Graphical Models 75 (2013) 231–246

Contents lists available at SciVerse ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

n-Dimensional multiresolution representation of subdivision meshes with arbitrary topology ☆

Lionel Untereiner *, David Cazier, Dominique Bechmann

IGG Team, ICube, UMR 7357, CNRS, Universite de Strasbourg, Pole API, BP 10413, 300 Bd Sebastien Brant, 67412 Illkirch cedex, France

ARTICLE INFO

Article history:
Received 16 February 2012
Received in revised form 22 February 2013
Accepted 20 March 2013
Available online 9 April 2013

Keywords:
Multiresolution representations
Data structures
Subdivision volumes
Levels of detail
Boundary representation
Adaptive subdivision

ABSTRACT

We present a new model for the representation of n-dimensional multiresolution meshes. It provides a robust topological representation of arbitrary meshes that are combined in closely interlinked levels of resolution. The proposed combinatorial model is formalized through the mathematical model of combinatorial maps allowing us to give a general formulation, in any dimensions, of the topological subdivision process that is a key issue to robustly and soundly define mesh hierarchies. It fully supports multiresolution edition what allows the implementation of most mesh processing algorithms – like filtering or compression – for n-dimensional meshes with arbitrary topologies.

We illustrate this model, in dimension 3, with an new truly multiresolution representation of subdivision volumes. It allows us to extend classical subdivision schemes to arbitrary polyhedrons and to handle adaptive subdivision with an elegant solution to compliance issues. We propose an implementation of this model as an effective and relatively inexpensive data structure.

1. Introduction

An increasing number of applications involve the use of volume meshes supporting sophisticated operations like adaptive subdivision, simplification or remeshing. Such meshes are the keystone of the modeling of physical phenomena and the basis of numerical analysis and optimization [1]. For instance, local refinements and re-meshing techniques [2] have recently been proposed to optimize tetrahedral meshes for elastoplastic simulations. Likewise, isogeometric finite elements analysis [3] has been provided through Catmull–Clark subdivision of hexahedral meshes to improve the convergence of numerical simulations on CAD models with arbitrary topology. On the other hand, recursive subdivision of coarse meshes have been used to define tools for the dynamic deformation of geometric objects [4–6]. Generalizing these subdivision schemes, the Slow Growing Subdivisions [7] is defined in any dimension.

As regards surfaces, the multiresolution paradigm has become very popular. In this context, an object is no longer modeled as a single mesh, but as a sequence of nested meshes encoding distinct levels of details. Such sequences of meshes are obtained either through the subdivision of a given coarse mesh or through an iterative simplification of an initial fine mesh. These principles have respectively led to subdivision surfaces [8] and progressive meshes [9] which are the basis of many works in geometry processing and mesh compression. In this framework, the adaptivity of the representation is crucial as it allows strong reduction in the size of the multiresolution objects.

Finally, many works in the meshing community address the issue of automatic mesh generation and optimization. Among others, Delaunay and Voronoi tessellation are popular techniques [10,11]. They basically provide unstructured tetrahedral meshes or arbitrary polyhedral meshes, dual from each other. In higher dimension, they generalize

☆ This paper has been recommended for acceptance by Peter Lindstrom and Bruno Eric Levy.

* Corresponding author.
E-mail address: lionel.untereiner@unistra.fr (L. Untereiner).

1524-0703/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.gmod.2013.03.003

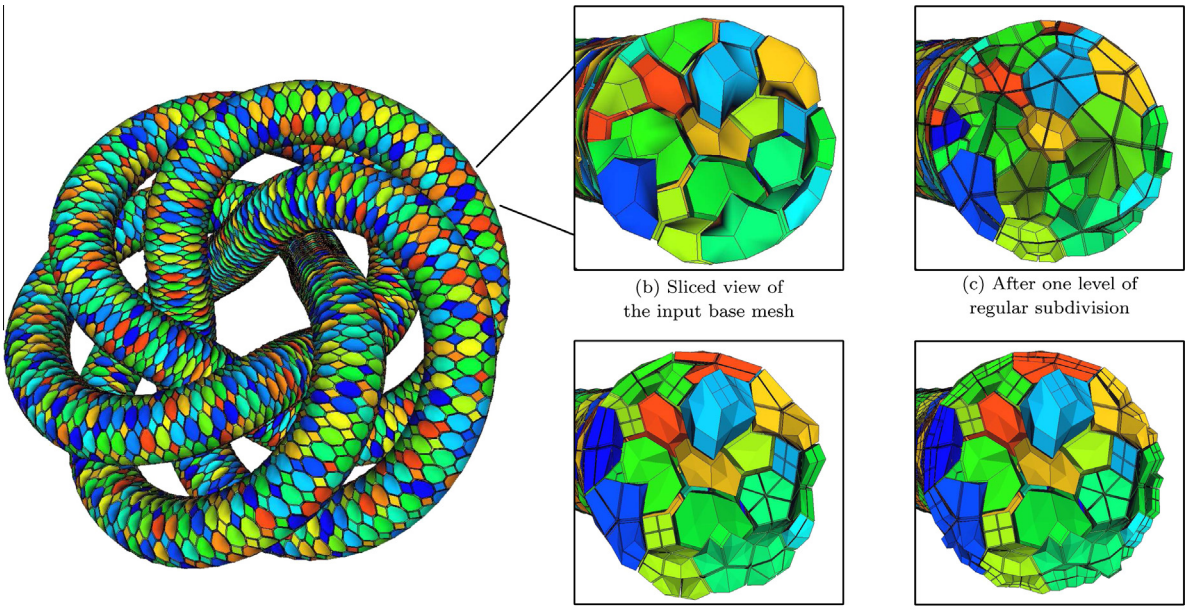


Fig. 1. Example of subdividing an arbitrary polyhedral volumetric mesh with our method. Colors of the cells were kept at a level to another in order to identify the subdivided cells. The adaptive subdivision consists, in this example, in refining the boundary cells at each level.

as simplicial meshes and their dual representations. How to generate such base meshes is not discussed further here and we refer the reader to [12] for a complete survey. Nevertheless, their increasing use as the basis of multiresolution applications strengthens the need for efficient and generic multiresolution representation of n-dimensional meshes.

Many data structures have be proposed to encode level of details for triangular or quadrangular subdivision of surfaces [13]. For higher dimensions, two kinds of representations stand out from the literature. The first ones are based on octrees. Benefiting from efficient implementation, they are limited to hexahedral meshes with ad hoc topologies. The second ones consist in hierarchies of tetrahedrons. Able to handle more general topologies, they are not real multiresolution representations as they maintain a unique mesh at the finest level of resolution as well as a set of coarsening operations that are applied on demand. In both frameworks, adaptivity generates non-conforming meshes that are not representable with usual data structures.

In the field of geometric modeling, the n-dimensional combinatorial maps made popular by the works of Lienhardt [14] and Dufourd [15], define a formal model[1] that possesses the sought-after genericity, expressiveness and potentials. It forms a solid basis for the targeted multiresolution model.

In the following, we present a new model for the representation of n-dimensional multiresolution meshes. This model extends the 2-manifold model presented in [17]. It defines a set of nested meshes representing closely interlinked levels of resolution that can be randomly and independently accessed. Each level of resolution models the topology of a n-dimensional manifold mesh. We present operations to build and traverse the cells of such hierarchies of meshes along with their neighborhoods.

The genericity of the n-dimensional maps allows the representation of a wide variety of meshes (Fig. 1) and subdivision schemes and naturally supports adaptive subdivision without compliance issues (see Fig. 6b and c). The proposed model fully supports multiresolution edition – i.e. every cells of every level of subdivision may store specific geometric embeddings or details – what allows the implementation of tools like analysis and synthesis of multiresolution data as required for many mesh processing algorithms.

Thanks to the formal model of combinatorial maps, complicated subdivision operations are decomposed into simpler ones. This allows us designing sophisticated algorithms using robust *low-level* topological operations – as for instance edge cutting or face splitting – acting on arbitrary meshes. We illustrate this point with a detailed presentation of a subdivision algorithm for arbitrary polyhedrons. That provides an efficient and elegant implementation of subdivision volumes.

The remainder of this paper is organized as follows. Section 2 reviews related work on subdivision volumes and multiresolution models. Section 3 presents and illustrates our multiresolution model. Sections 4–6 details the subdivision algorithm, the handling of adaptive and mixed subdivision schemes. In Section 7 the performances of our model in terms of time and space complexity are

[1] Here the term *model* refers to the underlying mathematical model that describes the meshes in terms of combinatorial topology [16]. Concrete implementations of this model provide data structures that encode the meshes.

discussed. Section 8 concludes with an electro-thermal simulation application.

2. Related work

2.1. Subdivision volumes

Given a volumetric base mesh M^0 and a subdivision process S, the recursive process defined by $M^{i+1} = S \cdot M^i$ produces what is usually called *subdivision volumes*. The built meshes converge to a limit mesh M^∞ guaranteed to be a smooth manifold. The subdivision process is usually separated into two stages. First, the topology of the mesh is refined, which comes to insert some *cells* (vertices, edges, faces or volumes) as well as their corresponding neighborhood relations.

Secondly, the geometry is updated: the subdivision stencil is applied to the inserted vertices for an interpolating scheme and to all vertices in case of an approximating scheme. The weights used in the subdivision stencils are designed to control the continuity of the generated surfaces and volumes. Many works focus on the convergence and the continuity of specific subdivision schemes [18,19].

Our work focus on the topological refinements and in particular on the primal schemes. For subdivision surfaces, primal schemes reference models are those of Loop [20] and butterfly [21,22] for triangular meshes and Catmull–Clark [23] for quad or polygonal meshes. In the volumetric case, the natural extension of those schemes consists in three steps. First, the faces of every polyhedron are subdivided. Secondly, new vertices are inserted inside all polyhedrons. Finally all inserted vertices are connected with edges to define the resulting subdivided polyhedrons (see Fig. 2).

