Maintaining Multiple Levels of Detail in the Overlay of Hierarchical Subdivisions

(Extended Abstract)

Paola Magillo, Leila De Floriani
Computer and Information Science Department (DISI), University of Genova
Via Dodecaneso, 35, 16146 Genova (Italy)
Fax: +39-10-3536699 - Email: {magillo, deflo} @ disi.unige.it

Abstract

We propose a formal definition of a hierarchical structure representing the overlay of two hierarchical subdivisions at multiple resolutions, and an efficient bottom-up algorithm to compute it. An important application is connected to map processing in geographic information systems.

1 Introduction

In this paper, we consider the overlay of plane subdivisions (a classical problem in computational geometry) and extend it to the case of hierarchically represented subdivisions.

The problem, in this new setting, has relevance in geographic information systems, where plane subdivisions are used to represent maps, and the combination (overlay) of two maps, describing different characteristics of the same spatial domain, is a fundamental operation. The need for multiresolution arises, because huge amounts of data are available, while not all application tasks require the same level of detail. Hierarchical subdivisions compactly encode decompositions of a plane domain at multiple resolutions, and efficiently support extraction of information satisfying a given level of detail; moreover, they support navigation across various abstraction levels. Thus, they represent an effective way of reducing memory and processing costs as well as an effective tool for map analysis.

Several multiresolution models have been proposed, depending on the type of geographic data for which they are designed. Here, we consider a simple, general-purpose model, called the *hierarchical subdivision*, which is based on the concept of a recursive refinement. Given two hierarchical subdivisions, we focus on the problem of building a hierarchical subdivision that provides a multiresolution representation of their overlay. We formally define the problem and give a bottom-up algorithm for it.

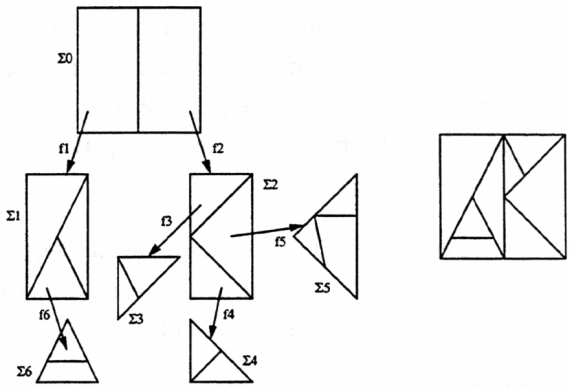

Figure 1: A hierarchical subdivision, and an expansion, computed under a condition c' such that $c'(f_1) = c'(f_2) = c'(f_3) = c'(f_6) = false$ and $c'(f) = true$ for any other face f.

2 Hierarchical Subdivisions

A *plane subdivision* is a triple $\Sigma \equiv (V, E, F)$ where:

- V is a set of points, called *vertices*;
- E is a set of closed straight-line segments, called *edges*, whose endpoints are vertices, and such that no two edges intersect, except in a common endpoint;
- F is a set of closed simply-connected polygonal regions, called *faces*, whose boundary is a union of edges, and such that any two faces intersect at most in their boundary;
- $\cup\{f \mid f \in F\}$ covers a simply-connected polygonal domain D, called the *domain* of the subdivision.

The number of edges and faces in a plane subdivision is linear in the number n of vertices.

Informally, a hierarchical subdivision is obtained by recursively decomposing a domain into a sequence of increasingly finer tesselations. The sequence of levels of detail obeys to a predefined set of conditions. For example, if a hierarchical subdivision is used to represents a political map, conditions could be defined by granularities of the administrative units (states, counties, cities, etc.). Starting from a subdivision Σ_0 representing the domain at the coarsest level, a face f of Σ_0 is refined, at the next level of detail, into a subdivision Σ_f, whose domain is f. This refinement process is recursively iterated.

