Simplicial-based multiresolution volume datasets management: an overview

Rita Borgo, Paolo Cignoni, and Roberto Scopigno

Istituto di Scienza e Tecnologia dell'Informazione (ISTI), Consiglio Nazionale delle Ricerche, Pisa, Italy - {*borgo, cignoni, scopigno*}*@isti.cnr.it*

Summary. The paper synthetically presents the methodologies proposed for the efficient multiresolution management of large volume datasets. In particular, we review those multiresolution approaches based on simplicial meshes. The methodologies available are classified in two main streams, according to the regular or irregular refinement kernel adopted to build the multiresolution representation. A comparison, highlighting respective strength and weakness of the two classes of methods, is proposed and discussed.

1 Introduction

Many approaches have been proposed to support multiresolution management of volume datasets: naïve sub-sampling, wavelet techniques, hierarchical space subdivisions (e.g. octrees), and simplicial decompositions. The term multiresolution is often used to indicate either discrete or continuous level of detail (LOD) representations. We will cover mostly the second aspect, and therefore we point our attention to those methods that allow to manage selective refinements (or the inverse operation, i.e. selective coarsening) in a dynamic manner, according to run time requirements. Simplicial meshes have been often used in the visualization of volume datasets. The simplicity of the basic cell allows to easily manage isosurface extraction (field is linearly interpolated, no ambiguity) and to implement in an efficient manner direct volume rendering (DVR) solutions [6]. Moreover, tetrahedral-based DVR solution can now be implemented using off-the-shelf graphics hardware, gaining impressive speed-ups with respect to software solutions. Therefore, simplicial decompositions have been often considered in the design of multiresolution methods, not only because they are easy to render but also because they easily adapt to different shapes or to the data field structure/topology. This paper presents an overview and a comparison of the different approaches for simplicial-based multiresolution proposed in the context of volume visualization. We subdivide the existing methods in two main classes, which depend on the refinement kernel used to manage the selective refinement/coarsening: *regular* or *irregular*. Regular techniques starts from a coarse regular base domain and apply recursive regular refinement, resulting in large meshes organized as uniform grid patches. On the other hand, irregular techniques are

independent from the topology of the underlying mesh; not being forced to follow a regular subdivision scheme (vertices can be added in any order) irregular techniques result to be more flexible and suitable to resolve complex geometric features and geometry changes.
The paper is organized as follows: Section 2 introduces some basic concepts in multiresolution data management and presents a general representation framework. Then, some representative irregular and regular approaches are presented in a synthetic manner in Sections 3 and 4. A comparative evaluation of the approaches presented is given in Section 5 and concluding remarks in Section 6.

2 Multiresolution representations for simplicial meshes

Current multiresolution models for generic tetrahedral meshes can manage a large set of level of details in a flexible, efficient and compact way. The main idea behind these methods is to exploit, in some way, the information that can be collected during the simplification or the refinement of a volume dataset. The assumption is that we use an iterative simplification or refinement algorithm that progressively reduces/refines the dataset by means of *small* and *local* modifications. This sequence of updates can be organized in structures which encode (either explicitly or implicitly) all the reciprocal temporal dependencies and makes it possible to apply those updates in many different orders. The main goal is to support very fast selective refinement/simplificaction, according to the user dynamic requests on selected portions of the domain.
These data representation approaches are strongly related with the multiresolution and simplification techniques for three-dimensional *surface* meshes [14], that is the domain where those ideas were firstly developed. Many different approaches have been presented to extend the multiresolution data management approach to volume datasets. A general framework which can be used to encompass all of them is the *Multiresolution Simplicial Model* (MSM), introduced by De Floriani, Puppo and Magillo [10] as a multidimensional extension of the 2D 1/2 structure described by Puppo in [24]. We give a brief intuitive description of the MSM scheme in the following subsection and we send back to [10] for a more formal description.

The Multiresolution Simplicial Model (MSM) The main idea of the MSM is that we can infer a multiresolution model by storing as small subsets of tetrahedra Σ_i called *fragments* the sequence of local modifications performed by an iterative simplification algorithm. Obviously these modifications are not independent and the relative dependencies among these fragments can be coded in a partial ordering $\prec$. In practice this partial ordering represents the fact that a given modification of the mesh cannot be done before another one, usually because they modify a common portion of the

mesh and have to be executed in a given order. This partial ordering can be graphically described as a *direct acyclic graph* (DAG), where nodes represent fragments and arcs encode the $\prec$ relation. In Fig. 1 we show an example of a simple two-dimensional MSM encoded as a DAG. Intuitively speaking, the

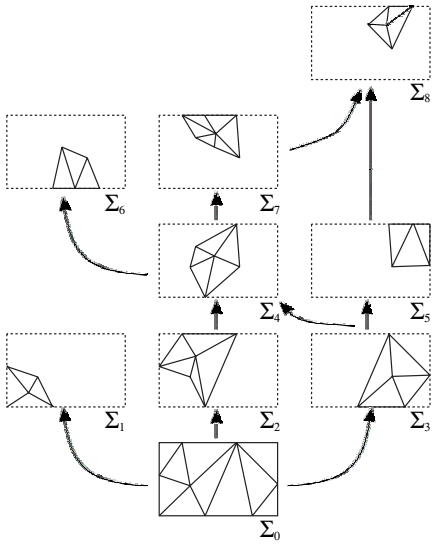


Fig. 1. The DAG describing a simple two-dimensional MSM.