The main proposed schemes for subdivision volumes are the Loop scheme adapted to tetrahedral [24] and octahedral [6] meshes, the Catmull–Clark scheme generalized to polyhedral [4] meshes and the adaptation of the $\sqrt{3}$ scheme to tetrahedral meshes [25]. Pascucci generalized the $\sqrt{2}$ scheme to any dimensions [7] with the $\sqrt[3]{2}$ scheme that supports adaptive subdivision. As we focus on the subdivision process – through topological refinement – we do not discuss further the geometrical properties of the mentioned subdivision schemes and simply use, in the following, the scheme proposed by MacCracken et al. [4].

The data structure used in the mentioned works imply limitations on the refinement process. First, they are specific to one kind of mesh – usually tetrahedral/octahedral meshes or hexahedral meshes – and thus limit the subdivision rules to those that generate representable polyhedrons. Secondly, the related schemes rarely support adaptive refinements because they imply compliance issues. Finally, these data structures only encode the current mesh and none of them provide true multiresolution tools. Our approach provide an elegant solution to these issues.

2.2. Multiresolution data structures

Many studies dealing with multiresolution meshes have focused on tetrahedral meshes. In this context, a

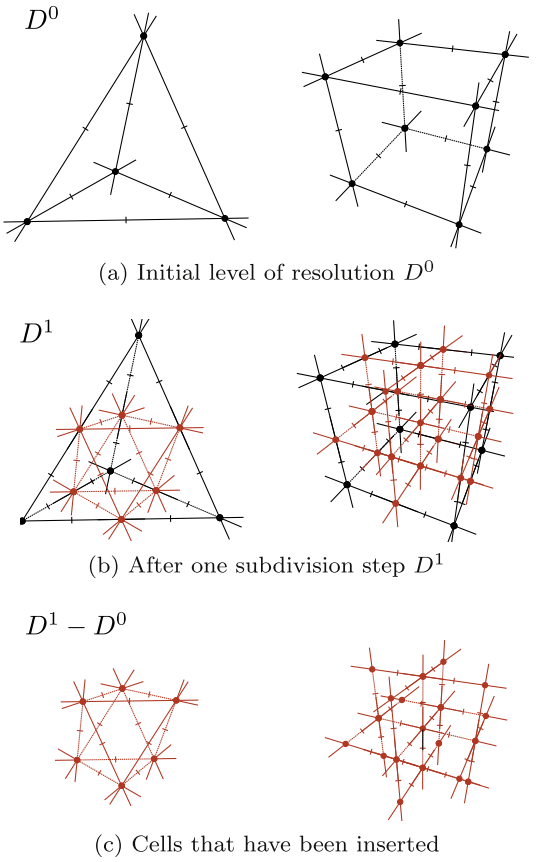


Fig. 2. Tetrahedral and hexahedral subdivisions.

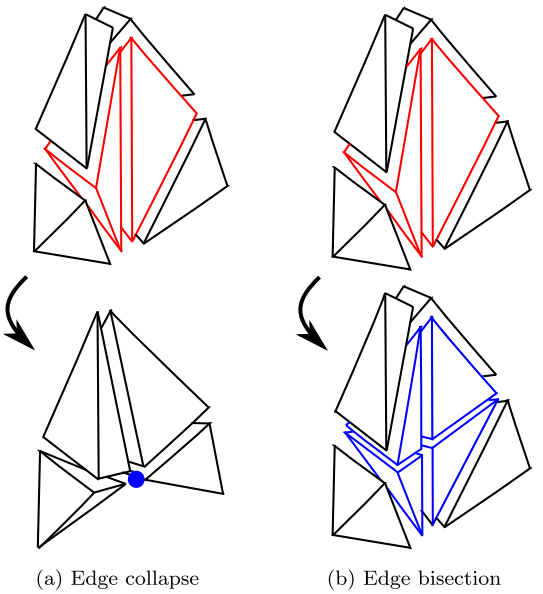


Fig. 3. Tetrahedral hierarchies using edge collapse or edge bisection. The deleted or subdivided tetrahedrons are shown in red. The resulting cells (vertex or tetrahedron) are drawn in blue.

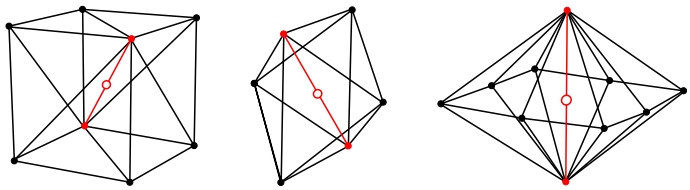
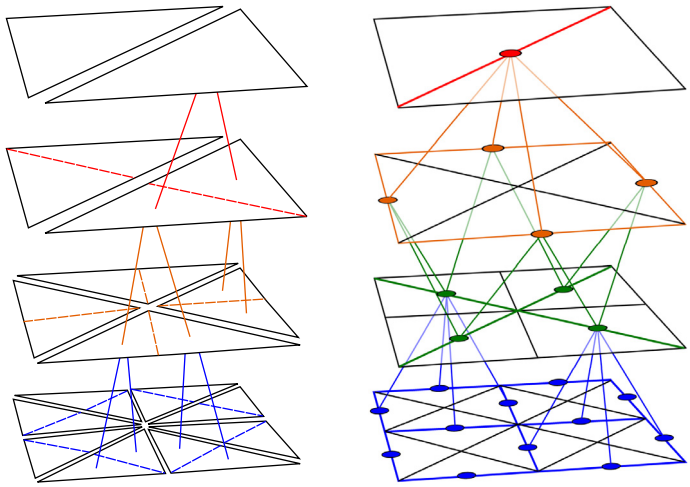


Fig. 4. Three classes of diamonds in 3D formed by 6, 4 and 8 tetrahedrons and respectively 8, 6 and 10 vertices. The bisected edge is shown in red.



(a) Triangles dependencies are encoded as a forest of binary trees.

(b) Diamonds dependencies are encoded as a DAG.

Fig. 5. Comparison in 2D between two data structures whose hierarchy is generated by an edge bisection operation.

regular hierarchy may be generated by mesh decimation using edge collapse operators (see Fig. 3a). The half-edge tree proposed in [26] encode such hierarchy in a binary forest of vertices. Historically, this work is among the first that proposed real multiresolution management of tetrahedral meshes. This approach is well suited for progressive transmission, but not for multiresolution edition.

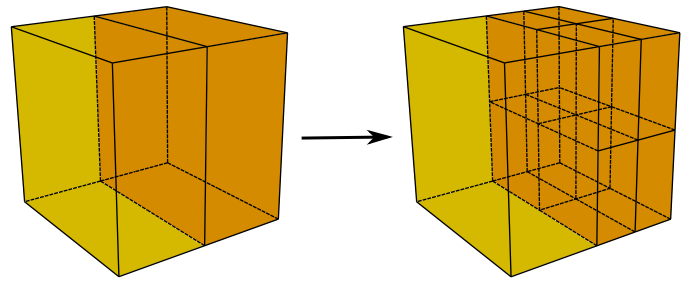
Other approaches allow a tetrahedral hierarchy construction by bisecting tetrahedrons along their longestedge [27] (see Fig. 3b). Two ways to encode such hierarchies have been proposed: binary forests of tetrahedrons (see Fig. 5a for a 2D example) or directed acyclic graphs of diamonds [28]. A diamond is the set of simplexes incident to a bisected edge (see Fig. 4). The nested relationship between simplexes induce a hierarchical relationship between the diamonds (see Fig. 5b with a 2D example).

Both hierarchies of tetrahedrons are encoded as a base mesh enriched by a sequence of collapse or subdivision operations. To be accessed, a level of resolution have to be built, applying a subset of the operations on the base mesh. Thus, random access to distinct levels requires many operations that prevent interactive edition of such data structures. Moreover these approaches prohibit the storage at each level of resolution of the details data that are needed for many geometry processing algorithms.

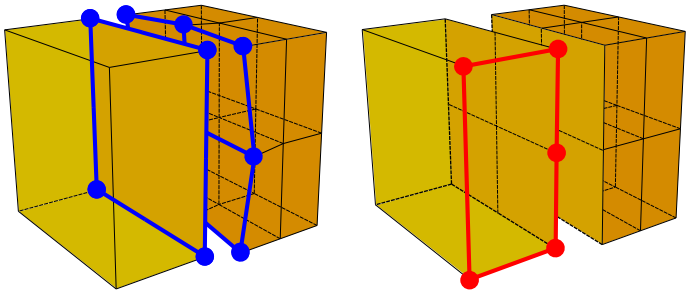
Other data structures have been proposed in the specific context of subdivision volumes. Here again, only the subdivided mesh is represented without level of details. McDonnell et al. [19] use a simplified version of the radial-edge data structure which is a generalization of the winged-edge structure to arbitrary complexes. Bajaj et al. [18] use a recursive representation of a d-dimensional cube (a d-cube), where a 0-cube is a vertex index and a d-cube ($d > 0$) a list of two $(d-1)$-cubes. This light representation does not contain neighborhood information. Thus, computing the new vertices positions has a complexity of $O(n^2)$ where n is the number of vertices in the mesh, making it unsuitable for interactive mesh processing.

Meanwhile, a conventional solution to encode multiresolution subdivision surfaces is to use a quadtree as a data structure [9]. This solution can be extended to the volumetric case by using an octree to encode multiresolution subdivision volumes, which is naturally derived from the nested hierarchy of volumes generated by the subdivision process. But this structure is not used in practice for multiresolution subdivision volumes because it does not represent all the cells of a mesh (vertices, edges, faces, volumes) as explained below.