More formally, a sequence $c_0, \ldots, c_h$ of conditions is given, such that c_i implies c_{i-1} for all $i = 1, \ldots, h$. A *hierarchical subdivision*, based on sequence $c_0, \ldots, c_h$ is a pair $\mathcal{H} = (\mathcal{S}, \mathcal{E})$, where $\mathcal{S} = \{\Sigma_0, .., \Sigma_m\}$ is a collection of plane subdivisions, such that:

- for every $1 \leq i \leq m$, Σ_i contains more than one face;
- for every $1 \leq j \leq m$, there is exactly one index i, with $0 \leq i < j$, and one face $f_j \in \Sigma_i$, such that the domain of Σ_j covers face f_j; we say that Σ_i is the *parent* of Σ_j and Σ_j is a *child* of Σ_j; moreover Σ_j is called the *direct expansion* of face f_j;
- Σ_0 satisfies condition c_0;
- if $\Sigma_i \in \mathcal{S}$ satisfies conditions $c_0, \ldots, c_j$, for some $0 \leq j \leq h-1$, then all children of Σ_i satisfy conditions $c_0, \ldots, c_j, c_{j+1}$;
- a subdivision Σ_i has no children if and only if it satisfies all conditions $c_0, \ldots, c_h$;

and $\mathcal{E}$ is a collection of hierarchical links between subdivisions in $\mathcal{S}$:

- for every pair of subdivisions Σ_i and $\Sigma_j \in \mathcal{S}$, such that the domain of Σ_j covers a face $f_j \in \Sigma_i$, $\mathcal{E}$ contains an oriented arc (Σ_i, Σ_j);
- such an arc is labelled with face f_j.

A hierarchical subdivision $\mathcal{H}$ can be described by a tree having nodes in $\mathcal{S}$ and labelled arcs in $\mathcal{E}$, where Σ_0 is the *root*. A face which is refined in the hierarchy is called a *macroface*, otherwise it is called a *simple face*. The total size of a hierarchy has been shown to be linear in the number of its simple faces [DeF92].

The condition c_k such that $k = \max\{q \mid \Sigma_i$ satisfies conditions $c_0, \ldots, c_q\}$ is called the *level* of subdivision Σ_i, and denoted by $level(\Sigma_i)$. The *level* of a face f_j is defined as $\max\{q \mid f_j$ satisfies conditions $c_0, \ldots, c_q\}$. For a face f_j belonging to a subdivision Σ_i, $level(f_j) \geq level(\Sigma_i)$; if f_j is a macroface and Σ_j is its direct expansion, then $level(\Sigma_i) \leq level(f_j) < level(\Sigma_j)$.

A hierarchical subdivision $\mathcal{H}$ implicitly represents a family of subdivisions, called the *expansions* of $\mathcal{H}$, each describing the domain at a certain level of detail. Any subdivision of the family can be extracted from $\mathcal{H}$ through a traversal of its tree structure: starting from the root subdivision Σ_0, each macroface is recursively replaced with its direct expansion. The recursive replacement process stops when all collected faces satisfy some condition c' (defining the desired level of detail), which is given within the specific application (see Figure 1). The resulting expanded subdivision is called the *expansion of $\mathcal{H}$ under condition c'*, and denoted by $exp(\mathcal{H}, c')$.

To model a single step of the recursive replacement process, we define a *refinement* operation, which takes two subdivisions Σ_i and Σ_j, such that the domain of Σ_j is a face $f_j \in \Sigma_i$, and builds another subdivision, that we denote by $\Sigma_i[f_j/\Sigma_j]$, by replacing macroface f_j in Σ_i with subdivision Σ_j.

We denote by $V(\Sigma)$, $E(\Sigma)$, $F(\Sigma)$ the set of vertices, edges and faces, respectively, of a subdivision Σ, and by $E(f)$, the set of boundary edges of a face f. The subdivision $\Sigma' \equiv \Sigma_i[f_j/\Sigma_j]$ is defined as follows:

- $F(\Sigma') \equiv F(\Sigma_i) - \{f_j\} \cup F(\Sigma_j)$;
- $V(\Sigma') \equiv V(\Sigma_i) \cup V(\Sigma_j)$;
- $E(\Sigma') \equiv E(\Sigma_i) - E(f_j) \cup \{e \in E(\Sigma_j) \mid e$ internal edge of $\Sigma_j\} \cup \{e'' \equiv e \cap e' \mid e \in E(f_j), e' \in E(\Sigma_j)$ and $e \cap e'$ is a segment$\}$.

The *expansion* of a hierarchical subdivision under a certain condition c' is recursively defined as follows:

- if every face of Σ_0 satisfies c', then the expansion is the same as the root subdivision Σ_0;
- otherwise, we consider all the arcs (Σ_0, Σ_j), labeled with a macroface f_j, such that $c'(f_j)$ is false. For every such arc, we recursively compute the expansion Σ'_j of the subtree rooted at Σ_j under condition c'. The expansion of $\mathcal{H}$ under condition c' is obtained from Σ_0 by simultaneously replacing each macroface f_j by Σ'_j.