fragments describe a portion of the dataset domain at a certain resolution and, by combining them carefully, we can reconstruct many different meshes. For example, let us examine an iterative refinement procedure on a simplicial mesh; the set of simplexes, derived from the substitution of a complex with a more refined one, can be considered as a fragment combined over the existing complex. In this approach the lowest element of the DAG is the coarsest representation of the dataset, and the elements above represent the refinements done onto the mesh (Fig. 1). The combination of fragments can be done through an operator $\oplus$ that replaces the simplexes in the lower fragments with the one in the subsequent fragment, holding the hypothesis that they have some common intersection; the order in which these fragments must be combined is the one induced by the partial ordering. The triangulation resulting from the combination of fragments $\Sigma_0, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_7$, satisfying a consistent order, is shown in Fig. 4. Note that any other consistent order of combination (e.g. $\Sigma_0, \Sigma_3, \Sigma_2, \Sigma_4, \Sigma_7$) builds the same triangulation.
The MSM model is an abstract way of conceiving a generic multiresolution model. In practice, the naive approach to directly map the MSM model into a data structure, representing explicitly the DAG of the set of fragments,

does not allow to attain the utmost space and computational efficiency. The importance of the MSM is that it allows to explain and compare all the current *real* multiresolution data structures under a common framework: they can be seen as smart and compact way of coding the elements of a MSM (the DAG and the fragments). For example, multiresolution models based on edge collapses usually represent the MSM fragments in a completely implicit way by storing just the collapse action instead of the explicit list of simplexes modified by the collapse.
In the following two sections we describe the two main classes of approaches: the methods based on *irregular* or *regular* refinement kernels.

3 Irregular Refinement Techniques

Since these multiresolution methods are based on an *irregular* simplification or refinement kernel, we shortly review in the next section the techniques for the simplification or refinement of a tetrahedral complex.

3.1 Simplification of a simplicial complex

One of the first approaches for the simplification of tetrahedral datasets was proposed in [4]. It exploits a basic coarse-to-fine refinement strategy, an early technique widely used for approximating natural terrains [13, 9]. An on-line algorithm for Delaunay tetrahedralization is used together with a selection criterion to refine an existing Delaunay mesh by inserting one new vertex at a time. The selection strategy at each iteration is aimed to refine the tetrahedron that causes the maximum error in the current approximation (considering both field value interpolation and mesh warping): the point v_{max} in the initial dataset V holding the maximum error becomes a new vertex of the current simplicial decomposition Σ_i. Adding a point implies to search for the tetrahedron in Σ_i that contains v_{max}, to split it on the new vertex v_{max} (Fig. 2) and to apply a sequence of flipping actions until the resulting mesh satisfies again the Delaunay criterion. This refinement procedure always converges, since the number of points in V is finite. Unfortunately, this approach is limited to datasets whose domain is convex, because the result of a Delaunay tetrahedralization is always convex.
An extension of this approach has been proposed in [6] to deal also with non-convex *curvilinear* datasets: the Delaunay tetrahedralization is computed in the computational domain (i.e. the underlying convex grid), while its image through lifting gives the corresponding mesh in the physical domain. The refinement method described above is difficult to adapt to the case of generic *non-convex irregular datasets*. Major difficulties arise in finding an initial coarse mesh which approximates the original domain Ω of the dataset (Delaunay triangulation is not applicable to non-convex polyhedra) and in the

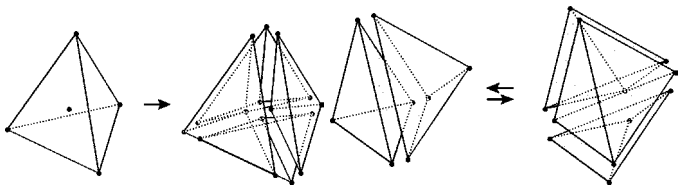


Fig. 2. A tetrahedra *split*, due to the insertion of a new vertex in the mesh, is shown on the left. On the right we show a *flip* action that substitutes two adjacent tetrahedra with three tetrahedra and viceversa.

estimation of the warping of the mesh boundary. Experience in the approximation of non-convex surfaces through 2D triangular meshes suggests that a *simplification* technique might be more appropriate to the case of non-convex irregular volume datasets (see, for example, [19, 15]). In the simplification approach we start with an input simplicial mesh decomposition of the dataset V; then, vertices are iteratively discarded as long as the error introduced by removing them does not exceed a given accuracy threshold.

Gross and Staadt [27] present a simplification technique based on collapsing an edge to an arbitrary interior point, and propose various cost functions to drive the collapsing process. Cignoni et al. [6] propose an algorithm based on collapsing an edge to one of its extreme vertices (called half-edge collapse, see Fig. 3), in which the simplification process is driven by a combination of the geometric error introduced in simplifying the shape of the domain and of the error introduced in approximating the scalar field with fewer points. This approach has been extended in [3] by defining a framework for the unified management of the two errors (related to the geometric domain warping and the scalar field interpolation) and by proposing some techniques to forecast or to evaluate efficiently such errors. In fact, selecting a vertex to be removed involves an estimation of how much the error will be increased; the vertex causing the smallest increase is usually selected at each iteration. An exact estimation of error variation can be obtained by simulating the deletion of all vertices in the current mesh. This would be computationally expensive, since, assuming that we are working with a small Euler characteristic and border as in [18] we can state that each vertex has approximately $20 \sim 24$ incident tetrahedra on average. This may involve relocating many points lying inside such tetrahedra. The use of heuristics to estimate apriori how a vertex removal affects error and warping is therefore a wiser choice [3]. Such an estimation is computed for all vertices before decimation starts, and it has to be updated during simplification for all those vertices affected by a change (i.e. when one or more of its incident tetrahedra change).