All the mentioned combinatorial models are limited to a specific class of polyhedrons whose faces degrees are fixed (square or triangle). That limits the possible subdivisions to those that avoid the emergence of non-conforming surfaces (see Fig. 6a and b). A common solution in the context of surfaces is to add special cells around the so-called T-



(a) Adaptive refinement of a hexahedron



(b) In an octree (c) In a 3-map

Fig. 6. Adaptivity management of a hexahedral refinement (a). With an octree (b), non-conforming generated faces cause the appearance of a topological hole visible in blue (vertices incident to this hole were moved to make them visible). In a 3-map (c) compliance is maintained as arbitrary polyhedrons can be handled. The generated polyhedron has 9 faces, 5 four-sided faces and 4 five-sided faces (one is visible in red in c).

junctions. In the volumetric case, the resulting combinatorics of cells is intricate and therefore the computation of neighborhood. On the other hand, this limits the possible refinement rules as they have to ensure that the generated volumes are representable with those special cells.

Finally, the octree based representations are the only ones that provide real multiresolution capability. But, by nature they impose an orientation of the meshes along three axes. This make difficult to properly handled complex geometries that do not feat to an hexahedron.

3. Multiresolution combinatorial model

A n-dimensional mesh is a discretization of a region in a manifold cellular complex composed of cells at different dimensions (vertices, edges, faces, volumes, ...) connected by adjacency relations.

The facet-edge data structure [29] is one of the first that has been proposed to implement this combinatorial model in dimension 3. A formalization of this structure has been proposed [15,14], leading to the mathematical model of n-dimensional combinatorial maps. A n-map M is a $n+1$-tuple $(D, \alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ where D is a finite set of darts abstracting the notion of oriented half-edges, α_0, α_1, ..., α_n are relations betweens darts such that:

- α_0, α_1, α_{n-2} are involutions on D: $\forall i < n-1$, $\alpha_i(\alpha_i(x)) = x$, i.e. $\alpha_i(x) = y \Rightarrow \alpha_i(y) = x$. In other words, darts are paire-wise linked.
- α_{n-1} is a permutation on D. In dimension 3, that implies that relation α_2 cyclically links faces around the edge they are incident to.

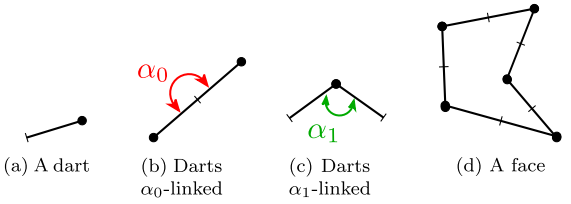


Fig. 7. Drawing conventions for a 3-map.

- $\alpha_j \circ \alpha_i$ is an involution $\forall i < n-2$ and $\forall j$ such that $i+1 < j \leqslant n-1$. Thus, $\alpha_2 \circ \alpha_0$ is an involution and both darts of an edge are symmetrically linked (see Fig. 8).

In dimension 3, following the conventions of Fig. 7, two darts linked with α_0 make a *simple edge* (see Fig. 7b) and two darts linked with α_1 make a *simple vertex* (see Fig. 7c). A face (see Fig. 7d) is a cycle of simple edges linked by simple vertices.

The α_2 relation links two faces along a simple edge (see Fig. 8). A cycle of α_2-linked faces forms an *edge* and the set of simple vertices that share α_2-linked simple edge forms a *vertex*. In higher dimension, the α_i relation links two i-cells along the $(i-1)$-cells of their boundaries. More details about the definition of i-cells and their neighborhood relations can be found in [15].

These topological relations define ways to traverse the cells of a map – its vertices, edges, faces and volumes. The main cell traversals in dimension 3 are illustrated Fig. 9. The edges of a face are traversed through the

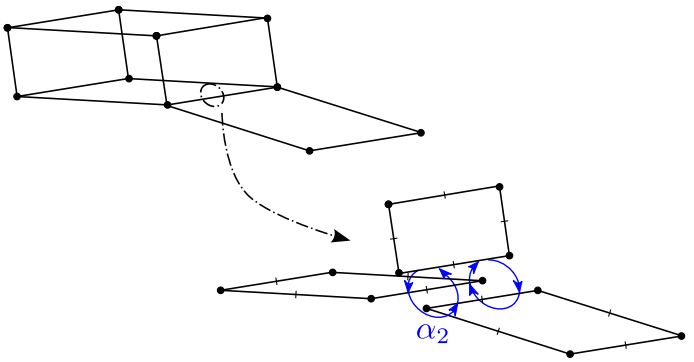
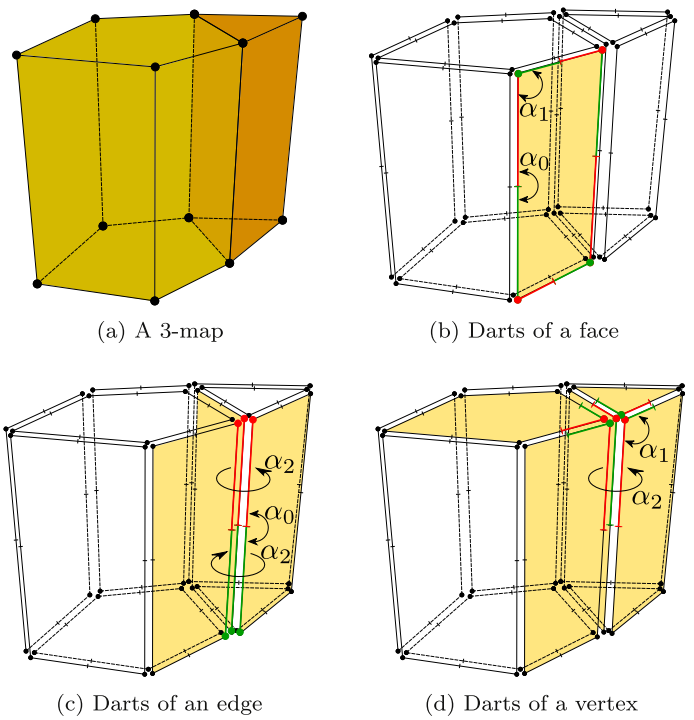


Fig. 8. Focus on faces linked with α_2.



(a) A 3-map (b) Darts of a face

(c) Darts of an edge (d) Darts of a vertex

Fig. 9. The main topological cell and their traversal in a 3-map. During this walkthrough, the cells of higher dimension are accessed in turn (some of them are highlighted in yellow above).

composition of relations α_0 and α_1 (see Fig. 9b). The faces incident to an edge are traversed through the α_2 relation as well as the volumes incident to these faces. Finally, the edges incident to a vertex are traversed, in a nonordered way, alterning the α_1 and α_2 relations (see Fig. 9d).

The n-dimensional multiresolution combinatorial maps extend the multiresolution 2-maps introduced in [17]. A multiresolution map M is formelly defined as follows:

$$M = \left(\{D^i\}_{i\geqslant 0}, \{\alpha_0^i\}_{i\geqslant 0}, \{\alpha_1^i\}_{i\geqslant 0}, \ldots, \{\alpha_{n-1}^i\}_{i\geqslant 0}\right)$$

where for all $i \geqslant 0$, $M^i = \left(D^i, \alpha_0^i, \alpha_1^i, \ldots, \alpha_{n-1}^i\right)$ is a n-map representing the mesh at level i. Intuitively, a multiresolution 3-map represents a hierarchy of volume meshes.

To avoid any redundancy, the darts of coarser levels are reused in the finer levels. Thus, the sequence $\{D^i\}_{i\geqslant 0}$ is composed of sets such that for all $i \geqslant 0$, $D^i \subset D^{i+1}$. In other words, each D^i contains the darts inserted until level i. This leads to a data model in which the memory cost is controlled as shown in Section 7. This aspect has already been shown in Fig. 2 where[2] the red darts correspond to the cells inserted during the subdivision and the black darts are those that are reused from one level to another.

The $\{\alpha_k^i\}_{i\geqslant 0}$ represent the versions of the relations α_k ($k \in \{0, 1, \ldots, n-1\}$) for different resolution levels of the

[2] For interpretation of color in Figs. 2, 6, 9, 16, and 17, the reader is referred to the web version of this article.

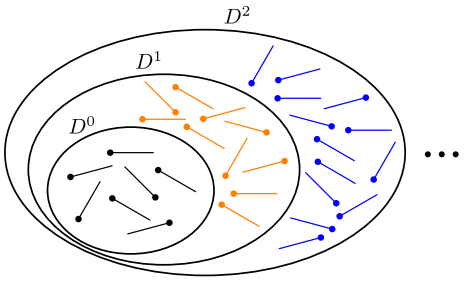


Fig. 10. Sets of darts from a multiresolution map.

n-map. They are constructed so that the α_k^i restriction to darts from level i make up a valid n-map.

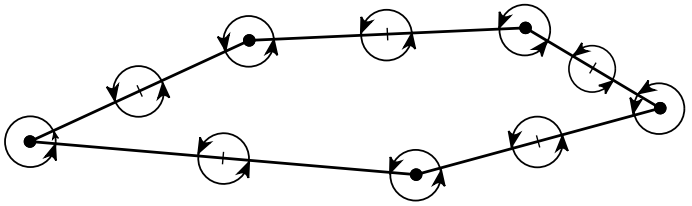
Specifically, in a multiresolution map, the darts are arranged in a sequence of nested sets as shown in Fig. 10. As the topological relations $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ change from one resolution level to another, each dart stores the values of its relations for each level in which it exists.

