Variable Resolution Triangulations

E. Puppo
Istituto per la Matematica Applicata – Consiglio Nazionale delle Ricerche
Via de Marini, 6 (Torre di Francia) – 16149 Genova, ITALY
Email: puppo@ima.ge.cnr.it
URL: http://www.ima.ge.cnr.it/STAFF/puppo-en.html

July 8, 1998

Abstract

A comprehensive study of multiresolution decompositions of planar domains into triangles is given. A general model is introduced, called a *Multi-Triangulation (MT)*, which is based on a collection of fragments of triangulations arranged into a directed acyclic graph. Different decompositions of a domain can be obtained by combining different fragments of the model. Theoretical results on the expressive power of the MT are given. An efficient algorithm is proposed that can extract a triangulation from the MT, whose level of detail is variable over the domain according to a given threshold function. The algorithm works in linear time, and the extracted representation has minimum size among all possible triangulations that can be built from triangles in the MT, and that satisfy the given level of detail. Major applications of these results are in real-time rendering of complex surfaces, such as topographic surfaces in flight simulation.

Keywords: multiresolution decomposition, triangulation, Level of Detail, surface representation.

1 Introduction

Multiresolution geometric models support the representation and processing of geometric entities at different *Level of Detail (LOD)*. Such models are useful to handle geometric data efficiently, depending on specific application needs. The case of topographic surfaces is especially attractive for its impact on geographic information systems - where terrains are analyzed and manipulated in the context of computationally intensive problems; and virtual reality - where terrains are visualized in a real-time environment. For instance, rendering in flight simulation can be made faster by providing at each frame a representation whose resolution is decreasing gradually with distance from the viewpoint.

A *multiresolution model* is a model that can provide different representations, depending on the level of detail required. In order to be effective, such a model must fulfill requirements that have been outlined by several authors [4, 6, 7, 13, 16, 21], and can be summarized as follows:

- *continuity through domain:* there cannot be cracks or aliasing due to abrupt transition between different resolutions;
- *continuity across resolution:* abrupt changes should be avoided in changing a representation into another at a close LOD;
- *adaptivity:* the number of primitives in a representation for any given LOD should be as small as possible;
- *efficiency:* a representation at a given LOD must be extracted in short (real) time.

In the case of *polyhedral terrains*, a plane domain is tessellated with polygonal patches (usually triangles), and an elevation is given to each vertex; a linear patch is defined for each polygon, which interpolates the elevations of its vertices. Continuity through domain is achieved if the underlying subdivision is *conforming*, i.e., adjacent regions share exactly edges and vertices on common borders. Continuity across resolution is especially important when representations change through time (e.g., adapting to a moving viewpoint). In order to achieve it, the model must provide many different representations of the same object.

The latter two requirements are intrinsically constrained by the complexiy of underlying problems. Terrain approximation, as studied by Agarwal and Suri [1], is defined as follows: given a set of points S sampled from a bivariate function, and an input threshold $\varepsilon > 0$, compute a piecewise-linear function of minimum complexity (i.e., formed by a minimum number of triangular patches) that fits the elevation of each point of S within an error ε. Agarwal and Suri prove that such a problem is NP-hard. It is open whether or not the problem admits a polynomial solution if output vertices are constrained to lie at input points. In practice, this means that adaptivity shall be relaxed in order to achieve efficiency. There exist only two algorithms computing a suboptimal solution in the unconstrained case that guarantee a bound with respect to optimal [1, 2]. Such algorithms have high time complexity, they are hard to code, and they do not seem to give much better results than other methods on real-world data (see, e.g., experimental results in [2]). More practical methods use heuristics (see, e.g., [12, 20] for surveys). Many of them are based on *local modification*, i.e., on an operation that replaces a (small) group of triangles with another group of triangles covering the same area. Some examples of local modifications are depicted in Figure 1.

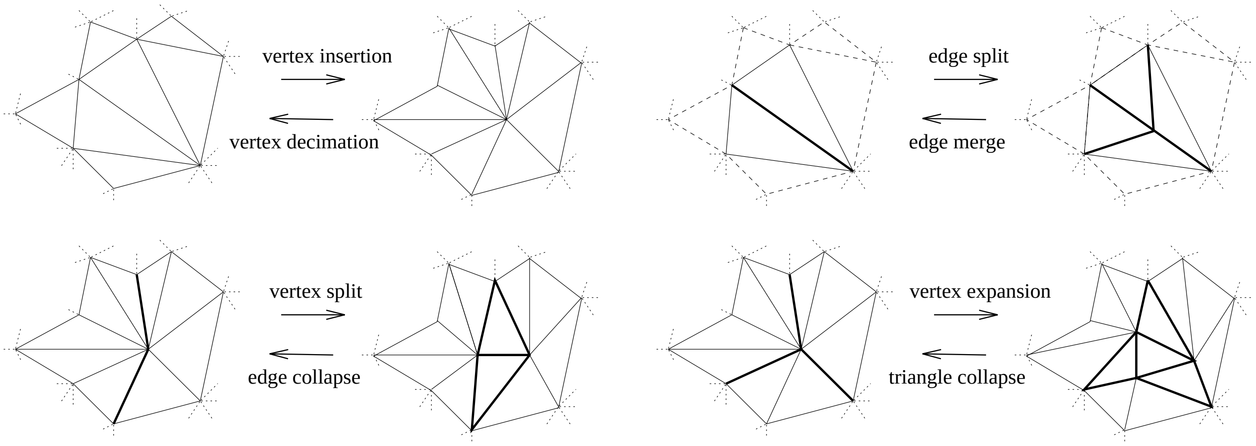

Figure 1: Some examples of local modifications.

The iterative application of local modifications to an initial triangulation leads to an approximate representation. It is easy to show that the problem of surface approximation remains substantially unchanged, and that the same algorithms apply, even if we are given a different threshold ε_i for each datum p_i of S, i.e., for a LOD variable through space [17].

In principle, it is possible to obtain a representation fitting any given LOD by the straight application of an approximation algorithm. However, in order to be effective, a method supporting visualization should provide a number of graphic primitives not larger than that supported by the graphic hardware at frame rate, and it should be able to do this in real time. Approximation algorithms are far too slow to be used on-line. Even a method working in time linear in the size of input would not be efficient enough to process in real time huge datasets with millions of points (at least, not on state-of-the-art workstations). An effective method should rather achieve linear time in the size of its output.

At the state of the art, multiresolution models are based on the possibility to extract different representations from a given structure, which is obtained off-line, and either explicitly or implicitly stored. The model presented here, called a *Multi-Triangulation (MT)*, is meant as a unifying framework over this class of multiresolution models, and the results proved on it are easily applied to all such models.

1.1 Related work

Early multiresolution models were based either on nested subdivisions (e.g., quadtrees, hierarchical triangulations), or on sequences of independent layers providing representations at increasingly finer resolutions (see, e.g., [8] for a survey). Nested models had problems either in the extraction of conforming subdivisions, or in the shape of output elements (slivery triangles) [7]. Lately, further models of this class have been proposed [16], which can be applied to uniformly distributed data (grids) only, and and can provide domain-continuous adaptive representations made with right triangles. Layered models, on the other hand, did not support the extraction of representations whose LOD is variable through space, also known as *selective refinement*.

Selective refinement from non-nested hierarchies was studied first, and independently, in [4, 6], and later in [13, 15, 21].

The hierarchical representation proposed by de Berg and Dobrindt [6] is a sequence of Delaunay triangulations at increasingly coarser level of resolution. The model is built through a technique developed earlier by Kirkpatrick for arbitrary triangulations [14]: each layer is obtained from the previous one by removing an independent set of vertices of bounded degree. Such a construction induces hierarchical links between groups of triangles belonging to successive layers, in the form of a *directed acyclic graph (DAG)*. The algorithm for selective refinement is based on a top-down visit of the DAG, and works in linear time in its output size. The algorithm does not guarantee that the extracted triangulation satifies the required LOD everywhere (see Section 4 for further discussion).