Trotts et al. [28] perform half-edge collapse as well. They control the quality of the simplified mesh by estimating the deviation of the simplified scalar field from the original one, and by predicting the increase in deviation caused by a

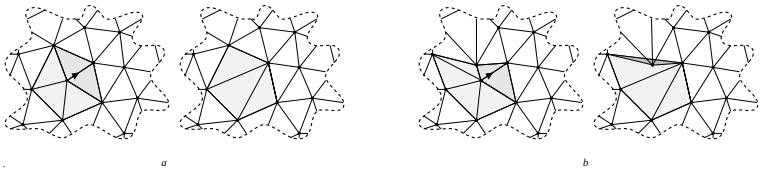


Fig. 3. Half-Edge collapse in 2D: (a) a valid collapse; (b) an inconsistent collapse.

collapse. They also provide a mechanism to bound the deformation of the domain boundary. Another approach for the simplification of generic simplicial complexes, called Progressive Simplicial Complex (PSC), has been proposed by Popovic and Hoppe [23], as an extension of the Progressive Meshes (PM) approach [19]. The PM structure basically encodes the sequence of edge-collapse transformations; the PSC codifies in a similar manner a sequence of more general edge-collapse transformations. It should be noted that while the PSC are quite general, they have been conceived for the management of possibly degenerated 2D surfaces rather than simplicial complexes representing a volume dataset, so the conditions of legality of a sequence of vertex-split operation of a generic complex are not specified, and the problem of evaluating the approximation error introduced in the volume field representation is not considered.

3.2 From simplification to Multiresolution Models

Each simplification algorithm described in Section 3.1 can be used to build a multiresolution representation. Either a decimation or a refinement algorithm working on a tetrahedral complex produce an "historical" sequence of tetrahedra, namely all tetrahedra that appear in the progressively simplified/refined mesh Σ during its construction. An historical sequence can be also viewed as the sequence of all subdivisions of the whole domain that are obtained through changes, or as an initial subdivision plus a sequence of fragments reflecting the local changes iteratively done to the mesh, which can be partially overlapping and are pasted one above the other to update the existing structure. For example, if we follow the *refinement* heuristic, the initial coarse triangulation is the starting mesh. When we insert a new point v_i in the complex (Fig.2), the new tetrahedra that are built form a new fragment Σ_i; the corresponding fragment replaced by Σ_i is constituted by the tetrahedra that were destroyed by the insertion of v_i. Following the MSM framework, all these fragments (represented by a tetrahedral complex covering a small part of the whole domain Ω) are arranged in a DAG where the order relation between fragments is dependent on their interferences in 3D space. The minimum fragment Σ_0, the coarsest representation of our mesh, has no incoming arcs. Similarly all the triangles on the top of the DAG $\mathcal{S}$, rep-

resenting the dataset at its full resolution, have no outcoming arcs pointing to new fragments.
A simple data structure to encode a generic MSM was presented in [8]. A much more compact data structure has been proposed for three-dimensional tetrahedral MSM built by a sequence of general edge collapses [5]. This latter structure, customized to the needs of volume visualization, requires three times less storage space with respect to a simple indexed data structure encoding the original mesh at full resolution, and 5.5 times less space than a data structure for the original mesh encoding both connectivity and adjacency information (as required, e.g., by direct volume rendering algorithms based on cell projection [30]). This kind of structure have to maintain, more or less explicitly, the set of all the vertices of $\mathcal{S}$, the set of all tetrahedra $\Sigma_{\mathcal{S}}$, the set of all fragments in $\mathcal{S}$, the $\prec$ relation among them and for each tetrahedron, the reference to the vertices forming it and the accuracy provided by this cell when we interpolate any point in its interior from the field values hold by the cell vertices.