Concretely, in a map with l_{max} subdivision levels, a dart that has been inserted in the ith level, stores an array of $l_{max} - i$ versions of these relations. Thus the darts from the finest level that represent about 87% of the darts (seven darts over eight in the best case as detailled in Section 7), store just one version of the α_k relations. Indeed, the last darts generated by the subdivision process are the most numerous. The coarse level darts are fewer (because the base mesh has the fewest darts) and store l_{max} versions of these relations. This is what makes our model so competitive as regards to the memory cost. This concept is illustrated Fig. 11 for only one face for readability.

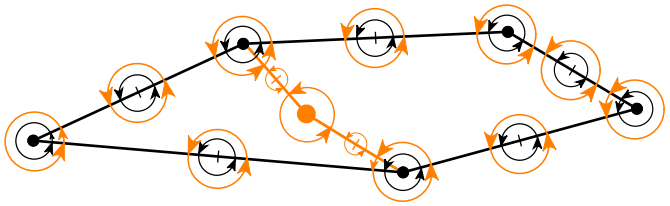
Robust n-dimensional low-level topological operations acting on arbitrary meshes have been presented in [15,14]. In our multiresolution model, each level of details is represented as a n-map and so naturally support all usual topological operators. Building a hierarchy of n-dimensional meshes is thus straightforward. Starting from a hierarchy with l levels of details, level $l+1$ is built by first duplicating the lth level and applying the mentioned low-level operators to subdivide the required cells. The difficulty comes from the increasing combinatorics of the cells sizes and neighborhoods that have to be traversed during such subdivision. The subdivision algorithm is presented in the following section.

There are several ways to implement the model of multiresolution maps. The simplest one is the following. A dart is a record that contains an array of pointers to other darts. There is one pointer for each topological relation and the size of those array depend on the level in which the dart has been inserted. A map is then an array of such darts. This method is simple and adapted for rapid prototyping, but it induces memory fragmentation and requires some optimizations to efficiently handle large meshes.

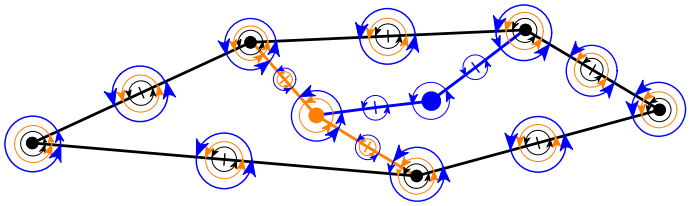
Another implementation consists in using groups of arrays. Here, darts are encoded as integer (indexes in those arrays). There is one group of arrays for each level of details. Each group contains one array for each topological



(a) 0-level relations



(b) 1-level relations



(c) 2-level relations

Fig. 11. Illustration of the storage of different versions of the topological relations in a face of a multiresolution 3-map.

relation. This implementation is available in C++ in the CGoGN [30] library. It improves the memory usage and is compatible with specific data structure used on GPU such as OpenGL's Vertex Buffer Object (VBO) – used for the rendering of such meshes – or other specific implementation with out-of-core issues.

The combinatorial relations that link darts and their combinations represent all possible incidence relationships between cells. They allow optimal traversals of complex neighborhoods for each cell (vertices, edges, faces, volumes). Our implementation is well-balanced between memory cost and efficiency. Nevertheless, in the case of meshes with fixed combinatorics (e.g. tetrahedrons or hexahedrons) we can be more efficient. In this case, the darts may be allocated by blocks corresponding to the fixed combinatorics of the cells. For example, in the case of tetrahedral meshes, we can allocate directly enough darts for each face or even for each tetrahedrons. The incidence relations in those cells can be expressed in an implicit manner.

The formal expression we use for our topological model implies that the algorithms we present below remains the same whatever the used implementation.

4. Arbitrary polyhedron subdivision

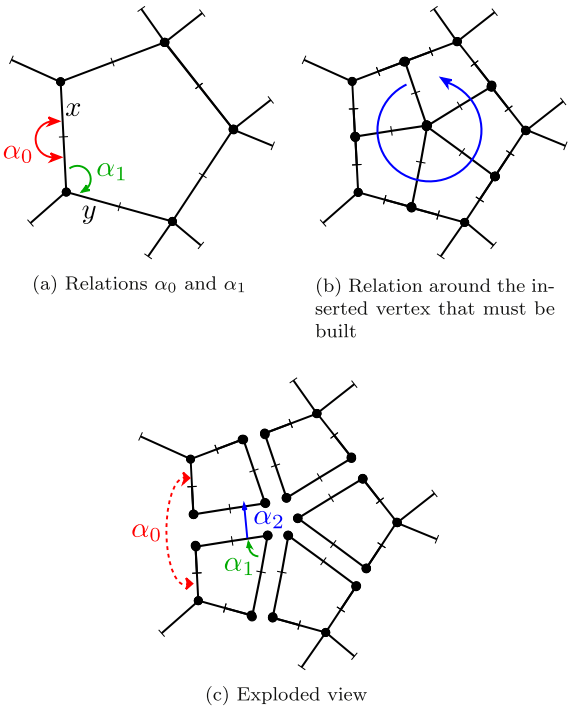
We propose in this section a subdivision scheme suitable for any kind of polyhedron, thus unifying the subdivision schemes described above. The objective is to demonstrate the genericity qualities of our topological model. We will see later how this genericity allows an adaptive subdivision of any volume meshes, solving all compliance issues correctly.

The subdivision method consists in inserting a vertex at the polyhedron center. This vertex is connected by edges to the centers of the faces. Then each vertex in the center of a face is connected to the edge midpoints making up the face. After this first subdivision step, the faces linking all these edges with the volumes bounded by these faces must be built. Finally, the last stage consists in generating all adjacency and incidence relations between all the created cells.

When looking at arbitrary polyhedrons (i.e. those which have an arbitrary number of faces and degree of each face) this last stage is by far the most complex combinatorially. The works on volume subdivision previously mentioned are limited to the subdivision of hexahedrons into 8 octants, resulting in a case where the topology of the inserted cells is invariant. We show below how the topological information present in our mathematical model can efficiently retrieve the order and orientation of added cells.

4.1. Edge order around a vertex

At first we consider the subdivision of a face and the order of edges around the inserted vertex. Fig. 12a shows a pentagonal face with the relations α_0 and α_1 in red and green. Fig. 12b shows this face subdivided with a central vertex, which involves finding around the central vertex the edge order (represented by the blue arrow).



(a) Relations α_0 and α_1

(b) Relation around the inserted vertex that must be built

(c) Exploded view

Fig. 12. Relations to rebuild in a subdivided face.

Fig. 12c shows an exploded view of the generated faces. It can be noticed that the combination $\alpha_1 \circ \alpha_2$, from the relation α_2 connecting two adjacent faces and the relation α_1 between two edges of a face, allows the traversal of the edges as desired. Thus, to reconstruct the order of the vertices, it is sufficient to generate the correct α_2 relation between the added faces. We notice that these relations are directly linked to relations with α_0 from subdivided edge darts (see red dotted arrow in Fig. 12c). The strength of our algorithm is the ability to alternate between primal and dual representation to propagate this information during this subdivision step. Let us note that for this to happen, we need two levels of subdivision to be able to coexist in the model.

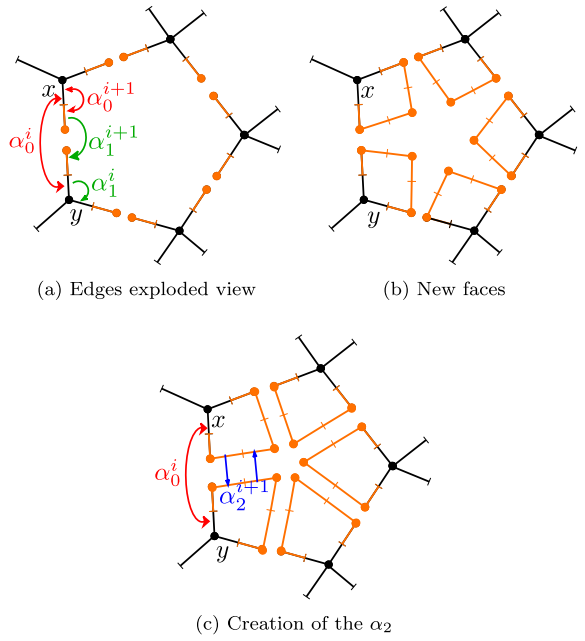
Now, we describe the algorithm for an i-level face subdivision. As a first step, all edges are cut into two by adding two $i+1$ level darts as shown in orange in Fig. 13a. For all level i, this step reconstructs the relations as follows (with $y = \alpha_1^i(\alpha_0^i(x))$):

$$\alpha_1^{i+1}\left(\alpha_0^{i+1}(x)\right) = \alpha_0^{i+1}\left(\alpha_1^i(y)\right) \tag{1}$$

The vertices from the original face form separate corners visible in Fig. 13a. The second step consists in forming the emerging faces by connecting the vertices of level $i+1$ by two edges, meaning 4 darts at level $i+1$ (Fig. 13b).

