertex Clustering method [90] simplifies a mesh by the detection and unification of cluster of nearby vertices (detected with a simple process based on on discrete gridding and coordinates truncation). Each cluster is replaced by a unique vertex, whose geometrical position may be the center of mass of the cluster or the vertex with maximal weight in the cluster (vertex weights are defined to emphasize the relative perceptual importance). Then, an *elimination* process updates the mesh topology by removing duplicate or degenerate edges and triangles which were formerly incident in the clusterized vertices, and now in the same unique cluster vertex.

The algorithm has been implemented and included in the IBM 3D Interaction Accelerator
(see the web at http://www.research.ibm.com/3dix/ and http://www.research.ibm.com/3dix/ description03a.html).

Simplified meshes accuracy has been improved in [72] by adopting enhanced vertex weighting and floating cell clustering (i.e. space decomposition is not uniform but driven by high-weight vertices locations), and by rendering with thick lines all of the faces which collapsed into edges.

Evaluation
The error introduced is bounded (in the Hausdorf distance sense) by the user-controlled accuracy, which drives cluster size selection. The method is of simple implementation and fast, but it is not concerned with preserving the topology or geometric accuracy of the original mesh. It may produce very crude approximations.
Times: no time is reported.

3.2.5.2. Perceptually Driven Clustering (Reddy). The Perceptually Driven Poligon Reduction method [86] adopts an approach similar to that of Rossignac and Borrel (clusterization of edges based on discrete gridding) to merge pairs of edges into a single entity. The main objective of this proposal is to enforce the clusterization and simplification process with a finer control of the perceptive quality of the simplified meshes, i.e. to define simplified meshes which, under some conditions, could be undistinguishable from the original one (e.g. viewed with a given zoom–out factor). Edge merge is therefore driven by a surface curvature test weighted by edge lenght.

Evaluation
The error introduced is bounded by the cluster size selection. The method is enough simple to implement, but it is not concerned with preserving the topology or geomeric accuracy of the original mesh.
Times: time are reported only relative to a very simple object, and are in the range 0.1-0.05 KTr/sec on an SGI Indigo2 Reality Station.

3.2.5.3. Simplification using Quadric Error Metrics (Garland and Heckbert). The Surface Simplification using Quadric Error Metrics method [40] is based on the iterative contraction of edges pairs (a generalization of edge contraction), which allows also to join unconnected regions of the model. The atomic action, *vertex pairs contraction*, may be conceived as a less general vertex clustering action. As simplification proceeds, a geometric error approximation is maintained at each vertex of the current model. The error approximation is represented using quadric matrices, extending a previous aproach [88]. For each vertex v, the error is the sum of squared distances to the planes incident in v (or associated to v after some contractions which have generated v). Instead of explicitly storing plane sets (as in [88]), authors represent the error sum by an error quadric matrix. After each contraction action, the planes associated to the contracted vertices are merged by simply adding the two matrices. This imply both efficient storing (a 4x4 matrix for each vertex) and efficient evaluation of error by using simple vector–matrix operations.
The algorithm is incremental, because all the valid pairs are

selected in a preprocessing phase and inserted in a heap sorted upon minimum cost; quadric error matrices are also computed in this phase. Then, a valid pair is iteratively extracted from the heap, contracted and the heap is updated by re-computing the cost of all adjacent pairs.
The main advantages of this approach are computational efficiency and generality. It allows the simplification of disconnected or non-manifold meshes. If maintaining topology is not a must, then this solution permits drastic simplification of models via easy *aggregation* of disconnected components or by closing topological holes.

Evaluation
Topology is not preserved, but non-manifold and disconnected objects are managed.
It is an incremental method, extremely efficient both in terms of simplification times and quality of the mesh produced (the improvement against other clustering approaches is great).
Times: few seconds needed to simplify the *bunny* mesh, from 69K triangles to a 100-faces model on a SGI Indigo2 R10000 195MHz.

3.2.6. Wavelet-based Methods

The wavelet decomposition approach seems very promising for surface simplification (and, moreover, multiresolution cames for free). A difficulty in adopting wavelet decomposition is the need to provide a regular, hierarchical decomposition. A wavelet approach has been proposed to manage either regularly gridded meshes [42] or more generical meshes [36, 8].

3.2.6.1. Multiresolution Analysis via Wavelet (Eck, De Rose et al.). The Multiresolution Analysis method [70, 36, 71] is based on the wavelet mutiresolution approach, i.e. on the use of a simple base mesh plus a sequence of local correction terms, called wavelet coefficients, which capture the detail present in the object at various resolutions. It provides a solution for the so-called *remeshing problem*, i.e. given an arbitrary input mesh M build a simplified mesh K^j which is guaranteed to be within a prescribed tolerance ε_1 and which satisfy the subdivision connectivity restriction (needed for wavelet decomposition).

This approach deals with meshes of arbitrary topological type, and it provides: bounded error, a compact multiresolution representation, and mesh editing at multiple resolution scales.

The approach is composed of three main steps:

1. *Partitioning:* the imput mesh M is divided in a (small) number of triangular regions $T_1, \ldots, T_n$, and the associated low resolution triangulation built on it is called the *base mesh* K^0; a Voronoi diagram and Delaunay triangulation approach is adopted;
2. *Parametrization:* for each region T_i, a local parametrization is built on the corresponding face of the base mesh K^0;
3. *Resampling:* j recursive quaternary subdivision of the base mesh faces are performed to build mesh K^j, and the coordinates of the new mesh K^j are obtained by mapping K^j vertices in R^3 using the parametrization built in the previous phase.

An extension of this approach has been proposed to manage the approximation of both geometry and surface color [8].

Evaluation
The wavelet surface reconstruction is not simple, because the surface has to be remeshed to support a regular subdivision connectivity. This may introduce error in the higher levels of detail and topology must remain fixed at all levels of detail.
The representation is lossy (i.e. even at the higher level of decomposition, the representation is approximated under error ε_1). Moreover, the number of faces of model K^j is in general twice the number of faces of the original mesh (and K^j is an approximation of the latter).
Topology is preserved, but the topology of the model must remain fixed at all levels of detail. Preserving sharp corners or edges is not easy.
Regarding easy of implementation, a sophisticated math knowledge is required.
Times: efficiency is low both in the construction of the multiresolution representation (remeshing) and in the reconstruction by resampling of an approximated meshes K_j, respectively done at a rate of 0.04 KTr/sec and 0.03 KTr/sec on an SGI Onyx ws.

3.2.7. Simplification via Intermediate Hierarchical Representation

3.2.7.1. Octree-based Simplification (Andujar et al.). The Automatic Generation of Multiresolution Boundary Representations method [2] produces a mutiresolution boundary representation by converting the input mesh into an intermediate MDCO octree representation, and then converting back the octree in boundary representation. The hierarchical structure of the octree allows easy reconstruction of models $P_1, \ldots, P_n$ at different level of details, where representation P_k is built by pruning octree's node at level greater than k.
Given an octree purged at level k, two approaches are proposed to reconstruct a simplified boundary representation. The first builds a *face octree* from the leaf gray nodes of the purged octree, with tolerance lower or equal to the leaf nodes' diagonal lenght. The face octree is then converted in *brep*. The second approach detects and merges the planar and connected regions associated to the leaf gray nodes.

Evaluation
Approximation error depends on both the rasterization precision adopted for the construction of the octree representation

and the threshold used in the *octree–to–brep* conversion, and the composition of these two is additive. The effective control on the approximation is not very precise.
Topology is not preserved. Vertices coordinates are not a subset of the initial ones.
Times: no times are reported.

3.2.7.2. Voxel-based Simplification (He et al.). The Voxel-based Simplification method [50, 51] (called also *Controlled Topology Simplification* in the revised paper) produces a multiresolution triangle mesh hierarchy by converting the input mesh into a pyramid of raster volumes. The input objects are first 3D rasterized to built a voxel representation, and then a pyramid of reduced resolution volumes is built by the progressive elimination of the high-frequency features (i.e. adopting a standard image–processing approach). A surface-fitting technique (e.g. marching cubes [69]) is applied on the volume datasets to produce different LOD meshes.
But converting into voxel representation and then fitting an isosurface might even produce a fitted mesh which is larger in size than the original one (e.g. in the case of objects with large planar regions). To cope with this problem, authors adopt an adaptive fitting approach based on the *Splitting Box* algorithm [76] in the revised version of the paper [51].

Evaluation
The algorithm works only on closed surfaces.
Approximation error depends on both the rasterization precision adopted for the construction of the voxel representation and the threshold used in the adaptive surface fitting, and the composition of these two is additive. The effective control on the approximation is not very precise.
Topology is not preserved. Vertices coordinates are not a subset of the initial ones.
The speed of the method is low and the production of both very compact and accurate simplified models seems hard to be obtained.
Times: times measured on a SGI Onyx 2xR4400 100MHz; surface extracted in 100-200 sec. using the adaptive fitting algorithm; 3D rasterization times are not reported.

3.3. Discussion

The different approaches adopted to measure the *approximation error* introduced in the simplification process are presented and characterized in Subsection 3.3.1. The results of an empyrical comparison on the accuracy provided by different simplification code are presented in Subsection 3.3.3. The empyrical comparison was done using *Metro*, a simple tool for measuring the "disparity" of two surface meshes, introduced in Subsection 3.3.2.

3.3.1. Simplification Error Evaluation

This section presents the various techniques for evaluating and bounding the approximation error introduced in the mesh simplification process. A keen control of the approximation accuracy is critical, for example to prevent highly perceivable discrepancies between different LODs or to produce simplified and hopefully nearly indistinguishable representations of the highly complex meshes acquired via range scanners.
A definition of the *approximation error* between two meshes is given in Subsection 2.2.2 – paragraph "Error on manifold surfaces", definition (3). A L_2 norm has been adopted to measure error in some simplification approaches [57, 59]. Here, we consider error measures based on the L_∞ norm, which has been adopted more often [21, 63, 11] and can be informally stated as follows.

Definition 3.1 Given two piecewise linear objects M_i and M_j, M_i and M_j are ε*-approximations* of each other **iff** every point on M_i is within a distance ε of some point of M_j and every point on M_j is within a distance ε of some point of M_i.

The approximation error is managed in many different manners by the various simplification approaches. A characterization may be based on the policy chosen to bound the approximation error:

1. approaches which support **locally bounded** errors, i.e. the approximation accuracy is known around each surface entity (e.g. most of the mesh decimation methods [97, 99, 3, 11, 63, 88]);
2. approaches which only support **globally bounded** approximation errors, i.e. the accuracy is known only for the entire simplified mesh (e.g. the simplification envelopes [21], superfaces [61] and clustering approaches [90], methods based on the conversion into an intermediate hierarchical representation [2, 51]);
3. approaches which control accuracy with **other criteria**, which are not compatible with Definition 1; usually, curvature is taken into account to define a global bound on the surface (e.g. geometric optimization [56], triangle removal decimation [48], mesh simplification [1]);
4. approaches which **do not evaluate** the approximation accuracy (and are generally driven by the user-required simplification rate).
   (e.g. re-tiling [103]; methods based on the evaluation of an energy function [59, 57] may be included in this class, if we do not consider the energy function as a valid measure of the approximation error, as defined in Def.1);

Methods of class (1), *locally bounded*, are generally iterative methods based on a sequence of local updates to the mesh geometry/topology. For each iteration, the current mesh M_i is slightly modified to produce mesh M_{i+1}. Modifications are limited to the two patches T_i and T_i', which (a) surround the decimated/collapsed/flipped element e_i, and (b) share the border. In this case, different methods have been proposed to evaluate, at each step, the variation in the local error bounds:

- **local evaluation**; we evaluate only the approximation introduced by replacing patch T_i with T_i'; either:

– using a fast *approximated* approach, e.g. measuring the distance of the decimated vertex from the average plane to patch T_i [97] (Figure 4.a), **or**
– using a *precise* approach, which is based on the observation that the mutual projection of the two patches T_i and T_i' segments the associated hole into pieces within which both geometries vary linearly [3]. Thus, it suffices to compute errors at the intersections of the projected edges; the maximal of these errors gives an upper bound of the local error (see Figure 4.b);

- **propagation-based** evaluation; at each step, we assign to each new face/vertex in T_i' the sum of the current *local evaluation* of the approximation error and the maximal error associated with the faces/vertices in T_i [3, 11];
- **global evaluation**; we directly estimate the approximation introduced by representing with the simplified patch T_i' the corresponding section of the initial mesh M_0. Many approaches have been proposed; they can be divided into two classes:

– *approximate* approaches;

  ◦ if all removed vertices are stored with the current simplified face f onto which they "project" (i.e. which are at the shortest distance from f), a global error approximation is the maximal distance from these vertices to f [99, 11] (see Figure 4.c). This criterion is efficient, but returns an underestimation because the L_∞ *mesh–to–mesh* distance might not be located on one of the initial mesh vertices; moreover, the criterion is very imprecise in the first simplification steps when few, or even none, of the removed vertices are associated with each simplified face;
  ◦ another approach has been proposed for methods based on edge collapsing [88]. At initialization time, for each vertex we store the list of planes where the faces incident in the vertex lie. Planes lists have to be maintained and updated during simplification (after each edge collapse action, the two lists associated with the extremes of the collapsed edge are merged). The error is then evaluated at each step as the maximum distance between the new collapsed vertex position and all the planes in the vertex list. The evaluation of distances is much more efficient in [40], where *quadric error matrices* are used. This approach returns an upper-bound for the approximation error, but in some cases the bound might be considerably over-estimated with respect to the actual error.

– *precise* approaches; these compute the Hausdorff distance between the original and the simplified mesh (*mesh–to–mesh* distance). This can be done either: by maintaining trace, during simplification, of all the original faces that map to simplified faces, and then computing *face–to–mesh* distances by performing, if needed, an adaptive decomposition of faces [63]; or, by extending Bajaj et al.'s local method [3], to compare patch T_i' with the corresponding section of the initial mesh M_0. In both cases, computing times are now proportional to the complexity of meshes T_i' with respect to the corresponding M_0 subsection (i.e. the more we proceed with simplification, the higher the complexity of each section of M_0 which is associated with the current patch, and the higher the processing cost for computing edge intersection and distances);

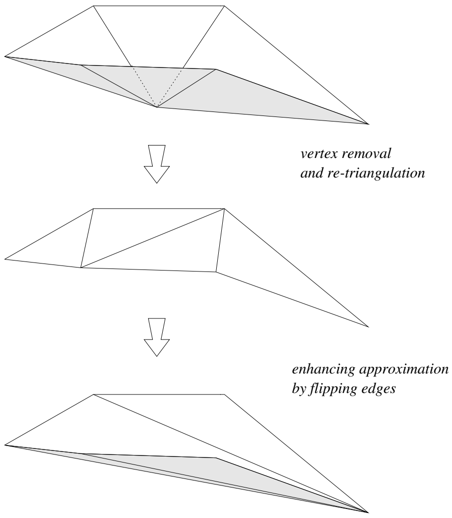


Figure 5: *The example shows how flipping may improve mesh quality.*

In the case of incremental methods, the accuracy of the simplified mesh may be improved by adopting a greedy approach based on edge flipping [11, 3], operated over the filling patches (Figure 5). But in order to effectively improve the approximation accuracy, edge flipping has to be driven by the evaluation of the *global error* variation caused by each potential flipping, and not only by a simpler equiangularity test.

Most of the methods reviewed offer no immediate provision to accurately control the perceptual effect of the degradation, because in most cases the approximation introduced into simplification has no immediate interpretation in terms of *visual degradation* [86]. Perceivable visual degradation may be caused either while visualizing a single simplified representation (e.g. in the case of eccessively approximated representation, loss of topology features, fuzziness of the simplified surface, etc.), or while changing the level of detail, the so called *inter-frame flickering* which is common if the meshes in the LOD representation present large visual differences.
Defining a measure for visual degradation is no easy task and is being hotly debated. Driving simplification by pre-

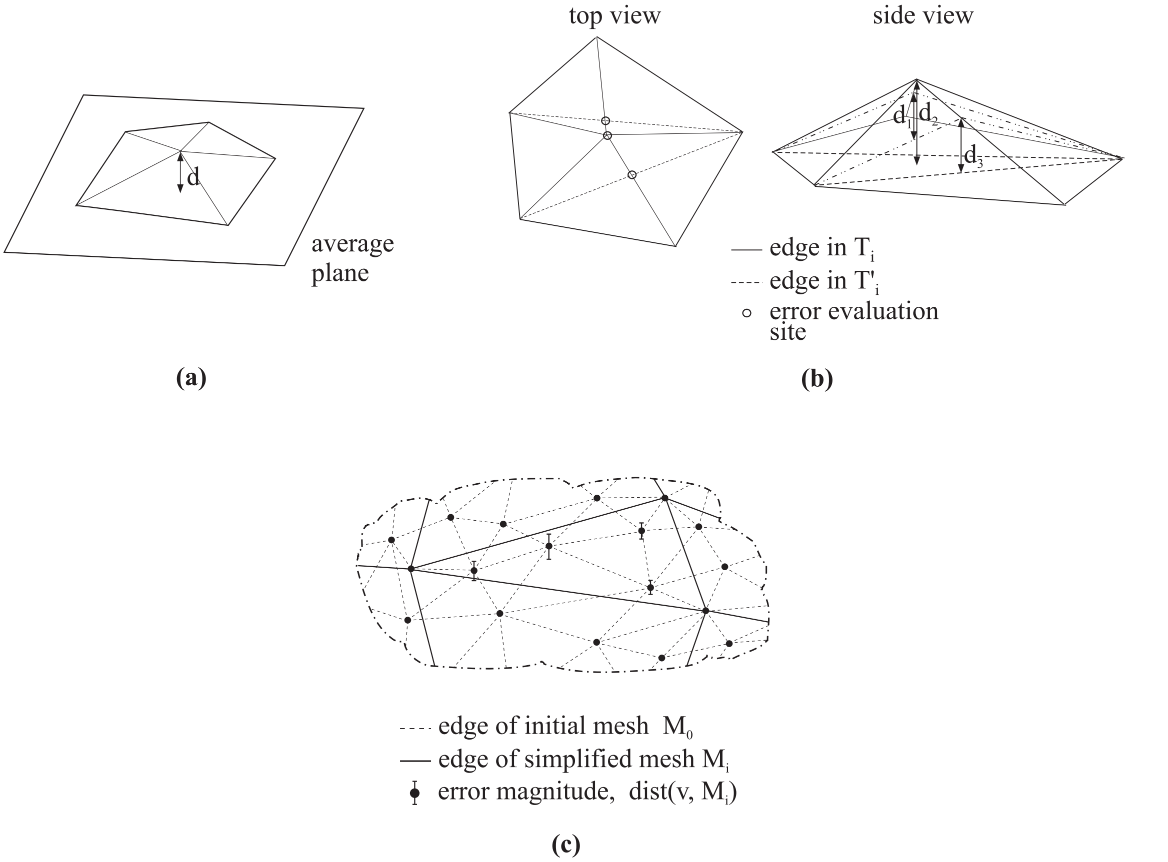


Figure 4: *Various methods to evaluate approximation error: (a) approximated local , (b) precise local, (c) approximated global.*

serving curvature and sharp edges gives good control on the appearance of the shape, one reason being that most renderers draw elementary components by shading colors according to surface normals [86]. But taking into account the shape is not enough: pictorial information (color or texture) is an important factor in perception, and therefore color discontinuities have to be managed carefully [86, 57, 8, 98].

3.3.2. The Metro tool

Due to the many approaches adopted to evaluate simplified mesh accuracy, a uniform and general tool for the evaluation of approximation precision is needed to compare the results of different simplification methods. For this reason we developed an ad-hoc tool, called *Metro*.

The first release of *Metro* was described in [20]. The current version of the tool, rel. 2.0, has been completely re-designed in order to increase precision in the evaluation of mesh accuracy, improve efficiency (it is now nearly ten times faster), and reduce memory allocation.

Metro numerically compares two triangle meshes S_1 and S_2, which describe the same surface at different levels of detail. It requires no knowledge of the simplification approach adopted to build the reduced mesh. *Metro* evaluates the difference between the two meshes on the basis of the *approximation error* previously stated in Definition 1. It adopts an approximate approach based on surface sampling and the computation of *point–to–surface* distances. The surface of the first mesh (hereafter *pivot* mesh) is sampled, and for each elementary surface parcel we compute a point–to–surface distance with the not–pivot mesh. *Point–to–surface* distances are computed efficiently by using a bucketed data structure for the representation of the non-pivot mesh.
The idea is therefore to adopt an integration process over the surface; the sampling resolution characterizes the precision of this integration (users may select the *sampling step size*). Sampling on the surface is achieved by adopting a classical incremental scan conversion approach or a Montecarlo sampling approach.
At the end of the sampling process, we switch the pivot and not–pivot mesh and execute sampling again, to get a symmetric evaluation of the error (but we observed that when a sufficiently thin sampling step is adopted, for example

0.01% of the bounding box diagonal, nearly equal values were obtained whatever mesh was chosen as the pivot).

Metro returns both *numerical* and *visual* evaluations of surface meshes "likeness" (Figure 6 shows a snapshot of its GUI). Most of the numerical results (mesh surface area, feature edges total length, mean and maximum distances between meshes, mesh volume) are reported in the tables in Section 3.3.3 . Error is also *visualized* by rendering the higher resolution mesh with a color for each vertex which is proportional to the error. A histogram reporting the error distribution is also visualized.

The error evaluated by *Metro* may be affected by finite numerical precision, although double precision is adopted in numerical computations. An "ad hoc" management has been provided for a number of dangerous cases, such as nearly coincident vertices, facets with small areas, and very elongated triangles.

3.3.3. Empirical Evaluation of Simplification Codes

Five representative (and available) simplification codes, listed below, have been tested on three datasets, which represent three main classes of data. Results have been reported in a submitted paper [18]. We briefly report in the following the comparison methodology and some discussion of the results obtained.

The test meshes used were: *bunny*, representative of range-scanned meshes (available at http://www-graphics.stanford.edu/data/); *fandisk*, representative of CAD models (available at http://research.microsoft.com/research/graphics/hoppe/); and *femur*, representative of medical surfaces fitted on volume datasets (courtesy of Istituto Ortopedico Rizzoli (IOR), Bologna, Italy).