The hypertriangulation proposed by Cignoni et al. [4] is also based on Delaunay triangulations, and is built either through iterative vertex decimation, or through iterative vertex insertion starting from a coarse mesh (see Figure 1, upper left part). The hierarchy deriving from triangle interferences is also a DAG, which is maintained implicitly through an embedding of triangles in a higher-dimensional space. The data structure is based on triangle adjacency, rather than interference. The algorithm for selective refinement builds the solution through a greedy technique that adds triangles by traversing the domain in breadth-first order. The algorithm guarantees that the LOD is satisfied everywhere, but the worst-case time is $O(n \log n)$, where n is the size of the model.

The progressive meshes (PM) proposed by Hoppe [13] are multiresolution models for representing generic manifold surfaces. A PM is obtained by reversing an iterative simplification process based on edge collapse: a coarse mesh is maintained, followed by a sequence of vertex split operations (see Figure 1, lower left part). A simple algorithm for selective refinement scans the sequence, performing split operations, and skipping a split whenever its corresponding portion of mesh satisfies the LOD already. Skipping a split may prevent performing other splits that occur later in the sequence, and would be necessary to achieve the required LOD. This may lead to poor results, because the LOD will be hardly satisfied. Hoppe outlined a remedy that traces vertex dependencies through the sequence on-line. Neither the correctness, nor the computational complexity of this method were analyzed. It seems computationally expensive, because it involves backtracking. A similar approach was proposed independently by Klein and Straßer [15] for a model built through a decimation technique applied on a Delaunay mesh. Later, Xia and Varshney [21] extended the data structure for PM by maintaining a hierarchy of vertex dependencies, and used them to recover all splits necessary to achieve the required LOD. Selective refinement is performed through a bottom-up search of the tree, which requires visiting the whole structure even to obtain the simplest possible mesh.

Here, we borrow two key ideas from such works. The first idea is that a multiresolution model can be obtained as the evolution af a mesh through a process of *iterative local modifications*. Each local modification changes a small portion of a mesh with a group of triangles that we call a *fragment*. Such an idea was at the basis of all models mentioned above.

The second idea is that a non-nested hierarchy of meshes can be represented as a *DAG of fragments, where arcs encode spatial interference* among different fragments. This concept was used first by Kirkpatrick [14] to obtain a spatial indexing structure for point location. It was applied explicitly in [6], and more implicitly in [4] to perform selective refinement.

Although the model proposed in this work is more general than those outlined above, both the data structure described in Section 4.1, and the algorithm proposed in Section 4 are reminiscent of those proposed in [6] (differences are discussed in Section 4).

1.2 Contributions

This work extends and fully formalizes results that were proposed in preliminary version in [18], in the same period and independently of works in [13, 15, 21].

On the basis of the ideas outlined in the previous section, we propose a framework which is obtained by abstracting on specific construction rules (local modifications), while considering only the intrinsic relationships among basic constituents. A Multi-Triangulation is defined as a DAG of generic fragments, satisfying certain properties, which permit to combine them in many different ways to cover the domain. In particular, we prove the following results:

- any possible triangulation made of triangles of an MT can be obtained by combining fragments that lie above some *cut* of the DAG, and such a triangulation is obtained by collecting triangles that label arcs in the cut (Theorem 3.5);
- given a threshold for the LOD, it is possible to extract from the MT the smallest triangulation among those made of triangles in the MT that satisfy the threshold; the extraction takes linear time in the size of the DAG above the cut giving the result; under certain conditions on the structure of the MT, time is linear in the output size, hence optimal (Theorem 4.1).

The proposed extraction algorithm has a straightforward implementation, and runs efficiently. Its practical performances are discussed in a companion paper [10].

All previous multiresolution models based on subdivisions fit in our framework, and can be interpreted as special cases of it. Each specific model is characterized by the local modifications adopted in constructing it, its data structure, and the traversal algorithm for selective refinement (whenever applicable). Such issues are discussed briefly in Section 5, and in more detail in companion papers [9, 20].

2 Preliminaries

A *triangulation* T is tessellation of a polygonal region of the plane into triangles that is *conforming*, i.e., for each two triangles of T their intersection is either empty, or it is coincident with a vertex, or an edge of both triangles. The *size* of T is the number of its triangles, denoted $|T|$; the *domain* of T is the portion of plane covered by its triangles.

Let T and T_i be two triangulations, such that the domain of T_i is contained in the domain of T. T_i is said *compatible* over T if there exists a subset of triangles $\overline{T}_i$ of T such that T_i and $\overline{T}_i$ have the same domain, and the tessellation obtained by substituting T_i to $\overline{T}_i$ in T is still conforming (i.e., it is a triangulation). In this case, T_i is called a *fragment* (with respect to T), and $\overline{T}_i$ is called the *floor* of T_i. T_i is said *minimally compatible* over T if no subset of T_i is compatible over T.

A *local modification* of T through a fragment T_i is an operation that replaces triangles of $\overline{T}_i$ with triangles of T_i in T. Such an operation is denoted $T \oplus T_i$, and its result is a triangulation. The

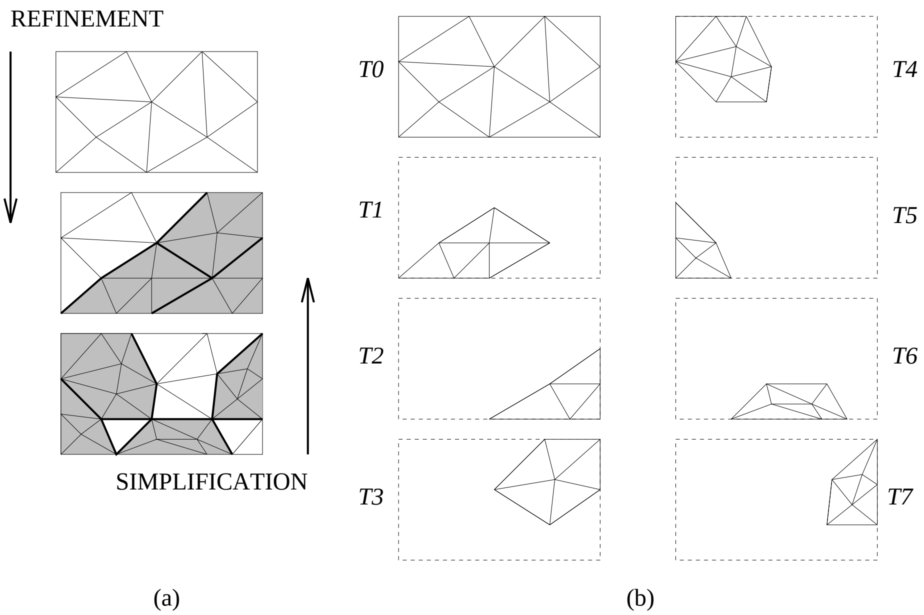

Figure 2: (a) A set of three triangulations at increasing resolution, and (b) its interpretation as a sequence of local modifications.

modification is called either a *simplification*, or a *refinement*, depending on whether T_i has less or more triangles than $\overline{T}_i$, respectively.

Given a triangulation T_0, and a sequence of fragments $T_1, \ldots, T_k$, if for all i a fragment T_i is compatible over the result of $\oplus_{r=0}^{i-1} T_r = T_0 \oplus T_1 \oplus \ldots \oplus T_{i-1}$ then we say that $\mathcal{T} = (T_0, \ldots, T_k)$ is a *compatible sequence* of modifications. The *size* of a sequence is the sum of sizes of triangulations composing it. The triangulation obtained by performing all modifications is said *combination* of the sequence, and denoted $\oplus\mathcal{T}$ for short. The floor of a generic element T_i in a compatible sequence is intended as its floor on the triangulation $\oplus_{r=0}^{i-1} T_r$ obtained by combining all fragments preceding it in the sequence.

3 Multi-triangulations

In order to have an intuition of our framework, let us consider first the sequence of triangulations depicted in Figure 2a. All triangulations decompose the same rectangular domain, and each of them can be obtained from the one preceding it (or following it) through refinement (respectively, through simplification) by replacing triangles in the shaded area. We are interested in combining triangles that appear in this sequence to form *all possible triangulations that can be made of such triangles.* Key observations to move from the sequence to our framework are the following:

1. thick edges in each shaded triangulation of Figure 2a belong also to the triangulation preceding it in the sequence;

2. the shaded portion of each triangulation can be subdivided by thick edges into fragments, each of which is minimally compatible over the previous triangulation: such fragments form a compatible sequence, as in Figure 2b.