The final step (see Fig. 13c) consists in sewing by α_2 at level $i+1$ the pairs of edges attached to darts connected by α_0 at level i (black darts). For all level i, this step reconstructs the relations as follows (with $y = \alpha_1^i(\alpha_0^i(x))$):

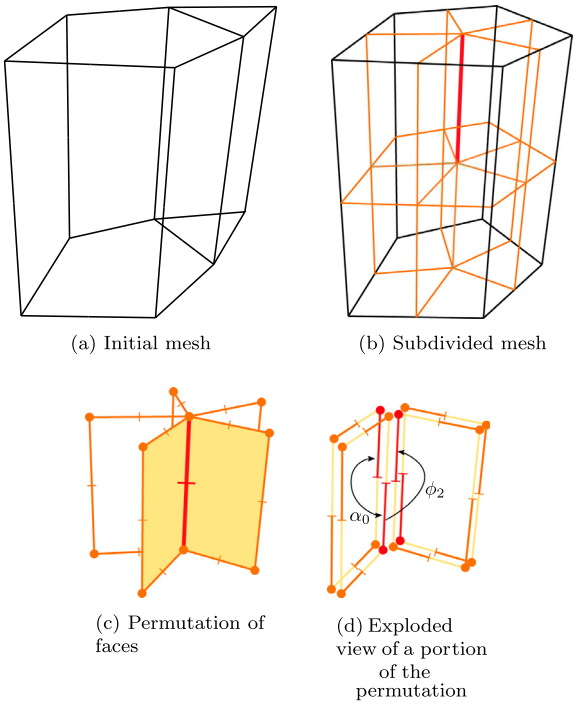
$$\alpha_2^{i+1}(\alpha_1^{i+1}\left(\alpha_0^{i+1}(x)\right) = \alpha_1^{i+1}(\alpha_0^{i+1}\left(\alpha_1^i(y)\right) \tag{2}$$



(a) Edges exploded view (b) New faces

(c) Creation of the α_2

Fig. 13. Face subdivision steps.



(a) Initial mesh (b) Subdivided mesh

(c) Permutation of faces (d) Exploded view of a portion of the permutation

Fig. 14. Faces permutation around an edge.

Finally, in a fully implicit manner, the order of edges around the central vertex is reconstructed regardless of the original face degree, so that this algorithm is generic because it allows subdividing any face whatever its degree. This principle is extended to the volume in the next section.

4.2. Face order around an edge

We now consider the subdivision of a volume as shown in Fig. 14. In particular, we focus on the order of faces around the edge highlighted in red Fig. 14b. We note $\phi_2 = \alpha_2 \circ \alpha_0$. We know from the definition of a 3-map that this relation is an involution, which means that $\phi_2(\phi_2(d)) = d$ for every dart d. In the same way that the order of edges around a vertex was divided into two relations, we decompose the permutation α_2 to rebuild it in $\alpha_2 = \phi_2 \circ \alpha_0$. Fig. 14c shows the faces around the edge highlighted in red. Fig. 14d shows an exploded view of the faces around the same edge. We observe that the combination $\phi_2 \circ \alpha_0$ allows the traversal of the faces in the desired manner. Thus, to reconstruct the sequence of faces, it is sufficient to correctly generate the relations α_0 between added faces.

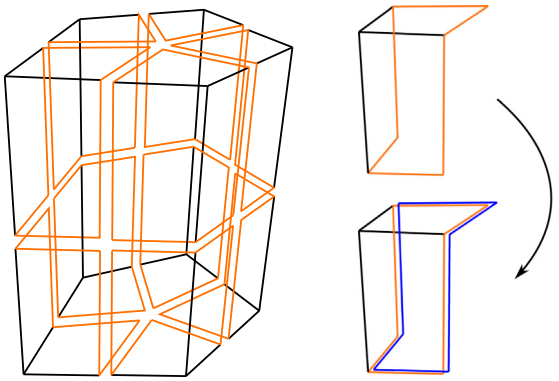
4.3. Topology of a subdivided polyhedron

The tools presented above are used to elegantly define any subdivision of a polyhedron. First, the faces are subdivided by adding a central vertex. At this step, sewing edges around every central vertex is not performed. This results in an umbrella (or corner) composed of quadrangular faces (see Fig. 15a) around each initial vertex. In a second step, this umbrella is automatically completed to form the future volume. To do this, a topological face is created along the umbrella's edges (see Fig. 15b). This topological face is then in its turn subdivided by adding a central vertex which is positioned in the center of the subdivided volume (see Fig. 15c).

At this point, we generate a volume per initial vertex of the polyhedron (see Fig. 15d). Relations left free can now be completed so as to order the internal faces from the created volumes around their common edge (see green arrow Fig. 15d). Specifically, for each pair of vertices from the initial polyhedron, we consider the two darts x and y sewn by α_0 at level i. Their images by α_0^{i+1} are the vertices of the two incident faces. These two faces are merged by sewing one out of two darts with the relation ϕ_2 introduced above. After this step, all the topological relations between created cells (vertices, edges, faces, volumes) are reconstructed, which demonstrates how much this algorithm is generic, since it can subdivide any volume whatever its degree.

4.4. Subdivision in higher dimension

The subdivision process naturally extents to any dimension in a recursive way. We give here an insight of the algorithm. As before, we define the relation $\phi_{n-1} = \alpha_{n-1} \circ \alpha_0$ that represent the cycling order of $n-1$-cells around of $n-2$-cells in a n-dimensional map. The subdivision of an hyper-volume v of dimension n at a level of resolution l begins with the subdivision of the $n-1$ cells forming its boundary. The sewing of the generated $n-1$-cells around the central vertex is not done. We obtain sets of linked $n-1$-cells forming $n-1$ dimensional umbrellas around



(a) Creation of the umbrellas

(b) Closing an umbrella with a topological face

(c) Subdivision of the topological face

(d) Volumes after subdivision

Fig. 15. Topological subdivision.

the initial vertices of v with open boundaries where the sewing have been prevented. Those umbrella are closed with $n-1$-cells acting as covers. Those covers are then subdivided at dimension $n-1$. That generates all the cells that are needed to build the central vertex. Those cells are sewn with relation α_0. To find the couples of cells that have to be sewn at level $l+1$, the α_0 relations at level l are queried. Thanks to the combinatorics of n-maps, that implicitly reconstruct the introduced ϕ_{n-1} relation and thus all required neighborhood relations between the cells inserted at level $l+1$.

5. Generic adaptive subdivision

The idea of an adaptive subdivision is to choose, according to given criteria, the polyhedrons to subdivide and those that will not be subdivided. To do this, it is necessary to treat correctly the cells, edges or faces, incident to polyhedrons having different subdivision levels. Otherwise, compliance issues already mentioned can occur. As described before, combinatorial maps have a strong asset to maintain this compliance. Unlike data structures based on incidence graphs or data structures describing explicitly

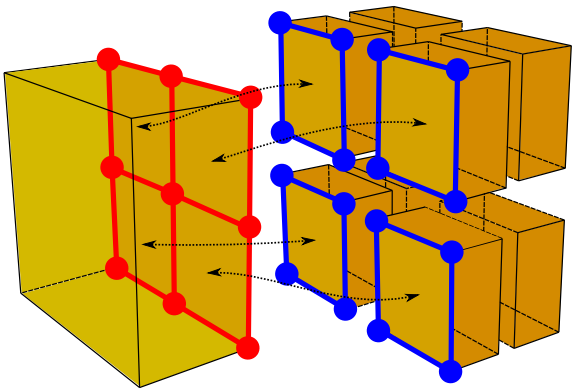


Fig. 16. One face shown at two levels of subdivision.

the cells (such as tetrahedral data structures), the cells of a map are defined implicitly. These implicit definitions are illustrated Section 3 with the topological cell traversal functions figure.

To be more precise, in a multiresolution 3-map, a face between two polyhedrons is common to both. The subdivision from a level i to a level $i+1$ of one of the polyhedrons leads to the subdivision of the common face on both sides. So, the darts of this face seen from the not subdivided polyhedron are traversed at the level i. The face is still seen as being not subdivided. Conversely, the darts of this face seen from the subdivided polyhedron are traversed at the level $i+1$. Thus, the same face is seen as being subdivided. Specifically, the darts path shows n faces of level $i+1$, where n is the degree of the initial face. For example in Fig. 16, the face adjacent to the two polyhedrons is seen as four 4-sided faces in the polyhedron on the foreground (shown in red) but also as one 4-sided face for each of the four hexahedrons in the background (shown in blue).

As exposed above, a face can be seen in a different way following the level from which we look at this face. It is necessary to know and be able to compute the level of this face. Let us define $lev(b)$ the level of dart b. So, to determine the level of a face f in the i-level 3-map, we start from one of its darts b with the smallest level of insertion into the map $lev(b) \leqslant i$. Indeed, the level of this face is necessarily between the minimum of insertion levels of its darts and the level i from the considered 3-map. In order to test from the lowest to the highest level, we start with one of the lowest inserted dart. Firstly, we test if the darts from the face of dart b at level $lev(b)$ are contained in face of b at level i. If yes, then f is a face of level $lev(b)$ inside the map of level i. Otherwise, we test if the darts of the face of b at level $lev(b)+1$ are contained in the face of b at level i. We loop over this level test until finding the lowest level for which all darts from the face of b are contained in the face f at the level i. At most, this level is the level i itself.