The following simplification codes were compared, using the *Metro* tool (see Subsection 3.3.2) on a SGI Indigo2 R4400 200MHz CPU, in terms of the size of the meshes produced, the approximation quality, and the running times:

1. **Mesh Decimation** [97]; code provided in the Visualization Toolkit 1.3 (VTK) by Bill Lorensen, Ken Martin and William Schroeder (http://www.cs.rpi.edu/∼martink/);
2. **Simplification Envelopes** [21], rel. 1.2; code developed at the Department of Computer Science of the University of North Carolina, code courtesy of Jonathan Cohen et al. (http://www.cs.unc.edu/∼cohenj);
3. **Multiresolution Decimation** [11]; code Jade rel. 2.0, implemented by the Visual Computer Group of CNUCE/IEI-C.N.R. (http://miles.cnuce.cnr.it/cg/jade.html - Jade rel. 2.0 has been slightly improved in terms of approximation error management with respect to the description and results reported in [11]);
4. **Mesh Optimization** [59]; code developed by Hugues Hoppe et al., Univ. of Washington (http://research.microsoft.com/research/graphics/hoppe/);
5. **Progressive Meshes** [57]; code developed by Hugues Hoppe, Microsoft inc. (http://research.microsoft.com/research/graphics/hoppe/)

We initially also planned to test a representative of commercial tools, i.e. the Polygon Reduction Editor available under *SGI Cosmo Worlds* (available at http://www.sgi.com/Products/cosmo/worlds/CosmoWorlds_UG/Reference/polyed.htm). This simplifier seems to be based on the *clustering* approach, and presents a simple GUI which allows the user to set threshold values to delete points by curvature, edges by length and triangles by area (see Subsection 5.1.2). The simplification process is driven by a *trial and error* approach, and the quality of the mesh produced depends on the skill (and the luck) of the user. The production of results of a quality comparable to the simplified meshes produced using the codes above appeared to be nearly impossible.

The results relative to the evaluation of the approximation error are summarized in the graphs in Figure 7. In the graphs on the left we plot the *maximal error* (E_{max}) evaluated by Metro on simplified meshes of different sizes. The *average error* (E_{avg}) is reported in the graphs on the right. For all graphs, the simplified mesh size is mapped on the X axis, and the error is mapped on the Y axis.

As we expected, the best results in terms of *average error* were given by the Progressive meshes and mesh Optimization codes (which are based on an L_2 metric over the object surface, meaning that they try to minimize the root mean square error). On the other hand, methods based on the L_∞ metric produce better results when we consider the *maximal error*.
It is noticeable that Simplification Envelopes and Multiresolution Decimation produce the best results when high accuracy is needed (i.e. for reduction factors not higher than 75%).

Finally, all of the solutions tested share a common weakness: they are defined to work on a single, topologically–sound mesh. This is not the general case in rendering CAD models or in virtual reality sessions, where we may need to simplify scenes or objects composed by multiple components, with a not topological–clean composition between components. New solutions are required for these applications to provide increased generality and robustness. First attempts in this direction have been recently proposed [73, 40].

The original bunny mesh is shown in Figure 8, toghether with three simplified meshes obtained with the Multiresolution Decimation code [11] at different approximation. Top-right: 14.339 faces (≈ 20% of the original), 7.278 vertices, error 0.001% of the mesh's bounding box diagonal . Bottom-left: 6.901 faces (≈ 10%), 3.559 vertices, error 0.005%. Bottom-right: 1.361 faces (≈ 2%), 789 vertices, error 0.1%. Flat shading is used in all of the images, to enhance mesh disparity.

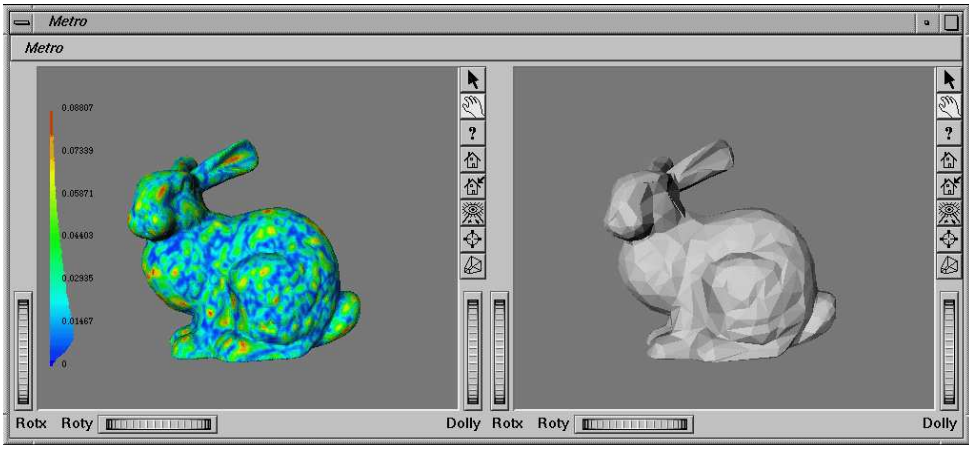


Figure 6: *The* Metro *tool.*

4. Multiresolution Meshes

The concept of *Level Of Detail (LOD)* is generically related to the possibility of using different representations of a geometric object (a surface, a volume, an image, etc.), having different levels of accuracy and complexity.

In case the object geometry is represented with a mesh, the level of detail of a single representation is directly proportional to the size of the mesh. In the simplest case, a LOD representation consists of a collection of meshes of different sizes, each representing the object at a given resolution. Each mesh is defined independently of the others, and it is tagged with some *range* of detail, which is used as a filter to select a mesh from the LOD representation, according to the needs of any given application.

A LOD representation can be built easily by the repeated application of any simplification algorithm with different parameters of simplification. Such a simple framework is a straightforward application of the mesh simplification technology, and it has been incorporated into leading edge graphics languages and packages, such as OpenInventor^{TM} [43, 106], and VRML [105]. However, a simple LOD representation has three main drawbacks:

1. since each mesh is stored independently, the number of levels of detail must necessarily be small, in order to maintain memory requirements into reasonable bounds;
2. because of the previous limitation, not only there is just a modest possibility of adapting resolution to application needs, but also changes between two consecutive levels are abrupt, hence causing undesirable "popping" effects in visualization during the transition from a level to another;
3. the resolution of each mesh is uniform over the whole object it represents, hence there is no possibility of grading the resolution of large objects (e.g., terrains, buildings, ships, planes, cars) that might span several ranges of distance from an observer within the same view.

In order to overcome such limitations, more sophisticated models have been developed in the literature, which we will call *multiresolution meshes*. Informally, a multiresolution mesh is a model that can provide a high number (virtually, a continuous range) of meshes representing a single object at different resolutions. The number of different levels of detail that such a model can provide will not be fixed a priori, but it will rather be proportional to the size of the mesh representing the object at the maximum possible resolution. The desirable properties of a multiresolution model are:

- the model must support efficient query processing, i.e., it must be possible to extract a mesh from the model at a given resolution in short (real) time;
- the size of the model must not be considerably higher than the size of the mesh at the highest resolution it can provide;
- any mesh extracted from the model must be "continuous", i.e., it is important to avoid cracks due to abrupt transition between different LODs within a single mesh;
- the transition between different meshes extracted from the model must be as "smooth" as possible, i.e., it is important to avoid abrupt changes when moving from a mesh to another at a close LOD.

For historical reasons, we will distinguish between *domain-uniform models*, i.e., multiresolution models that only support the extraction of meshes whose resolution is uniform over the whole object represented, and *domain-variable models*, i.e., models that also support the extraction of meshes whose resolution is gradually changing over different zones of the object represented. In general, we can assume that the desired level of detail/accuracy/resolution of a mesh can be specified by a *threshold function* defined either in the space embedding the object represented, or on

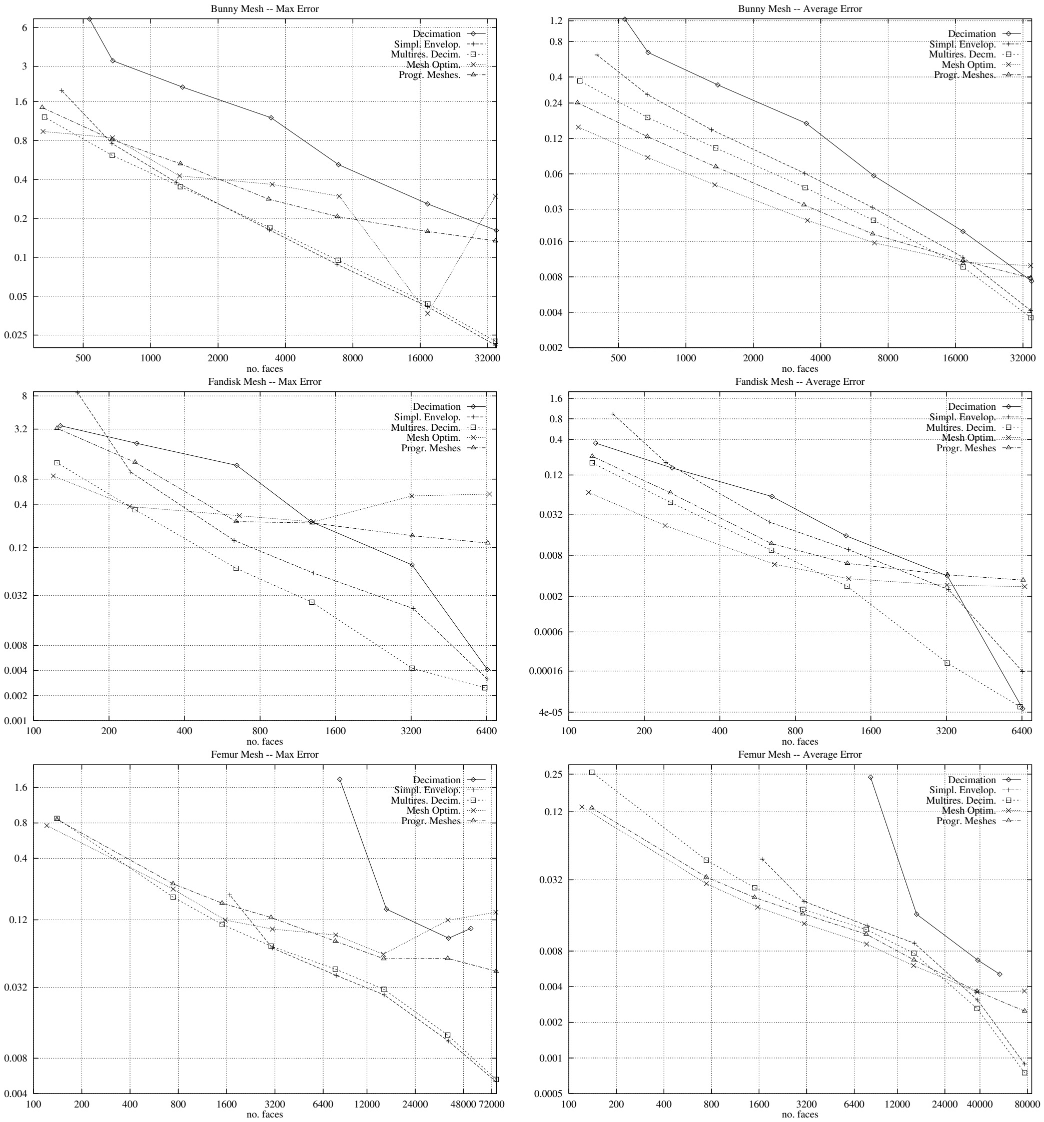


Figure 7: *The graphs show the performance of the various simplification codes on the three meshes.*

the object itself. Such a function gives, for instance, either the maximum error allowed, or the required density of the mesh at each point in space. Domain-uniform models only support threshold functions that are constant over the whole domain/object, while variable-resolution models support arbitrary threshold functions, or some special cases like distance-increasing functions. Variable-resolution models are all very recent.

Before reviewing the literature on multiresolution models, we provide a framework encompassing all models, which will help us understanding general structures and properties, as well as comparing the features of different models.

4.1. A general framework

In this section, we describe the *Multi-Triangulation (MT)*, a structure that was introduced in [82, 83], which provides a general framework for multiresolution meshes. Indeed, all multiresolution models proposed in the literature can be interpreted as special cases of the MT, hence they will be reviewed in the next section according to this framework.

The general idea underlying the MT is that any multiresolution mesh can be built through local operations that progressively modify an initial mesh through either refinement or simplification, and that local modifications can be arranged into a partial order according to their dependencies.

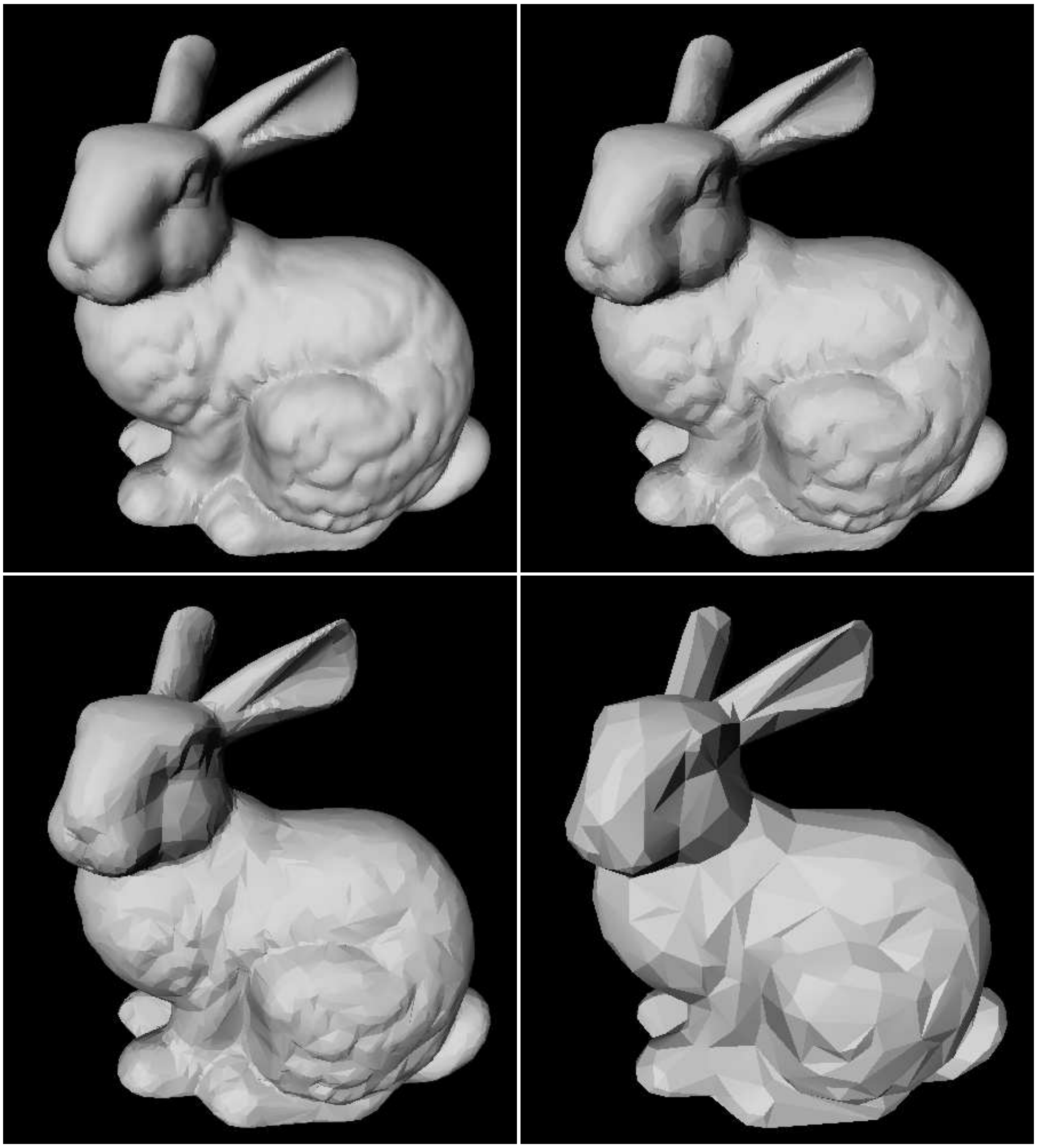


Figure 8: *The original bunny mesh (top-left, 69.451 triangles and 34.834 vertices) and three meshes simplified by the Multiresolution Decimation code. Top-right: 14.339 faces (≈ 20%), 7.278 vertices, error 0.001%. Bottom-left: 6.901 faces (≈ 10%), 3.559 vertices, error 0.005%. Bottom-right: 1.361 faces (≈ 2%), 789 vertices, error 0.1%.*

In order to have an intuition of the framework, let us consider first the simple LOD model given by the sequence of meshes depicted in Figure 9a. All meshes decompose the same rectangular domain into triangles, an each mesh can be obtained from the one preceding it (or following it) through some local modification. In other words, the sequence records the evolution of a mesh through a process of either refinement or simplification, as indicated by the arrows. Although in the example each local modification consists of insertion/deletion of one or more vertices, local modifications could be any combination of generic operations that modify the mesh locally, such as vertex split, edge collapse, triangle collapse, etc. Indeed, the framework is independent of the kind of local operations adopted.

The key observations to move from this LOD model to the concept of MT are the following:

1. the sequence of meshes can be viewed as an initial mesh, namely the mesh at coarsest resolution, plus a sequence of local modification, where each modification is specified by the portion of mesh that is modified, called a *fragment*, as in Figure 9b.
2. some fragments in the sequence are mutually independent (for instance, we can apply modification 2 even if modification 1 is not performed), while other modifications have dependencies (for instance, modification 4 cannot be applied, unless modification 1 occurs first).

The second observation suggests that fragments can be arranged into a partial order, which can be represented by a rooted DAG, having its root at the initial mesh, and one arc between each pair of nodes such that the start node of the arc has some triangle "covered" by the fragment corresponding to the end node (see Figure 12).

In the following, we will give more formal definitions and results on the Multi-Triangulation, by abstracting on its construction technique, while relying exclusively on the concept of local modification.

4.1.1. The Multi-Triangulation

For the sake of simplicity, the MT is described here for decomposing planar domains, while its extension to generic surface meshes, and to higher dimensions is straightforward (see [35, 27] for details on such extensions). Formal proofs of the properties reported in the following can be found in [83].

Given two triangulations T_i and T_j, we define their *interference* $\otimes$, and their *combination* $\oplus$, as follows:

$$T_i \otimes T_j \;=\; \{t \in T_i \mid \exists t' \in T_j,\ int(t) \cap t' \neq \emptyset\}, \qquad (1)$$
$$T_i \oplus T_j \;=\; T_i - (T_i \otimes T_j) \cup T_j, \qquad (2)$$

i.e.: interference is the set of triangles in the first triangulation that overlap with the second triangulation; combination is the mesh obtained by joining triangles of the second mesh, with those of the first mesh that do not interfere with them, as if the second mesh were "pasted" on top of the first one. Note that the combination is neither commutative nor associative.

If $T_i \otimes T_j \neq \emptyset$ then we say T_i and T_j are *interfering*, otherwise we say they are *independent*. If $T_i \oplus T_j$ is also a triangulation, and $\Delta(T_i \oplus T_j) = \Delta(T_i) \cup \Delta(T_j)$, then T_j is said *compatible over* T_i, and $T_i \oplus T_j$ is said a *modification* of T_i. Examples of interference and combination, which also illustrate compatibility, are given in Figure 10.

Given a sequence of triangulations $T_0, \ldots, T_k$ we define its *combination* as the successive combination of its elements $\oplus_{i=0}^{k} T_i = T_0 \oplus T_1 \oplus \ldots \oplus T_k$. We say that $T_0, \ldots, T_k$ is a *compatible sequence* if $\forall j = 1, \ldots, k$, T_j is compatible over $\oplus_{i=0}^{j-1} T_i$.

Definition 4.1 (Multi-Triangulation) Let Ω be a (compact) polygonal domain in $\mathbb{R}^2$. A *multi-triangulation (MT)* on Ω is a poset $(\mathcal{T}, \leq)$ where $\mathcal{T} = \{T_0, \ldots, T_h\}$ is a set of triangulations, and $\leq$ is a partial order on $\mathcal{T}$ satisfying the following axioms:

1. $\Delta(T_0) \equiv \Omega$, and $\forall i = 1, \ldots, h$, $\Delta(T_i) \subseteq \Omega$;
2. $\forall i, j = 0, \ldots, h,\ i \neq j$,
   a. $T_i \prec T_j \Rightarrow T_i \otimes T_j \neq \emptyset$;
   b. $T_i \otimes T_j \neq \emptyset \Rightarrow T_i$ is in relation with T_j (i.e., either $T_i < T_j$ or $T_j < T_i$);
3. the sequence $T_0, \ldots, T_h$ of all elements of $\mathcal{T}$ defines a consistent total order of $\mathcal{T}$ and it is a compatible sequence.

The elements of $\mathcal{T}$ are called *fragments*. The 2-set $T_{\mathcal{T}} = \cup_{i=0}^{h} T_i$, i.e., the set of all triangles of the multi-triangulation, is called the *associated 2-set* of $\mathcal{T}$. Given any subset $\mathcal{T}' \subset \mathcal{T}$, the total order of its elements consistent with $T_0, \ldots, T_h$ will be called the *default order*.

Fragments of an MT will be used to build different triangulations of Ω through combination. The order relation will be used as a sort of dependence between triangles, meaning that if $T_i < T_j$, then T_j cannot be used to build a triangulation, unless also T_i is used. Note that precedence does not necessarily imply interference: if $T_i < T_j$, we can have $T_i \prec \ldots T_k \prec \ldots T_j$ with $T_i \otimes T_k \neq \emptyset$, $T_k \otimes T_j \neq \emptyset$, while $T_i \otimes T_j = \emptyset$ (see Figure 11). Hereafter, an MT will be denoted simply by its set of fragments $\mathcal{T}$, while the order $\leq$ will be omitted, whenever no ambiguity arises.

An MT can be represented by a DAG, having one node per fragment of the MT, and such that there is an arc in the DAG from T_i to T_j if and only if $i < j$, and there is some triangle of T_i that belongs to $\oplus_{k=0}^{j-1} T_k$, while it does not belong to $\oplus_{k=0}^{j} T_k$ (i.e., it is "obscured" by triangles of T_j). We can also label arc (T_i, T_j) with the set of triangles of T_i that are obscured by T_j. Figure 12a shows the DAG describing the MT corresponding to the LOD model depicted in Figure 9. It is easy to see that T_0 is a least element of $\mathcal{T}$, hence the corresponding DAG is rooted at T_0.