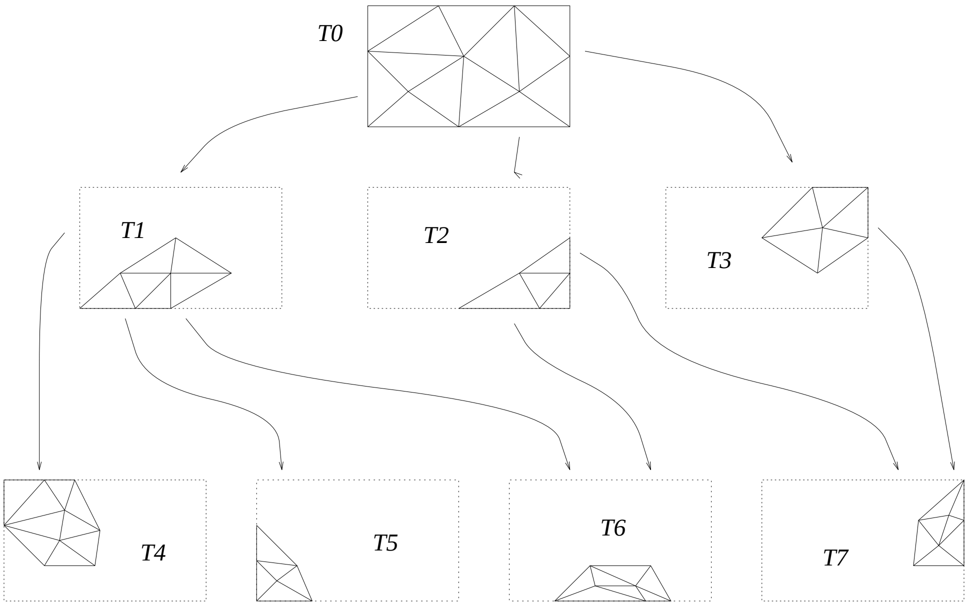

Figure 3: The DAG describing the MT correspondig to the sequence in Figure 2: the three levels of the DAG contain fragments corresponding to the three layers in Figure 2(a)

3. some fragments in the sequence are mutually independent (for instance, we can apply T_1, T_2, and T_3 in any order), while other fragments have dependencies (for instance, T_4 cannot be applied, unless T_1 is applied first). Note that two fragments from the same level in Figure 2a are always independent, and that two fragments may be independent even if they belong to different levels (for instance, T_4 and T_5 are independent from T_2 and T_3).

The last observation suggests that fragments can be arranged to form a a DAG (i.e., a partial order), having its root at the initial mesh, and arcs between pairs of dependent nodes (see Figure 3). This DAG provides essential information to combine triangles that appear in the model.

Definition 3.1 *A* multi-triangulation (MT) *is a Directed Acyclic Graph (DAG) having as nodes the elements of a compatible sequence of triangulations $\mathcal{T} = (T_0, \dots, T_h)$, and satisfying the following axioms:*

1. dependency: *there exists an arc from T_i to T_j if and only if some triangle of T_i belongs to the floor of T_j;*

2. minimality: *every fragment T_i is minimally compatible over its floor;*

3. edge permanence: *if e is an edge common to two fragments T_i and T_j, and T_j is an ancestor of T_i, then e is an edge of the floor of T_i;*

4. non-redundancy: *there are no duplicate triangles, i.e., each triangle belongs to exactly one fragment.*

The dependency axiom follows directly from geometric interferences of fragments, and it specifies the structure of the DAG; minimality and edge permanence axioms have a technical meaning that

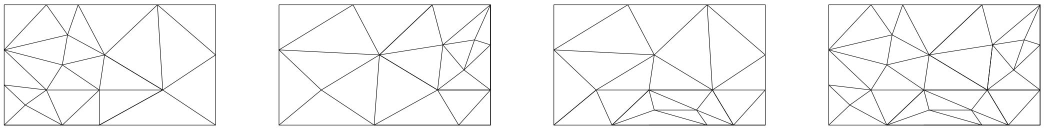

Figure 4: Four out of the possible 25 combinations that can be extracted from the MT of Figure 3.

will be clarified in the sequel (see Theorem 3.5 and discussion preceding it); the non-redundancy axiom is simply aimed to exclude useless fragments made of one triangle. Note that dependency and minimality can be achieved starting with any compatible sequence (non-minimal fragments can be subdivided in a canonical way), while edge permanence and non-redundancy are not necessarily verified by an arbitrary sequence.

Proposition 3.2 *Each triangle of an MT either belongs to the floor of exactly one fragment, or belongs to the combination of the whole sequence.*

Proof: Let t be a triangle, and let T_i be the fragment containing it (T_i is unique for non-redundancy). Let us assume first that a fragment T_j exists that contains t in its floor. In this case, t must belong to the combination of all fragments preceding T_j in the sequence, hence it cannot belong to the floor af any of them; also, t will not belong to the combination of the sequence truncated after T_j, hence it cannot belong to the floor of any fragment following T_j. If no such fragment T_j exists, this means that no fragment in the whole sequence will eliminate t from the mesh, hence t must belong to the final combination. □

It is possible to obtain different triangulations not only by truncating the sequence at some intermediate element, but also by considering any subsequence that is compatible with the partial order corresponding to the DAG, i.e., such that *a modification is made only if all modifications corresponding to its ancestors in the DAG have been made before.* Some such combinations for the MT in Figure 3 are shown in Figure 4.

In the following, we show how such combinations can be obtained as cuts in a *lattice representation* of the MT that is obtained as follows:

- a further *drain* node is added to the DAG, which corresponds to the combination of the whole sequence;
- an arc is added from a node T_i to the drain if and only if T_i has some triangle in the triangulation corresponding to the drain (i.e., T_i has some triangle that does not belong to the floor of any fragment);
- each arc in the lattice is labeled by the triangles of its source node that overlap with the fragment in its end node.

Note that in this framework each triangle of the MT belongs to the label of exactly one arc of the DAG: each triangle of a fragment is labeling some arc outgoing from it, and each triangle in the floor of a fragment is labeling some arc incoming at it. The lattice representation of the MT