For example, for the face f_1 (see Fig. 17), we start with the dart b_1 inserted in the 0-level map and then we begin the traversal of the face of b_1 at level 0 (darts in black). Since all the darts are contained in the face of b_1 at level 2 (darts black, orange, blue). In other words, we traverse the whole face and come across only darts of level 0. We

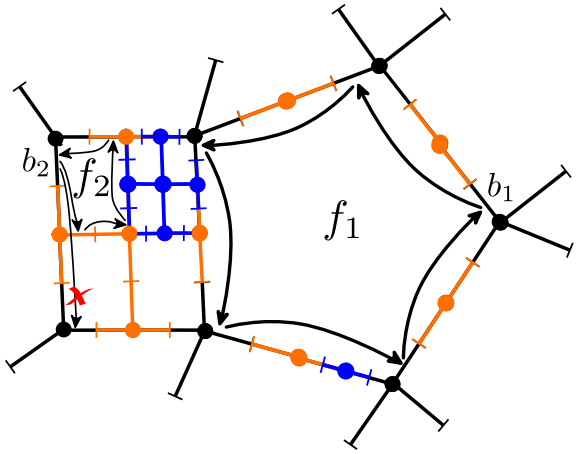
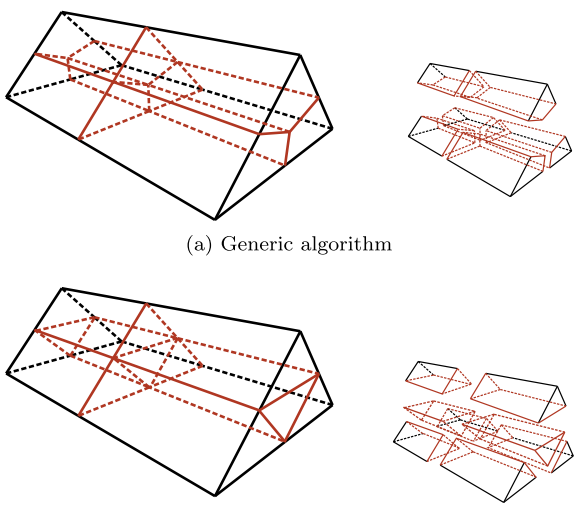


Fig. 17. Example of face level computing: illustration for the faces f_1 and f_2, starting respectively with dart b_1 and b_2.

can conclude that f_1 is a face of level 0. For the face f_2, we start from the dart b_2 inserted in the map at level 0. The next dart in the face of b_2 at level 0 is not contained in face of b_2 at level 2. So, we continue to traverse the face of b_2 at level 1 and we see that all darts from the face are contained in the face of b_2 at level 2. We can conclude that f_2 is a face at level 1. In other words, with the darts of level 0, we change face but not with darts of level 1. To remain the same face we have to consider it at level 1.

Thus, to describe clearly what an adaptive subdivision formally is, we have to define the notion of subdivision level of a cell in a map of level i. We call $cell_k^i(b)$ the set of darts of the k-cell from dart b at level i. We define the subdivision level of the k-cell from dart b to be equal to the lowest level j such as $cell_k^j(b') \subset cell_k^i(b)$, with $b' \in cell_k^i(b)$. Specifically, the inclusion in the formula above means that the level of a cell is defined as the smallest level such that the border of the cell can be subdivided but not the cell.

Using this information on the level of a cell, the adaptive subdivision of a polyhedron can be written fairly simply. To subdivide an i-level polyhedron at level $i+1$ the algorithm starts by subdividing each face of the polyhedron. The subdivision of a face starts with the searching for its level. Indeed, if the face is already above level $i+1$ because the adjacent polyhedron is already subdivided then the algorithm proceeds to the next face. If not (face at level i), this face queries the level from the edges composing it. Indeed, if one edge of this face is already at level $i+1$ because the adjacent polyhedron is already subdivided, the algorithm proceeds to the next edge. The edges at level i are subdivided to level $i+1$. The face can then be subdivided at level $i+1$. When this first step is performed, we then use the tools presented in the previous sections to build and link topologically at level $i+1$ the inner faces of future volumes. Once the process is complete we obtain polyhedrons linked with topological relations at level i and level $i+1$. By applying the algorithm in this way, adaptivity in the hierarchy is managed transparently.



(a) Generic algorithm

(b) Algorithm adapted to the degree of the faces

Fig. 18. Subdividing a prism.

6. Results

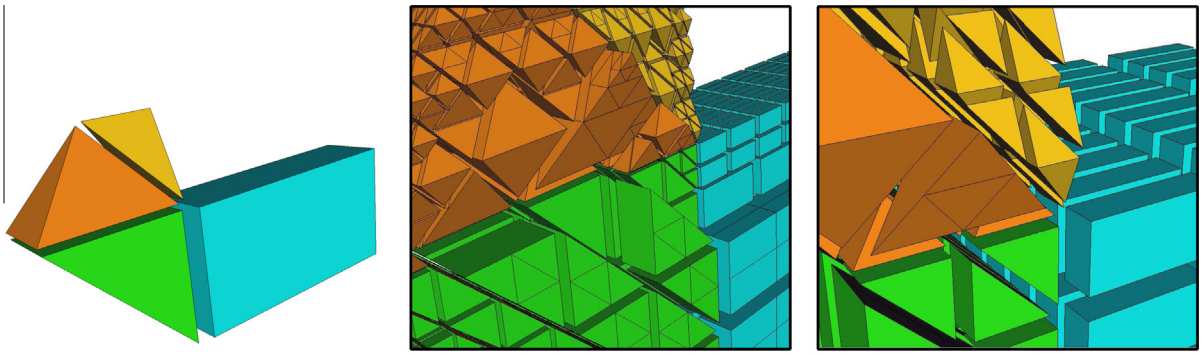
The subdivision algorithm we described is a general way to adaptively subdivide each polyhedron in a volumetric mesh or each hyper-volume in a n-dimensional mesh. We show here examples that demonstrate the functionalities and features of our data structure and adaptive refinements. The geometry of the examples below, except the last figure, is computed using linear interpolation in order to highlight their topology.

The examples in Figs. 19 and 20 we apply our subdivision algorithm over a wide variety of polyhedral cells. The adaptive subdivision is illustrated in Figs. 19 as we choose to subdivide the cells that lie on the boundary of the mesh.

The genericity of our model allows us the mixing of loop subdivision on triangular faces and quadrangular subdivision on the other ones. Thus, a prism is no longer divided into six hexahedrons, but into eight prisms (cf. Fig. 18). This is essential to keep a certain regularity in the mesh elements as often required by finite elements methods. Fig. 20a shows an exploded view of the complete initial example mesh with a genius greater than 0. After four steps of regular subdivision (see Fig. 20b and c), we can see that all cells are subdivided in polyhedral cells without any compliance issues. The geometry follows the subdivision scheme of [4].

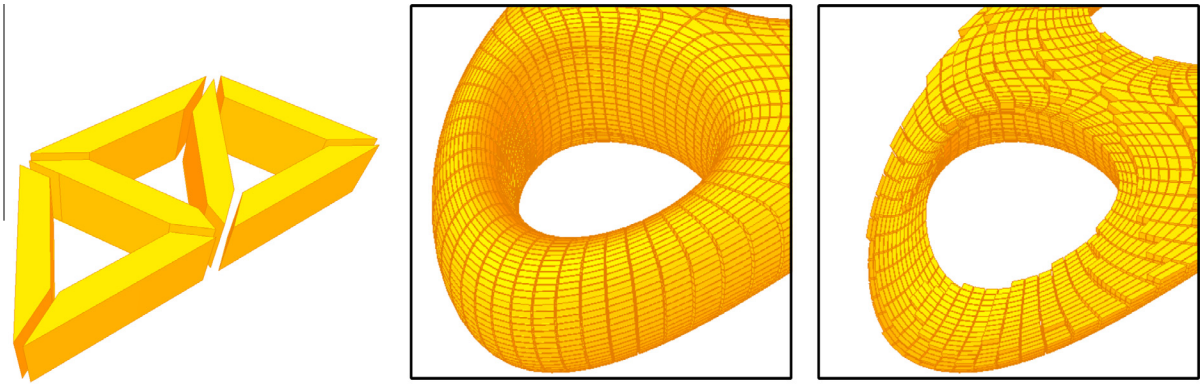
7. Complexity evaluation

In this section, we evaluate the memory cost and complexity of the topological queries of our model. This study is done for 3-dimensional maps. We compare with data structures showing equivalent capabilities: the forest of octrees [31]. First, we compare time complexity between the two structures. Then, finally, we compare space



(a) Exploded view of the initial polyhedral mesh (b) Cutted view of the inside polyhedrons (c) Closer view of the boundary cells

Fig. 19. Visualization of an arbitrary polyhedral mesh with adaptive subdivision.



(a) Exploded view of the intial polyhedral mesh (b) Geometry using [4] after 4 levels of regular subdivision (c) Cutted view after 4 levels of regular subdivision

Fig. 20. Visualization of a polyhedral mesh with regular subdivision.

complexity without hierarchical information and with their multiresolution extension.

7.1. Time complexity

Adjacency queries are the operations that are most performed in multiresolution subdivision volumes.

They are useful when applying a subdivision scheme as well as for the execution of more complex algorithms.

In a forest of octrees, each hexahedron or tetrahedron of the initial mesh is the root of an octal tree. This structure can be seen as an undirected acyclic graph. Neighborhood queries in this kind of structure are solved in constant amortized time. The classical algorithm is to ascend the tree until the parent root and then look for the adjacent cell. Adjacency queries at the root level are solved in $O(log(n))$ where n is the maximum level (i.e. the size of the deepest tree).

For a multiresolution 3-map, each mesh of the hierarchy represents a 3-map directly accessible. Adjacency queries are executed in constant time. The complexity of more advanced algorithms such as browsing adjacent edges is linear in the size of the considered neighborhood. Our data model is optimal for all current operations.