An MT has always the following property, which allows

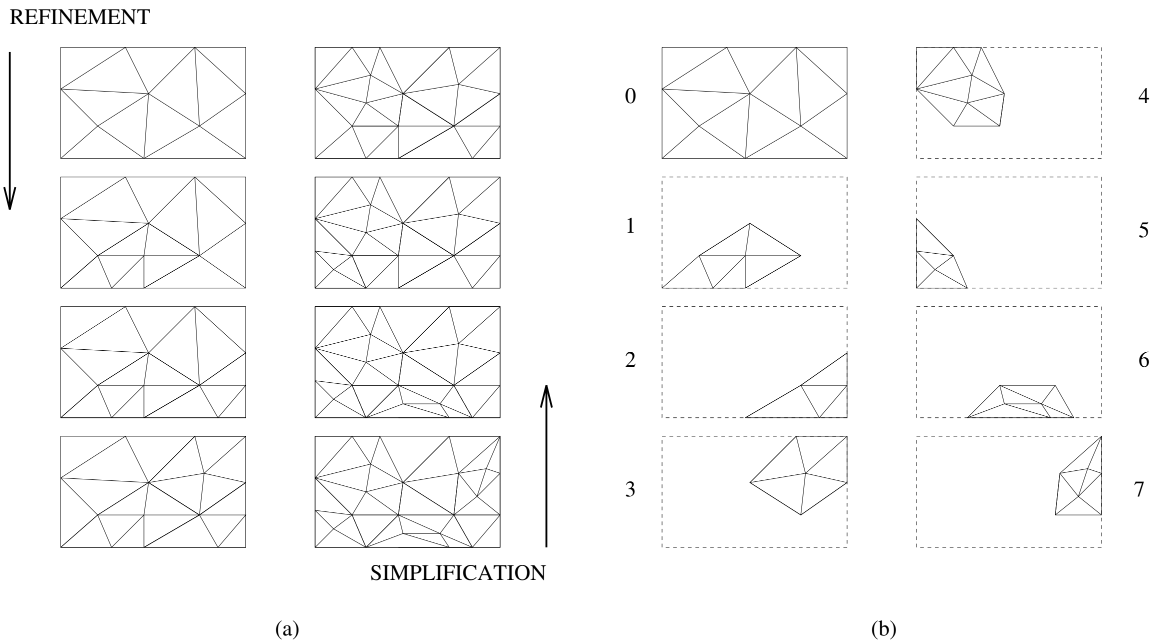


Figure 9: *A simple LOD model (a), and its interpretation as a sequence of local modifications (b).*

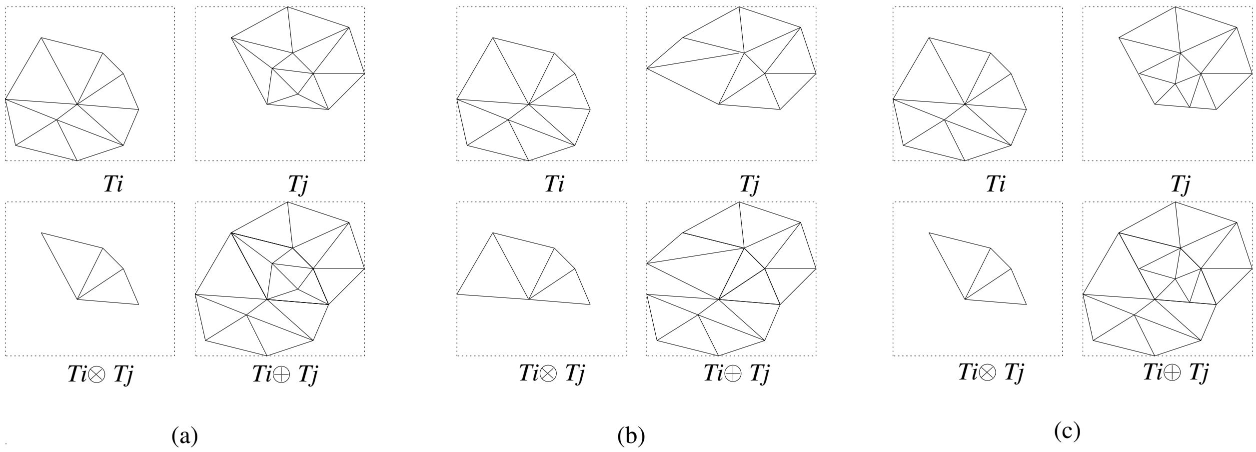


Figure 10: *Examples of interference and combination (each fragment is included in a dotted reference frame): T_j is compatible over T_i (a); T_j is not compatible over T_i because the domain of the result does not cover the union of the two domains (b); T_j is not compatible over T_i because the result is not a triangulation (c).*

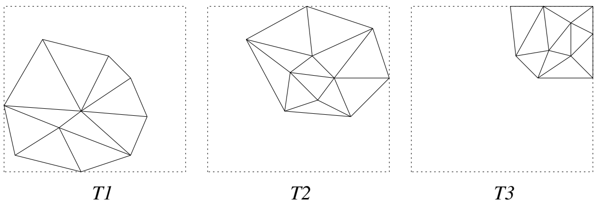


Figure 11: *Precedence does not imply interference: $T_1 \prec T_2 \prec T_3$, hence $T_1 < T_3$, while $T_1 \otimes T_3 = \emptyset$.*

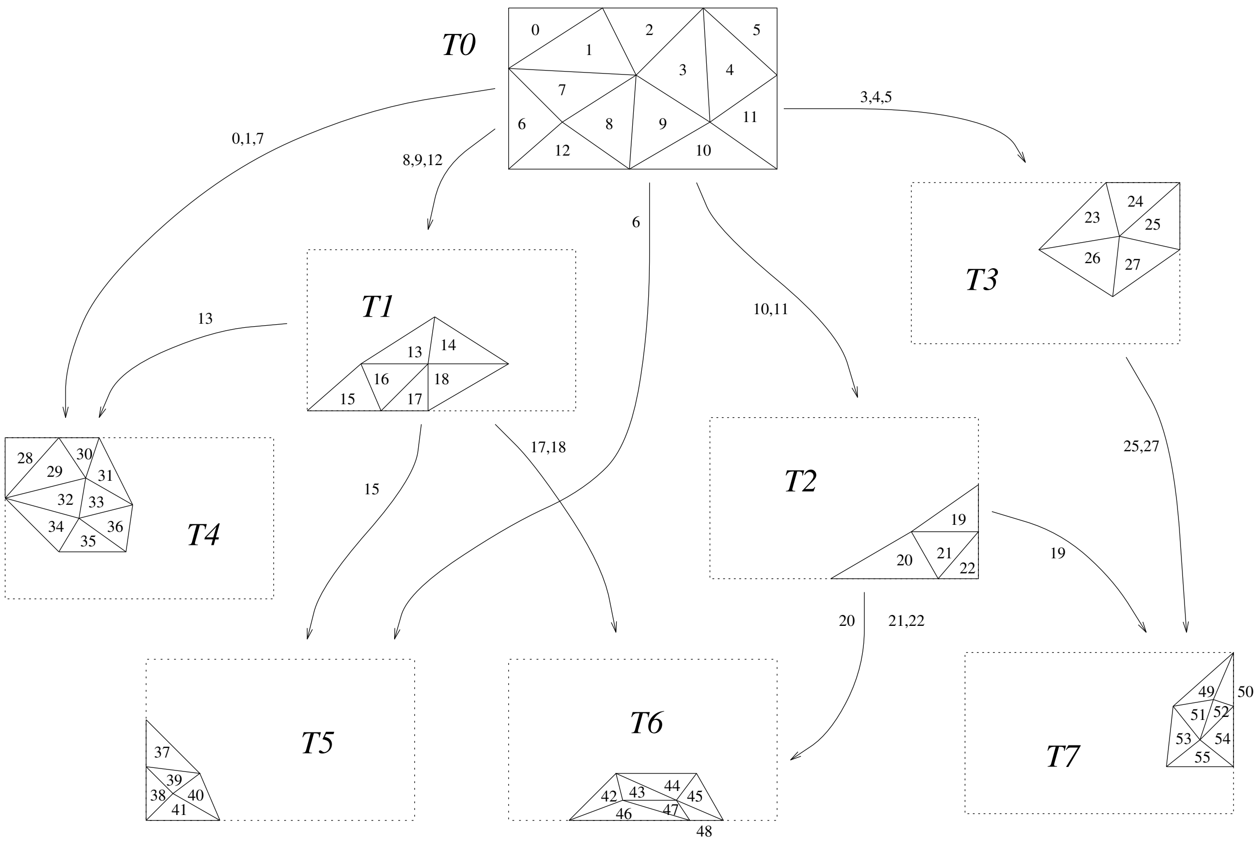


Figure 12: *The DAG describing the MT correspondig to the LOD sequence in Figure 9.*

us to relate meshes that can be extracted from it to its lower sets:

> *Given a lower set $\mathcal{T}'$, any consistent order of its elements form a compatible sequence, and the combination of any such sequence always gives the same mesh; in particular, we can always obtain such a mesh by adopting the default order.*

Henceforth, the combination obtained from any consistent order of a lower set $\mathcal{T}'$ will be called the combination of $\mathcal{T}'$, and it will be denoted $\oplus\mathcal{T}'$. The combination $\oplus\mathcal{T}$ of the whole set $\mathcal{T}$ will be called the *top* of $\mathcal{T}$. In Figure 13, four among the 25 possible meshes that can be extracted from the MT depicted in Figure 12 are shown.

We already remarked that a lower set is always related to a cut in the DAG identifying it. We can further obtain a nice relation between the mesh obtained by a lower set in an MT, and the set of triangles labeling the arcs in the corresponding cut. To this purpose, we extend the DAG to a lattice, by adding a node corresponding to the top T of $\mathcal{T}$, and an arc (T_i, T) from each fragment that has some triangle belonging to the top, suitably labeled with such triangles. With such a representation, we have the following:

> *The combination of a lower set produces a mesh composed of the collection of triangles labeling the corresponding cut in the lattice.*

Figure 14 shows the lattice representation of the MT of Figure 12 with a cut, corresponding to the first mesh depicted in Figure 13.

The number of different meshes that can be extracted by cutting an MT will be said the *expressive power* of $\mathcal{T}$. Due to the combinatorial nature of lower sets (or cuts), we can expect that the expressive power is a fairly large number, much higher than the number h of meshes in the initial LOD model, provided that fragments are sufficiently small, and not too much dependent from each other. However, so far we have only seen that each cut in the MT corresponds to a mesh. We wonder next whether *any* possible mesh that can be obtained by assembling triangles of the MT (not necessarily through combination!) does correspond to a cut or not. If the affirmative case, we obtain a sort of maximality on the expressive power of an MT. In fact, maximality is verified for an MT under some suitable conditions, which are very reasonable, and verified by all multiresolution meshes known in the literature. We investigate next such properties.

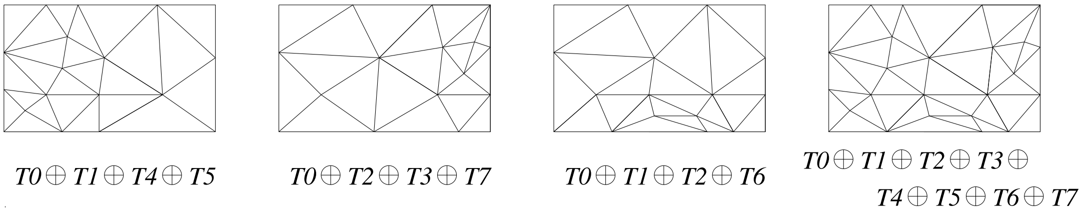


Figure 13: *Four out of the possible 25 coverings that can be extracted from the MT of Figure 12, together with the combinations generating each of them.*

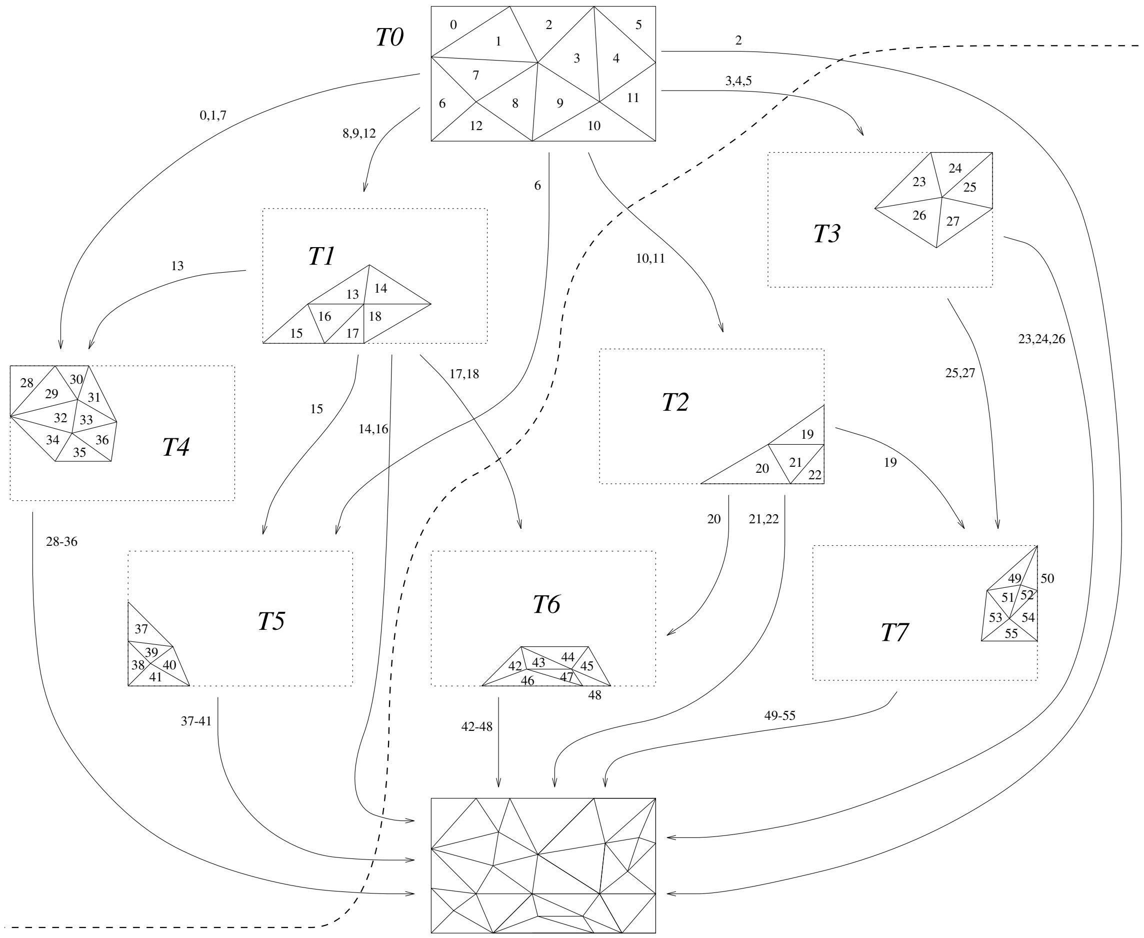


Figure 14: *The lattice representation of MT of Figure 12, with a cut defining a lower set that generates the first mesh of Figure 13.*

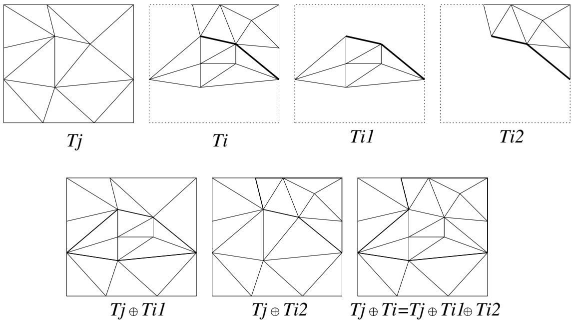


Figure 15: *T_i is not minimally compatible over T_j because each of the two subsets T_{i_1} and T_{i_2} of T_i is compatible over T_j. Triangulations $T_j \oplus T_{i_1}$ and $T_j \oplus T_{i_2}$ cannot be obtained from T_j and T_i, while $T_j \oplus T_i$ is obtained as $T_j \oplus T_{i_1} \oplus T_{i_2}$.*

We call *support* of a fragment T_i of $\mathcal{T}$ the combination of its sub-closure (recall that the sub-closure of an element is its down-closure minus the element itself): in practice, the support is the smallest triangulation (in terms of the lower set defining it) over which fragment T_i can be combined. The set of triangles of the support that are interfering with T_i, i.e., those labeling its incoming arcs in the lattice, is called the *floor* of T_i: in practice, the floor is composed by those triangles that are obscured by T_i once it is combined with its support.

Definition 4.2 (Canonical form) If a fragment T_i is compatible over another fragment T_j, and no subset $T_i' \subset T_i$ is compatible over T_j, then T_i is said *minimally compatible* over T_j. A multi-triangulation $\mathcal{T}$ is in *canonical form* if every fragment T_i of $\mathcal{T}$ is minimally compatible over its support.

In practice, if a triangulation is in canonical form, there is no possibility to combine only a portion of a fragment with the fragments preceding it. This fact has relevance to the expressive power, as shown in Figure 15. However, any MT can be transformed into another MT in canonical form having the same associated 2-set: this is done by breaking each fragment into pieces, each of which is minimally compatible over its support. Therefore, we will always assume that an MT is in canonical form.

Definition 4.3 (Non-redundancy) A multi-triangulation $\mathcal{T}$ is *non-redundant* if

1. there are no duplicate triangles, i.e., each triangle of the associated 2-set belongs to exactly one fragment;
2. $\forall i, j = 0, \ldots, h$, if e is an edge common to T_i and T_j, and $T_j < T_i$, then e is an edge of the floor of T_i.

The meaning of non-redundancy is the following. Triangles, which are the atoms of the structure, are not replicated in different fragments to warrant that the size of the MT is the same as the size of its associated 2-set. Edges can be replicated, since they provide an interface for combining triangulations. However, if different triangulations share a common edge, they must form a sequence in the DAG encoding the poset. If an MT is redundant, there can exist meshes that cannot be obtained through combination (see Figure 16). On the other hand, if an MT is in canonical form and non-redundant, then its expressive power is maximum, as stated in the following theorem.

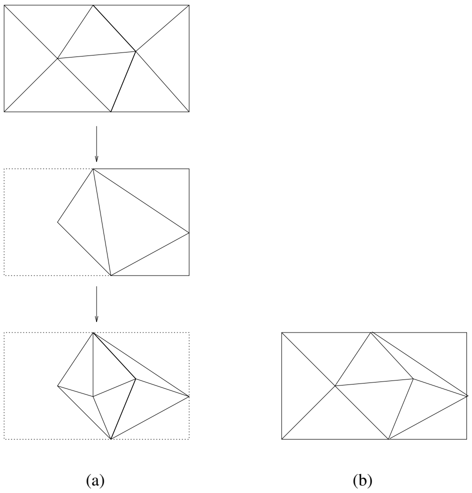


Figure 16: *A redundant MT (a): the triangulation in (b) is made from some of its triangles, but it cannot be obtained through combination.*

Theorem 4.4 Let $\mathcal{T}$ be a non-redundant multi-triangulation in canonical form. Then for any triangulation T generated by its associated 2-set $T_{\mathcal{T}}$ there exists a lower set $\mathcal{T}' \subseteq \mathcal{T}$ such that $T = \oplus \mathcal{T}'$.

Although properties given so far hold for generic MTs, in practical cases we will be interested in structures that allow us to control the resolution of a mesh by suitable cuts. To this purpose, we will focus on monotone MTs, defined as follows.

Definition 4.5 (Monotone MT) A multi-triangulation $\mathcal{T}$ is *increasing* if and only if for every pair of lower sets $\mathcal{T}'$ and $\mathcal{T}''$, $(\mathcal{T}' \subset \mathcal{T}'') \Rightarrow |\oplus \mathcal{T}'| < |\oplus \mathcal{T}''|$. Similarly, a multi-triangulation $\mathcal{T}$ is *decreasing* if and only if for every pair of lower sets $\mathcal{T}'$ and $\mathcal{T}''$, $(\mathcal{T}' \subset \mathcal{T}'') \Rightarrow |\oplus \mathcal{T}'| > |\oplus \mathcal{T}''|$. A multi-triangulation which is either increasing or decreasing is said *monotone*.

It is straightforward to verify that a multi-triangulation is increasing [decreasing] if and only if the size of each fragment is larger [smaller] than the size of its floor.

A further interesting property of MTs is that also the collection of all floors, plus the top, is an MT, having the same associated 2-set, hence generating the same set of triangulations. It is also true that an MT is increasing [decreasing] if and only if its reverse is decreasing [increasing]. Since the reverse of an MT generates the same triangulations, and has an opposed monotonicity, we can always consider increasing MTs : a decreasing MT can be dealt with by working on its reverse. In fact, all multiresolution meshes that we will analyse in the following can be interpreted as increasing MTs, even though some of them are constructed as decreasing structures, and then reversed.

We finally give some definitions that are useful to evaluate the complexity of algorithms that traverse an MT.

Definition 4.6 (Linear growth) An increasing multi-triangulation $\mathcal{T}$ has *linear growth* if for each lower set $\mathcal{T}' \subseteq \mathcal{T}$ the ratio between the size of $\mathcal{T}'$ and the size of its combination is bounded by a constant. A decreasing multi-triangulation has linear growth if and only if its reverse has linear growth.
(Bounded width) A multi-triangulation $\mathcal{T}$ has *bounded width* if the number of triangles labeling arcs outgoing from a fragment is bounded by a constant.
(Logarithmic height) A multi-triangulation $\mathcal{T}$ has *logarithmic height* if the maximum length of a path from the source to the drain is logarithmic in the number of arcs of $\mathcal{T}$.

We will see in the examples that linear growth, bounded width, and logarithmic height are desirable properties since they permit to achieve optimal time complexity in visiting the structure. In particular, linear growth is fundamental to achieve optimal time complexity (i.e., linear in the output size) for algorithms extracting a mesh at a given resolution from an MT, while bounded width and logarithmic height are important for other traversal operations, such as point location (see [82, 27] for details).

4.1.2. Data structures

In this section, we classify information that describe an MT, which can be stored in a data structure. As we will see in the next section, the different kinds of information stored in the various data structures represent, together with the special cases of MT implemented, important characteristics of the various multiresolution meshes. Relevant information about a multi-triangulation include:

- *Geometric information*: all geometric elements forming an MT are simplices (vertices, edges, triangles), and the geometry of a simplex is completely defined by the set of its vertices, which also uniquely determines the boundary of the simplex; therefore, it is sufficient to store geometric information (coordinates) only for the vertices of an MT.
- *Structural information* about the simplexes composing the fragments in the model: all simplexes are defined combinatorially in terms of tuples of vertices; since a mesh is a regular complex, its structure is completely defined by the set of its maximal simplexes (triangles), which can be encoded either explicitly, or through some structural rule that implicitly provides the structure of each fragment.
- *Topological information*, which encode adjacency, boundary and coboundary relations among the simplexes of an MT. We distinguish between *local topology*, which is concerned with topological relations between simplexes within the same fragment, and *global topology*, which considers relations between simplexes not necessarily belonging to the same component (global topology includes local topology). Most important topological links are adjacencies between maximal simplexes (triangles).
- *Spatial interference information*, which relate fragments and simplexes that have spatial interference, according to the definitions given in the previous section. Interference information can be encoded either explicitly or implicitly, either between pairs of interfering components, or between all pairs of interfering simplexes (triangles) belonging to such components.
- *Additional information*, which might be needed to relate an MT to a specific application. For instance, the accuracy (error) of each fragment or simplex in the model might be stored. This kind of information is often crucial to the complexity of a data structure, and to the efficiency of algorithms manipulating it.