In order to store a hierarchical subdivision $\mathcal{H}$, we encode each node Σ_i of $\mathcal{H}$ into a two-dimensional specialization of the cell-tuple data structure proposed by Brisson [Bri93], and explicitly represent hierarchical links between each macroface and its direct expansion.

A $2D$ cell-tuple structure, representing a plane subdivision Σ, consists of a pair $(\mathcal{C}, \mathcal{S})$, where $\mathcal{C}$ is

a collection of basic elements, called *cell tuples*, and $\mathcal{S} = \{switch\_vertex, switch\_edge, switch\_face\}$ is a family of *switch operators* defined on cell-tuples. A cell-tuple is (in the two-dimensional case) a triple (v, e, f) where v, e and f are, respectively, a vertex, an edge and a face of Σ, such that v is an endpoint of e, and e is a boundary edge of f. Since an edge has two endpoints and at most two adjacent faces, the total number of cell-tuples is at most four times the number of edges, that is $O(n)$.

The switch operators link each cell-tuple to other three cell-tuples, in the following way:

- $switch\_vertex(v, e, f) = (v', e, f)$, where v' is the other endpoint of e;
- $switch\_edge(v, e, f) = (v, e', f)$, where e' is the other edge incident in v and belonging to the boundary of f;
- $switch\_face(v, e, f) = (v, e, f')$, where f' is the other face adjacent to e.

Four cell-tuples, connected through operators *switch_vertex* and *switch_face*, implicitly represent an edge together with its extreme points and adjacent faces; a vertex, together with all its incident edges and faces, corresponds to a chain of cell-tuples connected with operators *switch_edge* and *switch_face*; finally, a chain of cell-tuples connected with operators *switch_edge* and *switch_vertex* defines a face and all its boundary edges and vertices.

Geometric information are stored by enriching each cell-tuple (v, e, f) with a pointer to the coordinates of vertex v. Each subdivision Σ_i in the hierarchy is represented by an individual cell-tuple structure, where geometric information are global, i.e., cell-tuples belonging to different subdivisions, but containing the same vertex, point to the same coordinate pair.

Every node Σ_i of the hierarchy stores its level $\in \{c_0, \ldots, c_h\}$. Since faces are not explicitly represented in the cell-tuple structure, we store information about macrofaces and their expansions separately. Every node Σ_i of the hierarchy stores a list containing, for each macroface f_j of Σ_i, the level of f_j, a pointer to one of the cell-tuples of the form (v, e, f_j) in Σ_i, and a pointer to the direct expansion Σ_j of f_j; this list is sorted according with the increasing level of macrofaces.

Moreover, we store links from tuples (v, e, f), with f macro, to tuples of the direct expansion of f. Let t be a cell-tuple, in a subdivision Σ_i, corresponding to a triple (v, e, f), where f is a macroface, and let Σ_j be the direct expansion of f. Then t points to the unique cell-tuple t' in Σ_j corresponding to a triple (v, e', f'), where $e' \subseteq e$ and $f' \subset f$. Storing parent-child links in tuples is necessary to retrieve the expanded subdivision under a given condition (since it allows to to merge the cell-tuple structure representing a child subdivision into the cell-tuple structure of its parent), and it adds only a constant term to the memory requirements of each tuple. Our data structure is represented in Figure 2.

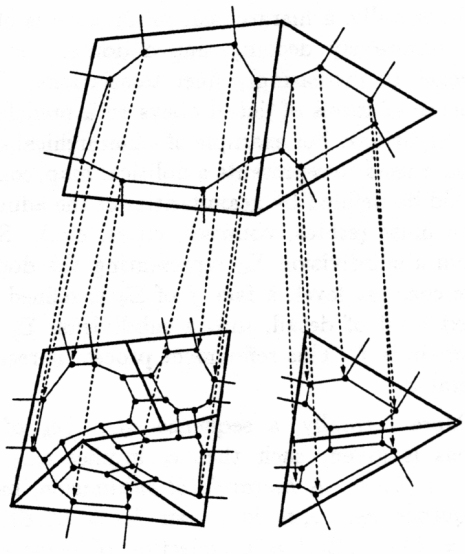

Figure 2: The data structure encoding a hierarchical subdivision: points represent tuples, segments connecting them represent switch operators; hierarchical links between the tuples of a macroface and the tuples of its direct expansion are shown as dotted lines.