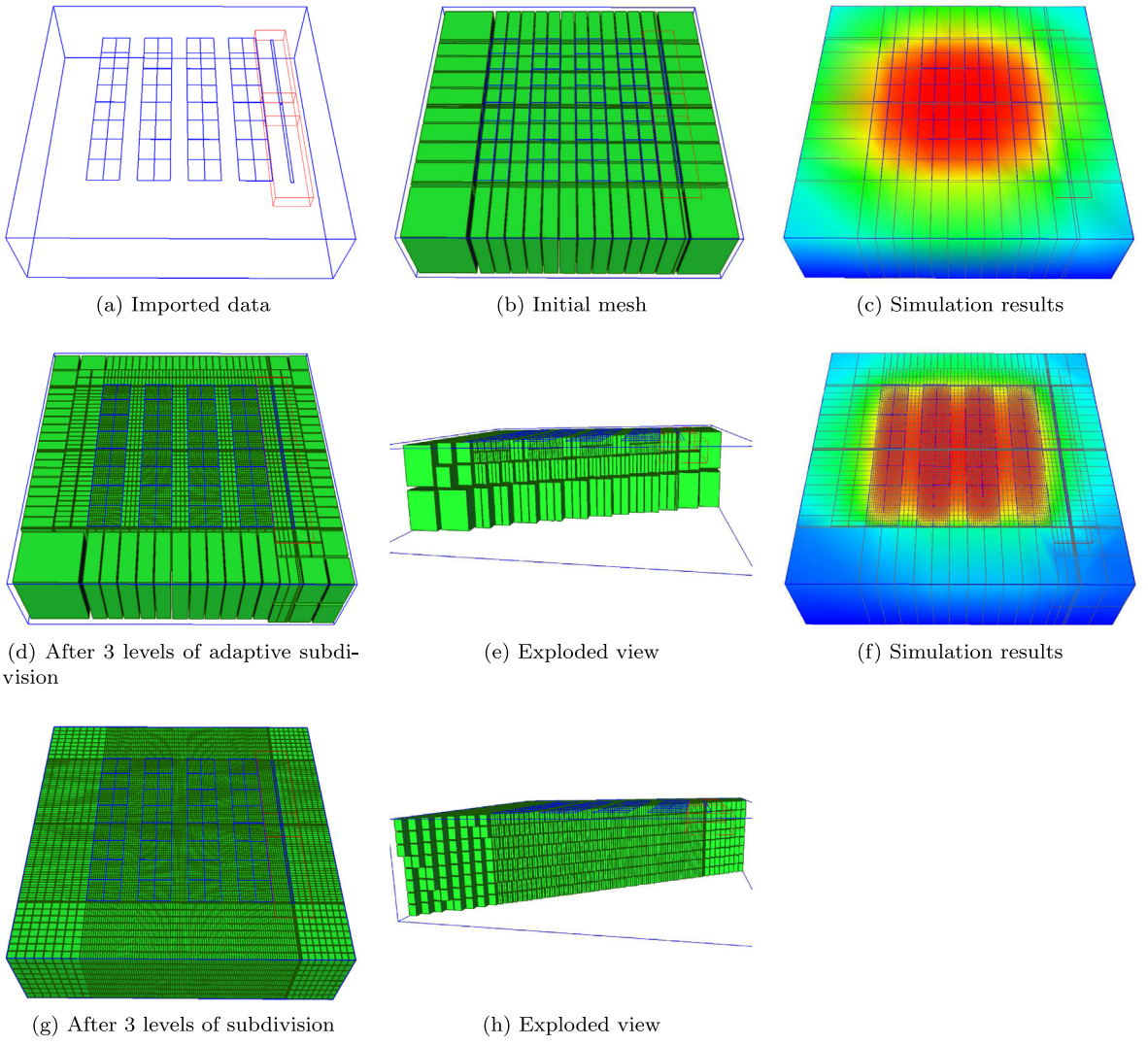
7.2. Space complexity

In this section we compare the estimated cost of a multiresolution 3-map with a hexahedral octal trees forest within a regular subdivision process. This is the worst case in terms of memory space. However, to clarify our purpose, we begin by separating the cost of the structures themselves from their multiresolution extension.

7.2.1. Standard cost

Let n be the number of darts in a 3-map. To count the total cost of the topological information we need to count the number of pointers stored for each dart. The computation for relations α_0, α_1 and α_2 is quite simple, since each dart has a single link. So there are a total of n pointers stored in each of the three relations. We also have to count for each dart a pointer to a structure storing the vertex embeddings. This gives us a cost of $4 \times n$ pointers.

In a forest of octrees, the roots represent the coarsest mesh. Their cost is the number of hexahedrons present in



(a) Imported data (b) Initial mesh (c) Simulation results

(d) After 3 levels of adaptive subdivision (e) Exploded view (f) Simulation results

(g) After 3 levels of subdivision (h) Exploded view

Fig. 21. Visualization of the application pipeline. The first row shows the imported data, the generated mesh and the simulation of this mesh. The second row shows the same information for 3 levels of adaptive subdivision. The third row shows these information for 3 levels of global subdivision.

the mesh. However it should be noted that, unlike the combinatorial maps, this is not a complete topological representation because there is no relation linking all cells of the mesh together. Indeed, the edges are not represented in this kind of structure. Thus, as explained above, topological holes appear at the boundaries between areas of different levels.

In order to complete the topological information available in this kind of structure, we must add minimum information describing the topology of a polyhedron to ensure that the data structure defines a volumetric object. Each hexahedron contains a list of its faces with, for each face, a list of edges (4 pointers per face then 24 pointers for six faces) and finally for each of the 12 edges, there are two pointers to the vertices (24 more pointers). We must add to this count eight pointers to an array of vertices embedding. With h being the number of hexahedrons in the mesh, we obtain a cost of $56 \times h$ pointers.

A hexahedron being composed of 24 darts in a 3-map, we know that $h = \frac{n}{24}$. The ratio between the two data structures at this stage is therefore:

$$\frac{4 \times h}{\frac{56}{24} \times h} \simeq \frac{12}{7} \simeq 1,7 \tag{3}$$

The combinatorial maps structure is at this step 70% more expensive in terms of space. This is natural because, unlike combinatorial maps, the forest of octrees is missing topological information. We have added only the minimum information to describe a volume. Indeed, all neighborhood requests are not available immediately but must be computed. For example, to find all faces around an edge we must first find the adjacent volume of one face of this edge. Then in the list of faces of this volume, we must look at the identifier of the corresponding edge. Finally, we

must start again. In a combinatorial 3-map, the darts of a face are linked together by the volumetric relation α_2 that is sufficient to be able to traverse volumes.

7.2.2. Multiresolution case

Let n be the number of darts in a multiresolution 3-map. Let n_0 be the number of darts in the mesh at level 0, and k the maximum number of levels of the multiresolution structure.

Each hexahedron is cut into eight new hexahedrons; the number of darts is multiplied by a factor 8 at each step of subdivision. We have: $n = n_0 \times 8^k$.

To count the total cost of the topological information we need to count the number of pointers stored for each dart. The computation of relations α_1 and α_2 is quite simple, since each dart has a unique link α_1 or α_2. So there is a total of n (or $n_0 \times 8^k$) pointers stored in each of the two relations.

For the relation α_0, we have to sum the sizes of the pointer arrays contained in each dart. The size of this array is based on the darts insertion level. Note that $\frac{7}{8}$ darts are inserted at maximum resolution and thus have only one link α_0, $\frac{7}{8}$ of the other darts (i.e. $\frac{7}{64}$) have two links ...; for i between 1 and k there is $n \times \frac{7}{8^i}$ darts whose array has i elements. The darts describing the level 0 have $k+1$ elements in their array. The total number of items in the arrays of the relation α_0 for all darts is:

$$n_0 \times (k+1) + n_0 \times 7 \sum_{i=1}^{k} i \times 8^{k-i} \tag{4}$$

Including pointers α_1, α_2 and a pointer to the vertices embeddings, the total number of pointer is:

$$3 \times n_0 \times 8^k + \left(n_0 \times (k+1) + n_0 \times 7 \sum_{i=1}^{k} i \times 8^{k-i} \right) \tag{5}$$

The sum of the previous expression can be identified with the power series: $\sum_{n \geqslant 0} i \times x^i = \frac{x}{(1-x)^2}$, defined for $|x| < 1$. Neglecting the terms of the series such as $i > k$, we obtain:

$$\sum_{i=1}^{k} i \times 8^{k-i} \simeq 8^k \times \frac{\frac{1}{8}}{\left(1-\frac{1}{8}\right)^2} \tag{6}$$

Which can be substituted to:

$$3 \times n_0 \times 8^k + \left(n_0 \times (k+1) + n_0 \times 8^k \times \frac{8}{7} \right) \tag{7}$$

This equation simplifies to: $\frac{29}{7} \times n_0 \times 8^k$

The roots of the forest of octrees store 9 pointers for hierarchical information to the sons. In the forest of octrees, a node requires nine pointers for hierarchical information (i.e. eight for the sons and one for the parent). As we discussed in the previous section, each root and each node of the forest of octrees also stores 56 pointers to topological and geometrical information. To summarize, we obtain 64 pointers for each root and 65 pointers for each node.

Let h_0 be the number of volumes at 0-level (i.e. the root of the tree) and k the maximum resolution level. As the number of volumes is multiplied by a factor 8 at each subdivision level, the total number of stored pointers is:

$$h_0 \times 64 + 65 \times h_0 \times \sum_{l=1}^{k} 8^l \tag{8}$$

The sum can be identified to the power series: $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{i-1}$. We express it as:

$$\sum_{l=1}^{m} 8^l \simeq \frac{8^{m+1}-1}{7} - 1 \tag{9}$$

As we know that $h_0 = \frac{n_0}{24}$, our equation simplifies to: $\frac{65}{21} \times n_0 \times 8^k$.

The ratio between the two data structures decreases to:

$$\frac{\frac{29}{7} \times n_0 \times 8^k}{\frac{65}{21} \times n_0 \times 8^k} \simeq \frac{87}{65} \simeq 1,3 \tag{10}$$

The multiresolution combinatorial maps structure is in the end only 30% more expensive in terms of space than the a forest of octrees in the multiresolution case. This additional cost is largely compensated for by the genericity of representable polyhedrons with a combinatorial map and the completeness of topological information.

8. Application

As alluded to the introduction, multiresolution volumetric meshes can be used in the simulation of physical phenomena. In this manner, we began a collaboration with a research team in physics and microelectronics.