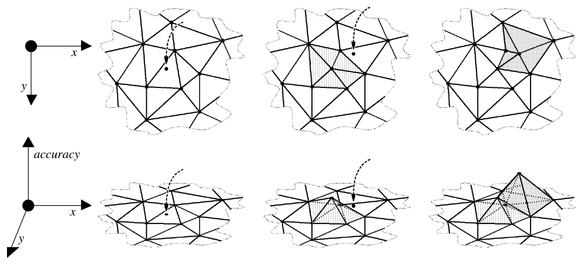


Figure 17: *The three-dimensional interpretation of local modifications in a hypertriangulation.*

In order to better understand the nature and complexity of topological, and interference information, we describe some further structures that give alternative views of the MT. In [19], an inceasing MT is interpreted as a simplicial complex embedded in 3D space, called a *hypertriangulation*, by assigning a third coordinate to each vertex of the MT, corresponding to a “resolution” axis (i.e., the more refined the fragment, the higher the third coordinate of its vertices). With such a structure, each fragment can be seen as a “dome” resting on its floor along its boundary (see Figure 17). It is readily appearent from such a structure the importance of triangle adjacencies through edges, which permit to weld triangles of different fragments to form meshes. Each edge in the structure may have different triangles incident at it, subdivided in two groups, namely to its left and right side, respectively; triangles of each group are in order of increasing refinement (see Figure 18).

Interference links between components are directly represented by the arcs in the DAG (or lattice) representation; interference links between triangles can be obtained by combining information provided by the arcs, and by the triangles

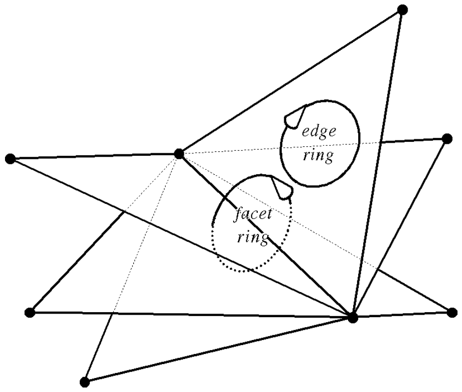


Figure 18: *Triangles incident at an edge are subdivided in two groups, each sorted by increasing resolution.*

labeling them. In [35], a graph representing interference between triangles is obtained as an "explosion" of the DAG of fragments, obtained by replacing every node T_i with as many nodes $t_1 \dots t_p$ as the triangles in T_i, and by transferring to each t_j $(j = 1, \dots, p)$ the arcs of T_i which correspond to interferences involving t_j. In practice, in such a structure the concept of fragment is lost, while each triangle is explicitly related to the triangles that either obscure it, or are obscured by it in the MT.

4.2. Review of multiresolution meshes

Multiresolution meshes can be subdivided in two main classes, according to their structure, that we will conventionally call *tree-like models*, and *historical models*, respectively:

- *tree-like models* are based on nested subdivisions, according to the basic idea that resolution can be refined by recursively subdividing a region (a triangle, a patch) into a set of smaller regions covering it exactly; the hierarchy of nested regions can be described by a tree; in order to give an interpretation of such models as MTs, it is necessary to perform a suitable translation of their tree structure into a DAG;
- *historical models* rely on either refinement or simplification techniques for their construction, and they essentially store the evolution of a mesh throughout the refinement/simplification process; being based more closely on the concept of local modification, such models are all straightforward specialization of the MT.

In the following, we review most relevant models that appeared in the literature in the two classes. The order we follow in the review is meant to improve comprehension, rather than being strictly historical.

4.2.1. Tree-like models

Tree-like models have been studied for a long time, and they have been adopted in many applications, and in several fields. Intuitively, a tree-like model encodes a nested subdivision of an initial mesh, assumed as the coarsest representation of the modeled object, through a recursive refinement process: at each step of refinement, a simplex of the mesh is refined independently into a local mesh, through a decomposition rule that is characteristic of each specific model. The hierarchy of meshes is represented as a tree where each node is a local mesh, and each arc corresponds to a refinement operation.

In general, a tree-like model cannot be interpreted directly as an MT whose components coincide with the nodes of the tree representing it. Let us consider the refinement of a single simplex (triangle) t of a current mesh. If the boundary of the mesh T_t refining it is not elementwise coincident with the boundary of t (i.e., with its three edges), then such a refinement cannot be considered a local modification, according to the definition given in Section 4.1.1, since T_t would not be compatible over the mesh containing t (see, for instance Figure 20).

In other words, two interfering nodes of the tree may not be compatible because their common border is subdivided differently. Therefore, in order to obtain the MT corresponding to a given tree-like model, nodes of the tree must be clustered to form compatible components [35]. More precisely, an MT $\mathcal{T}$ representing a tree-like model is obtaineded as follows:

- the fragments of $\mathcal{T}$ are obtained from the nodes of the tree by iteratively joining pairs of nodes corresponding to complexes in which two adjacent triangles have been refined by decomposing a common edge consistently;
- relation $<$ is defined by imposing that, for every arc (n_1, n_2) of the tree, if node n_1 was included into a component $T_i \in \mathcal{T}$, and node n_2 was included into a component $T_j \in \mathcal{T}$, then $T_i < T_j$ must hold. In other words, each node in the DAG representing the MT inherits all arcs incoming to, and outgoing from the nodes that have been clustered to form it.

Since an MT has maximum expressive power, we know that any mesh that can be extracted from a tree-like model will necessarily correspond to a cut in the MT obtained through clustering. Loosely speaking, if edges of triangles are split too often in the refinement process generating the tree-like model, then the nodes of the tree tend to be clustered into large fragments, hence allowing only a reduced number of cuts, i.e., supporting a reduced number of LODs. Moreover, since the relations between different nodes that must be custered to form a fragment are not encoded in the tree, algorithms for extracting generic meshes (e.g., meshes at variable resolution) from tree-like models will usually be involved.

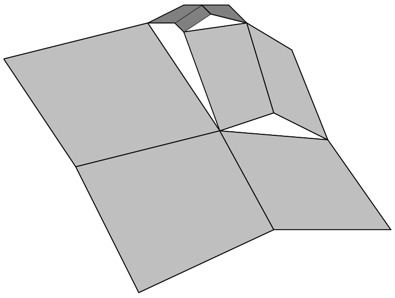


Figure 19: *A quadtree surface model is affected from cracks.*

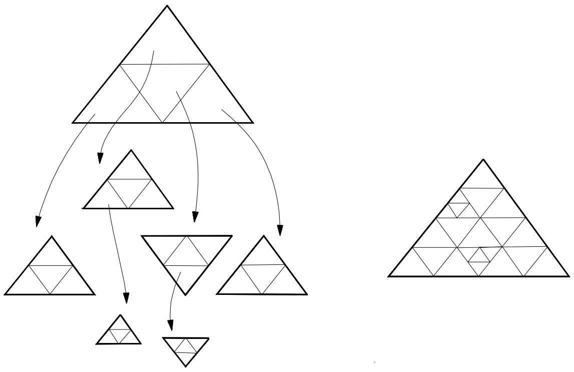


Figure 20: *A quaternary triangulation.*

4.2.1.1. Quadtrees. The *quadtree* is a general spatial structure based on the recursive quaternary subdivision of a square domain into quadrants, obtained by splitting the square with two orthogonal lines through its center [91]. The corresponding hierarchy can be represented by a quaternary tree, where each node corresponds to a mesh made of four square cells, while its children represent the four meshes refining each such cell.

If a quadtree is used as a base mesh to represent a surface, a bilinear patch of surface is associated to each quadrant of a mesh. Such a representation is straightforward for the case of scalar fields, like terrains [10]; it can be adapted to parametric patches whose computational domain can be a square; but it is not directly applicable to manifold surfaces, unless the surface represented is homeomorphic to a plane domain.

Despite the simplicity of the quadtree, and a number of good properties it has (especially as a spatial index), surface representation with such a model has a big drawback. In order to extract a mesh from a quadtree, the tree can be visited top-down, and meshes corresponding to each node can be either collected, or recursively refined, according to the desired level of detail. However, if a mesh is extracted, which combines nodes from different levels of the tree, the resulting surface will have cracks (see Figure 19). Such a kind of mesh will be said *non-conforming*.

Although the quadtree is not a simplicial model, the MT framework can be easily generalized to the case of rectangular meshes, hence encompassing also this structure. If the MT representation of a quadtree is computed, it is easily seen that all nodes at each level of the tree must be clustered to form a single component, hence giving a pyramid of full rectangular meshes. Indeed, all edges of a quadrant are always split during refinement, thus each quadrant must be clustered with its siblings to the same fragment. The resulting MT reduces to a simple LOD model, with a single fragment per level, and each fragment providing a whole, independent, surface mesh. Those meshes are the only conforming ones that can be extracted from the quadtree. Note that this fact prevents from extracting *adaptive* meshes, i.e., meshes whose level of detail is different over different portions of the domain.

The quadtree can be generalized to an arbitrary dimension, in order to model higher dimensional objects, by maintaining similar characteristics (and problems).

4.2.1.2. Quaternary triangulations. The *quaternary triangulation* is a straightforward generalization of the quadtree to a triangular mesh. Starting from a triangular universe, the recursive splitting rule consists of subdividing each triangle into four by joining the midpoints of its edges (see Figure 20). The interpretation of such a structure as MT is completely analogous to that of the quadtree. A single fragment per level of the tree is obtained through clustering, hence only one conforming mesh per level can be extracted.

The quaternary triangulation is widely used in domain decomposition for the finite element methods, and it has been applied also to surface representation in the case of terrains [41]. More recently, such a decomposition rule has been used in the context of manifold (free-form) surfaces, to support multiresolution based on the *wavelet transform* (see Chapter 6).

4.2.1.3. Adaptive hierarchical triangulations. The main problem of quadtrees and quaternary triangulations is that all edges of a regions are split when the region is refined. In the context of terrain modeling, other tree-like models have been developed, which try to preserve the possibility of performing adaptive refinement, by letting some edges "survive" across different levels of the tree.

A diametrically opposed philosophy performing refinements that never split edges, propagates long edges from the coarsest level down to the most refined level, hence producing horrible meshes full of slivery triangles [25]. A compromise has been proposed in [93], adopting an error-driven refinement rule that permits to split a triangle according to the five patterns depicted in Figure 21, i.e., by inserting a point

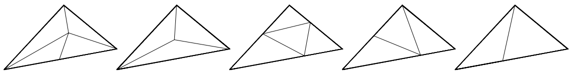


Figure 21: *Heuristic rules for refining a triangle.*

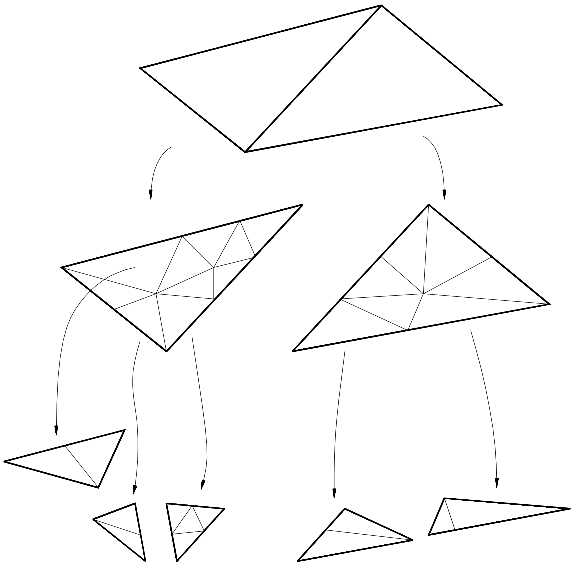


Figure 22: *A hierarchical Delaunay triangulation.*

on one or more of ite edges, and/or inside the triangle itself. In this way, edges that survive through different levels can provide an interface to weld triangles from different levels to form meshes at variable resolution. However, such a possibility was not exploited in [93]. In [34], another model was proposed, in which an error-driven refinement of a triangle is performed by inserting an arbitrary number of new vertices (selected through an error-driven criterion) inside a triangle, and/or on some of its edges, and by computing the refined mesh as the Delaunay triangulation of such a set of vertices (see Figure 22). Also in this case, some edges may survive across different levels, hence giving a reasonably large set of fragments in the corresponding MT, which permit to perform adaptive refinement. Such structures are increasing and non-redundant MTs in canonical form, but they might not have the desirable properties of linear growth, bounded width, and logarithmic height, at least in theory. However, they have a good behavior in practical cases. Their major drawback is still the presence of some slivery triangles, due to the coexistence of an irregular subdivision with a nesting structure. The model proposed in [34] has been extended to 3D in [4], and applied to volume data visualization.

In [34], a comprehensive study of tree-like models is presented, covering several issues about data structures, neighbor finding, and mesh extraction. Two algorithms for extracting meshes at variable resolution from tree-like models

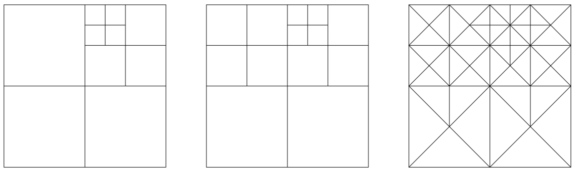


Figure 23: *A restricted quadtree is built by first balancing the tree locally, then triangulating each quadrant.*

are also presented. The first algorithm is a simple top-down visit of the tree, which accepts a triangle as soon as it satisfies the required level of detail. The resulting structure is a non-conforming subivision called a generalized triangulation, in which some triangles are added new vertices along their edges (just as in quadtrees and quaternary triangulations). A triangulation of such generalized triangles is performed next to conform the mesh, hence obtaining a continuous triangulated surface. The whole algorithm is completed in time linear in its output size. Note that the final triangulation step produces triangles that might not be part of the original structure, hence the resulting mesh is not necessarily one of the meshes that can be extracted from the corresponding MT. The main drawback, in this case, is that there is no control on the accuracy of such new triangles, hence the accuracy of the final structure might be worse than desired.

The second algorithm is essentially an adaptation of an earlier algorithm working on hypertriangulations [19], which will be reviewed in Section 4.2.2.5. In this case, the accuracy of the result is warranted, but the algorithm works only for a class of distance-increasing threshold functions, and its computational complexity is suboptimal. The implementation is somehow involved because of neighbor finding operations across different levels of the tree.

4.2.1.4. Hierarchies of right triangles. In the attempt to overcome the problem of cracks in quadtrees, a modified structure, called a *restricted quadtree*, was introduced in [104], and applied to the representation and rendering of free-form patches. A restricted quadtree is essentially a quadtree in which adjacent leaves are allowed to differ for no more than one level. All leaves form a mesh which is conformed by triangulating each quadrant according to predefined patterns: a quadrant is subdivided into four right triangles by its diagonals, and each such triangle is further subdivided into two right triangles by joining the center of the quadrant with the opposite edge, only if the adjacent quadrants along such an edge are at a deeper level than the current quadrant (see Figure 23). An efficient algorithm to build such a structure from terrain data has also been described in [33].

Note that such a construction technique gives a conforming mesh only at the leaf level. If a coarser mesh is extracted from the quadtree, then the resulting mesh must be suitably balanced, and triangulated a posteriori, hence giving a result whose triangles are not part of the original structure.

A way to overcome this drawback is to consider the whole hierarchy as formed by right triangles. In [53], a hierarchy of right triangles refining a square domain was introduced, and used to give representations of images at variable resolution. The hierarchy starts with a triangulation of the square made of four right triangles, obtained by splitting the square through its diagonals. Then, a tree is built by recursively splitting each triangle into two right triangles, obtained by joining the midpoint of its longest edge with the opposite vertex. The resulting hierarchy permits to grade refinement through the domain, thus supporting variable resolution.

The structure and power of this model is better understood by considering its corresponding MT, depicted in Figure 24. In this case, each fragment is formed by four right triangles (two right triangles for border fragments), and all have either the shape of a square, or of a diamond (of a right triangle, for border fragments). Each triangle is matching along each of its short edges with one triangle at the same level, and with another triangle at the next level, and along its long edge with one triangle at the same level, and with another triangle at the previous level. Each fragment is characterized by its central vertex, hence the MT can also be interpreted as an order relation establishing dependencies among vertices. It is easy to verify that this MT is in canonical form, increasing, non-redundant, and it has linear growth, bounded width, and logarithmic height.

Although in [53] recursive splitting has been only used as a simplification mechanism, full exploitation of this kind of hierarchy as a support to variable resolution has been proposed independently in [67], and in [37], for the purpose of terrain visualization in flight simulation. In both such papers, algorithms for extracting a reduced mesh satisfying the required LOD for a given eye position are presented, which are based on recursive refinement and hierarchical dependencies of vertices in the mesh.

In [37], a data structure that maintains explicitly the tree of triangles is adopted, where for each triangle only its accuracy is stored, while geometry is provided implicitly by its position in the tree. A mesh at variable resolution is extracted by traversing the tree top-down, and recursively refining triangles that do not satisfy the threshold function: at each refinement operation, vertex dependencies are propagated trhogu the tree, and further refinements are forced on adjacent triangles in order to ontain a conforming mesh.

In [67], only the raw grid of vertices (i.e., geometrical information) is stored, and a mesh at a given resolution is extracted on-the-fly by a bottom-up visit of the tree, and upward propagation of vertex dependencies. In this case, the basic idea is that a pair of adjacent triangles can be merged into their parent if the consequent loss of accuracy does not exceeds the threshold function. Since accuracies of triangles are not stored, and computing them exactly is too an expensive operation, heuristics are adopted to estimate them on-the-fly. Further heuristics are used to purge large portions of the domain through simple operations. See also Section 5.2.2.

In summary, the main advantage of hierarchies of right triangles is that extremely compact data structures can be used, because the hierarchy is implicitly defined on the basis of fixed patterns: there is no need to encode structural, topological, and interference information explicitly. Each triangle can be in fact treated as a code, which can be manipulated algebraically in order to obtain its geometry, its adjacencies, and its interferences (see [53, 37] for details).

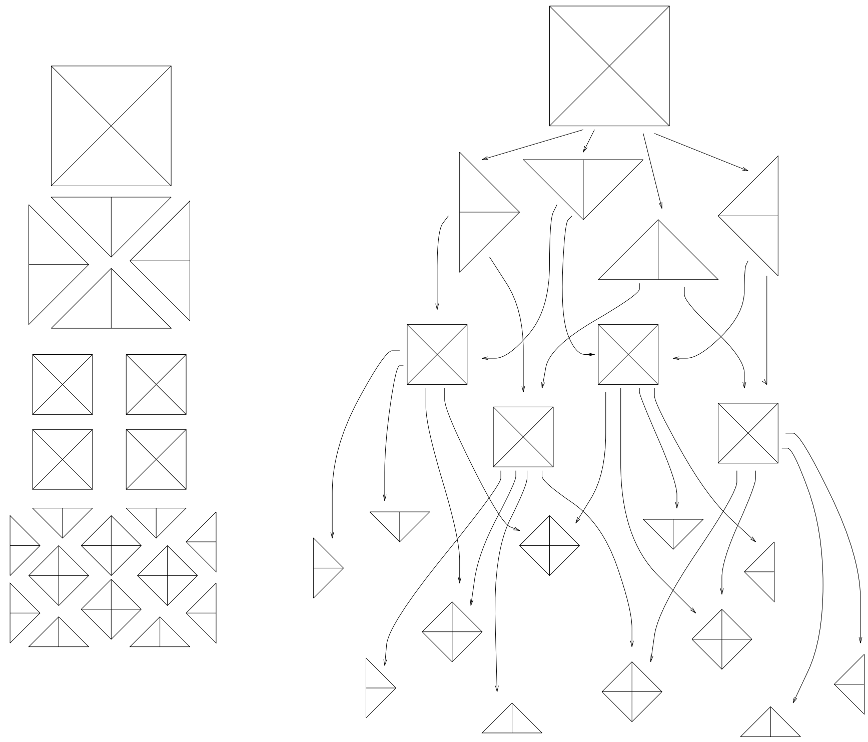
The extreme regularity of this structure, besides being its main advantage, is also its main limitation. In the case of scalar fileds, data points must necessarily be regularly distributed on a square grid; in the case of parametric surfaces, each patch must have a square computational domain, therefore it is difficult to handle patches with complicated trimming curves; in the case of closed manifold surfaces, the structure cannot be applied directly: if the surface has the topology of a sphere, the structure can be generalized by starting from an octahedron as base mesh, while it is not clear how to extend the framework to model surfaces with a more complex topology.

This kind of hierarchy is extensible to the higher dimensional cases, to model hypersurfaces produced by scalar fields over (hyper)cubic domains. Higher dimensional models maintain similar characteristics of regularity, and easy symbolic manipulation (see [52] for details on the 3D case).

4.2.2. Historical models

Historical models are multiresolution meshes whose structure is obtained by recording the evolution of a mesh through a sequence of local modifications. Let us consider one among the many mesh simplification algorithms based on local modifications (see Chapter 3). Each single simplification step performed by the algorithm can be considered to generate a new fragment, namely the fragment composed of all triangles that appear in the mesh after simplification. Such fragments together with the usual interferences between triangles that replaced by modifications, define a decreasing MT, rooted at the mesh at full resolution. The corresponding increasing MT is obtained by reversing the structure, or, equivalently, by considering each fragment as formed by the triangles that disappear from the mesh at a single simplification step. Conversely, an increasing MT can be obtained directly by considering fragments generated at single steps of a refinement algorithm. All historical models can be considered to fall into this class of construction techniques. The main differences between the various models are:

1. the simplification/refinement criterion adopted to build the structure;



(a) (b)

Figure 24: *An MT based on right triangles: the fragments shown level by level (a), and the corresponding DAG (b).*

2. the amount of information stored;
3. the operations supported, and the efficiency of algorithms that implement them.

The review of models in this class not only gives an interesting insight of the evolution from LOD to variable-resolution models, but it also allows us to discuss the effectiveness, and efficiency of multiresolution meshes, and on the necessary traedoff between space and time complexity.

4.2.2.1. Delaunay pyramids. In its original formulation [24], the Delaunay pyramid is a LOD model for terrain surfaces, made of a sequence of Delaunay triangulations, and obtained on the basis of a predefined sequence of error tolerances $\varepsilon_0 > \ldots > \varepsilon_h$. The first triangulation represents the surface at an accuracy ε_0, and each other triangulation is obtained from the one preceding it in the sequence, by applying the refinement algorithm described in [38], until the next accuracy is met. All levels are encoded in a global data structure, which stores all spatial interferences between triangles of consecutive meshes, in order to support navigation (e.g., point location).

In [5], a pyramid with many levels is considered, having triangles that "survive" across different levels, and a more compact data structure is proposed, which avoids replicating surviving triangles. Such a data structure implicitly introduces the concept of fragment, which is exploited in a later work [32] to extract meshes at variable resolution, by adapting a general algorithm proposed for MTs in [82] (see part Explicit and implicit MTs in Section 4.2.2.4).