The algorithm for extracting an expanded subdivision under a given expansion condition c' performs a preorder traversal of the tree, and operates according to the definition of expansion (see [Mag95] for a detailed description). The time complexity is linear in the size of the extracted subdivision.

3 The Overlay Problem

Given two plane subdivisions Σ_1 and Σ_2, the *overlay* of Σ_1 and Σ_2 is a plane subdivision Σ' whose vertices, edges and faces are obtained by intersecting those of Σ_1 and of Σ_2. If $\Sigma_1 \equiv (V_1, E_1, F_1)$ and $\Sigma_2 \equiv (V_2, E_2, F_2)$, the overlay of Σ_1 and Σ_2 is $\Sigma' \equiv (V', E', F')$, where:

- $V' = V_1 \cup V_2 \cup \{P \mid P \equiv e_1 \cap e_2, e_1 \in E_1, e_2 \in E_2, e_1 \cap e_2$ is a point$\}$,
- $E' = \{e' \mid e' \equiv f_1 \cap e_2, f_1 \in F_1, e_2 \in E_2, f_1 \cap e_2$ is a segment$\} \cup \{e' \mid e' \equiv e_1 \cap f_2, e_1 \in E_1, f_2 \in F_2, e_1 \cap f_2$ is a segment$\} \cup \{e' \mid e' \equiv e_1 \cap e_2, e_1 \in E_1, e_2 \in E_2, e_1 \cap e_2$ is a segment$\}$,

- $F' = \{f' \mid e' \equiv f_1 \cap f_2, f_1 \in F_1, f_2 \in F_2, f_1 \cap f_2 \neq \emptyset, f_1 \cap f_2$ is a region$\}$.

The size of the overlay of two subdivisions of size n_1 and n_2, respectively, is $O(n_1 + n_2 + k)$, where k is the number of intersections between the edges, and it is $O(n_1 n_2)$ in the worst case.

Here, we focus on the problem of efficiently computing a multiresolution model representing the overlay of two hierarchical subdivisions.

Let $\mathcal{A}$ and $\mathcal{B}$ be two hierarchical subdivisions, built based on two sequences of conditions $\alpha_0, \ldots, \alpha_a$ and $\beta_0, \ldots, \beta_b$, respectively. We select a sequence $\gamma_0, \ldots, \gamma_c$ of pairs of conditions: for every $0 \leq i \leq c$, $\gamma_i \equiv \alpha_{p(i)}$ AND $\beta_{q(i)}$ where $0 \leq p(i) \leq a$ and $0 \leq q(i) \leq b$. Indices $p(i)$ and $q(i)$ must be chosen in such a way that γ_i implies γ_{i-1} for all $1 \leq i \leq c$. This is achieved by imposing that, for every $1 \leq i \leq c$, $p(i) \geq p(i-1)$, $q \geq q(i-1)$, and at least one of the two inequalities is strict.

Our aim is building a hierarchical subdivision $\mathcal{C}$, based on the sequence of conditions $\gamma_0, \ldots, \gamma_c$, such that, for every i, the expansion of $\mathcal{C}$ under condition γ_i is the same as the overlay of $exp(\mathcal{A}, \alpha_{p(i)})$ and $exp(\mathcal{B}, \beta_{q(i)})$.

If n_a and n_b are the sizes of the two hierarchies $\mathcal{A}$ and $\mathcal{B}$, the worst-case size of $\mathcal{C}$ is $n_c = O(n_a + n_b + k)$ where k is the number of intersections between the expansions of $\mathcal{A}$ and $\mathcal{B}$ under conditions $\alpha_{p(c)}$ and $\beta_{q(c)}$, respectively.

4 An Algorithm to Compute an Overlay Hierarchy

A key observation is that the overlay hierarchy $\mathcal{C}$ cannot be built top-down efficiently. A top-down construction involves superimposing pairs of nodes from $\mathcal{A}$ and $\mathcal{B}$ corresponding to the selected pairs of conditions, and then simplifying the resulting structure to a hierarchical subdivision. This approach may compute the same faces several times (see [Mag95] for an example of this fact). In the worst-case, $\Theta(n_a)$ faces are found $\Theta(b)$ times. If b is $\Theta(n_b)$, the complexity raises to $\Theta(n_a n_b)$, even when the actual size of the overlay hierarchy is linear in $n_a + n_b$.