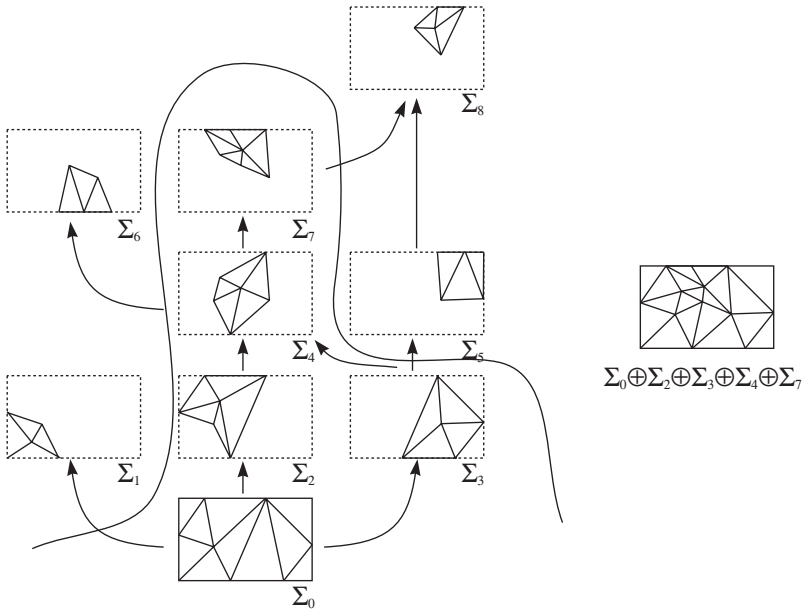


Fig. 4. Extracting a mesh at constant/variable resolution means detecting a proper subset of the fragments, $\mathcal{S}' \subset \mathcal{S}$, corresponding to a cut in the DAG.

3.3 Extracting a variable resolution model

Algorithms for the extraction of a variable resolution model from an MSM have been presented in [24, 5]. If we assume that our MSM is by construction

monotone (i.e. every refinement action improves the accuracy of the intermediate mesh), the extraction of a *variable resolution model* can be simply performed by defining a boolean acceptance function $c(\sigma)$ which evaluates the accuracy of each tetrahedron σ (or, in reality, of each fragment). The algorithm for the extraction of a variable resolution model builds incrementally the desired solution by adding new fragments to a current solution, performing a breadth-first traversal of the DAG representing the MSM. The traversal starts from the coarsest fragment Σ_0, root of the DAG, and fragments above the current solution are progressively traversed and marked (see Fig. 4). The current solution is maintained as a list of tetrahedra Σ_{Out}. For each fragment Σ that we encounter in the traversal of the DAG, the following two loops are executed:

- we search for fragments before Σ, still not visited and, if found, they are added to the traversal queue Q. All the fragments before Σ can be found by checking, for each tetrahedron $\sigma \in \Sigma$, if the corresponding lower fragment Lower(σ), has been marked.
- for each tetrahedron $\sigma \in \Sigma$, if it satisfies the acceptance function $c(\sigma)$ then σ is added to the current solution, else we add the upper fragment of σ to the traversal queue Q and mark it to be removed from the solution.

The correctness of this algorithm has been proved in [8].

4 Regular Refinement Techniques

A multiresolution data representation can be built by starting from some sort of regularly shaped entities and using a regular subdivision pattern. Simple mathematical rules withhold the basis of a regular subdivision scheme that is usually applied in a recursive manner. The adoption of a predefined rule to perform the subdivision introduces some constrains on the topology of the local region to be refined/simplified, and adapts well mainly on regular dataset.
The simplest approach to perform a regular subdivision on regular voxel grids is the *octree* scheme, but the disadvantages of the octree-based subdivision are also well known (mainly, the discontinuities introduced in the common frontier of cells of different resolution). A common approach to ensure continuity is to adopt methods based on the use of simplicial decompositions, as we have seen in the previous section. Simplicial decompositions can also be used to represent regular datasets and to implement regular subdivision kernels on those datasets. Regular refinement approaches based on the simplicial decomposition have been proposed to manage terrain data represented by regular grids (e.g. [11]), and are nowadays the most common solutions for highly efficient multiresolution terrain rendering. Conversely, just a few approaches have been proposed to manage volume datasets [22, 31, 17, 21, 20]

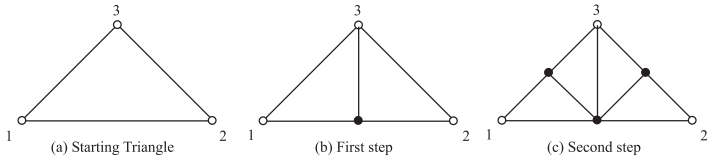


Fig. 5. A 2D example of the Rivara's 4T subdivision approach.

and to accomplish a good balance between efficiency of the subdivision technique and easiness of implementation.
Essentially, the refinement heuristic consists of a recursive subdivision of the volume data, driven by a set of mathematical rules which guarantee a progressive improvement of the accuracy of the intermediate representations in an error-controlled manner. The subdivision process in general starts by subdividing the bounding box of the volume (i.e. a single hexahedral cell) in simplicial cells. The subdivision proceeds recursively and can be described either as a per vertex-adding process or, analogously, as a cell subdivision process. Each step picks up a vertex from the original dataset and divides the cell containing it (or a group of adjacent cells) in two (or more) simplicial cells. Because of the regular and hence predictable parametric structure of the refinement process, these techniques always generate meshes with bijective mapping between the coarsest levels and the finest levels. Therefore, going from a coarse to finer level corresponds in executing a regular subdivision of simplices allowing for smooth and continuous changes between different levels of details. The positive advantages of regular refinement techniques are basically their elegant mathematical formulation and the simplicity of the rules for generating different representations (which often allows to avoid explicit representation of the mesh topology). Recursive subdivision schemes usually produce hierarchical multiresolution representations, very efficient to process and store; moreover, regular refinement techniques often guarantee an almost everywhere regular structure of the variable resolution meshes extracted. In the next paragraphs we introduce in detail some of the main contributions in the field of regular refinement techniques.