The Delaunay pyramid is directly applicable to the case of scalar fields, and it has been extended to the 3D case for volume visualization in [12]. The framework could be extended to deal with parametric surfaces, by using a constrained Delaunay triangulation in parameter space. An extension to manifold surfaces is hardly feasible due to the difficulty of reproducing the Delaunay refinement framework on free-form meshes sembedded in 3D space.

The interpretation of a Delaunay pyramid as an MT is straightforward. In the original formulation, the pyramid is formed by a small number of layers, and eahc layer is a mesh completely different from the previous one in the sequence, hence there is just one fragment per layer. The advantage over quadtrees and quaternary triangulations is that in this case the tessellation used at each level is adaptive. In the formulation of [5], the DAG describing the MT can be built directly from the pyramid data structure. The resulting MT is analogous to the *explicit MT* described in a later section: it is increasing, non-redundant, and in canonical form; in theory, it might have not linear growth, bounded width, and logarithmic height, though in practice it shows a good behavior.

Such drawbacks were eliminated in an alternative model proposed in [23], which is essentially a Delaunay pyramid built bottom-up, starting at the full resolution mesh: each level is obtained from the previous one through a decimation process which eliminates an independent set of vertices of bounded degree. In this way, the corresponding MT is formed by a large number of fragments, namely one for each decimated vertex, and has linear growth, bounded width, and logarithmic height, hence supporting point location in optimal time. In [23], an algorithm was also described, for extracting in time linear in its output size a representation at variable resolution based on an arbitrary threshold function. The algorithm is based on a top-down traversal of the pyramid, and on a greedy construction of the result. Unfortunately, the greedy approach, which accepts a triangle in the solution as soon as its accuracy satisfies the threshold function, does not warrant that the desired threshold will be fulfilled everywhere: indeed, at a later stage, the algorithm might not be able to refine some triangles that do not satisfy the threshold, because not fragment dependencies can be fulfilled.

The construction algorithm adopted in [23] gives theoretically good features to the resulting model, but in practices it achieves a lower compression (i.e., it needs more triangles to get to a certain accuracy) with respect to the greedy technique adopted for the original Delaunay pyramid. In [27], different techniques for building a multiresolution mesh based on decimation are analysed in the more general context of MTs. Experimental results and theoretical analysis show that the best results from both the theoretical and practical point of view are obtained with a modified decimation technique that eliminates maximal sets of independent vertices, by selecting such vertices among those that cause the least error increase. However, results obtained with a simple greedy decimation (or refinement) achieve almost the same quality in terms of growth, width, and height of the structure. Therefore, the performance of traversel algorithms in the practical cases is almost the same, independently of the construction algorithm.

4.2.2.2. Linear sequences. In [19], it was noted that by tracing the change of accuracy of a mesh through a process of refinement or simplification, it is possible to assign a *life* to each simplex, namely the range of accuracies through which the simplex "survives" in the mesh. Life of simplexes can be used as a filter: if all simplexes that appear during refinement/simplification are stored, each tagged with its life, then the collection of simplexes of a mesh at uniform resolution can be obtained by a simple scan of the sequence. In [5] and [14] more efficient data structures were proposed that support filtering from the sequence in optimal time.

Following a similar principle, an extremely compact model based on the Delaunay triangulation was proposed in [65] to model parametric and manifold surfaces. In this case, a greedy refinement or decimation through on-line insertion/deletion of vertices into/from a Delaunay triangulation is performed (in computational space for parametric surfaces, and on a local planar projection for manifold surfaces). In terms of the corresponding MT, this structure is completely analogous to Delaunay pyramids with many levels. The main characteristics of this multiresolution mesh is in its data structure: only the simplest mesh, plus the sorted sequence of vertices inserted during refinement, or removed during decimation, are stored, each tagged with the approximation error of triangles incident at such vertex at time of its insertion. All structural, topological, and interference information are assumed implicitly given thorugh the Delaunay construction rule. The great advantage in terms of storage requirement is paid in terms of efficiency in the traversal algorithms: an algorithm for extracting a domain-uniform mesh must scan the sequence of vertices and explicitly build the mesh through an on-line Delaunay triangulation algorithm, until the desired accuracy is met; an algorithm for extracting domain-variable meshes is also outlined, which requires intricate tests based on the circumcircle criterion (hence involving numerical computation) in order to trace vertex dependencies.

Independently, a similar model, called *progressive meshes (PM)* was proposed in [57], based on a different simplification strategy. In this case, the basic local modification is *edge collapse*, which is reversed into *vertex split* to obtain an equivalent refinement sequence (see Figure 25). The corresponding MT is in canonical form, non-redundant, and increasing. In the worst case, it might not have linear growth, bounded width, and logarithmic height, but it has good behaviour in practice. As for the previous model, the data structure adopted is implicit and very cheap in terms of storage: only the initial mesh, and the sequence of vertex splits are sotred; each vertex split requires one new vertex, plus three indices to existing vertices. The extaction of a domain-uniform mesh is simpler than in [65], because no numerical computation is required, while topology of the existing mesh is modified through symbolic updates; an algorithm for domain-variable meshes is just outlined in [57], which requires involved traversals of the list to find vertex dependencies on-line. Progressive meshes have been used for different applications, such as terrain surfaces, manifold surfaces, and color images.

4.2.2.3. Hierarchies of vertices. In the attempt to obtain efficient algorithms for extracting domain-variable meshes from a PM, more sophisticated data structures encoding also vertex dependencies were proposed in [110], and more recently in [58].

The basic observation behind these structures is that a vertex v can be split while extracting a mesh only if all other vertices that surround v in the original PM sequence belong to the current mesh already. If the PM is interpreted as an MT, the previous condition can be restated as follows. The DAG desribing the MT can be seen as a vertex hierarchy:

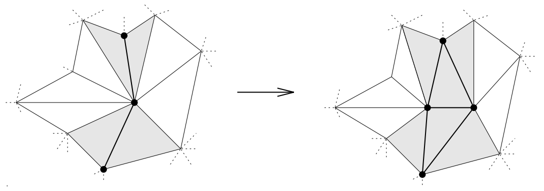


Figure 25: *An operation of vertex split: vertex v splits to form an edge, thick edges are duplicated, and two new triangles appear; the shape of each triangle already incident at v changes because the two new vertices move away from v.*

indeed, each fragment has the shape of the right side, and a floor like the left side of Figure 25; this means that the two internal vertices of a fragment depend on the central vertex of its floor, and on all boundary vertices of the fragment. Such vertices have been inserted in the current mesh already if and only if the down closure of the current fragments has been traversed already (i.e., all corresponding fragments have been combined already).

In [110], a PM is built by collapsing independent sets of edges, following a technique similar to that used in [23] to build a Delaunay pyramid bottom-up. In this way, a binary balanced tree is obtained, where the children of each vertex v are the endpoints of the edge that collapsed in v. The corresponding MT has linear growth, and logarithmic height, but it does not necessarily has bounded width. In the MT terminology, the tree of vertices only encodes dependencies btween central vertices of a fragment and its floor. The remaining dependencies with boundary vertices of the fragment are encoded separately. This data structure is somehow equivalent to, though more involved than, the data structure for implicit MTs described below.

Variable resolution extraction from such a structure is performed in two steps: first, the binary tree is traversed bottom-up, and vertices crresponding to too detailed fragments are discarded on the basis of some LOD criterion; next, vertex dependencies are traced back from the remaining vertices, in order to include all vertices that must be necessarily inserted. The extracted mesh is obtained by performing vertex splits only at selected nodes, in the original order of the PM. The main problem of such an algorithm is that the hierarchy is visited bottom-up, hence all vertices of the model must be analyzed to obtain even the simples possible mesh.

In a recent paper [58], a similar binary tree (actually, a forest) of vertices is used, which is built top-down in a preprocessing step by traversing the PM. Remaining vertex dependencies are stored locally to each node, by maintaining all vertices marked, and triangles shaded in Figure 25. It is claimed in [58] that such information is sufficient to determine all vertex dependencies. An algorithm is proposed, which extracts a mesh at variable resolution by traversing the tree top-down, and forcing vertex split whenever the resolution condition requires it. A split operation is performed by recursively managing vertex dependencies through a stack. Such an algorithm is essentially analogous to that working on implicit MTs that will be described in the following. Although neither its correctness, nor the minimality of the extracted mesh are proved, it produces qualitatively good results and it can achieve real time performance.

4.2.2.4. Explicit and implicit MTs. Besides being a theoretical framework, an MT can be directly implemented through suitable data structures, and used to extract meshes at variable resolution, and to perform other traversal operations (see [35] for details on operations).

In [82] a general algorithm for extracting a mesh at variable resolution from an MT is proposed, which works by traversing the MT starting at its root, visiting the DAG in breadth-first order, and marking all triangles that cannot be part of the solution. A queue of fragments that must be visited is maintained, which is initialized with the root, while triangles selected for a potential solution are added to a list. After traversal, such a list will contain all triangles of the solution, plus some extra (marked) triangles that are purged through a single scan. In [82], a formal proof is given that the algorithm extract the smallest possible mesh for a given threshold function, and that runs in time linear in the size of the lower set generaing the mesh. If the MT has linear growth, then the time complexity is output-sensitive optimal, i.e., linear in the size of the extracted mesh. Independendtly, a similar algorithm was proposed in [7], achieving similar results.

Such an algorithm requires a data structure based on interferences that directly implements the lattice of Figure 14, namely it encodes all fragments and triangles, and for each fragment two lists of triangles forming it and its floor, respectively, and for each triangle the fragment holding it, and the fragment having it in its floor. No topological information is encoded, while each triangle is tagged with its approximation error. An MT encoded with such a structure will be called an *explicit MT*.

In a recent paper [27], an improved version of the algorithm working on explicit MTs were proposed, which works in a single step (no final list scan is required), and is interruptible, namely it always maintains a current valid mesh, which can be returned at any time if the algorithm is interrupted before its completion. This feature is relevant in real time applications. Results presented in [27] show that real time performance can be achieved on reasonably large meshes.

A dynamic algorithm is under development, which is based on the possibility to update a current cut, by moving it up and down through the DAG, in order to adapt the resolution of a current mesh to a changing threshold function [28]. With such an algorithm, even better performances might be achieved in dynamic scenarios like flight simulation.

A possible drawback of explicit MTs is their storage requirement. Indeed, the number of triangles in the structure can be relevant, compared with the number of its vertices, hence the storage might be much larger that that needed, e.g., by linear structures. In [29], an alternative data structure, called *implicit MT* is proposed, which can be used to encode any MT built through refinement/simplification rules that are implicitly defined, such as PMs, or historical Delaunay sequences; In this case, a *raw* DAG is maintained, where for each node a minimum amount of information is stored, necessary to perform the local modification corresponding to the node. For instance, in the Delaunay case, it is sufficient to store the central vertex, wile for PMs, it is sufficient to store the same information present at each element of the PM sequence. In other words, such a data structure is obtained by taking elements of a linear structure, and arranging them in the partial order that defines the corresponding MT.

As shown in [29], the space occupied by an implicit MT is not much higher than that of a corresponding linear sequence, while efficient extraction algorithms for variable resolution meshes can be applied, because intereferences among fragments are encoded explicitly in the data structure. However, since triangles forming a mesh must be recomputed explicitly during extraction, time performance is affected. In the experiments presented in [29], the extraction algorithm runs on the average ten times faster on the explicit data structure, than on the implicit one. A futher drawback, which is common also to *linear models*, and *vertex hierarchies*, is that the minimal elements for which accuracy is encoded in the data structure are fragments, rather than triangles. This fact prevents the extraction algorithm from the possibility to select some triangles from a fragment, while discarding other triangles, on the basis of the accuracy of each single triangle. As a consequence, the extracted mesh might be not the minimal one that satisfies the threshold function. In the experiments presented in [29], a mesh extracted from the implicit structure is, on average, twice larger than the corresponding mesh extracted from the explicit MT.

Explicit and implicit MTs have been applied to terrain surfaces, and manifold surfaces [27], and their application to parametric surfaces is under development [31].

4.2.2.5. The hypertriangulation. The *hypertriangulation* proposed in [19] is somehow unique in the context of multiresolution meshes, since it is the only model based on topological information. The main principle behind this structure comes from the embedding of fragments of an MT into a higher dimensional space, as explained in Section 4.1.2, and shown in Figure 17. The data structure based on triangle adjacencies (see Figure 18) permits to traverse the structure by moving through the domain, and across levels of resolution, while no interference information is encoded. Each triangle in the hypertriangulation is tagged with its *life*, defined as above.

Such a data structure is exploited to design an algorithm for extracting variable resolution representations for a special class of threshold functions, namely those that are monotonically increasing with distance from a given viewpoint, such as those needed in flight simulation. The algorithm is based on the incremental construction of a mesh, starting at a triangle near the viewpoint, and proceeding on the surface iteratively in breadth-first order: at each cycle, the border of the current mesh is considered, and a new triangle is extracted through adjacencies from the hypertriangulation, which is adjacent to the current mesh, and extends it from outside its border. While moving away from the viewpoint, the algorithm exploits the sorted sequence af adjacencies at each edge to extract triangles from progressively coarser levels of detail.

This traversal technique ensures that the extracted model will satisfy the threshold function everywhere, but the computational complexity is suboptimal in the worst case, i.e., $O(n \log n)$, where n is the size of the hypertriangulation. However, in the experiments shown in [19] good performance were achieved on reasonably large datasets.

Hypertriangulations built from a Delaunay refinement technique [38], have been used to model terrain surfaces in [19]; hypertriangulations built from a decimation technique [11] were used for representing manifold surfaces in the context of an interactive multiresolution modeler [15]. The hypertriangulation can be directly extended to higher dimensional cases, though the size of the data structure based on topological information can grow exponentially with the dimension of the complex.

4.3. Discussion

We finally give a qualitative evaluation of the models that we reviewed, based on some issues that have relevant impact in surface modeling, processing, and visualization. Our discussion does not intend to be exhaustive. Many different issues can be brought either in favor or against each model or class of models. We follow here a track that derives both from the arguments used by the different authors that proposed such models, and from the comparative study we have carried out while trying to fit such models to our framework.

4.3.0.6. Modeling issues.

- *Adaptivity and data distribution.* A main purpose of multiresolution models is data compression for intermediate levels of precision. Models that are based on irregular subdivisions offer in general better possibilities in adapting to surface characteristics, thus, they usually achieve better data compression. Note also that models based on irregular subdivisions admit any kind of distribution of data, while models based on regular subdivisions require regularly distributed data.
  In this view, quadtrees, and their variants (including trees of right triangles), can be considered less adaptive than all others, since they are based on regular subdivisions. Historical models in general offer excellent adaptivity: the

high independence of local modifications enhances the possibility to adapt refinement locally as the accuracy increases, while the strict hierarchy of tree-like models imposes boundaries at each refinement, which could prevent the possibility of improving the adaptivity on a local basis.

- *Expressive power.* It is important that a multiresolution mesh provides many possible meshes at different resolutions that can be obtained by combining its components. As we already discussed, the expressive power is reletad to the fact that the MT corresponding to a given model is in canonical form and non-redundant, and that its fragments are sufficiently small, and can be combined in many possible ways to obtain meshes. All models reviewed correspond to non-redundant MTs in canonical form.
  Concerning the size of fragments, quadtrees, quaternary triangulations, and the original Delaunay pyramid with few levels have a poor behavior, since they are made of few large fragment, each corresponding to a whole mesh, hence they have the expressive power of a simple LOD model. Tree-like models based on irregular triangulations have better characteristics, though fragments can become considerably large, unless such models are built by performing only small refinements between consecutive levels. All other models correspond to MTs with small fragments, each corresponding to local modification that involves an expectedly small number of triangles, hence they are expected to have a high expressive power. MTs with bounded width warrant that the size of a fragment is always bounded by a small constant, hence giving a further theoretical support to such a feature.

- *Shape.* The shape of cells in the mesh should be maintained as much regular as possible. In particular, regions with elongated shape (slivers) should be avoided. This issue has impact in visualization, and in numerical processing.
  Models based on regular subdivisions exhibit the best features in terms of shape, since they guarantee the same shape (e.g., square, right triangle) for all regions. Tree-like models based on irregular subdivisions are often affected by slivers: in general, slivers are a consequence of refinements that avoid edge splitting in order to let some edges survive across different levels of resolution. Therefore, in the case of tree-like models, either the mesh gets corrupted by slivers, or the multiresolution mesh results with a small expressive power.
  For historical models, the shape of cells (simplexes) is highly dependent on the local modification criterion adopted in building the structure. Criteria based on the Delaunay triangulation, either on a local basis, as in [11, 65], or on a global basis, as in [23, 24, 27] are aimed to maintaining triangles as more regular as possible in retriangulating the mesh because of vertex insertion or removal. Edge collapse used in PMs [57, 110] might increase the number of edge indicences at some vertices, hence obtaining sharp angles at the corresponding triangles. However, the visual effect of meshes obtained with this technique appears reasonably good.

- *Conforming meshes.* As outlined previously, it is important that a mesh extracted from a multiresolution mesh is conforming, i.e., it defines a continuous surface (or hypersurface).
  Quadtrees and quaternary triangulations have a poor behaviour in this respect, since no mesh extracted from them are conforming, except those corresponding to regular grids. Restricted quadtrees give a conforming mesh only at the highest level of resolution, while extracting a conforming mesh at an intermediate level involves an unwieldy procedure. The remaining tree-like models provide conforming meshes through suitable extraction procedures.
  All historical models provide only conforming meshes through suitable extraction algorithms.

- *Storage.* The storage cost of a multiresolution model is important to evaluate its goodness. Storage requirements are due not only to the intrinsic structure of a model, but primarily to the data structure adopted to encode it.
  Models based on regular subdivisions, such as quadtrees, quaternary triangulations, and hierarchies of right triangles achieve the best results in this respect, since it is possible to exploit their regular structure to maintain structural, topological, and interference information implicitly, while encoding only the vertices of the model in compact structures such as bidimensional arrays. The efficient computation of the structure of cells and fragments, topological links, and interferences can be performed by using locational codes and symbolic computation. Tree-like structures based on irregular triangulations require encoding structural information explicitly at each node of the tree, together with hierarchical links among nodes. For the sake of efficiency in traversal operations, it might be also necessary to encode adjacencies between nodes that conform at their common boundaries, hence requiring a high storage cost.
  For historical models, there is a great variety of data structures of increasing complexity and efficiency. Linear structures are the most compact ones, since they encode only vertices and a small amount of additional information; implicit MTs and vertex hierarchies also encode interference information; explicit MTs and Delunay pyramids encode both interference information and structural information on fragments; hypertriangulations do not encode interference information, but encode topological information, which might require even more space. Empirical evaluations reported in [29] show that implicit MTs require about twice, and explicit MTs require about seven times the space occupied by linear models for an equivalent MT (linear models are taken as reference, because they are the most compact structures for encoding MTs based on irregular triangulations).

4.3.0.7. Processing issues. Processing issues refer to the operations one wants to perform on a multiresolution mesh. Such issues have great impact on the design and implementation of systems, and they are often related to some of the modeling features that we have just discussed.

- *Construction.* All construction algorithms are essentially based on the simplification technology, hence their performance can be evaluated on the basis of the discussion made in Chapter 3. The worst case time complexity of such algorithms is seldom significative, since it is usually much pessimistic with respect to the practical performance. However, since all models can be built off-line in reasonable time, we can consider all of them valid in this respect. Nevertheless, because of the huge amount of data that must be processed in some applications (e.g., terrain data), the possibility of building different parts of a model independently seems a critical issue. Tree-like models, and especially quadtrees and their variants, seems most suitable to support data partition, provided that suitable data structures for secondary storage are developed. Historical models, on the contrary, being based on a global approach, suffers from the need for maintaining always all data available.
- *Extraction of a mesh.* This is the principal operation that is performed on a mutliresolution mesh for the purpose of visualization. The most relevant issues about algorithms implementing it are the ability of extracting meshes either at uniform or variable resolution, and their time performance.
  Simple and efficient algorithms for extracting meshes at uniform resolution have been proposed for all models reviewed. In some cases, such as for quadtrees, quaternary triangulations, and the Delaunay pyramids, such algorithms are almost trivial; for most other models, algorithms are based on some simple traversal of the data structure, and collection of simplices that form the resulting mesh. Such an operation is more expensive for linear sequences that do not encode simplices explicitly, due to the need of computing the mesh on-line from its vertices.
  Algorithms for extracting meshes at variable resolution have been proposed only for most recent models, though it is possible to extend such techniques also to earlier models. Due to the poor expressive power of quadtrees and quaternary triangulations, variable resolution can be obtained from these models only by "cheating", i.e., by extracting first a non-conforming mesh through a simple traversal of the hierarchy, and triangulating non-conforming regions in a second step, to obtain a conforming mesh. As remarked already, a mesh obtained in such a way is not among those generated by the corresponding MT, and there is no control over its structure. For tree-like models based on irregular triangulations, a correct algorithm for variable resolution extraction needs that suitable links are maintained in the data structure to reconstruct either tragments of the corresponding MT, or adjacencies of the corresponding multitriangulations, hence making computation unwieldy.
  For the case of historical models, and for hierarchies of right triangles, we can outline three basic classes of algorithms, on the basis of the MT interpretation of each model:
  1. algorithms based on top-down traversal of the DAG;
  2. algorithms based on bottom-up traversal of the DAG;
  3. algorithms based on breadth-first traversal of the domain.

  Most algorithms belong to the first class. The first such algorithm was proposed in [23] for the Dlauanay pyramid built bottom-up: such an algorithm has an optimal time performance, but cannot warrant that the extracted mesh fulfills the required threshold everywhere. Such a problem was fixed in the algorithms proposed in [82, 7, 27] for the explicit MT, which achieve optimal performance, while guaranteeing that the extracted mesh is the smallest one satisfying the threshold everywhere. Similar results are obtained on a hierarchy of right triangles by the algorithm proposed in [37], which is performs a top-down visit of the tree of triangles, together with a downward propagation of vertex dependencies. Algorithms proposed for the implicit MT [29], and for the PM with vertex hierarchies [58] follow a similar technique by achieving optimal time performance, but they cannot warrant the minimal size of the result. Algoritms for linear structures proposed in [57, 65] also belong to this class: even such algorithms cannot guarantee the minimality of the result, and, moreover, they are much slower than the previous ones because of the need of computing both the mesh structure, and fragment interferences online.
  Only two algorithms have been proposed in the second class : the one proposed in [110] works on a PM with vertex hierarchies, while the one proposed in [67] works on a hierarchy of right triangles. In these cases, it is possible to achieve the minimum size of the result, but the time performance is not output-sensitive: it is always necessary to traverse the whole multiresolution mesh to obtain even the simplest possible mesh. It should be pointed out that the approach followed in [67] contains lots of heuristics and approximate error evaluations, aimed to speedup traversal of the dataset, which make the algorithm fast, but might lead to a mesh that either does not fulfill the threshold function everywhere, or is larger than necessary.
  An intersting extension of the first two classes is the dynamic algorithm proposed in [28], which exploits the possibility to traverse the DAG describing an MT both topdown and bottom-up, starting at a current cut, in order to update a current mesh to a changing threshold.
  The third class of algorithms require topological information. Hence, only an algorithm of this class, working on hypertriangulations has been proposed [19]. Such an algorithm supports only distance-increasing threshold functions, it cannot warrant the minimality of the result, and

it has suboptimal time complexity, though practical performance show a good behavior. This algorithm has also ben adapted to work on irregular tree-like models [34].