Thus, we must proceed bottom-up. We first compute the overlay of $exp(\mathcal{A}, \alpha_{p(c)})$ and $exp(\mathcal{B}, \beta_{q(c)})$, and then build the upper levels of $\mathcal{C}$ through iterative generalization steps, driven by the sequence of conditions $\gamma_0, \ldots, \gamma_c$. Each generalization step deletes edges of $\mathcal{A}$ and $\mathcal{B}$ which represent too fine details and merges remaining edges and faces accordingly.

The expansions of $\mathcal{A}$ and $\mathcal{B}$ under condition $p(c)$ and $q(c)$ are computed by the extraction algorithm described in [Mag95]. During the extraction, the following information are also computed and stored:

- for every vertex v of $exp(\mathcal{A}, \alpha_{ca})$, its *level*, defined as the level of the coarsest node of $\mathcal{A}$ where v appears;
- for every cell-tuple $t \equiv (v, e, f)$ of $\mathcal{A}$, the *proper level* of t, defined as the level of vertex v, and the *original level* of t, defined as the level of the coarsest node in $\mathcal{A}$ where we find a tuple (v', e', f') such that $e \subseteq e'$;
- similar information for vertices and tuples of $exp(\mathcal{B}, \beta_{q(i)})$.

Only levels relevant to the overlay (i.e., levels α_k or β_j which appear in some γ_i) are considered. A level α_k which does not appear in any γ_i is "rounded" to α_p where $p = \min\{q \geq k \mid \alpha_q$ appears in some $\gamma_i\}$, and similarily for β_j. The above defined levels are related to $\mathcal{A}$ and $\mathcal{B}$. We trasform these levels into levels related to $\mathcal{C}$ in the following way. If a tuple t has level α_k in $\mathcal{A}$, then the level of t in $\mathcal{C}$ is γ_i where $i = \min\{i \mid p(i) = k\}$; similarily for the tuples of $\mathcal{B}$.

The overlay of $exp(\mathcal{A}, \alpha_{p(c)})$ and $exp(\mathcal{B}, \beta_{q(c)})$ can be computed by using any of the existing algorithms for overlapping plane subdivisions [Boi92, Cha94, Mai88, Gui87]. An optimal $O(n+k)$ time complexity is obtained in the algorithm by Guibas and Seidel [Gui87]. While constructing the overlay, levels must be maintained and updated:

- the level of an intersection point between an edge e_1 of $exp(\mathcal{A}, \alpha_{p(c)})$ and an edge e_2 of $exp(\mathcal{B}, \beta_{q(c)})$ is the finer level between the original level of e_1 and the original level of e_2. This means that each intersection point is considered at the coarsest level where it occurs.
- intersection points split edges of $exp(\mathcal{A}, \alpha_{p(c)})$ and $exp(\mathcal{B}, \beta_{q(c)})$ into pieces. The tuples corresponding to each piece maintain the same original level of the original edge. The proper level of every new tuple is, as usual, the level of the vertex contained in it.

Once the finest overlay has been constructed, the bottom-up computation of the hierarchy starts. At each step, we collect from the current *working subdivision* (initially, the overlay of $exp(\mathcal{A}, \alpha_{ca})$ and $exp(\mathcal{B}, \beta_{ba})$), the tuples corresponding to the nodes of $\mathcal{C}$ at level γ_i, and insert them in the overlay hierarchy; we do this for i decreasing from c to 0. A working example of hierarchy construction is shown in Figure 3.