4.1 4T Algorithm

The 4T algorithm has been introduced first by Rivara [26] and successively extended to n-dimension by Plaza and Carey. This technique can be conceived, in the 2D case, as a triangle refinement based on the longest edge bisection criteria, applied in two steps (see Fig. 5). The refinement process can be applied selectively, but it is important to maintain the refined mesh *conforming*, i.e. the subdivision of a cell on edge e will force the subdivision of all the ones sharing the edge e.
Plaza and Carey extended the 4T algorithm to 3D meshes by working on the *"skeleton"* of the mesh to be refined, i.e. they introduce bisection points on

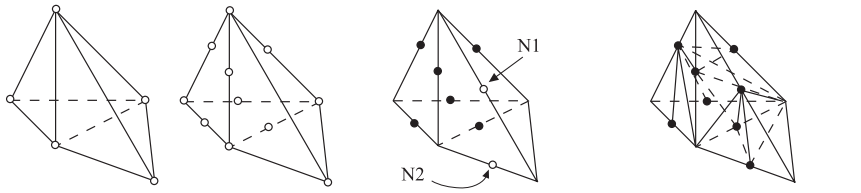


Fig. 6. Application of the 3D subdivision algorithm of Plaza and Carey.

all edges of a cell to be refined. To guarantee the conformity of the mesh, they induce the subdivision of all the simplices incident on a bisected edge. A mesh is considered *conforming* if the intersection of two elements consists of a common face or a common edge or a common vertex or it is empty. The algorithm applies the following actions until no more cells are selected for subdivision:

- Step 1: select the subset of tetrahedra S' which need to be refined, and add a node at the midpoint of each edge of each selected $t \in S'$;
- Step 2: the conformity of each $t \in S - S'$ is checked; for each non-conforming t, a bisection node is added at the midpoint of each non-conforming edge of t and also at the midpoint of the longest edge of t (Fig. 6.c);
- Step 3: each t containing bisection nodes is properly subdivided (Fig. 6.d).

Tetrahedra are selected for refinement following an error indicator (only measures of the shape quality of the mesh are considered in the paper) or to guarantee conformity. The introduction of other bisection vertices in the cell split to guarantee conformity (see nodes *N1* and *N2* in Fig. 6.c) is an empirical solution to prevent bad-shaped tetrahedra. Unfortunately, the conformity is obtained at expenses of a significant propagation of each error-driven split; a refinement cannot be very local and tight under the 4T refinement rule, but extends to a much larger area.

4.2 Multiresolution Tetrahedral Framework (MTF)

The Multiresolution Tetrahedral Framework (MTF) [31] also adopts the longest edge bisection criteria in the development of a framework for tetrahedral meshes refinement. The MTF approach starts from the bounding box of the regular volume dataset (seen as a hexahedral cubic cell), which is subdivided into 12 tetrahedra as follows: the center point of the cell is connected to all the 8 vertices forming 6 pyramids, each pyramid is then divided into two tetrahedra by splitting the base along the diagonal. After this initial step the algorithm proceeds as a regular tetrahedra subdivision scheme. Each tetrahedra is recursively subdivided on the midpoint of its longest edge.
The tetrahedra generated by the subdivision can be grouped in three main classes (see Fig. 7.a-b):

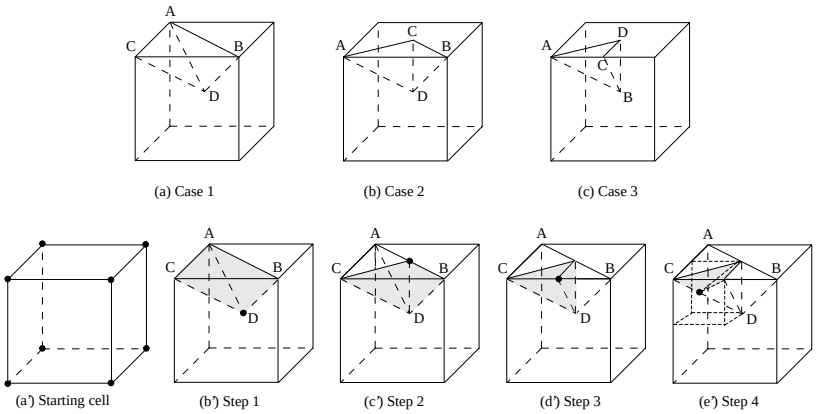


Fig. 7. Zhou et al. approach (MTF): a-b)the three types of tetrahedra generated by the subdivision rule; a'-b')The cyclic cubic cell subdivision.

- *Class 1*: there is only one face parallel to a coordinate plane and there exists only one edge l of that face not parallel to any coordinate axis;
- *Class 2*: there is only one face parallel to a coordinate plane and there exists only one edge l of that face parallel to a coordinate axis;
- *Class 3*: there are two faces parallel to a coordinate plane (the edge that does not belong to any of these two faces is denoted by l).

In all three cases, the edge AB is the edge previously called l, that is the one split along the midpoint in the subdivision process. We show the entire subdivision process in Fig. 7.a'-e'. We start with the initial subdivision of the cubic cell into 12 tetrahedra all belonging to Class 1 (Fig. 7.b'). Those tetrahedra are then subdivided producing tetrahedra belonging to Class 2 (Fig. 7.c'), which are again subdivided producing tetrahedra belonging to Class 3 (Fig. 7.d'). With a further subdivision of a Class 3 tetrahedra we obtain cells of Class 1, and therefore the configuration recursively returns to the initial subdivision step. It is worth noting that each type of tetrahedra belongs to a different step of the subdivision process (modulo 3) and that the overall process has a cyclic behavior.
The MTF approach preserves explicitly the topology of the finest level mesh and this constitutes a problem (excessive storage cost) in the representation of very large datasets .