- *Zooming.* Tree-like models favor operations like focusing over a specific area of interest, through a top-down traversal of the tree: the area of interest is localized on a given subdivision, and only those regions that fall in such an area are considered for further expansion at the desired level of resolution. Regular models, being based on subdivisions with a constant number of regions, are better suited to this task.
  Not all historical models support zooming: essentially, efficient zooming techniques require interference information. Only Delaunay pyramids, and explicit and implicit MTs provide such information. The remaining models give small or no support at all.
- *Navigation.* A generalization of zooming is navigation of the structure. Many tasks (especially interactive manipulation) require navigating the structure both through different levels of resolution, and across domain. Navigation across domain requires topological information.
  Regultree-like models can be traversed afficiently by exploiting adjacencies and interferences implied by their regular structure. Hierarchical Delaunay triangulations explicitly encode interference, local topology, and adjacency information between siblings, therefore supporting navigation.
  Among historical models, only the hypertriangulation encodes topological information: global adjacencies in such a structure can be also used to move across levels of resolution.
- *Geometric queries.* The possibility of performing efficiently geometric queries, such as point location, ray shooting, clip, etc., enhance the application of multiresolution meshes in the modeling and rendering contexts.
  Regular tree-like models naturally offer a spatial indexes that support such operations with high efficiency through standard algorithms (e.g., point location can be done in logarithmic time) [92]. Similar techniques can be applied to irregular tree-like models, though with a lower theoretical efficiency [26].
  Point location can be also performed efficiently on Delaunay pyramids, and explicit MTs, and optimal logarithmic time complexity is achieved in the Delaunay pyramid built bottom-up, and in general on MTs with bounded width and logarithmic height. More complex queries need both interference and topological informations, hence they are not fully supported by any historical model. A more detailed analysis of requirements for supporting geometric queries on MTs is given in [35].

5. Applications

5.1. Applications to graphics and virtual reality

LOD representation is an important resource in interactive graphics and virtual reality systems, and it will be widely adopted in the near future in all interactive applications. An overview of LOD support in Virtual Reality solutions is proposed in [85]. The support given for the LOD representation in OpenInventor and VRML2.0 is presented briefly in the following subsection.

Given a 3D scene, the LOD representation of the objects contained has to be built. Modern toolkits or modeling environments provide tools for the controlled simplification of polihedral meshes. Some of them will be reviewed in Subsection 5.1.2.

Moreover, the tools for the construction of variable resolution meshes out of multiresolution models are presented in Subsection 5.1.5.

5.1.1. LOD in OpenInventor and VRML 2.0

Open Inventor™ [43, 106] is an object-oriented toolkit for developing interactive, 3D graphics applications (see the Web at: http://www.sgi.com/Technology/Inventor.html). It also defines a standard file format for exchanging 3D data among applications.

Later on, Open Inventor served as the basis for the VRML (Virtual Reality Modeling Language) [105] standard (see the Web at: http://vag.vrml.org/), a file format for describing interactive 3D objects and worlds. VRML is designed to be used on the Internet, intranets, and local client systems. VRML is capable of representing static and animated dynamic 3D and multimedia objects with hyperlinks to other media such as text, sounds, movies, and images. VRML browsers, as well as authoring tools for the creation of VRML files, are widely available for many different platforms.

The *Open Inventor* classes of database primitives include shape nodes (for example, sphere, cube, cylinder, quad mesh), property nodes (for example, material, lighting model, textures, environment), and group nodes (for example, separator, level-of-detail, and switch).

In particular the *SoLevelOfDetail* node, using the OpenInventor terminology, allows you to specify the same object with varying levels of detail or complexity, and provides hints allowing to automatically choose at rendering time the appropriate version of the object. The children of this node are arranged from highest to lowest level of detail. The size of the objects when projected into the viewport determines which child to use. This node is very useful for applications requiring the fastest rendering possible. It has one field, the *screenArea* field, which specifies the area on the screen to use for comparison with the bounding box of the level-of-detail group. To determine which child to traverse, Inventor computes the 3D bounding box of all children in the level-of-detail group. It projects that bounding box onto the viewport and then computes the area of the screen-aligned rectangle that surrounds the bounding box. This area is then compared to the areas stored in the screenArea field.

Analougously, the VRML 2.0 LOD grouping contains any number of child objects, referred to as *levels*.

```
LOD {
    exposedField    MFNode level []
    field           SFVec3f center 0 0 0    # (-∞,∞)
    field           MFFloat range []        # (0,∞)
}
```

The main difference between OpenInventor and VRML is the different policy adopted to drive the selection of the rendered level of detail: VRML level selection depends on the object distance from the viewpoint.

The *level* field contains a list of nodes which represent the same object or objects at varying levels of detail, ordered from highest level of detail to the lowest level of detail. The *range* field specifies the ideal distances at which to switch between the levels. The *center* field is a translation offset in the local coordinate system that specifies the centre of the LOD node for distance calculations.
The number of nodes in the level field shall exceed the number of values in the range field by one (i.e., N+1 level values for N range values). The range field contains monotonic increasing values that shall be greater than 0. In order to calculate which level to display, first the distance is calculated from the viewer's location, transformed into the local coordinate system of the LOD node (including any scaling transformations), to the center point of the LOD node. The LOD node evaluates the integer function $L(d)$ to choose a level for a given value of d (where d is the distance from the viewer position to the centre of the LOD node).

Specifying too few levels will result in the last level being used repeatedly for the lowest levels of detail. If more levels than ranges are specified, the extra levels are ignored. An empty range field is an exception to this rule. This case is a hint to the browser that chooses the level automatically to maintain a constant display rate.

5.1.2. Construction of LOD representation

Given a framework with visualization toolkits or graphics libraries which support LOD representation, from the user point of view a very handy tool should build automatically the sequence of representations of a given object which optimize its visualization in the framework of visualization toolkits or graphics libraries which support LOD representation.
The tools which are currently available show more limited capabilities. They only support the construction of simplified representations of the objects and their composition in a LOD model. The choice of the different levels of detail (i.e. the complexity of each single model, and the degree of geometrical/visual approximation) is therefore left to the user.

5.1.2.1. SGI Cosmo Worlds. Cosmo Worlds (see on the Web at: http://cosmo.sgi.com/worlds/developer/DataSheet/) supports the creation and editing of the 3D models that make up a virtual world. Among the tools provided, the *Optimization Tools* suite gives the user total control for streamlining visual response and interactivity, optimizing playback performance, or speeding downloads of complex scenes. The Polygon Reduction Editor, the Polygon Builder, the Inline Editor, and the LOD Editor provide users with a choice of methods for simplifying the displayed or downloaded scene.

The Poligon Reduction Editor reduces polygon count of an object or group of objects in the scene, while trying to keep the shape's integrity. Using the sliders (see Figure 26) user may experiment with different types of polygon reduction:

- *delete points by curvature*; deletes points and retriangulates the shape to compensate for lost points. Measures the dihedral angle between each adjacent pair of triangles and removes those triangle pairs that have a common dihedral angle less than the value specifyed with the slider;
- *discard triangles by area*; finds the area of triangles in the object and gets rid of the smallest triangles;
- *discard edges by length*; calculates the lengths of all edges of the object and discards the smallest;
- *merge initial coordinates*; merges nearby points into a single point if they are within a certain distance of each other.

The results of simplification actions are presented in the editor's graphic window.
All methods above are based on a clustering approach. This provides good efficiency, but low quality of the simplified meshes produced (see Subsection 3.3.3).

Once multiple levels of detail have been created, the LOD representation can be built using the Cosmo's Level of Detail Editor (Figure 27). User may select the distances at which to display each representation, or the selection of the ranges may be omitted, thus creating a "performance LOD" grouping that allows the browser to select the appropriate level for optimal performance. In this case, the browser selects the level with the lowest index that it can render while still maintaining an acceptable frame rate. Some browsers may ignore this option.

A new set of simplification tools has been recently announced in the framework of the *OpenGL Optimizer* application programming interface (API) (see the Web at: http://www.sgi.com/Technology/OpenGL/optimizer/presentation.html). The OpenGL Optimizer API is under developement to meet the demands of visualizing complex CAD data sets. This API is built on top of OpenGL.
The OpenGL Optimizer modular structure is composed of 5 distinct sections. The section labeled "Simplifiers" will contain the technology that enables to automatically generate and utilize different LOD.

5.1.2.2. IBM Interaction Accelerator. IBM 3D Interaction Accelerator is a workstation-based interactive soft-

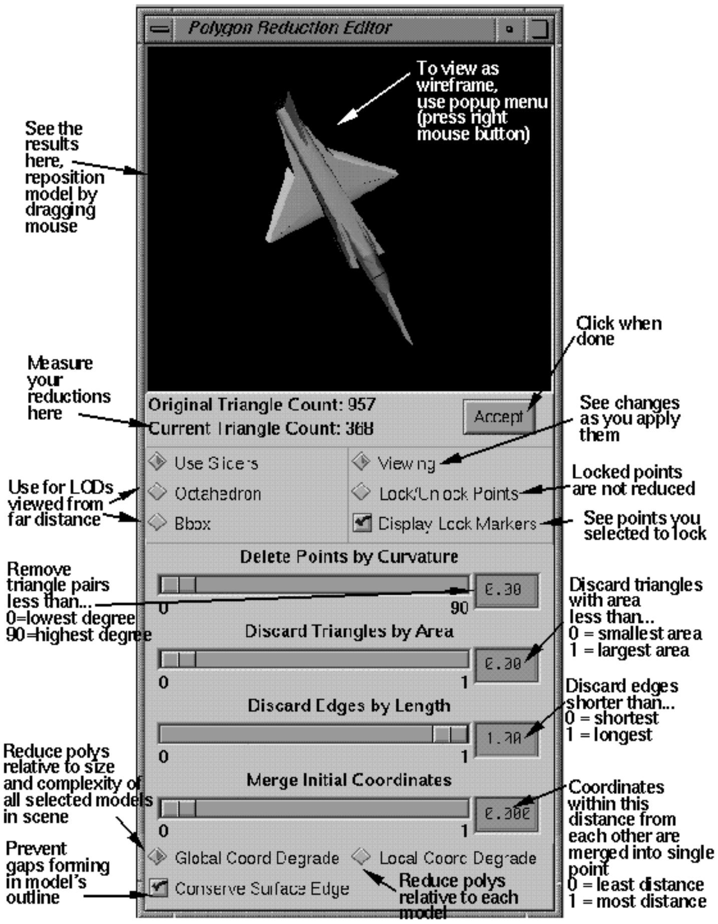


Figure 26: *The graphic interface of the Polygon Reduction Editor of Cosmo Worlds (image courtesy of SGI inc.).*

ware product that enables real-time visualization and inspection of very large and highly complex mechanical and architectural CAD models (see the Web at: http://www.research.ibm.com/3dix/). It includes a simplification module, based on the Vertex Clustering algorithm [90]. An example of a mesh at different levels of detail, obtained with the Interaction Accelerator software, is presented in Figure 28.

5.1.2.3. IMCompress. IMCompress 2.1 is a an automatic polygon reduction tool, included in the PolyWorks integrated line of software tools for building 3-D polygonal models from 3-D digitizer data (see the Web at: http://www.innovmetric.com./anglais/page_ed.html#IMEdit or http://www.innovmetric.com/anglais/page_pol.html#POLYWORKS). It guarantees true 3-D tolerances between compressed and original models. It also automatically preserves local topology and surface edges of models. It is completely automatic (may be executed as a command-line library, or it may also be used interactively when called up from IMEdit system). It requires only one input parameter, which is the desired reduction level. Reduction levels are specified as true 3-D tolerances in model units, or as the desired numbers of triangles. Several reduction levels may be specified in a single pass of the program for generating LOD representations.
It supports preservation of textures (e.g. pictorial info, surface color) and materials. The IMTexture module generates texture maps for compressed models from the color per vertex information of the original model (see Figure 29).

5.1.2.4. HP DirectModel. The DirectModel toolkit has been recently announced by Hewlett Packard (see the Web at: http://hpcc920.external.hp.com/wsg/ prod-

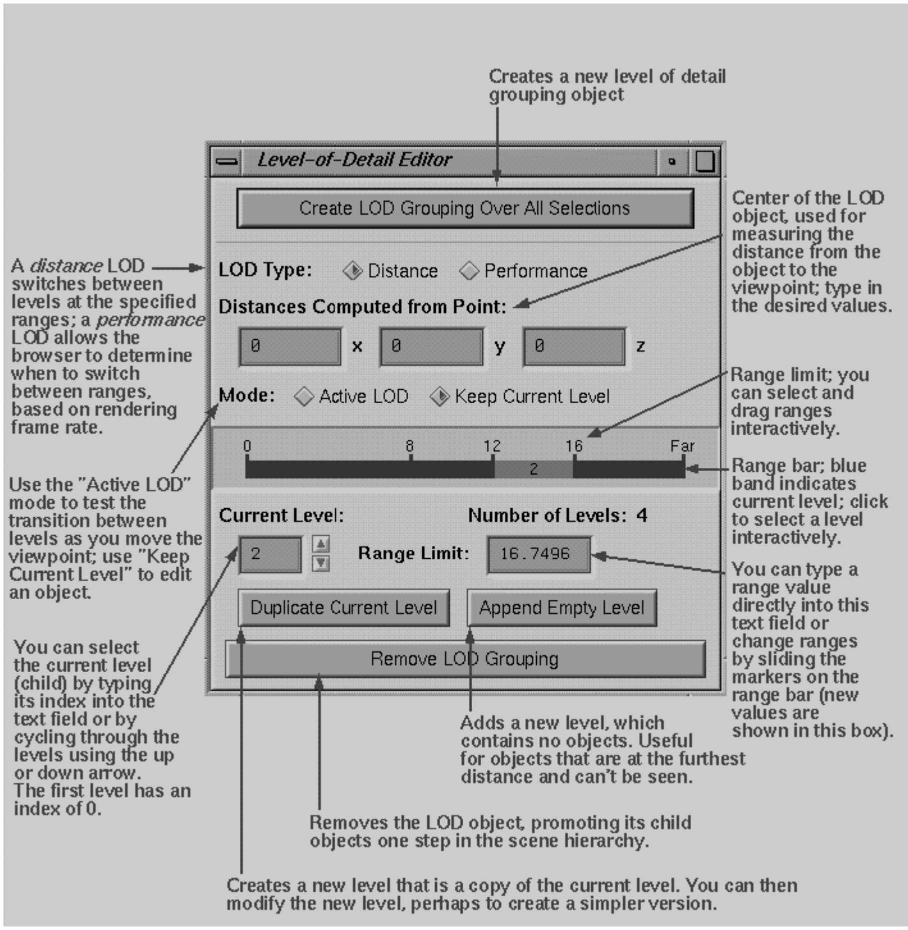


Figure 27: *The graphic interface of the Level of Detail Editor of Cosmo Worlds (image courtesy of SGI inc.).*

ucts/grfx/dmodel/overview.htm). DirectModel is a toolkit for creating technical 3D graphics applications. Its primary objective is to provide the performance necessary for interactive rendering of large 3D geometry models. DirectModel is implemented on top of traditional 3D graphics APIs and provides the application developer with high-level 3D model management, advanced geometry culling and simplification techniques.

The Direct Model toolkit provides four simplifiers: Octree-based spatial hierarchy (based on the clustering approach), convex hull (useful for simplifying models that could be viewed from a distance), bounding box (the object is replaced by a coloured bounding box) and tri-stripper (converts disjoint collections of triangles into tri-strips for efficient rendering). They can be used to simplify polygonal models in a pre-processing stage prior to rendering the model. A mesh decimator is also in the process of being added to the toolkit.

An interesting feature of DirectModel is the extensibility of the toolkit. Other simplifiers can be plug directly into the toolkit by sub-classing the simplifier base class, and can be used from within the application programs.

5.1.3. Controlling color and other mesh attributes

The preservation of attribute discontinuities during mesh simplification is an important characteristic, that only a few papers have taken into account carefully [86, 57, 8, 98]. The control of the pure geometric approximation is not sufficent in many applications to assure that the required accuracy is fulfilled.

The color/pictorial information defined over the mesh (e.g. by setting explicitely the color of each vertex of a high resolution mesh, as it is in the case of meshes returned by range scanners) can been preserved during simplification by building a mapping between the original mesh vertices and the simplified mesh faces [98]. Once the mesh has been simplified, a texture is built for each face using the colors of the associated removed vertices. The simplified face textures may

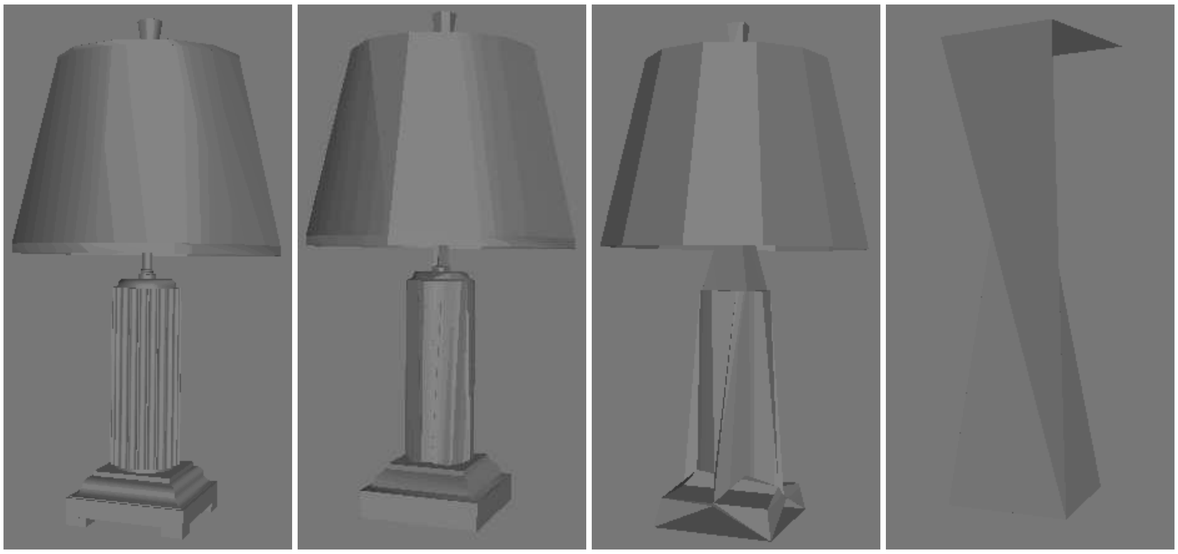


Figure 28: *Mesh simplification with IBM Interaction Accelerator, original object on the left (image courtesy of IBM inc.); mesh sizes: 10,108 facets, 1,383 facets), 474 facets, and 46 facets.*

be stored in a very efficient manner in a single texure map (see Figure 29).

Another type of mesh attribute can be the sampling of a field over the mesh vertices (e.g. the value of a physical variable like temperature, potential, pressure, etc.). In order to use the field values in visualization (e.g. mapping field values to color, or by computing iso-lines), a simplification code must take into account the value of the field while reducing the complexity of the mesh [94, 57].

5.1.4. Progressive transmission of geometric data

The diffusion of three-dimensional web applications, and the distribution of 3D data on the net, require that geometries should be transferred as fast as possible. The LOD representation gives a incomplete solution to this problem, because its main design goal is to speedup visualization. If the problem is how to transfer effiently geometrical data, a different answer can be the design of *progressive transmission* methodologies, analogously to how images are managed. If we send a 3D mesh on a (slow) communication line, one would like to receive, and render locally, progressively better approximations of the model.
A first, naive possibility is to transmit a sequence of LOD models, in order of increasing complexity. But in this way the information transferred is redundant, and to be able to visualize a single model users must wait until the entire mesh level has been completely transmitted.

A first proposal for a real progressive trasmission of 3D geometries was proposed on top of the *Progressive Meshes* scheme [57]. A very rough approximation is sent firstly, and then a sequence of update records are transferred, which enable the remote receiver to refine progressively the initial rough mesh. At each instant of time, the remote user is able to visualize a current mesh which is locally reconstructed on the base of the update records.
A similar approach can be adopted with all incremental simplification methods based on local updates and global error control.