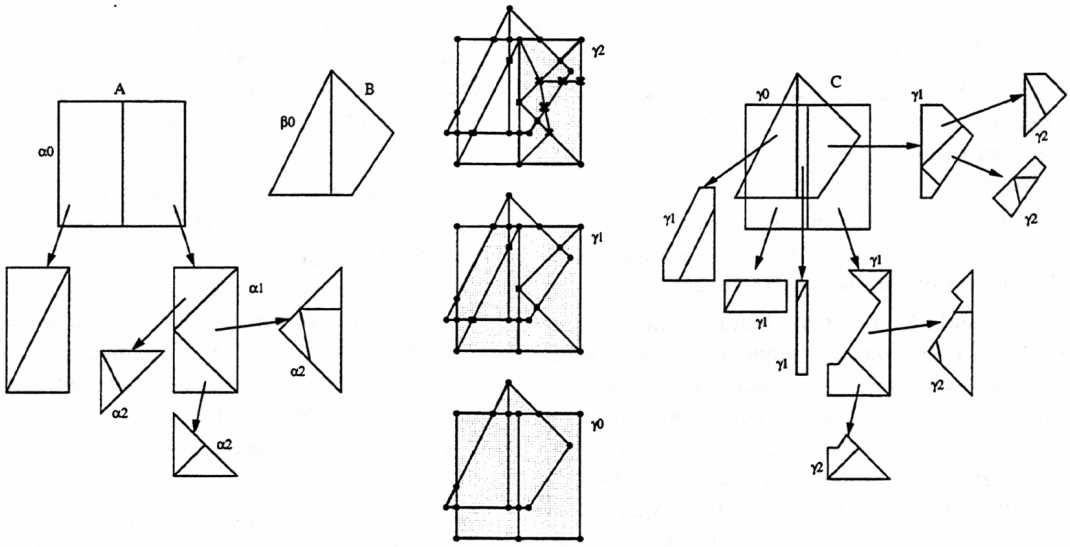

Figure 3: Construction of the overlay hierarchy. From left to right: the two input hierarchies ($\gamma_i = \alpha_i \wedge \beta_0$, for $0 \leq i \leq 2$); the three generalization steps ($\bullet$ = vertices at level γ_0, $\blacksquare$ = vertices at level γ_1, $\times$ = vertices at level γ_2; at each step, the shaded faces are saved); the final result.

Let i be the current level to be constructed. The nodes of $\mathcal{C}$ at level γ_i are the maximal connected components of tuples from the working subdivision such that:

- contain at least one tuple with proper level equal to γ_i (the determination of components actually starts from these tuples);
- the tuples are connected through *switch_vertex* and *switch_edge* (define boundaries of faces);
- the tuples are connected through *switch_face* (link to adjacent face) only if both tuples involved have original level equal to γ_i.

These components are saved to form the nodes of $\mathcal{C}$ at level γ_i. More precisely, for every connected component we initialize a node of the output hierarchy with level = γ_i and whose cell-tuple structure contains a copy of the tuples belonging to the connected component. Later we explain how these new nodes are organized into a hierarchy, i.e., how the list of the macrofaces of a node is created and how the tuples related to a macroface are linked to the appropriate tuples of the node corresponding to the direct expansion of the face.

Actually, we must use some special care to avoid saving a face if its vertices at level γ_i do not define true corners in the polygonal area covered by the face. Details, not discussed here, are contained in [Mag95].

After saving level γ_i of the overlay, we update the current working subdivision by removing all vertices and edges which appeared at level γ_i but do not appear at level γ_{i-1} any more. We first eliminate all tuples with original level = γ_i (type1-tuples) and then eliminate all tuples with original level < γ_i but proper level = γ_i (type2-tuples). The elimination of type1-tuples correspond to true edge deletion, while the elimination of type2-tuples corresponds to merging two adjacent collinear edges into a single one by removing their common vertex.

A type1-tuple t is connected through *switch_vertex* and *switch_face* to other type1-tuples (since they share the same supporting edge). On the contrary, *switch_edge* can connect t to some tuple t' which is not to be deleted: the *switch_edge* link form t' must be updated. We consider maximal chains of tuples of type1 conncted through *switch_edge* and *switch_face* links, alternately (these tuples turn round a vertex). If the chain is not closed, it starts and ends into two tuples t_1 and t_2 which are not deleted: we connect t_1 and t_2 through *switch_edge*.

A type2-tuple t is connected through *switch_edge* and *switch_face* to other type2-tuples (since they share the same vertex), while *switch_vertex* can connect t to some tuple t' which is not to be deleted. We consider maximal chains of tuples of type2 conncted through *switch_edge* and *switch_vertex* links, alternately (these tuples follow the boundary of a face). The chain starts and ends into two tuples t_1 and t_2 which are not deleted: we connect t_1 and t_2 through *switch_vertex*.