4.3 Slow Growing Subdivision (SGS)

The Slow Growing Subdivision (SGS), introduced by Pascucci [20], extends the subdivision criteria at the base of the MTF subdivision to generical

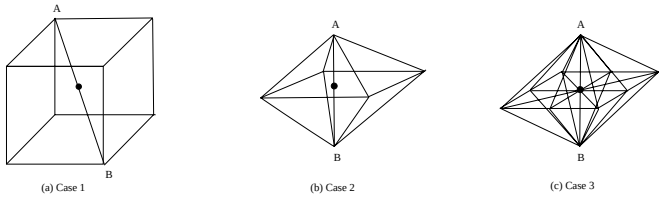


Fig. 8. The three cell types of the Slow Growing Subdivision (SGS) approach.

n-dimensions and, moreover, it is designed to manage huge datasets and presents peculiar features. Analogously to the other approaches, the SGS framework requires a first phase where the decomposition is performed from the coarser cell to the finest ones, with the objective of estimating the degree of accuracy given by any intermediate decomposition, and a run-time phase, where the decomposition is performed according to some application-specific constraints (view-dependent, adaptiveness, error-based criteria).
The starting point of the SGS subdivision is again the volume bounding box, a cubic cell which is divided into 6 pyramids obtained by connecting the boundary faces with the cell center. The SGS approach does not simply divide the cube into tetrahedra, as it happens in MTF. The original idea introduced by the SGS is to setup a cyclic set of *join&split* actions: each of these actions consists in merging some cells produced in the previous subdivision step, forming a new entity; and then splitting this new entity by inserting a new dataset vertex (the vertex introduced is always the center point of the split entity). The SGS method defines three different type of entities (here called *diamonds*, to differentiate them from the tetrahedral or pyramidal cells produced by the split of those entities). Therefore, the second step in the SGS cyclic subdivision process is to build new *diamonds* by joining each of the six pyramids produced in step 1 with the corresponding pyramid generated in an adjacent cube (see Fig. 9.a). This new entity corresponds to an *octahedral-shaped diamond*, whose center corresponds to the center of one of the faces of the cube. This entity is split by inserting the center point and by connecting all its 6 vertices with the center point, thus dividing the octahedral-shaped diamond into eight tetrahedra. This eight tetrahedra correspond each to the eighth part of different *hexadecahedral-shaped diamonds* whose centers correspond to the midpoints of the initial cubes edges (Fig. 9.b). Finally, the sub-cells produced by the subdivision of the hexadecahedral-shaped diamonds will recombine with adjacent ones to form again *cubic-shaped diamonds*, having edge length equal to 1/2 of the initial cubic-shaped cell (Fig. 9.c). This concludes the cycle, which can be repeated until we reach the maximal resolution of the input dataset.
The cells cyclically generated by the subdivision can be grouped in three main classes (Figure 8):

- Class 1 diamond: the cell is *cube-shaped*;

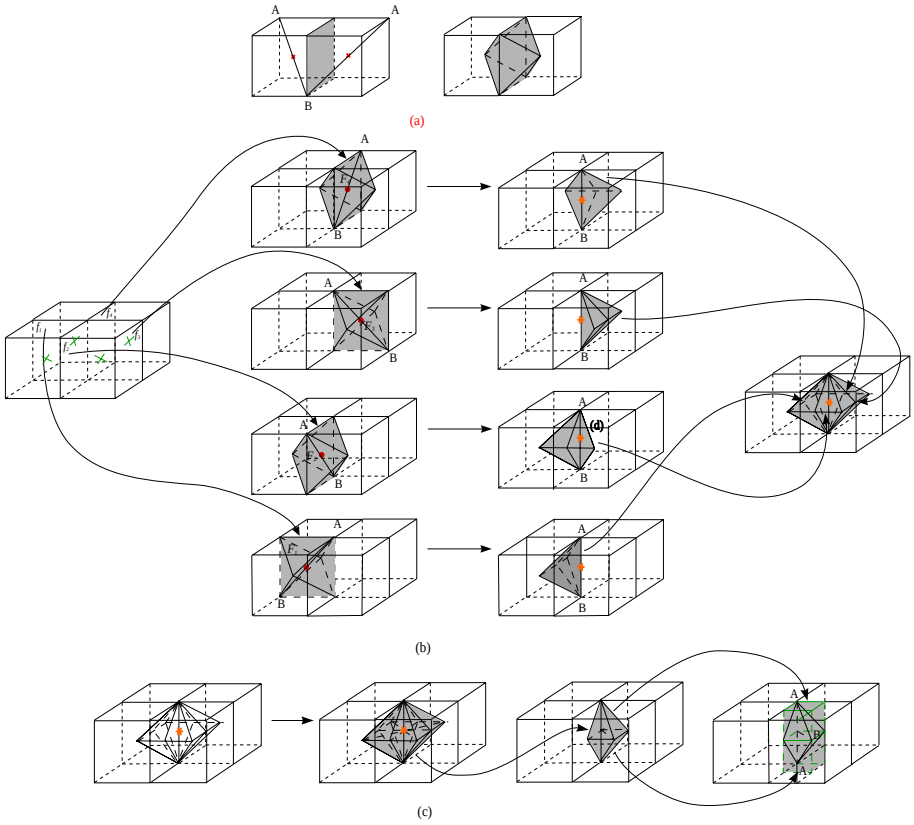


Fig. 9. SGS refinement: a) from the initial cubic cell to a class 2 diamond; b) a class 3 diamond generated for the join of portions of four class 2 diamonds; c) class 1 cells are produced by the subdivision of class 3 diamonds.