One of their research topics is the development of an electro-thermal simulator integrated in a standard electrical CAD environment. This simulator will be used to perform the electro-thermal analysis to validate the performance of planar or 3D integrated systems in their environment. The thermal behavior is modeled using a finite element approach and the 3D thermal network is build from the information stored in the electrical layout (i.e the chip dimension, components location, etc). To keep the number of meshes to a minimum while keeping the accuracy, a fine mesh has to be set near areas where there is significant change in the temperature profile (i.e. area with high power density) and a coarse mesh is used in areas where the temperature profile is flat or with a small gradient. The areas where a fine mesh is needed are referred as influence area.

We show here the test chip [32] used to validate experimentally the simulator. This chip is made of two resistors placed on each sides and a 4×4 matrix of temperature sensors is placed across the chip. The Fig. 21 illustrate the pipeline used in this application from the imported data (first column) to the visualization of the simulation results (last column). As we can see in the first row, we begin by importing their data which appears as a set of bounding boxes. The components location over the chip is shown in the blue areas on Fig. 21a and the influence areas are shown in red on Fig. 21a. Next, we generate a mesh adapted to the imported data (576 vertices and 255 volumes) and embed the information about the components and influence area location in the mesh (see

Fig. 21b). Finally, the mesh is exported and is used to run the simulation with their tool. For visualization purpose, we just import simulation results on the vertices (see Fig. 21c).

The second row shows the result after 3 levels of adaptive subdivision of the polyhedrons included in the influent areas (see Fig. 21d and e for an exploded and clipped view). This mesh with 29,510 vertices and 19,218 volumes can easily be simulated in the CAD environment. Besides, comparing to the initial mesh, we can see that the adaptive subdivision enhance the accuracy of the simulation results.

The third row shows the result after three levels of a regular subdivision of the whole mesh (see Fig. 21g and h for an exploded and clipped view). In our context, this kind of mesh with 149,193 vertices and 130,560 volumes is to large to be used to run a simulation.

To conclude, thanks to our adaptive subdivision the number of vertices decreases by 80%, the number of volumes decreases by 85%. With our method we allow the simulation of this kind of chips in a standard electrical CAD environment.

In the future, we will be looking to create meshes for 3D integrated circuits by stacking multiple tier of planar circuits in one topologically consistent mesh and subdivide each of them according to each heat area. This will make possible the electro-thermal analysis of complex 3D integrated circuits. To reduce even further the thermal network and still keeping the accuracy, an equivalent network can be build by selecting only the thermal critical paths between each component. Our topological structure can also be used for efficient path traversals through the levels of resolution as seen in the previous sections.

9. Conclusion and future work

The multiresolution combinatorial n-maps defined in this paper provides a powerful framework for the representation of arbitrary multiresolution meshes and specifically of subdivision volumes. They overtake classical data structures thanks to their genericity and their ability to support arbitrary topological subdivisions that are the keystone to robustly and soundly define mesh hierarchies.

We demonstrate that this multiresolution model benefits from controlled memory costs. All low-level topological queries perform in constant time and thus all neighborhood queries run in linear time according to the number of cells in those neighborhood. This allows very efficient implementations of this model, as competitive as other less general data structures.

A longer term perspective is to extend simplification and merging operators – like edge collapses or more generally cell collapses – to define progressive meshes in arbitrary dimension. This would enable to build mesh hierarchies from iterative simplification processes and provide filtering and signal analysis tools to n-dimensional meshes.

Acknowledgements

We thank Sylvain Thery for helping us with the images. We also thank reviewers for helping improving this paper.

References

[1] G. Allaire, Numerical Analysis and Optimization: An Introduction to Mathematical Modelling and Numerical Simulation, Numerical Mathematics and Scientific Computation, Oxford University Press, 2007.
[2] M. Wicke, D. Ritchie, B.M. Klingner, S. Burke, J.R. Shewchuk, J.F. O'Brien, Dynamic local remeshing for elastoplastic simulation, ACM Trans. Graph. 29 (2010) 49:1–11.
[3] D. Burkhart, B. Hamann, G. Umlauf, Iso-geometric finite element analysis based on Catmull–Clark subdivision solids, Comput. Graph. Forum 29 (5) (2010) 1575–1584.
[4] R. MacCracken, K.I. Joy, Free-form deformations with lattices of arbitrary topology, in: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques – SIGGRAPH '96, New Orleans, 1996, pp. 181–188.
[5] S. Capell, S. Green, B. Curless, T. Duchamp, Z. Popović, A multiresolution framework for dynamic deformations, in: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '02, ACM, New York, NY, USA, 2002, pp. 41–47.
[6] S. Schaefer, J.P. Hakenberg, J. Warren, Smooth subdivision of tetrahedral meshes, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing – SGP '04, Nice, France, 2004, pp. 147–154.
[7] V. Pascucci, Slow growing subdivisions (sgs) in any dimension: towards removing the curse of dimensionality, Comput. Graph. Forum 21 (3) (2002) 451–460.
[8] L. Velho, D. Zorin, 4–8 Subdivision, Comput. Aided Geom. Des. 18 (5) (2001) 397–427.
[9] L. De Floriani, L. Kobbelt, E. Puppo, A survey on data structures for level-of-detail models, in: N.A. Dodgson, M.S. Floater, M.A. Sabin (Eds.), Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization, Springer, Berlin Heidelberg, 2005, pp. 49–74.
[10] J. Tournois, C. Wormser, P. Alliez, M. Desbrun, Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation, ACM Trans. Graph. 28 (3) (2009).
[11] B. Lévy, Y. Liu, Lp centroidal voronoi tesselation and its applications, in: ACM Transactions on Graphics (SIGGRAPH Conference Proceedings) Patent Pending – FR 10/02920 (filed 07/09/10).
[12] P. Frey, P.-L. George, Mesh Generation, John Wiley & Sons, 2008.
[13] L. DeFloriani, L. Kobbelt, E. Puppo, A survey on data structures for level-of-detail models, Advances in Multiresolution for Geometric Modelling, Series in Mathematics and Visualization (2004) 49–74.
[14] P. Lienhardt, Topological models for boundary representation: a comparison with n-dimensional generalized maps, Comput. Aided Des. 23 (1) (1991) 59–82.
[15] J.-F. Dufourd, Algebraic map-based topological kernel for polyhedron modellers: algebraic specification and logic prototyping, in: Int. Conf. Eurographics'89, North-Holland, 1989, pp. 649–662.
[16] F.H. Croom, Basic Concepts of Algebraic Topology, Undergraduate Texts in Mathematics, Springer, 1978.
[17] P. Kraemer, D. Cazier, D. Bechmann, Extension of half-edges for the representation of multiresolution subdivision surfaces, Visual Comput. 25 (2) (2009).
[18] C. Bajaj, S. Schaefer, J. Warren, G. Xu, A subdivision scheme for hexahedral meshes, Visual Comput. 18 (5-6) (2002) 343–356.
[19] K.T. McDonnell, Y. Chang, H. Qin, Interpolatory, solid subdivision of unstructured hexahedral meshes, Visual Comput. 20 (6) (2004).
[20] C. Loop, Smooth subdivision surfaces based on triangles, Department of Mathematics, University of Utah, Utah, USA, 1987.
[21] N. Dyn, D. Levine, J.A. Gregory, A butterfly subdivision scheme for surface interpolation with tension control, ACM Trans. Graph. 9 (1990) 160–169.
[22] D. Zorin, P. Schröder, W. Sweldens, Interpolating subdivision for meshes with arbitrary topology, in: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, ACM, New York, NY, USA, 1996, pp. 189–192.
[23] E. Catmull, J. Clark, Recursively generated b-spline surfaces on arbitrary topological meshes, Comput.-Aided Des. 10 (6) (1978) 350–355.
[24] Y.-S. Chang, K.T. McDonnell, H. Qin, A new solid subdivision scheme based on box splines, in: Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications, SMA '02, ACM, New York, NY, USA, 2002, pp. 226–233.
[25] D. Burkhart, B. Hamann, G. Umlauf, Adaptive and feature-preserving subdivision for high-quality tetrahedral meshes, Comput. Graph. Forum 29 (1) (2010) 117–127.

[26] E. Danovaro, L. de Floriani, P. Magillo, E. Puppo, D. Sobrero, N. Sokolovsky, The half-edge tree: a compact data structure for level-of-detail tetrahedral meshes, in: SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005, IEEE Computer Society, Washington, DC, USA, 2005, pp. 334–339.

[27] D. Luebke, B. Watson, J.D. Cohen, M. Reddy, A. Varshney, Level of Detail for 3D Graphics, Elsevier Science Inc., New York, NY, USA, 2002.

[28] K. Weiss, L. De Floriani, Simplex and diamond hierarchies: models and applications, Comput. Graph. Forum 30 (8) (2011) 2127–2155.

[29] D.P. Dobkin, M.J. Laszlo, Primitives for the manipulation of three-dimensional subdivisions, in: Proceedings of the Third Annual Symposium on Computational Geometry, SCG '87, ACM, New York, NY, USA, 1987, pp. 86–99.

[30] CGoGN, Combinatorial and Geometric Modeling with Generic *n*-Dimensional Maps. <http://cgogn.unistra.fr>.

[31] C. Burstedde, L.C. Wilcox, O. Ghattas, `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM J. Sci. Comput. 33 (3) (2011) 1103–1133, http://dx.doi.org/10.1137/100791634.

[32] J.-C. Krencker, J.-B. Kammerer, Y. Herve, L. Hebrard, 3d Electro-thermal simulations of analog ics carried out with standard cad tools and verilog-a, in: 2011 17th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC), 2011, pp. 1 –4.