To build hierarchical links, we proceed as follows. When we copy the tuples forming level γ_i into the the overlay hierarchy, we keep into the tuples in the working subdivision a link to their saved copies. While updating the working subdivision to pass from level γ_i to level γ_{i-1}, pointers contained in non-removed tuples are maintained. Later, when we save a tuple $t \equiv (v, e, f)$, with f macro, belonging to level γ_{i-1} of the overlay, t already contains a pointer to the appropriate tuple of the direct expansion of f. The macrofaces of a node of $\mathcal{C}$ (a connected component of tuples) can be found as cycles of tuples connected through *switch_0* and *switch_1*, and containing non-null pointers. The level of each macroface is computed as γ_{k-1}, where γ_k is the level of the direct expansion of the macroface.

It can be shown that our algorithm builds the overlay hierarchy $\mathcal{C}$ in $O(n_c)$ time, where n_c is the size of $\mathcal{C}$ (a complexity analysis is in [Mag95]).

5 Extensions

Hierarchical subdivisions use conservative refinement (an edge can be splitted into segments, but its geometry does not change), which ensures that all intersection points appearing in the upper levels of the overlay hiearchy also appear in the overlay at the highest resolution. This property is exploited by our construction algorithm. More general multiresolution models (e.g., [Ber94]) allow non-conservative refinement, by refining an edge into a polygonal chain. Our algorithm can be extended to deal with this kind of non-conservative refinement, provided that some consistency constraints are verified. If two edges e_a and e_b, belonging to hierarchy $\mathcal{A}$ and $\mathcal{B}$, respectively, intersect, then the chains refining e_a and e_b must intersect as well (i.e., every intersection point at a coarsest level must correspond to some intersection at finer levels). Our algorithm, as described in Section 4, correctly performs the simplification of a chain of non-collinear segments into a single edge during a generalization step. The non-trivial part is about intersection points, whose coordinates must be recomputed when a chain is is generalized into a segment; moreover, several intersection points can collapse into one. Levels of vertices, intersection points and tuples must also be redefined in an appropriate way [Mag96].

We are still investigating the possibility of extending the algorithm to multiresolution models where entities of lower dimensionality can be refined into entities of higher dimensionality (e.g., a point into a region) [Fra94, Pup95].

An open research problem is the extension to three or more dimensions, where hierarchical cell complexes are used for representing structured solid objects or scalar fields, and overlays describe the spatial interference between two objects or the distribution of two fields within the same space.

References

[Ber94] M. Bertolotto, L. De Floriani, E. Puppo, "Multiresolution Topological Maps", *Advanced Geographic Data Modelling*, Netherlands Geodetic Commission, Publication on Geodesy - New Series, 40, 1994, 179-190.

[Boi92] J.D. Boissonnat, O. Devillers, R. Schott, M. Tailland, M. Yvinec, "Application of random sampling to on-line algorithms in computational geometry", *D&C Geometry*, 8, 1992, 51-71.

[Bri93] E. Brisson, Representing geometric structures in d dimensions: topology and order", *D&C Geometry*, 9, 1993, 387-426.

[Cha94] B. Chazelle, H. Edelsbrunner, L.J. Guibas, M. Sharir, "Algorithms for bichromatic line-segment problem and polyhedral terrains", *Algorithmica*, 11, 1994, 116-132.

[DeF92] L. De Floriani, E. Puppo, "A hierarchical triangle-based model for terrain description", *LNCS* 639, 1992, 236-251.

[Fra94] A.U. Frank, S. Timpf, "Multiple representations for cartographic objects in a multi-scale tree - an intelligent graphical zoom", *Computer & Graphics*, 18, 6, 1994, 823-829.

[Gui87] L.J. Guibas, R. Seidel, "Computing convolutions by reciprocal search", *D&C Geometry*, 2, 1987, 175-193.

[Mag95] P. Magillo, L. De Floriani, "Maintaining multiple levels of detail in the overlay of hierarchical subdivisions", *Tech. rep. DISI-TR-95-13*, DISI - University of Genova (Italy), 1995, available at "http://www.disi.unige.it/" (see the first author's page).

[Mag96] P. Magillo, work in preparation.

[Mai88] H.G. Mairson, J. Stolfi, "Reporting and counting intersections between two sets of line segments", *Theoretical Foundations of Computer Graphics and CAD, NATO ASI Series*, F40, 1988, Springer-Verlag, 307-325.

[Pup95] E. Puppo, G. Dettori, "Towards a formal method for multiresolution spatial maps", *LNCS* 951, 1995, 152-169.

[Van93] P. Van Oosterom, *Reactive Data Structures for Geographic Information Systems*, Oxford University Press, 1993.