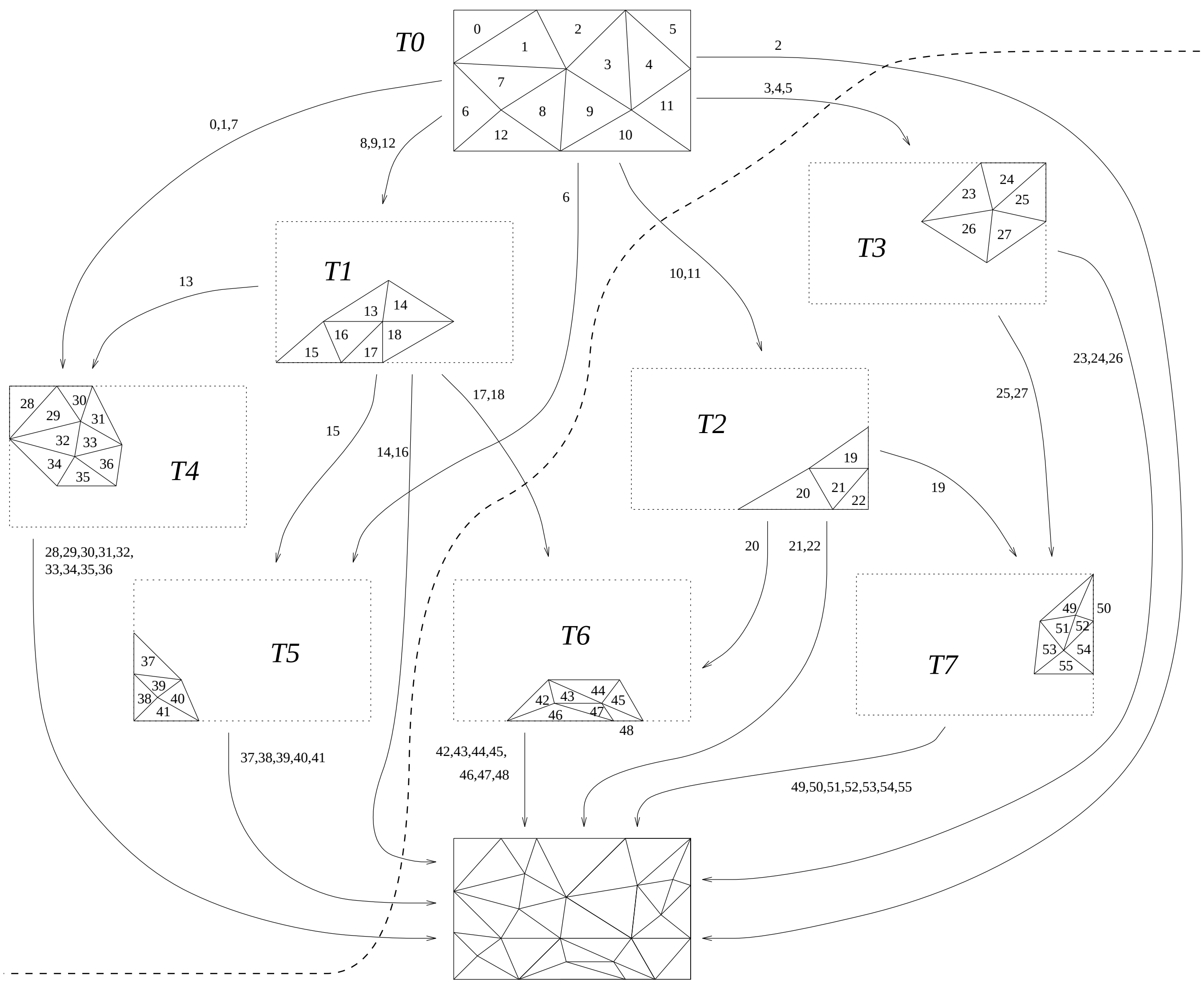

Figure 5: The lattice representation of MT of Figure 3, with a front corresponding to the first mesh of Figure 4.

in Figure 3 is shown in Figure 5. In the following, we will refer interchangeably to the sequence of fragments forming an MT, and to the corresponding DAG, and we will always work on its lattice representation.

A cut of a DAG is a set of arcs disconnecting it. A *front* $\mathcal{E}$ in an MT $\mathcal{T}$ is a cut that contains exactly one arc for each path from the root to the drain.

Proposition 3.3 *Let $\mathcal{E}$ be a front in an MT $\mathcal{T}$. Then: (i) $\mathcal{E}$ disconnects the DAG into two connected components, one containing the root (called the* upper part*), and another containing the drain (called the* lower part*); (ii) if a fragment T_i belongs to the upper part, then also all its ancestors in $\mathcal{T}$ belong to the upper part.*

Proof: The root and the drain belong to different connected components because all paths between them are disconnected by $\mathcal{E}$. In order to prove (i), let us consider a generic node T_i. There always exists a path in $\mathcal{T}$ from root to drain through T_i. Such a path is disconnected by $\mathcal{E}$ at exactly one arc, thus either T_i remains connected to the root, or T_i remains conected to the drain. In order to prove (ii), let T_j be an ancestor of T_i. There always exists a path in $\mathcal{T}$ from the root to the drain through T_j and T_i. If T_i belongs to the upper part, the such a path must be disconnected by $\mathcal{E}$ only between T_i and the drain, hence also T_j must belong to the upper part. □

The following proposition shows how fronts are related to triangulations extracted from the MT.

Proposition 3.4 *Let $\mathcal{T}$ be an MT, and let $\mathcal{T}_{\mathcal{E}}$ be the subsequence of $\mathcal{T}$ formed by the nodes in the upper part of a given front $\mathcal{E}$ (with the natural order). Then $\mathcal{T}_{\mathcal{E}}$ is a compatible sequence, and its combination is formed by all triangles labeling the arcs of $\mathcal{E}$.*

Proof: In order to prove that $\mathcal{T}_{\mathcal{E}}$ is a compatible sequence, it is sufficient to show that the floor of each fragment belongs to the combination of fragments preceding it in the sequence. We prove this by induction. Let $\mathcal{T}_{\mathcal{E}} = (T_0, T_{i_1}, \ldots, T_{i_h})$ be the given sequence. Since, by Proposition 3.3, all ancestors of T_{i_1} must appear before it in $\mathcal{T}_{\mathcal{E}}$, it follows that the floor of T_{i_1} is formed only of triangles of T_0, hence T_{i_1} is compatible over T_0. Let us assume that the sequence truncated at T_{i_j}, i.e., $(T_0, T_{i_1}, \ldots, T_{i_j})$ is a compatible sequence. Its combination is formed by all triangles that belong to this sequence, except those that belong to the floor of some fragment in the sequence itself. We note that all triangles forming the floor of $T_{i_{j+1}}$ must belong to the truncated sequence. Since, by Proposition 3.2, none of them can belong to the floor of a different fragment, thus they all belong to the combination of the truncated sequence.

We note now that every triangle of $\mathcal{T}_{\mathcal{E}}$ is labeling an arc either in the front $\mathcal{E}$, or in the upper part of $\mathcal{E}$. If a triangle t is labeling an arc in $\mathcal{E}$, then t cannot be modified by any other fragment in $\mathcal{T}_{\mathcal{E}}$, since it belongs to the floor of a fragment that lies in the lower part (i.e., it is eliminated by a modification that does not belong to $\mathcal{T}_{\mathcal{E}}$). Therefore it must belong to the combination of $\mathcal{T}_{\mathcal{E}}$. On the other hand, if t is labeling an internal arc of the upper part, then it belongs to the floor of some element of $\mathcal{T}_{\mathcal{E}}$, therefore it cannot belong to its combination. □

Performing a sequence of modifications is equivalent to sweeping a front downwards through the lattice. We start the sequence with a front that lies immediately below T_0, and we move it below a node when we perform the corresponding modification. This means that indeed fronts define all possible compatible sequences in an MT, hence *the number of different triangulations that one can build from compatible sequences of fragments is equal to the number of different fronts.* In the sequel, we will call this number the *expressive power* of the MT, and $\oplus\mathcal{E}$ will denote the triangulation corresponding to a given front $\mathcal{E}$.

Note that the proof of Proposition 3.4 does not involve minimality, edge permanence, and non-redundancy. In other words, this possibility of combining fragments holds for any arbitrary compatible sequence of modifications. However, if a sequence does not satisfy either minimality or edge permanence, there might exist some triangulation formed of triangles of the sequence that cannot be obtained by combining fragments (see Figures 6 and 7 for examples). On the contrary, the expressive power of an MT is maximum, as shown in the following theorem.

Theorem 3.5 *Let $\mathcal{T}$ be a multi-triangulation, and let T be any triangulation formed by triangles of $\mathcal{T}$ (i.e., each triangle of T belongs to some fragment of $\mathcal{T}$). Then there exists a front $\mathcal{E}$ such that $T = \oplus\mathcal{E}$.*

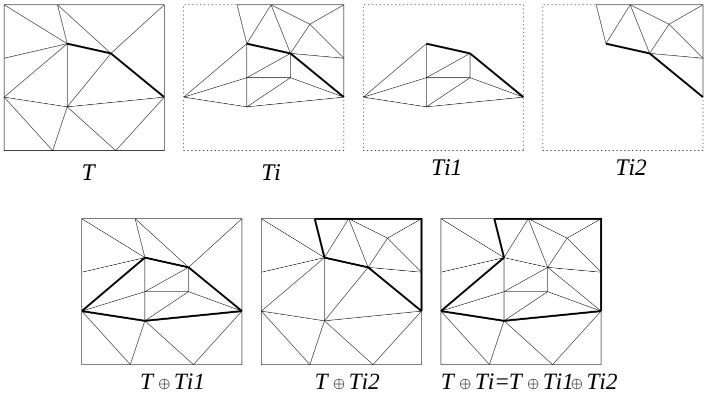

Figure 6: T_i is not minimally compatible over T because each of the two subsets T_{i_1} and T_{i_2} of T_i is compatible over T. Triangulations $T \oplus T_{i_1}$ and $T \oplus T_{i_2}$ cannot be obtained by combining T and T_i, while $T \oplus T_i$ is obtained as $T_j \oplus T_{i_1} \oplus T_{i_2}$.