5.1.5. Variable resolution modeling

5.1.5.1. Zeta - Resolution Modeling. **Zeta** is a prototypal system which supports the construction of variable resolution representations out of a multiresolution mesh; it gives a unified solution to both the selective refinement problem and its reverse, selective simplification[15]. Zeta has been designed according to a precise constraint: selective updates (either refinement or simplification) have to be operated as incremental updates over a variable resolution mesh. Moreover, the proposed approach must show an interactive response time.
Instead than simply taking into account the viewing space "impact" of the represented data, a variable resolution representation of an object can be designed following user's interpretation of the visual importance of different surface areas: e.g., higher precision is generally needed in the representation of the face than the chest of a cyber-actor.
For this reason the considerable similarity between *resolution modeling* and *shape modeling* has been pointed out in [15], in the sense that both should have been fulfilled through a tight interaction with the user. While the construction of the multiresolution representation is a process which can be simply made in an automatic and unattended way [11, 57], the

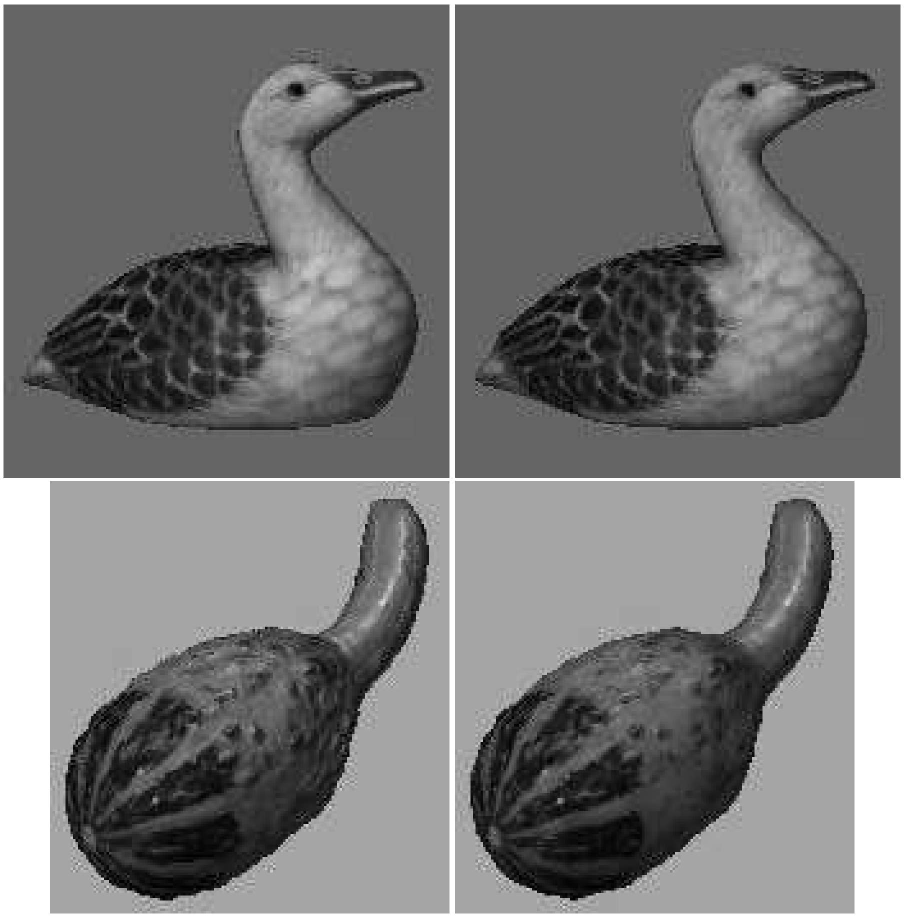


Figure 29: *Mesh simplification with IMCompress, the original meshes with colored vertices are on the left (top: 166 278 faces; bottom: 419 616 faces); the simplified ones, with texture mapped color, are on the right (in both cases, a 512x512 texture is mapped onto a 1000 faces model). Images courtesy of* InnovMetric Software *inc.*

resolution modeling phase generally involves a semantic interpretation of the data which cannot be fulfilled without human intervention.
Given this framework, the rationale of the Zeta prototype is to validate a new data structure and algorithms which allow the user to directly "model resolution". Standard CAD systems provide tools to assist users in the design of "shapes", but none of them actually provides the tools needed to manage resolution. The modeling framework can be conceived as composed of three separate stages: a first stage where the "shape" is designed in full detail (*shape modeling*), a second stage where the multiresolution representation is built automatically by means of recent surface simplification methods (*multiresolution encoding*), and a the third stage, *resolution modeling*, which supports the construction of different instances of the the input shape which are characterised by variable resolutions/details. Zeta implements a new approach to the last stage which is highly general, user-driven and independent on the particular simplification approach used to build the multiresolution representation. The approach proposed is based on the hypertriangulation multiresolution representation proposed in [19]. A set of kernel functionalities support: *a*) topological walking on the surface, *b)* efficient extraction of fixed resolution representation, *c*) unified management of selective refinement and selective simplification, *d*) easy composition of the selective refinement/simplification actions, *e*) no cracks in the variable resolution mesh produced, *f*) multiresolution mesh editing functionalities, and *g*) interactive response times.

The interface of *Zeta* is presented in Figure 30. The first release of *Zeta* is available on the World Wide Web at address http://miles.cnuce.cnr.it/cg/swOnTheWeb.html. In the snapshot, a mesh has been extracted at a low resolution (and rendered wire frame). Then, some refinement actions have been operated in the areas of the head and the arms (and the result is shaded in the figure).

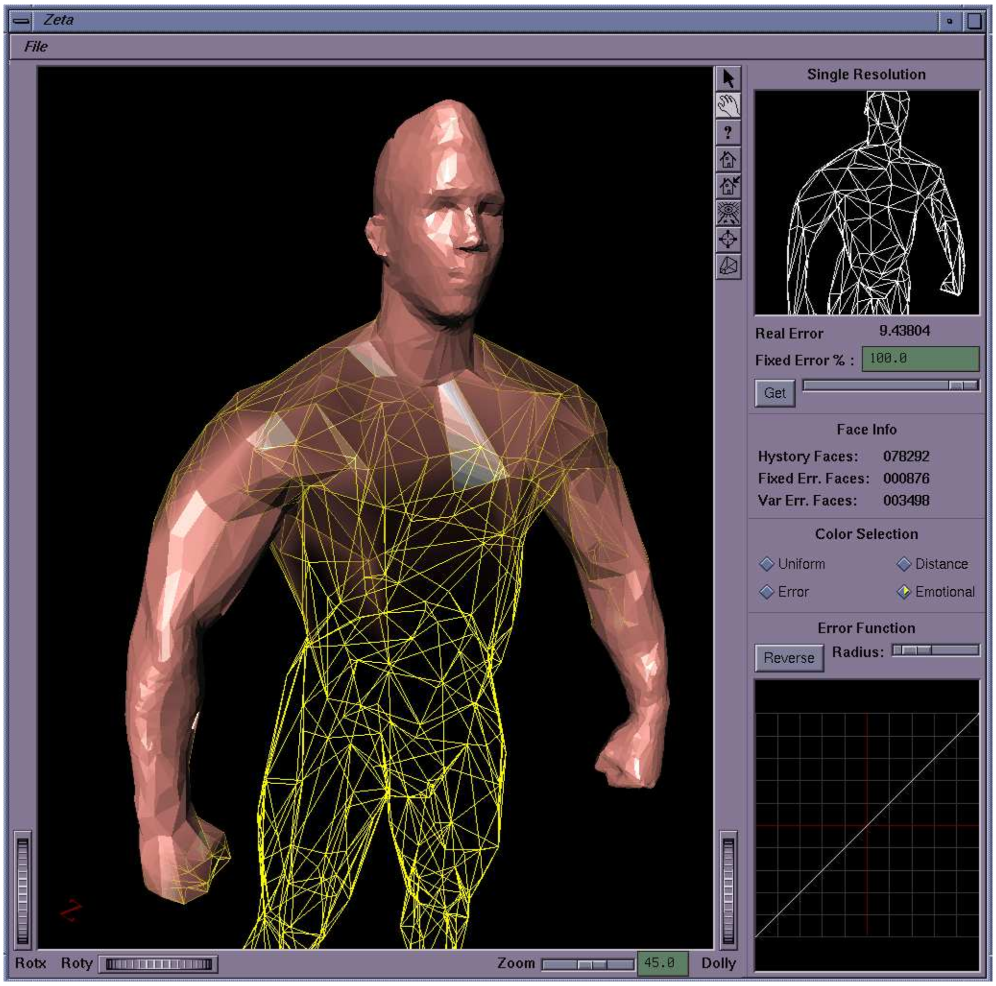


Figure 30: *The interface of* Zeta*, a resolution modeling prototypal system; the resolution modeling session started on a low resolution representation of a mesh distributed by Cyberware Inc. (the GMM model consists of 78k facets); then we selectively refined it (few composed refinement actions were operated in the areas of the head and the arms).*

Because the mesh refinement/simplification process is user-driven, a complex dialog session has to be managed. The *user* has control over: the action (*refinement* or *simplification*) that has to be operated; the *focus point* p_f on the current mesh, selected by picking the surface; the current *radius* r, which identifies the area size (to be refined/simplified) surrounding p_f on the current mesh; and the function *Err()* which determines, for each element of the mesh, the required increase/decrease of precision by taking into account the distance of the element from p_f. The *Err()* function can be interactively designed by the user using the lower-right graphic area in the Zeta window (Figure 30 and Figure 31.d and 31.e, where the default *Err()* function has been modified).
According to user inputs, the system modifies locally the current mesh, by decreasing or increasing the mesh precision in the mesh subsection of radius r and centred in p_f.

Six different stages operated on the rabbit mesh are presented in Figure 31, to highlight some *Zeta*'s capabilities. The mesh colors in the second, third and fourth clips represent the error of each mesh face (using a color ramp from blue to green).

5.1.5.2. Interactive Multiresolution Mesh Editing – CalTech. The Caltech Multi-Res Modeling Group has developed an *Interactive Multiresolution Mesh Editing* system [111].

The user manipulates high resolution geometry as if working in a patch based system, with the additional benefit of hierarchical editing semantics. Using sophisticated adaptive subdivision techniques coupled with lazy evaluation, a scalable editing system has been defined, currently running on both PC and SGI workstation hardware.
Through the use of subdivision and smoothing techniques

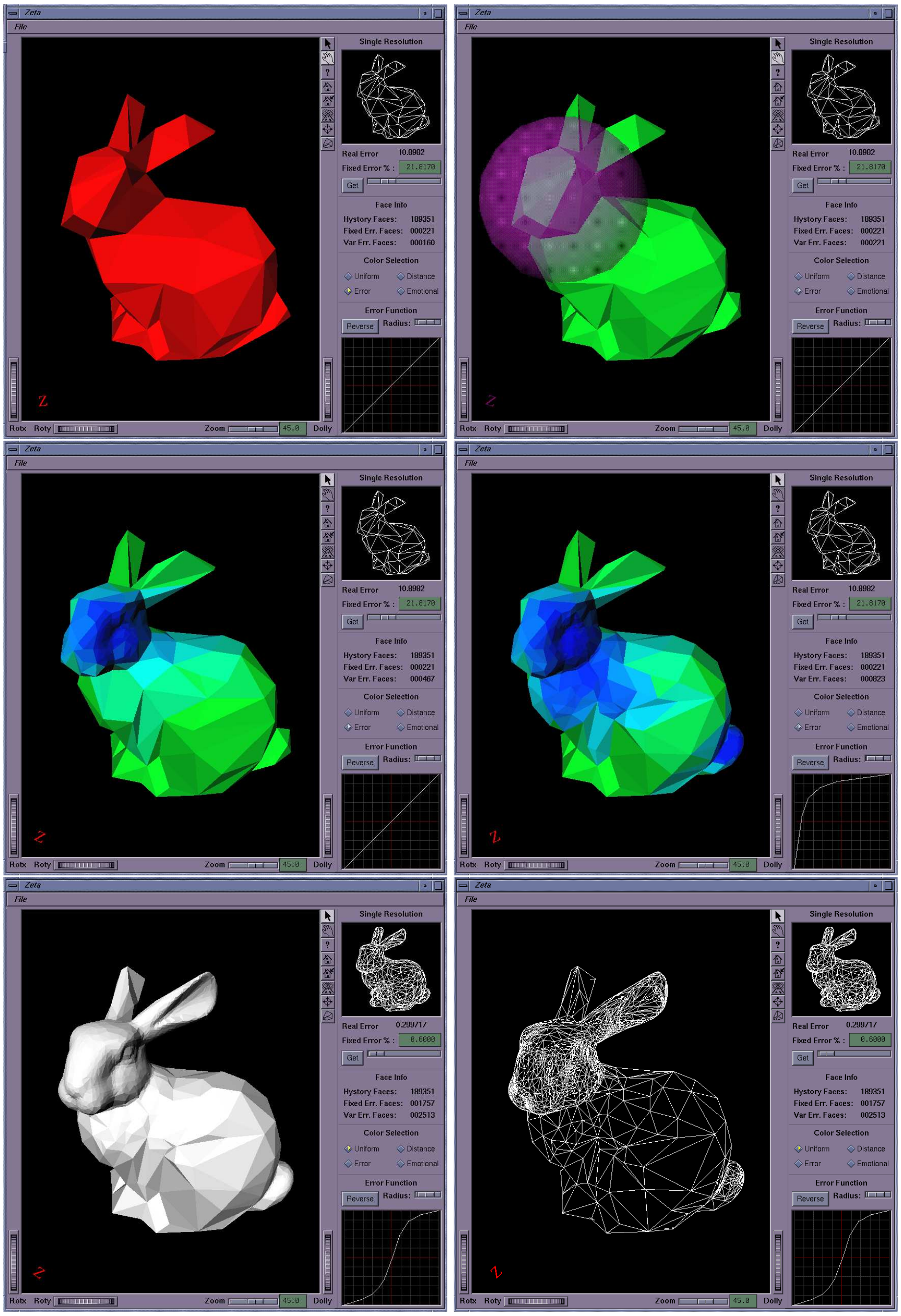


Figure 31: Some subsequent stages of a resolution modeling session: (a) selection of the initial fixed error mesh; (b) selection of the radius used (c) in the following selective refinement on the rabbit eye; (d) the *Err* function has been modified and two more selective refinements have been operated on the cheek and the tail; (e,f) shaded and wire frame rendering of the final variable resolution mesh.

the system supports large scale smooth edits as well as tweaking detail at the individual vertex level, while maintaining a concise internal representation. No smoothness constraints need to be maintained explicitly and all the usual artifacts of traditional patch based modelers, often visible during animation, are avoided.

Subdivision is a natural extension of the existing patch-based surface representations. At the same time subdivision algorithms can be viewed as operating directly on polygonal meshes, which makes them a useful tool for mesh manipulation.

Combination of subdivision and smoothing algorithms of Taubin [102] allows to construct a set of algorithms for interactive multiresolution editing of complex meshes of arbitrary topology. Simplicity of the essential algorithms for refinement and coarsification allows to make them local and adaptive, considerably improving their efficiency.

The images in Figure 32 show an example of an edited mesh obtained using this Multiresolution Editing system. The original is on the left (courtesy Venkat Krischnamurthy). The edited version on the right illustrates large scale edits, such as armodillo's belly, and smaller scale edits such as his double chin. All edits were performed at about 5 frames per second on an Indigo R10000 Solid Impact.

5.2. Applications to GIS and flight simulation

Terrain modeling in the context of geographical information systems, and flight simulators has probably been the first application of the simplification and multiresolution technology. Early techniques for refinement and simplification [38, 66] were meant as compression tools, designed with the explicit purpose of building compact approximated representations of terrains through triangular meshes (called Triangular Irregular Networks (TINs) in the GIS literature), starting from huge grids of regularly sampled data. Also most existing multiresolution models were initially designed to represent terrains [7, 19, 10, 23, 24, 34, 37, 41, 67, 93, 82].

On this basis, some prototypal systems that support either GIS operations, or visual flythrough, or both, have been developed, and proposed in the literature. Some such systems are reviewed in the following.

5.2.1. Multiresolution GIS

The application of multiresolution meshes to terrain processing in the context of GIS has been studied thoroughly by the joint group at DISI-University of Genova, and IMA-National Research Council. One prototypal system has been developed, and another one is under development.

5.2.1.1. HTIN. The HTIN (Hierarchical Triangular Irregular Network) system is based on the hierarchical Delaunay triangulation proposed in [34]. This system supports several interactive tasks that can be controlled through a unified graphical user interface.

The system contains an algorithm for building a multiresolution mesh through refinement, on the basis of an initial grid at high resolution, and of a sequence of error tolerances provided by the user. Once the hierarchical triangulation has been built, the system allows the user to perform interactively the following operations at a user-defined accuracy:

- extraction and visualization of a domain-uniform terrain mesh;
- extraction and visualization of contour lines;
- computation of overlays between a terrain mesh and other triangulated maps;
- visibility computations, such as horizon, and viewshed from a given point of view.

5.2.1.2. VARIANT. VARIANT (VAriable Resolution Interactive Analysis of Terrains) is a new prototype built on top of the explicit MT data structure. The current version includes just an algorithm to build an explicit MT through refinement from an initial grid of data at high resolution, and a viewer, which supports the real-time extraction and visualization of a terrain mesh. The viewer can extract a mesh either at domain-uniform resolution, or at domain-variable resolution, according to a threshold function increasing with distance from a viewpoint. The systems allows the user to interactively move the viewpoint in space, and it reacts by providing the mesh corresponding to each viewpoint position in real time. The system also supports visual flythrough controlled through the mouse. The GUI of VARIANT with a perspective view of a terrain is shown in Figure 33.

An extension of VARIANT with algorithms for general geometric queries, which are meant to provide a kernel for more complex GIS operations, is described in [30], and it is currently under development.

5.2.2. Flight simulation

Serious terrain datasets used in flight simulators can be really huge. Even simple simulators used for videogames can deal with considerably large terrain models. Most commercial products for flight simulation encode terrains using regular square grids, which can be easily partitioned on disk, and often adopt some naive multiresolution technique, such as performing grid subsampling for areas far from the viewpoint.

More advanced techniques, based on models discussed in Chapter 4, have been included so far only in academic prototypal systems. The models and algorithms proposed in [23, 19], and the VARIANT system described above achieve extraction times that might support frame rate on moderately large datasets (order 10^5 vertices). However, all such structures have been designed for the primary memory, while it

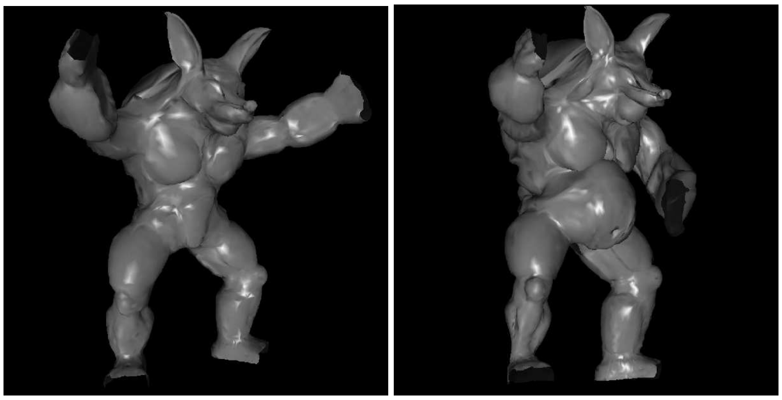


Figure 32: *Armodillo mesh: the original is on the left and the edited version is on the right (images courtesy of Caltech Multi-Res Modeling Group).*

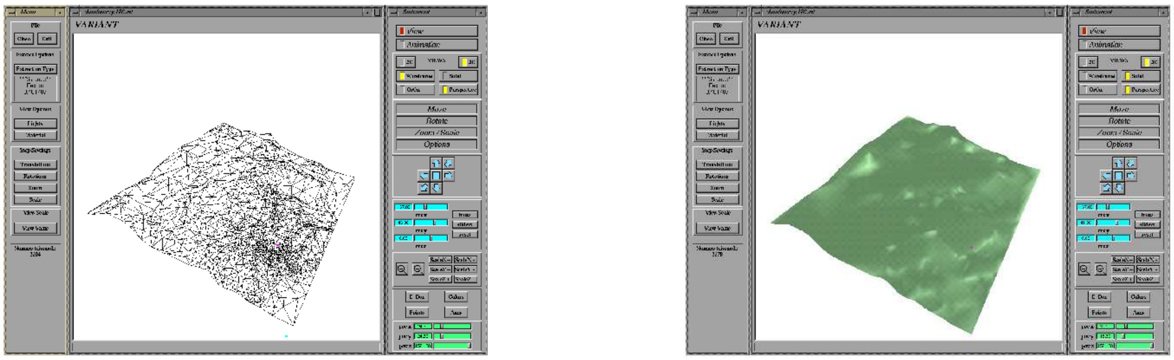


Figure 33: *The top view of a terrain mesh at variable resolution extracted by VARIANT (left); the GUI of VARIANT, with a perspective view of the same mesh (right).*

is still not clear yet how to support the construction, secondary storage, and efficient retrieval of models based on irregular triangulations for huge datasets. The compactness of data structures, and the management of secondary memory are crucial issues to handle larger datasets, e.g., about order $10^6 - 10^8$ vertices. This is perhaps one of the hottest research challanges in the field of multiresolution modeling.

Interactive flythough on huge dataset can be achieved by using multiresolution meshes based on hierarchices of right triangles. Indeed, due to the regular structure of such models, it is easy to design eficient data structures and procedure to store them on disk, and retrieve them efficiently. Two prototypal systems based on such models have been developed independently, which will be described in the following.

5.2.2.1. Vulture – CS Arizona. The Vulture system has been developed at the Department of Computer Science of the University of Arizona (see the web at http://www.cs.arizona.edu/people/will/rtin.html), and it is based on the Right Triangular Irregular Network (RTIN) described in [37].

The system allows the user to load a terrain grid, and it builds a hierarchy of right triangles in an initialization step (which takes a few seconds for datasets smaller than 10^5 vertices). The accuracy of each triangle in the hierarchy is evaluated and explicitly stored.

The system offers a GUI with functionalities similar to the VARIANT system described above. The user can control viewpoint position and view direction through mouse drag on a top view of the terrain. The system can perform extraction and visualization of a terrain mesh at variable resolution, according to a threshold function increasing with distance from the viewpoint. The extraction algorithm achieves real time performances on moderately large datasets (smaller than 10^5 triangles).

Apparently, the current data structure needs further optimization, since storage requirements for moderately large datasets is already quite high, which makes it not applicable in practice to huge datasets.

5.2.2.2. Display algorithm – Georgia Tech. Another system based on hierarchies of right triangles has been developed at the Georgia Institute of Technology. The system relies on an extremely compact data structure encoding only the raw grid ov elevation values, and on a bottom-up algorithm for extracting meshes at variable resolution. This algorithm can achieve frame rate on top level graphics workstations (SGI Onix RealityEngine^2) for datasets of huge size (about 10^7 vertices).

The basic principle on which the extraction algorithm relies is the recursive merge of adjacent triangles into their parents, whenever the loss of accuracy can be accepted. Merging two triangles means removing a vertex from the mesh. The error made in merging is estimated on-the-fly as the difference between the actual elevation of the removed vertex, and its interpolated elevation in the parent triangle. If the projection in screen space of such a difference is not larger than a given threshold, then merge can take place, otherwise the vertex must be preserved in the output mesh. Allowed merge operations are further constrained to vertex dependencies, that are traced through the implicit hierarchy to warrant that the result be a conforming mesh.

Error evaluation, though approximated, is an expensive task to be performed at each vertex of the original grid. For this reason, other heuristics are used, based on a block decomposition of the dataset, which are aimed to either coarsen or refine large blocks of data by a rough evaluation of their accuracy. In this way, most vertices are either discarded from, or included into the extracted mesh during a preliminary traversal based on block decomposition, while the error estimation must be performed only at the vertices whose "importance" cannot be evaluated in the context of block decomposition.

5.3. Applications to volume data

Volume datasets used in current applications have a common problem, the size of the datasets, which affects both storage requirements and visualization times. Therefore, interactive image generation from very large datasets could be not feasible, even with the use of fast graphic hardware and parallelism. Approximate rendering algorithms can give only a partial solution to this problem. It is also possible to manage data complexity by adopting an *approximate representation* of the dataset. The latter approach is more general because it remains totally independent of the rendering approach. The methodology in this case is therefore to work on *data simplification* rather than on *graphics output simplification*. A comparison between these two approaches is given in [16].

5.3.1. Simplification algorithms

The main approach to build an approximate representation of a tetrahedral dataset is to choose a subset of the original vertices and to build a new triangulation of (almost) the same domain. A naive *random subsampling* as proposed in [109] has serious drawbacks: there is no control on the accuracy of the simplified mesh; the technique is not adaptive, i.e. data density cannot vary over different regions of the domain.