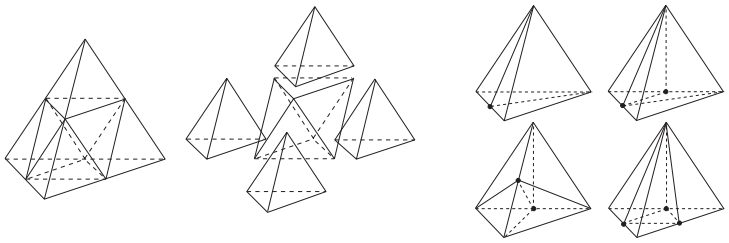
- Class 2 diamond: the cell is *hexadecahedral-shaped* and can be oriented along x, y or z axis;
- Class 3 diamond: the cell is *octahedral-shaped* and can be oriented along x, y or z axis.

The entire subdivision process is summarized by the three steps of the cyclic process in Fig. 9.a-c. Each diamonds is characterized by a center that corresponds to the midpoint of edge AB, which is the longest edge of the cell. The midpoint of edge AB is always selected as the dividing point and corresponds respectively to the center of the cube (Fig. 9.a), to the center of the cube faces (Fig. 9.b) and to the midpoint of the cube edges (Fig. 9.c). The subdivision of Class 1 cells (Fig. 8.a) produces cells belonging to Class 2 (Fig. 8.b) that, subdivided again, produce tetrahedra belonging to Class 3 (Fig. 8.c). After the subdivision of Class 3 cells the configuration recursively returns to Step 1. Similarly to the MTF approach, each type of diamond in the SGS subdivision

schema belongs to a different step of the subdivision process and the overall process has a cyclic behavior. The difference between the two approaches relies mainly in the cell shape and in the way a cell can be identified. An important advantage of the SGS's approach is the fact that each diamond is completely characterized (i.e. type, orientation and refinement level it belongs to) by the geometrical position of its center. This means that a mesh refinement process can be simply implemented by inserting dataset vertices according to a particular visiting order, and that the topology of the cells has not to be stored explicitly but can be simply reconstructed from the center points coordinates.
A variation to the SGS approach has been proposed by Gregorski et al. [16]; they perform the refinement directly on the tetrahedra that compose the diamonds, and each diamond is seen as an aggregate of tetrahedra and not as the "main" entity. They also exploit the regularity of the scheme to achieve good memory layout. The regularity of the SGS scheme allows to represent the data by storing just the initial volume dataset and a second grid with an error estimate for each voxel node. The subdivision process is regular and the cell produced have not to be represented explicitly, nor their topology. This organization of the data allows efficient storing and LOD's extraction and adaptive traversal. At *rendering time*, for example, for extracting an isosurface at a given view-dependant accuracy, the SGS subdivision process is performed recursively by simply reading voxel nodes from the dataset (according to the peculiar subdivision order) and reconstructing on the fly the new cells introduced. The recursion ends when no more refinement is needed or when we produce class 1 diamonds at the finer resolution level.

4.4 Red/Green Subdivision

The Red/Green Subdivision approach was first introduced by Bank et al. [29] for 2D meshes and successively extended to three dimensional tetrahedral meshes by Bey [1]. Grosso et al. [17] adopted a red/green subdivision schema to formulate a mesh refinement approach based on finite element analysis. Their approach must meet several constraints necessary to perform finite element computations: the mesh should be *conforming* (this condition prevents *hanging nodes*) and *stable* under some measure criteria (degeneracy like zero-angles must be avoided to guarantee stability of the numerical computations); moreover, the subdivision should guarantee *nestedness*, i.e. each element produced during subdivision is obtained by subdividing an element in a coarser level of the refinement sequence.
Since it satisfies the above requirements, the Red/Green refinement algorithm produces a stable and consistent partitioning of the space. The subdivision proceeds following two main rules: a *red*, or regular, subdivision for single elements subdivision, and a *green*, or irregular, subdivision for boundary elements. In the case of 2D meshes, the red refinement rule divides a triangle into four congruent ones by connecting the midpoints of its edges. The green



(a) red regular subdivision (b) green irregular subdivision

Fig. 10. Red/green subdivision of a tetrahedron.

refinement consists of simple bisections connecting one edge midpoint with the opposite vertex. In the case of 3D meshes, the regular refinement rule first cuts off four sub-tetrahedra at the vertices of the cell, as shown in Fig. 10.a, leaving an undivided octahedron whose subdivision is not unique and depends on the choice of one of the three possible diagonals. The subdivision of the octahedron is performed following a strategy based on affine transformation to a reference tetrahedron [1] and guarantees the result to be a stable regular refinement. For a tetrahedron there exist $2^6-2 = 62$ possible green refinement patterns which can be reduced to 9 using symmetry considerations. Grosso et al. restricts them to four (see Fig. 10.b), performing red refinement on all the remaining patterns. Green refinement rules are normally local, tetrahedra refined through a green rule are inserted only to satisfy border conditions and to avoid hanging nodes. To guarantee stability green refined tetrahedra can be refined only once and, if further subdivision is needed, the originally green refined tetrahedron is re-refined with a red rule.