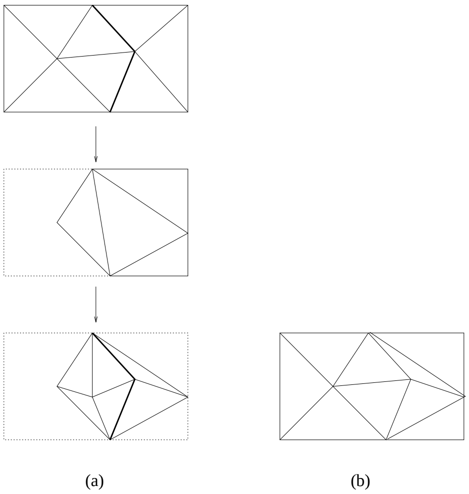

Figure 7: (a) A sequence that violates edge permanence at thick edges: the triangulation in (b) is made from some of its triangles, but it cannot be obtained by combining fragments.

Proof: We give a constructive proof that builds the compatible sequence $\mathcal{T}_\mathcal{E}$ corresponding to $\mathcal{E}$ level by level, from its leaves to its root. Let $\mathcal{T}_\mathcal{E}$ be initially empty. From the non-redundancy axiom, we know that each triangle of T must belong to exactly one fragment of $\mathcal{T}$. Therefore, we tag each triangle of T with the level in the DAG of the fragment containing it, and the maximum level of triangles in a triangulation is called its *height*. Let k be the height of T. We "demolish" T by generating a sequence of triangulations $T \equiv T^{(k)}, T^{(k-1)}, \ldots, T^{(0)} \equiv T_0$, where $T^{(i)}$ has a height i, and it is obtained from $T^{(i+1)}$ by deleting fragments at level $i+1$, and replacing them through their floors.

Let t be a triangle at level k, belonging to a fragment T_i: we claim that all other triangles of T_i must belong to $T^{(k)}$. Indeed, let T_i' be the maximal set of triangles of T_i that belongs to T, contains t, and such that its triangles are connected through edge adjacencies. Let e be an edge that lies on the boundary of T_i'. The following cases can arise:

- if e is a boundary edge of $T^{(k)}$, then it must also be a boundary edge of T_i;
- if the triangle of $T^{(k)}$ incident at e from outside T_i' is at the same level of T_i, then e must be a boundary edge of T_i, because fragments at the same level of $\mathcal{T}$ do not overlap;
- if the triangle of $T^{(k)}$ incident at e from outside the component is at a level lower than T_i, it follows from the edge permanence axiom that e must be an edge also of the floor of T_i.

Since T_i' is bounded by edges that all belong to the floor of T_i then T_i' must be compatible over it. It follows from the minimality axiom that T_i' must coincide with T_i.

Therefore, all triangles of the highest level form a set of fragments that we add to $\mathcal{T}_\mathcal{E}$. We can now obtain $T^{(k-1)}$ by substituting such fragments in $T^{(k)}$ with their floors. $T^{(k-1)}$ has obviously a height of $k-1$. We build the whole sequence defined above by iterating the procedure, while generating fragments that we add to $\mathcal{T}_\mathcal{E}$, until we get a triangulation of height zero, that must necessarily coincide with T_0. This last triangulation is also added to $\mathcal{T}_\mathcal{E}$. Clearly, reversing this procedure is equivalent to combining the elements of $\mathcal{T}_\mathcal{E}$, and the result of such an operation is indeed T. Hence $\mathcal{T}_\mathcal{E}$ is a compatible sequence. Its corresponding front is obtained by the sweeping procedure explained before. □

3.1 Monotone MTs

By now, we know that fronts permit to combine triangles of the MT in all possible triangulations they may form. However, we do not know yet how to select one specific triangulation for a given criterion. Since we are interested in performing selective refinement, in the following we consider a special class of MTs, corresponding to either refinement or simplification sequences.

Definition 3.6 *A multi-triangulation $\mathcal{T}$ is* increasing [decreasing] *if and only if the size of each fragment is larger [smaller] than the size of its floor.*

It is straightforward to see that a multi-triangulation is increasing [decreasing] if and only if for every two fronts $\mathcal{E}'$ and $\mathcal{E}''$, if $\mathcal{E}'$ lies above $\mathcal{E}''$ (i.e., $\mathcal{T}_{\mathcal{E}'}$ is contained in $\mathcal{T}_{\mathcal{E}''}$) then the size of $\oplus\mathcal{E}'$ is smaller [larger] than the size of $\oplus\mathcal{E}''$.

It is also easy to show that an MT can be "reversed" by substituting each internal fragment with its floor, and reversing all arrows in the lattice representation [19]. The resulting structure is also an MT, and it generates the same set of triangulations, being made of the same set of triangles. Furthermore, if an MT is increasing, its reverse will be decreasing, and vice-versa. Therefore, it is sufficient to study the case of increasing MTs, while a decreasing MT can be dealt with by working on its reverse. In the sequel, we will speak generically of *monotone* MT, meaning either increasing or decreasing MT.

Definition 3.7 *An increasing multi-triangulation $\mathcal{T}$ has* linear growth *if for any consistent subsequence $\mathcal{T}'$ the ratio between the size of $\mathcal{T}'$ and the size of its combination is bounded by a constant.*

In Section 4, we will see that linear growth is a desirable property since it permits to achieve optimal output time complexity in visiting the structure. The following proposition provides a sufficient (but not necessary) condition to linear growth that is easy to verify in practical cases. In Section 5 we will also discuss some examples of MT of practical interest that might not have linear growth in the worst case.

Proposition 3.8 *Let $\mathcal{T}$ be an increasing multi-triangulation. If the size of each fragment of $\mathcal{T}$ is larger than the size of its floor for at least a constant factor, then $\mathcal{T}$ has linear growth.*

Proof: For any $T_i \in \mathcal{T}$, let n_i be the size of T_i, and f_i be the size of its floor. Let $\mathcal{T}' = (T_0, T_{j_1}, \ldots, T_{j_k})$ be a compatible subsequence of $\mathcal{T}$. We have:

$$|\mathcal{T}'| = n_0 + \sum_{i=1}^{k} n_{j_i},$$

and, since the combination of a sequence is made of all triangles that do not belong to the floor of any of its elements,

$$|\oplus \mathcal{T}'| = n_0 + \sum_{i=1}^{k} (n_{j_i} - f_{j_i}).$$

Let $c > 1$ be a constant such that for all i it holds $\frac{n_i}{f_i} > c$. We have

$$|\oplus \mathcal{T}'| > n_0 + \sum_{i=1}^{k} \frac{c-1}{c} n_{j_i} > \frac{c-1}{c}|\mathcal{T}'|, \quad \text{and thus} \quad \frac{|\mathcal{T}'|}{|\oplus \mathcal{T}'|} < \frac{c}{c-1},$$

i.e., the MT has linear growth. $\square$

In particular, if an MT is increasing, and the size of each floor is bounded by a constant, then it has linear growth.

4 Extracting a triangulation at variable resolution

4.1 Data sctructure

In order to encode a multi-triangulation we adopt a data structure similar to that proposed to encode the hierarchical representation in [6], which is indeed a special case of MT. The data

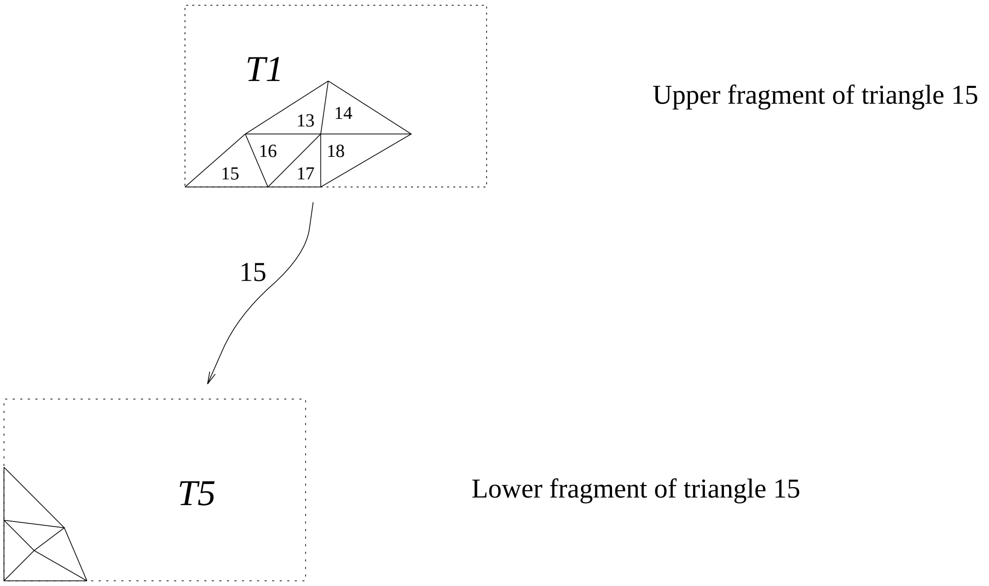

Figure 8: A portion of the MT in Figure 5. With respect to triangle 15: T_1 is the upper fragment, while T_5 is the lower fragment.

structure is based on interferences: it maintains directly relations between triangles and fragments, and indirectly relations between fragments.