An approach to the representation of regular volume datasets based on the use of a hierarchical recursive partition (an octree-like scheme) has been proposed in [108]. Each node is obtained by recursive subdivision: it holds a basis function to reconstruct the field, as well as a measure of both error and importance factors, which are used for selective traversal of the tree. The method cannot be extended to irregularly distributed data. Using such a structure as a *LoD* representation, by considering each tree level as a separate layer, is equivalent to use subsampling. A *multiresolution* representation is also possible, by selecting nodes at different levels, but the field may result discontinuous across different levels, thus causing unpleasant effects (e.g., aliasing in direct volume rendering, and cracks in isosurfaces).

Most of the adaptive methods that try to select the smallest set of point that approximate a dataset within a given error, developed for the simplification of 2D surfaces, can be extended to simplify volume data, but only a few experiments have been made [12, 49, 87].

A *LoD* representation based on tetrahedral decomposition was proposed in [12]. Independent simplified representations of a volume dataset at different levels of approximation were built by adopting a refinement technique.
This approach has been extended to provide a multiresolution model for volume data based on tetrahedral meshes and data simplification [14]. Two methods for building approximated tetrahedral meshes are proposed: a top-down method

that refines a coarse initial mesh by iteratively inserting vertices, and a bottom-up method that simplifies an initial mesh at the highest resolution by iteratively discarding vertices. Since both methods are based on iterative local mesh modifications, each of them produces a fine-grained sequence of meshes at increasingly finer (respectively, coarser) resolution. In other words, a high number of different tetrahedral meshes at different resolutions are obtained on the basis of a moderate number of tetrahedra, namely all tetrahedra that appear during successive updates. Such tetrahedra can be stored in a compact representation of a multiresolution model which supports fast on-line extraction of a mesh at arbitrary resolution. A multiresolution visualization system *TAn (Tetrahedra Analyser)* has been developed. The TAn prototype is available in the public domain at: http://miles.cnuce.cnr.it/cg/swOnTheWeb.html.

5.3.2. Multiresolution management

Many approaches have recently been proposed for the multiresolution management of surfaces, while multiresolution volume data management is still under developed. First steps in this direction were made in [17, 14], and further research has still to be done.

6. Uncovered subjects

Due to space and time limits in preparing these notes, we had to leave out some subjects that will be briefly addresses in this chapter.

6.1. CAD surfaces

Simplification methods, and multiresolution meshes need further specification to be applied to parametric surfaces in the context of CAD modeling systems.

The further problem, in this application, is that not only each parametric patch, but also its computational domain is approximated. Therefore, the simplification process must take care of preserving trimming curves, and of conformity between meshes representing different patches adjacent along them. This problem has consequences also in the construction of multiresolution meshes, since models representing each parametric patch must be welded properly along trimming curves at all levels of resolution.

A simplification algorithm for parametric patches is presented in [64], which is used in [65] to build a multiresolution mesh based on linear sequences. A technique for building a multiresolution mesh based on an explicit MT is presented in [31].

6.2. Non-uniformly dimensional meshes

Some work has been done in the literature on simplification algorithms and multiresolution meshes that permit to work with meshes that are not uniformly domensional, e.g., to describe scenes in the 3D space containing not only objects bounded by surfaces, but also “sticks” and isolated points. There are at least two such models, proposed in [90], and [80], respectively. The field of application of such models is mainly in modeling complex objects and scenes composed of many parts in virtual reality contexts.

In Chapter 3 we reviewed some such work from the point of view of simplification algorithms, while we did not review the corresponding multiresolution models.

A strong formalization of such models has not been given yet in the literature. The MT framework “as is” does not capture them, since it is based on regular complexes. On a different perspective, a first formalization of non-uniformly dimensional multiresolution models in the plane has been proposed in [84], for the special case of modeling geographic maps. By exploiting the principles presented in such a paper, it should be not difficult to extend the MT to deal with higher dimensional cases.

6.3. Multiresolution models based on wavelets

A relevant part of the literature on multiresolution is regarding methods based on *wavelets*. Wavelets are a mathematical tool for hierarchically decomposing a function in terms of a coarse overall shape, and of progressively finer details, encoded as a sequence of coefficients.

Wavelets have been applied in many contexts, such as images, curves, surfaces, and volume data. For the case of surfaces, wavelet methods usually rely on regular recursive subdivisions of a domain, such as the quaternary triangulation discussed in 4.2.1.2.

In Chapter 3 we discussed some simplification methods based on wavelets [36, 70, 71], and in Chapter 5 we described an application of wavelets to multiresolution modeling [111].

Further references on wavelets are the following: tutorials papers [74, 100, 101]; multiresolution representation of manifold surfaces [8, 42]; applications to volume data [77, 78, 79, 46, 107].

6.4. Image pyramids

In the computer vision literature, there have been many proposals of multiresolution methods for image analysis, based on hierarchical decompositions of the image space. The literature on this subject is often referred to as *image pyramids*.

From the point of view of this tutorial, an image pyramid is essentially a LOD model for images, where interference links between consecutive levels are maintained. In the literature, many different pyramids have been proposed, from the simple case of regular grids at different resolution, to irregular tessellation based, e.g., on Voronoi diagrams.

An excellent reference for this subject is the book [60].

6.5. Range images

Range images, acquired with range sensors such as laser scanners, or structured lights, have a structure similar to 2D height fields, such as terrains. The difference, in this case, is that a range image contains discontinuities (jumps) that must be preserved by algorithms that attempt to reconstruct the framed scene.

In the literature, there are some applications of refinement techniques to reconstruct surfaces from range images. In [9], image discontinuities are detected first, which are imposed as constraints to compute an initial triangulation of the image plane; such a triangulation is further refined by inserting vertices on-line until the approximation of data achieves a given accuracy. The resulting triangular mesh is an approximation of the scene; discontinuities are modeled by doubled edges. In [81], an unconstrained refinement algorithm is applied first, which produces an adaptive fragmentation of the scene; a relaxation algorithm is applied next that merges nearly coplanar triangles, thus eliminating false jumps and creases; the result is a polygonal mesh reconstructing maximal planar faces of objects, while preserving jumps and creases of the surface. This method can be generalized to handle curved objects by using higher polynomials instead of linear functions at each facet of the subdivision.

References

1. M.E. Algorri and F. Schmitt. Mesh simplification. *Computer Graphics Forum (Eurographics'96 Proc.)*, 15(3):78–86, 1996.
2. C. Andujar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Sole'. Automatic generation of multiresolution boundary representations. *Computer Graphics Forum (Eurographics'96 Proc.)*, 15(3):87–96, 1996.
3. C. L. Bajaj and D.R. Schikore. Error bounded reduction of triangle meshes with multivariate data. *SPIE*, 2656:34–45, 1996.
4. M. Bertolotto, E. Bruzzone, L. De Floriani, and E. Puppo. Multiresolution Representation of Volume Data through Herarchical Simplicial Complexes. In *Aspects of Visual Form Processing*, pages 73–82. World Scientific, Singapore, 1994.
5. M. Bertolotto, L. De Floriani, and P. Marzano. Pyramidal simplicial complexes. In *Proceedings 4th International Symposium on Solid Modeling*, pages 153–162, Salt Lake City, Utah, U.S.A., May 17-19 1995. ACM Press.
6. J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–356, 1988.
7. Peter J.C. Brown. A fast algorithm for selective refinement of terrain meshes. In *COMPUGRAPHICS 96*, pages 70–82. GRASP, December 1996. A longer version is available as Technical Report No. 417, Computer Laboratory, Cambridge University, CB2 3QG, UK, February 1997.
8. A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '96), ACM Press*, pages 91–98, Aug. 6-8 1996.
9. X. Chen and F. Schmitt. Adaptive range data approximation by constrained surface triangulation. In B. Falcidieno and T.L. Kunii, editors, *Modeling in Computer Graphics*, pages 95–114. Spriger-Verlag, Berlin Heidelberg, 1993.
10. Z.T. Chen and W.R. Tobler. Quadtree representation of digital terrain. In *Proceedings Auto Carto London*, pages 475–484, London, 1986.
11. A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. Technical Report C96-021, CNUCE – C.N.R., Pisa, Italy, July 1996. (to appear on *The Visual Computer*, 1997).
12. P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and rendering of volume data based on simplicial complexes. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 19–26. ACM Press, October 17-18 1994.
13. P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Multiresolution Modeling and Visualization of Volume Data. Technical Report C95-22, Istituto CNUCE – C.N.R., Pisa, Italy, July 1995.
14. P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Multiresolution Representation and Visualization of Volume Data. Technical Report C97-05, Istituto CNUCE – C.N.R., Pisa, Italy, January 1997.
15. P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno. Resolution modeling. Technical Report C97-02, CNUCE – C.N.R., Pisa, Italy, January 1997.
16. P. Cignoni, C. Montani, D. Sarti, and R. Scopigno. On the optimization of projective volume rendering. In P.Zanarini R. Scateni, J.J. van Wijk, editor, *Visualization in Scientific Computing 1995*, pages 58–71. Springer KG, Wien, 1995.
17. P. Cignoni, C. Montani, and R. Scopigno. Magic-Sphere: an insight tool for 3D data visualization. *Computer Graphics Forum*, 13(3):317–328, 1994. (Eurographics '94 Conf. Proc.).
18. P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. Technical Report 97-08, Istituto CNUCE – C.N.R., Pisa, Italy, June 1997.

19. P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer*, 13:(in press), 1997. (A preliminary version appeared on "Scientific Visualization '95", Proc. Inter. Symposium, World Scientific, pp.50-68).

20. P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. Technical Report B4-01-01-96, I.E.I. – C.N.R., Pisa, Italy, January 1996.

21. J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *Computer Graphics Proc., Annual Conf. Series (Siggraph '96), ACM Press*, pages 119–128, Aug. 6-8 1996.

22. P. Criscione, C. Montani, R. Scateni, and R. Scopigno. DiscMC: an interactive system for fast fitting isosurfaces on volume data. In M. Gobel, J. David, P. Slavik, and J. van Wijk, editors, *Virtual Environments and Scientific Visualization '96*, pages 178–190. Springer Computer Science, Wien, 1996.

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

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

25. L. De Floriani, B. Falcidieno, C. Pienovi, and G. Nagy. A hierarchical data structure for surface approximation. *Computers and Graphics*, 8(2):475–484, 1984.

26. L. De Floriani, G. Gattorna, P. Marzano, and E. Puppo. Spatial queries on a hierarchical terrain model. In *Proceedings 6th International Symposium on Spatial Data Handling*, pages 819–834, Edinburgh (UK), September 5-9 1994.

27. L. De Floriani, P. Magillo, and E. Puppo. Building and traversing a surface at variable resolution. In *Proceedings IEEE Visualization 97*, Phoenix, AZ (USA), October 1997. (to appear).

28. L. De Floriani, P. Magillo, and E. Puppo. A dynamic algorithm for extracting meshes from a multi-triangulation. Technical report, Department of Computer and Information Sciences, University of Genova, 1997. (in preparation).

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

30. L. De Floriani, P. Magillo, and E. Puppo. Variant - processing and visualizing terrains at variable resolution. Technical report, Department of Computer and Information Sciences, University of Genova, 1997.

31. L. De Floriani, P. Magillo, and E. Puppo. Visualizing parametric surfaces at variable resolution. In *Proceedings 9th International Conference on Image Analysis and Processing*, Firenze (Italy), Sept. 1997. (to appear).

32. L. De Floriani, P. Magillo, E. Puppo, and M. Bertolotto. Variable resolution operators on a multiresolution terrain model. In *Proceedings Fourth ACM International Workshop on Advances in Geographic Information Systems*, pages 123–130, Rockville, Maryland, November, 15-16 1996.

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

34. L. De Floriani and E. Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14(4):363–411, October 1995.

35. L. De Floriani, E. Puppo, and P. Magillo. A formal approach to multiresolution modeling. In R. Klein, W. Straßer, and R. Rau, editors, *Theory and Practice of Geometric Modeling*. Springer-Velrag, 1997. (to appear).

36. M. Eck, T. De Rose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Computer Graphics Proc., Annual Conf. Series (Siggraph '95), ACM Press*, pages 173–181, Aug. 6-12 1995.

37. W. Evans, D. Kirkpatrick, and G. Townsend. Right triangular irregular networks. Technical Report 97-09, University of Arizona, May 1997.

38. R.J. Fowler and J.J. Little. Automatic extraction of irregular network digital terrain models. *ACM Computer Graphics (Siggraph '79 Proc.)*, 13(3):199–207, Aug. 1979.

39. T.A. Funkhouser and C.H. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex environment. In *Computer Graphics Proc., Annual Conf. Series (SIGGRAPH 93)*, pages 247–254. ACM Press, 1993.

40. M Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '97), ACM Press*, 1997. (to appear).

41. D. Gomez and A. Guzman. Digital model for three-dimensional surface representation. *Geo-Processing*, 1:53–70, 1979.

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

43. OpenInventor Architecture Group. *Inventor Mentor: OpenInventor Reference Manual*. Addison Wesley, 1994.

44. A. Guéziec. Surface simplification with variable tolerance. In *Second Annual International Symposium on Medical Robotics and Computer Assisted Surgery*, pages 132–139, Baltimore, MD, November 1995.

45. A. Guéziec. Surface simplification inside a tolerance volume. Technical Report RC 20440, I.B.M. T.J. Watson Research Center, 1996.

46. B. Guo. A multiscale model for structured-based volume rendering. *IEEE Trans. on Visualization and Computer Graphics*, 1(4):291–301, December 1995.

47. M.J. De Haemer and M.J. Zyda. Simplification of objects rendered by polygonal approximations. *Computers & Graphics*, 15(2):175–184, 1991.

48. B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11(2):197–214, 1994.

49. B. Hamann and J.L. Chen. Data point selection for piecewise trilinear approximation. *Computer Aided Geometric Design*, 11:477–489, 1994.

50. T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel-based object simplification. In *IEEE Visualization '95 Proceedings*, pages 296–303. IEEE Comp. Soc. Press, 1995.

51. T. He, L. Hong, A. Varshney, and S. Wang. Controlled topology simplification. *IEEE Trans. on Visualization & Computer Graphics*, 2(2):171–183, 1996.

52. D.J. Hebert. Symbolic local refinement of tetrahedral grids. *Journal of Symbolic Computation*, 17:457–472, 1994.

53. D.J. Hebert and H.-J. Kim. Image encoding with triangulation wavelets. *Proceedings SPIE*, (2569(1)):381–392, 1995.

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

55. P. Heckbert and M. Garland. Survey of surface simplification algorithms. Technical report, Carnegie Mellon University - Dept. of Computer Science, 1997. (to appear).

56. P. Hinker and C. Hansen. Geometric optimization. In *IEEE Visualization '93 Proc.*, pages 189–195, October 1993.

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

58. Hugues Hoppe. View-dependent refinement of progressive meshes. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '97)*, 1997. (to appear).

59. Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '93)*, pages 19–26, 1993.

60. J.-M. Jolion and A. Rosenfeld. *A Pyramid Framework for Early Vision*. Kluwer Academic, Dordrecht (Netherlands), 1994.

61. A. D. Kalvin and R.H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE C.G.&A.*, 16(3):64–77, 1996.

62. A.D. Kalvin, C.B. Cutting, B. Haddad, and M.E. Noz. Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures. *SPIE Vol. 1445 Image Processing*, pages 247–259, 1991.

63. R. Klein, G. Liebich, and W. Straßer. Mesh reduction with error control. In R. Yagel and G. Nielson, editors, *Proceedings of Visualization '96*, pages 311–318, 1996.

64. R. Klein and W. Straßer. Large mesh generation from boundary models with parametric face representation. In C. Hoffmann and J. Rossignac, editors, *Proceedings 3rd ACM Symposium on Solid Modeling and Applications*, pages 431–440, 1995.

65. R. Klein and W. Straßer. Generation of multiresolution models from cad data for real time rendering. In R. Klein, W. Straßer, and R. Rau, editors, *Theory and Practice of Geometric Modeling (Blaubeuren II)*. Spinger-Verlag, 1997. (to appear).

66. J. Lee. Analysis of visibility sites on topographic surfaces. *International Journal of Geographic Information Systems*, 5(4):413–425, 1991.

67. P. Lindstrom, D. Koller, W. Ribarsky, L.F. Hodges, N. Faust, and G.A. Turner. Real-time, continuous level of detail rendering of height fields. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '96), ACM Press*, pages 109–118, New Orleans, LA, USA, Aug. 6-8 1996.

68. W.E. Lorensen. Marching through the visible man. In *Visualization '95*, pages 368–373. IEEE, 1995.

69. William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–170, July 1987.

70. Michael Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, Dept. of Computer Science and Engineering, U. of Washington, 1994.

71. Michael Lounsbery, Tony D. DeRose, and Joe Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, 1997.

72. K.L. Low and T.S Tan. Model simplification using vertex clustering. In *1997 ACM Symposium on Interactive 3D Graphics*, page to appear, 1997.

73. D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '97)*, 1997. (to appear).

74. S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. on Patt. Anal. and Mach. Intel.*, 11(7):674–693, 1989.

75. D. Moore and J. Warren. Compact isocontours from sampled data. In D. Kirk, editor, *Graphics Gems III*, pages 23–28. Academic Press, 1992.

76. H. Muller and M. Stark. Adaptive generation of surfaces in volume data. *The Visual Computer*, 9(4):182–199, 1993.

77. S. Muraki. Approximation and rendering of volume data using wavelet transforms. In *IEEE Visualization '92 Proc.*, pages 21–28, October 19-23 1992.

78. S. Muraki. Volume data and wavelet transforms. *IEEE C. G.& A.*, 13(4):50–56, July 1993.

79. S. Muraki. Multiscale volume representation by a dog wavelet. *IEEE Trans. on Vis. and Comp. Graph.*, 1(2):109–116, June 1995.

80. J. Popovic and H. Hoppe. Progressive simplicial complexes. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '97)*, 1997. (to appear).

81. E. Puppo. Segmentation/reconstruction of range images based on piecewise-linear approximation. In C. Braccini, L. De Floriani, and G. Vernazza, editors, *Image Analysis and Processing*, number 974 in Lecture Notes in Computer Science, pages 367–372. Springer-Verlag, 1995.

82. E. Puppo. Variable resolution terrain surfaces. In *Proceedings Eight Canadian Conference on Computational Geometry, Ottawa, Canada*, pages 202–210, August 12-15 1996.

83. E. Puppo. Variable resolution triangulations. Technical Report 12/96, Institute for Applied Mathematics, C.N.R., Genova (Italy), November 1996.

84. E. Puppo and G. Dettori. Towards a formal model for multiresolution spatial maps. In Max J. Egenhofer and John R. Herring, editors, *Advances in Spatial Databases*, number 951 in Lecture Notes in Computer Science, pages 152–169. Springer-Verlag, 1995.

85. M. Reddy. A survey of level of detail support in current virtual reality solutions. *Virtual Reality: Research, Development and Appl.*, 1(2):85–88, 1995.

86. M. Reddy. Scrooge: Perceptually-driven polygon reduction. *Computer Graphics Forum*, 15(4):191–203, 1996.

87. K.J. Renze and J.H. Oliver. Generalized unstructured decimation. *IEEE C.G.&A.*, 16(6):24–32, 1996.

88. R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum (Eurographics'96 Proc.)*, 15(3):67–76, 1996.

89. J. Rossignac, editor. *Geometric Simplification (ACM SIGGRAPH Course Notes No.35)*. ACM Press, 1996.

90. J. Rossignac and P. Borrel. Multi-resolution 3D approximation for rendering complex scenes. In B. Falcidieno and T.L. Kunii, editors, *Geometric Modeling in Computer Graphics*, pages 455–465. Springer Verlag, 1993.

91. H. Samet. *Applications of Spatial Data Structures*. Addison Wesley, Reading, MA, 1990.

92. H. Samet. *The design and Analysis of Spatial Data Structures*. Addison Wesley, Reading, MA, 1990.

93. Lori Scarlatos and Theo Pavlidis. Hierarchical triangulation using cartographics coherence. *CVGIP: Graphical Models and Image Processing*, 54(2):147–161, March 1992.

94. D. Schikore and C. Bajaj. Decimation of 2D scalar data with error control. Technical Report CSD-TR-95-005, CS Dept., Purdue University, 1995.

95. F. Schmitt, B.A. Barsky, and W.H. Du. An adaptive subdivision method for surface-fitting from sampled data. *Computer Graphics (SIGGRAPH '86 Proc.)*, 20(4):179–188, 1986.

96. W. Schroeder. Polygon reduction techniques. In *ACM Comp. Graph. Proc., Annual Conf. Series (Siggraph'95), Course Notes n. 30 (Advanced Techniques for Scientific Visualization)*, pages 1.1–1.14, Aug. 6-12 1995.

97. William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *ACM Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

98. Marc Soucy, Guy Godin, and Marc Rioux. A texture-mappping approach for the compression of colored 3d triangulations. *The Visual Computer*, (12):503–514, 1996.

99. Marc Soucy and Denis Laurendeau. Multiresolution surface modeling based on hierarchical triangulation. *Computer Vision and Image Understanding*, 63(1):1–14, 1996.

100. E.J. Stollnitz, T.D. DeRose, and D.H. Salesin. Wavelets for computer graphics: A primer, part 1. *IEEE Computer Graphics and Applications*, pages 76–84, May 1996.

101. E.J. Stollnitz, T.D. DeRose, and D.H. Salesin. Wavelets for computer graphics: A primer, part 2. *IEEE Computer Graphics and Applications*, pages 75–85, July 1996.

102. G. Taubin. A signal processing approach to fair surface design. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '95), ACM Press*, pages 351–358, Aug. 6-12 1995.

103. Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *ACM Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.

104. B. Von Herzen and A.H. Barr. Accurate triangulations of deformed, intersecting surfaces. *Computer Graphics (Siggraph 87 Proc.)*, 21(4):103–110, 1987.

105. *The Virtual Reality Modeling Language Specification - Version 2.0*, August 1996.

106. Josie Wernecke. *The Inventor mentor: programming Object-oriented 3D graphics with Open Inventor*. Addison Wesley, 1994.

107. R. Westermann. A Multiresolution Framework for Volume Rendering. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 51–58. ACM Press, October 17-18 1994.

108. J. Wilhelms and A. van Gelder. Multi-dimensional Trees for Controlled Volume Rendering and Compression. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 27–34. ACM Press, October 17-18 1994.

109. P.L. Williams. *Interactive Direct Volume Rendering of Curvilinear and Unstructured Data*. PhD thesis, University of Illinois at Urbana–Champaign, 1993.

110. J.C. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In R. Yagel and G. Nielson, editors, *IEEE Visualization '96 Proc.*, pages 327–334, 1996.

111. D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '97), ACM Press*, 1997. (to appear).