Three sets are maintained: the set of all triangles, the set of all vertices, and the set of all fragments. Each fragment contains: a list of (pointers to) triangles composing it; and a list of (pointers to) triangles composing its *floor*. The root element T_0 has an empty floor; the drain element is empty, while it has a floor containing all triangles of the combination of $\mathcal{T}$.

Therefore, each triangle t is referenced by two fragments: the fragment T_i containing t, called the *upper* fragment; and the fragment containing t in its floor, called the *lower* fragment (upper and lower refer to relative positions of the two fragments in the DAG, see Figure 8). Each triangle in the triangle set contains pointers to its upper and lower fragments, as well as pointers to its three vertices. Each vertex is simply characterized by its two cartesian coordinates.

The following operations can be implemented in the above data structure with linear complexity in their output size: FLOOR(T_i) returns the set of triangles composing the floor of a fragment T_i; LOWER(t), and UPPER(t) return the lower, and upper fragments of t, respectively; ROOT($\mathcal{T}$) returns the root T_0, while DRAIN($\mathcal{T}$) returns the drain element.

4.2 Algorithm

Selective refinement requires a test to either accept or discard the resolution of a given triangle. In order to be generic about the application, we assume that a Boolean condition $c()$ is defined on the triangles of $\mathcal{T}$, such that for a given triangle t, $c(t)$ is true if and only if the resolution of t is acceptable. Similarly, the notation $c(T)$ means that all triangles of a triangulation T satisfy $c()$. We assume that $c()$ can be evaluated at a single triangle in constant time, and we consider the following *selective refinement problem*:

> Given an increasing multi-triangulation $\mathcal{T}$, and a Boolean condition $c()$, find the smallest triangulation T made of triangles of $\mathcal{T}$ such that $c(T)$ is true.

From Theorem 3.5 we know that the solution will be given by some front in the DAG describing $\mathcal{T}$. Moreover, since $\mathcal{T}$ is increasing, the size of a front will increase when moving it from the root towards the drain. Our algrithm follows a technique similar to that adopted in [6], i.e., a top-down visit of the DAG. The main difference is that the algorithm in [6] can expand a fragment only if all triangles of its floor have been traversed already. For this reason, such an algorithm may need to accept triangles even if they do not satisfy the Boolean condition. On the contrary, our algorithm permits to perform a local modification whenever necessary by propagating fragment dependencies upwards through the hierarchy.

The algorithm traverses $\mathcal{T}$ 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 T_0, 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.

Traversal is performed through a loop controlled by the content of the queue. At each iteration, the current fragment is extracted from the queue, and triangles composing it and its floor are visited. Each non-marked triangle of the floor is marked, and if its corresponding upper fragment has not been visited yet, then such a fragment is added to the queue. Each non-marked triangle of the fragment is tested against condition $c()$. If a triangle passes the test, then it is added to the potential solution, otherwise it is marked, and its lower fragment is added to the queue. Note that a triangle can possibly be marked after its insertion in the potential solution, when triangles forming the floor of its lower fragment are considered. Traversal stops when either the queue becomes empty, or a triangle that fails the test cannot be refined further. In the latter case, the algorithm returns an empty solution. After traversal, the list of potential solution is scanned, and only triangles not marked are given in output.

In Figure 9 we give a detailed pseudo-code of the algorithm, which is based on the data structure described in the previous section. Besides the primitives on multi-triangulations outlined before, we make use of some standard procedures acting on generic lists. Let Q be a queue, and let e be a generic element, the following standard primitives are used: QUEUE_EMPTY(), IS_EMPTY(Q), DEQUEUE(Q), ENQUEUE(Q, e). Let T be a list, and let t be a generic element, the following standard primitives are used: LIST_EMPTY(), ADD(T, t), REMOVE(T, t). Note that REMOVE(T, t) removes the current element t of list T during list scan, hence it can be implemented with constant time complexity. Finally, we use primitives to mark and test generic elements: MARK(e), IS_MARKED(e), NULL(e). Both triangles and fragments can be marked: marking a triangle means that it cannot be part of the solution; marking a fragment means that it has been visited. All the primitives above can be implemented with constant time complexity.

In order to prove the correctness of the algorithm, let us first assume that it provides a non-empty solution. First, we show that the output T is indeed a triangulation satisfying $c()$. Next, we show that any other triangulation satisfying $c()$ is necessarily larger than T.

If a triangle is added to the list T during the algorithm (line 24), it necessarily satisfies $c()$. Therefore, it is sufficient to show that T is a triangulation. Let us consider the set $\mathcal{T}'$ of fragments visited by the algorithm: $\mathcal{T}'$ is the upper part of some front. Indeed, when visiting the floor of a

```
Algorithm EXTRACT(𝒯,c(),out T)
  begin
    local var Q : queue; T1,T2 : fragment; t : triangle;
    Q = QUEUE_EMPTY();
    T = LIST_EMPTY();
    T1 = ROOT(𝒯);
    MARK(T1);
    ENQUEUE(Q,T1);
    while not IS_EMPTY(Q) do
      T1 = DEQUEUE(Q);
      for every t ∈ FLOOR(T1) do
        if not IS_MARKED(t) then
          MARK(t);
          T2 = UPPER(t);
          if not (NULL(T2) or IS_MARKED(T2)) then
            MARK(T2);
            ENQUEUE(Q,T2);
          end if
        end if
      end for;
      for every t ∈ T1 do
        if not IS_MARKED(t) then
          if c(t) then
            ADD(T,t)
          else
            T2 = LOWER(t);
            if (T2 ==DRAIN(𝒯)) then T = LIST_EMPTY(); exit() end if;
            MARK(t);
            MARK(T2);
            ENQUEUE(Q,T2)
          end if
        end if
      end for
    end while;
    for every t ∈ T do
      if IS_MARKED(t) then REMOVE(T,t) end if
    end for
  end.
```

Figure 9: The algorithm for selective refinement.

fragment, we make sure that all fragments preceding it are also visited (lines 11-20). Let us now consider the set of triangles of $T_{\mathcal{T}'}$: each such triangle t belongs to T if and only if there does not exist a fragment of $\mathcal{T}'$ having t in its floor. Indeed, if there exists one such fragment T_i, then t would be marked when visiting the floor of T_i (line 13). Conversely, if no such fragment exists, then the lower fragment of t is never visited. This means that t cannot be marked, since marking t would either cause visiting its lower fragment (line 28), or be caused by visiting its lower fragment (line 13). Hence, t must be a triangle of T. In conclusion, the triangles of T are all and only those that have no lower fragment in $\mathcal{T}'$, therefore T is the combination of $\mathcal{T}'$.

From Theorem 3.5, we know that any triangulation made of triangles of $\mathcal{T}$ can be obtained by some compatible sequence $\mathcal{T}''$. In order to prove the minimality of T, we show that all fragments visited by the algorithm will necessarily belong to the sequence $\mathcal{T}''$ generating the solution. Proof by induction. Since $\mathcal{T}''$ is the upper part of some front, it must contain the root T_0 of $\mathcal{T}$. Now, let us assume that the first k fragments visited by the algorithm belong to $\mathcal{T}''$. We show that the $(k+1)$-th fragment T_j visited must also belong to $\mathcal{T}''$. There are two possible causes for visiting T_j. Either T_j is parent of a fragment already visited (lines 11-20), hence T_j must be part of $\mathcal{T}''$; or some triangle t belonging to a fragment of $\mathcal{T}''$, and to the floor of T_j, fails the test (lines 25-31). In this case, it means that t cannot be part of the solution, hence there must exist a fragment in $\mathcal{T}''$ that covers t. The only possible fragment is indeed T_j.

Now, let us suppose that $\mathcal{T}'$ is a proper subsequence of $\mathcal{T}''$. Since $\mathcal{T}$ is an increasing MT, this means that the size of $\oplus\mathcal{T}''$ would be larger than the size of $\oplus\mathcal{T}'$, which contradicts the fact that $\mathcal{T}''$ is a solution. Hence, we must have $\mathcal{T}' \equiv \mathcal{T}''$.

Finally, we show that if a sequence $\mathcal{T}''$ giving the solution exists, then the algorithm never goes past the front corresponding to $\mathcal{T}''$. Indeed, the only way to pass such a front is to visit some of the fragments immediately below it. This can only happen if at least one such node has some triangle in its floor (i.e., of the front) that does not satisfy $c()$ (lines 25-30). But this would contradict the fact that $\mathcal{T}''$ is a solution. In conclusion, the algorithm always finds an existing solution. It returns an empty set only when a triangle t has been found, which fails the test, and there exist no fragment to cover it: in this case, the MT does not provide any mesh sufficiently refined to satisfy the required condition over the whole domain.

It follows from the proof of correctness that a fragment is visited by the algorithm only if it belongs to the sequence $\mathcal{T}'$ generating the solution (provided that a solution exists). Each visited fragment is put in the queue at most once, indeed insertion in the queue is conditional on the fragment not being marked. The computational cost of each iteration of the **while** loop (lines 10-33) is linear in the number of triangles composing the current fragment, and its floor. Since each triangle can appear at most in one fragment and in one floor, the total number of operations performed by the **while** loop is linear in the size of $\mathcal{T}'$. Also, each triangle can be inserted in the output list at most once (line 24), hence also the final scan (lines 35-37) takes linear time in the size of $\mathcal{T}'$. Therefore, the algorithm runs in time linear in the size of $\mathcal{T}'$.

From the above proof we have the following theorem.

Theorem 4.1 *The selective refinement problem for an increasing multi-triangulation $\mathcal{T}$ can be solved in time linear in the size of the sequence generating the output (i.e., at most linear in the size of $\mathcal{T}$).*

If $\mathcal{T}$ has linear growth, then an existing solution can be found in time linear in its output size, hence optimal.

A completely analogous algorithm and proof are valid for a monotonically decreasing MT. In this case, its reverse is visited. Since the data structure encodes also the reverse MT, it is not necessary to recompute it explicitly [19]. In this case, running the algorithm for the reverse MT is equivalent to visit $\mathcal{T}$ starting at the floor of its drain, and swapping FLOOR and fragment, as well as UPPER and LOWER, in the code.

5 Applications

Applications of multi-triangulations to terrain modeling is straightforward. For each vertex in the model, its elevation z is encoded as an additional information. For each triangle t, the error ε_t in approximating data elevations may be encoded too. Such an error can be defined as the maximum difference between the actual elevation of each point p whose projection lies inside t, and its approximated elevation on the triangular surface patch corresponding to t (see, e.g., [8]).

Given a LOD function $\tau : \mathbb{R}^2 \rightarrow \mathbb{R}$, the condition for algorithm EXTRACT is defined as $c(t) = (\varepsilon_t \leq \min_{p \in t} \tau(p))$. For applications such as flight simulators, function τ is usually increasing with distance from the viewpoint. A possible function used in [4] is simply $\tau(p) = K|p - v|$, where v is the viewpoint, and K is a suitable constant. Other authors propose functions that depend on the size of error projected on the image [16].

An alternative is to make LOD dependent on the size of each graphical primitive in the output image: in this case, a triangle might be accepted, independently of its accuracy, when the area of its projection is smaller than a given threshold δ. Therefore, the condition for algorithm EXTRACT is defined as $c(t) = (\Delta(t) \leq \delta)$, where $\Delta()$ denotes the area of the triangle projected onto the image.

In either case, it is also convenient to restrict LOD to the view frustum: to this aim, condition $c()$ can be set to **true** for all triangles whose projections lie outside the frustum. Note that this is not equivalent to restrict the search in algorithm EXTRACT to triangles inside the frustum. Although in the latter case the algorithm might run faster, the output would be a triangulation of the portion of domain corresponding to the frustum, not necessarily compatible with a triangulation of the rest of the domain. In a dynamic scenario, this might lead to dramatic changes in the triangulation even for a small movement of the viewpoint, hence producing unpleasant visual effects (popping).

Another possible application of multi-triangulations is in domain decompositions for finite element methods. In this case, a possibility is that a domain must be decomposed into triangles whose size satisfies a user-defined density function [3]: algorithm EXTRACT could be used with a condition $c(t) = (r_t \leq \delta(o_t))$, where $\delta : \mathbb{R}^2 \leftarrow \mathbb{R}$ is the density function, r_t is the circumradius of t, and o_t is its circumcenter.

The practical performance of an MT is highly influenced by the structure of its fragments, and of their mutual interferences. Generally, an MT formed of many small fragments should have high expressive power, and support selective refinement efficiently. The structure of fragments depends on the construction technique adopted. An MT can be built easily through iterative algorithms based on local modifications, which follow directly from known techniques for surface simplification.

Most techniques are based on heuristics that try to select a good sequence of modifications to reduce the size of a mesh necessary to achieve a given LOD. It is easy to see that construction techniques proposed in [4, 6] (based on vertex decimation), and in [13, 21] (based on edge collapse) produce monotone MTs. Other techniques can be used as well (see Figure 1, and refer to [20] for details).

Linear growth is relevant to the performance of selective refinement. This is achieved by using local modifications that involve a number of triangles bounded by a constant. For instance, the decimation technique adopted in [6] always eliminates vertices of bounded degree. A similar approach could be adopted to build an MT by iteratively collapsing edges whose endpoints have bounded degree. More arbitrary selection techniques, such as those adopted in [4, 13, 15], might result more effective in reducing the number of vertices necessary to achieve a given LOD, but may lead to a quadratic growth in the worst case. Simplification techniques can be combined with the selection of independent sets of entities (e.g., vertices to remove in [6], and edges to collapse in [21]) in order to obtain a DAG whose depth is logarithmic in its size. This last feature is more relevant to traversal operations such as point location, rather than to selective refinement. Construction issues are discussed further, and different techniques are compared empirically in a companion paper [10].

Multiresolution models based on nested subdivisions can be also interpreted as special cases of an MT. Given such a model and its tree representation, the fragments of the MT are obtained by clustering nodes that have vertices along their common border. For a general discussion of this subject refer to [9, 20]. Here we just discuss the model proposed by Lindstrom et al. [16], which seems especially effective for the special case of data on a regular grid. In this case, a binary tree of right triangles is obtained by recursively splitting a triangle at the midpoint of its longest edge. An MT corresponding to this structure is obtained by clustering triangles of the same size that are adjacent through their shorter edges (see Figure 10). Such an MT is monotone, and has linear growth. Being based on a large number of small fragments, such an MT has high expressive power. A great advantage in this case it that an implicit pointerless data structure based on locational codes can be used [11].

6 Concluding remarks

The multi-triangulation is a framework for the multiresolution decomposition of plane domains that unifies several multiresolution surface models. A variety of MTs can be built and manipulated easily and efficiently through mesh simplification methods. The optimal time behavior of the algorithm for selective refinement, the simplicity of its implementation, as well as the minimality of the representation it extracts, make the MT a valid support to enhance the quality of surfaces that can be rendered in real time.

The application of MT to modeling and rendering topographic surfaces, and free-form surfaces is demonstrated in a companion paper [10]. Also the extension to higher dimensions of the MT is straightforward [9]. In particular, multi-tetrahedralizations can be used for applications in volume visualization, and finite element methods.

There is a number of open problems connected to these issues, both from a theoretical, and from a practical point of view. A first important problem is to perform selective refinement while guaranteeing a bound on the size of the output, with respect to the optimal solution. The MT gives only the best possible solution for a specific set of triangles, while approximating the optimal

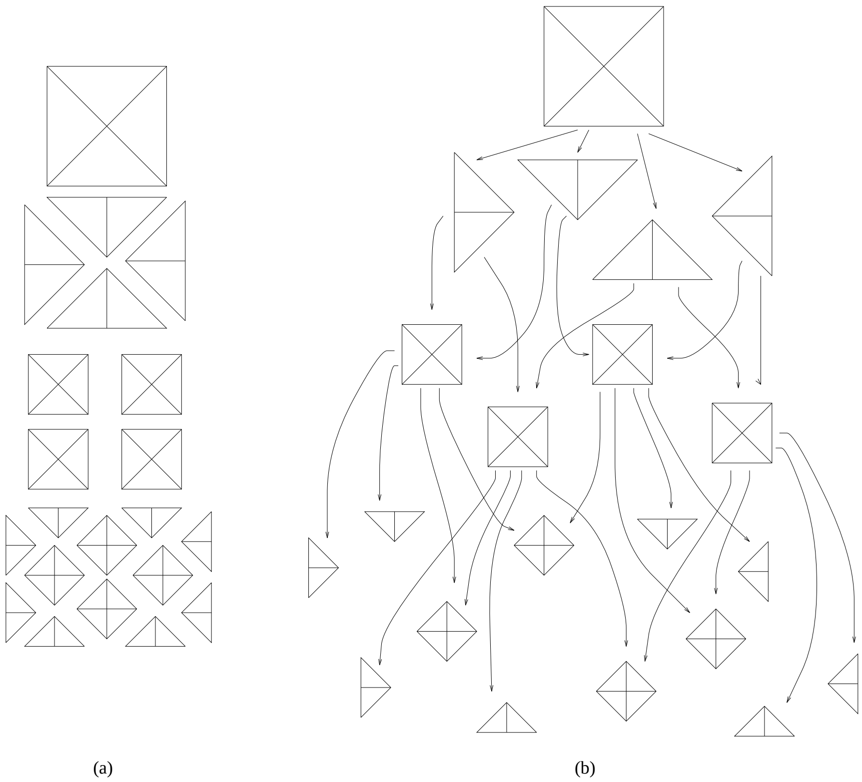

Figure 10: An MT based on right triangles: (a) the fragments shown level by level; (b) the corresponding DAG.

solution seems much a harder problem. There is certainly a large gap to fill between the few results on provably good approximations [1, 2], and empirically good models adopted in the applications.

Another problem is connected to free-form surface modeling. In this case, the straight generalization of the MT [10] (as well as any other existing model that supports selective refinement) may give surfaces affected from self-intersections, even when all meshes in the construction sequence have not this problem. Some simplification strategies exist that preserve the topology of the original surface [5], but it is not clear whether a model supporting efficient selective refinement can be built on that basis.

From the point of view of applications, there are at least two major problems. The first problem is to find an optimal balance between storage, and computational costs. Rich data structures – such as the one presented in this paper, and those in [4, 6, 21] – support efficient selective refinement, but may result expensive in terms of storage overhead. On the other hand, cheap data structures, e.g. those in [13, 15], have small overhead, but may penalize the efficiency of selective refinement. Moreover, there are other spatial operations that may be of interest in the applications (e.g., point location, intersection, navigation), which should be considered in designing a data structure for a multiresolution model. The second problem is that huge datasets used in practice, such as those needed in flight simulation, can hardly be maintained in primary memory. Therefore, efficient structures for storage into secondary memory, and retrieval from it should be designed.

Note. After this paper was submitted for publication, a large number of other works on multiresolution surface models appeared in the literature, both from the author, and from others. Most such works rely on extensions and variations either of the works in [13, 16], or of the work proposed here. All models to our knowledge still fit in the MT framework (generalized to arbitrary surfaces), hence all results of this paper can be applied to those models too.

Acknowledgments

Leila De Floriani, Paola Magillo and Paola Marzano are gratefully acknowledged for many helpful discussions and suggestions on the matter of this paper. This work has been partially supported by the National Research Council of Italy under Coordinate Project "Multiresolution Models for the Visualization of Multidimensional Fields".

References

[1] P.K. Agarwal, S. Suri, 1994, Surface approximation and geometric partitions, *Proceedings 5th ACM-SIAM Symposium on Discrete Algorithms*, pp.24-33. Extended version to appear in *SIAM Journal of Computing.*

[2] P.K. Agarwal, P.K. Desikan, 1997, An efficient algorithm for terrain simplification, *Proceedings 8th ACM-SIAM Symposium on Discrete Algorithms.*

[3] L.P. Chew, 1993, Guaranteed-quality mesh generation for curved surfaces, *Proceedings 9th ACM Symposium on Computational Geometry*, pp.274-280.

[4] P. Cignoni, E. Puppo, R. Scopigno, 1995, Representation and visualization of terrain surfaces at variable resolution, *Proceedings International Symposium on Scientific Visualization*, R. Scateni (Editor), World Scientific, Cagliari (Italy), September 1995. Extended version appeared in *The Visual Computer*, 13(5), 1997, pp.199-217.

[5] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, W. Wright, 1996, Simplification envelopes, *Proceedings SIGGRAPH 96*, in *Computer Graphics - Annual Conference Series*, ACM SIGGRAPH, New Orleans, LA, USA, August 4-9, 1996, pp.119-128.

[6] M. de Berg, M., K.T.G. Dobrindt, 1995, On the levels of detail in terrains, *11th ACM Symposium on Computational Geometry*, Vancouver (Canada), June 5-7, pp.c26-c27. Also available in long version as *Techical Report* UU-CS-1995-12, Utrecht University, Dept. of Computer Science, April 1995.

[7] L. De Floriani, E. Puppo, 1995, Hierarchical triangulation for multiresolution surface description, *ACM Transactions on Graphics*, 14, 4, pp.363-411.

[8] L. De Floriani, P. Marzano, E. Puppo, 1996, Multiresolution models for topographic surface description, *The Visual Computer*, Vol.12, No.7, pp.317-345.

[9] L. De Floriani, E. Puppo, P. Magillo, 1997, A formal approach to multiresolution modeling, *Geometric Modeling: Theory and Practice*, W. Straßer, R. Klein, R. Rau (Editors), Springer-Verlag, Berlin Heidelberg (D), pp.302-323.

[10] L. De Floriani, P. Magillo, E. Puppo, 1997, Building and traversing a surface at variable resolution, *Proceedings IEEE Visualization'97*, October 19-24, 1997, Phoenix, AZ.

[11] D.J. Hebert, H.-J. Kim, 1995, Image encoding with triangular wavelets, *Proceedings SPIE*, 2569(1), pp.381-392.

[12] P. Heckbert, M. Garland, 1997, Survey of polygonal surface simplification algorithms, *Siggraph'97 Course Notes.*

[13] H. Hoppe, 1996, Progressive meshes, *Proceedings SIGGRAPH 96*, in *Computer Graphics - Annual Conference Series*, ACM SIGGRAPH, New Orleans, LA, USA, August 4-9, 1996, pp.99-108.

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

[15] R. Klein, W. Straßer, 1997, Generation of multiresolution models from CAD data for real time rendering, *Geometric Modeling: Theory and Practice*, W. Straßer, R. Klein, R. Rau (Editors), Springer-Verlag, Berlin Heidelberg (D).

[16] P. Lindstrom, D. Koller, W. Ribarsky, L.F. Hodges, N. Faust, G.A. Turner, 1996, Real-time, continuous level of detail rendering of height fields, *Proceedings SIGGRAPH 96*, in *Computer Graphics - Annual Conference Series*, ACM SIGGRAPH, New Orleans, LA, USA, August 4-9, 1996, pp.109-118.

[17] G. Misund, 1996, Varioscale TIN based surfaces, 1996, *Proceedings 7th International Symposium on Spatial Data Handling*, Delft, Netherlands, August 13-16, 1996, pp.6.35-6.46.

[18] Puppo, E., 1996, Variable resolution terrain surfaces, in *Proceedings of the Eight Canadian Conference on Computational Geometry*, 12-15 August 1996, Ottawa (Canada), pp.202-210.

[19] Puppo, E., 1996, Variable resolution triangulations, *Technical Report* n.12-96, Istituto per la Matematica Applicata - C.N.R., November 1996.

[20] Puppo, E., Scopigno, R., 1997, Simplification, LOD, and multiresolution - Principles and applications, *Eurographics'97 Tutorial.*

[21] J.C. Xia, A. Varshney, 1996, Dynamic view-dependent simplification for polygonal models, *Proceedings IEEE Visualization'96*, 1996, pp.327-334.