5 Comparative Evaluation

A comparative evaluation of the two approaches (regular *vs.* irregular) is not straightforward. Both approaches present strength and weakness, different implementations of each method exist with the consequence that the comparison becomes much more complicated. To perform an evaluation we have then identified some issues which depend mainly from the visualization and data management requirements.

Adaptivity. We indicate with the term *adaptivity* how well each approach is able to produce compact and accurate representations of the initial model at different resolutions. The extraction of constant/variable resolution representations can be driven by any of the following objectives: (a) to give a good approximation of the dataset shape (i.e. preserving accurately its external or internal boundaries), or (b) to give an accurate representation of the field values encoded in the input dataset. In general irregular techniques allow

for better adaptivity, mainly due to the freedom in the choice of the best-fit refinement pattern which can be selected for each atomic refinement step. Irregular methods based on atomic refinements actions (e.g. edge split) allow to refine in a very selective manner the mesh, either preserving accurately the boundaries or by ensuring accurate representation of the original field values (by the interpolation of the field values on the mesh vertices). Moving from high to low resolution, and vice-versa, the sub-regions, onto which irregular techniques need to operate the update, can be restricted to a quite small number of tetrahedra. Conversely regular techniques need to propagate the change to a wider area, because in most cases those type of techniques allows at most one level of difference in the refinement of adjacent cells. This makes in general the representations, built with an irregular approach, more compact (no. number of cells required) than the one produced with a regular one. For the two-dimensional case this problem has been numerically evaluated and verified in [12], probably similar results could be obtained also for the three-dimensional case. Considering here just the number of cells is justified by the fact that most rendering algorithm have a time complexity that depends on this factor. Please note that the actual storage size does not only depend on the number of components (no. tetrahedral cells), but also by other factors (e.g. space complexity of the auxiliary structures, need to explicitly represent topology).

Flexibility. The term *flexibility* indicates to which extent each method allows to change the parameters used to drive the data refinement/simplification. This is a critical feature for the visualization of datasets with multiple scalar/vectorial field attributes like, for example dataset coming from simulations, where multiple parameters (pressure, temperature, velocity, etc.) should influence an error-based refinement criteria. Changing dynamically the refinement constraints is easier when using regular refinement techniques. The regular subdivision pattern allows, when needed, to execute a fast update of the refinement hierarchy. This is possible because there is no need to change the topology or the temporal ordering of the possible refinement actions (since those actions are defined statically by the selected regular refinement kernel choice). Conversely, the selection of which refinement "pattern" has to be applied and how the compatible sequences of refinement actions should be ordered, are error-driven in irregular methods and depend heavily on the field attribute and error evaluation criterion taken into consideration during the multiresolution representation construction. Therefore, a dynamic change in the parameters used to evaluate the accuracy often requires to rebuild the multiresolution data structure from scratch (and tetrahedral mesh simplification is a costly process on complex datasets).

Efficient data access. We indicate with *efficient data access* how well the data can be organized in memory to allow for an efficient data fetching. Irregular techniques usually require random access to the data on disk. On the

contrary, regular techniques can be easily organized in memory to guarantee locality in memory access, especially during the refinement actions which are known only at run-time. This issue plays a fundamental rule since the size of the datasets is increasing with a higher rate than the RAM memory available on standard computers. The need to structure data representation in an easy and regular manner is a crucial point to simplify the design of out-of-core solutions [2, 7], which are becoming a must in high-range scientific visualization.

Space complexity. The adoption of a regular schema allows for an efficient storing of the data, since it is in general possible to avoid to encode explicitly the extent and the topology of the mesh region under refinement and, often, also of the refined section. Conversely, irregular techniques often require the explicit representation of the refined mesh region and of the interdependency relationships that exist between the elements that make up the multiresolution structure. Larger is the amount of information that we can implicitly represent, more compact in space will be the representation schema. The speed of current computers allows us to trade some computation cycles (to reconstruct on the fly either topology or geometry) for saving memory locations and reducing secondary memory fetching overheads.

Implementation easiness. An initial guess would be that the implementation and the debugging of a regular approach should be simpler. Conversely, our experience in the implementation of both irregular [6, 3, 5] and regular methods [25] leads to a different evaluation. As far as the depth of the refinement chain increases, even on dataset composed by a few million points, the structural complexity of the refinement structure grows exponentially both for the regular and irregular case making quite hard the debugging for both approaches, and very complicated to produce a bug-free and robust code.

Generality of the method. We indicate with Generality of the method how well a given approach can manage different types of datasets. Irregular techniques show to be more flexible. Regular techniques in fact perform well only on regular datasets and result to be difficult to generalize to irregular ones. Re-sampling an irregular dataset on a regular grid is not convenient for several reason. It is common that the ratio between the sizes of the smallest and the largest cell in those dataset could be easily up to 1:10,000 and large cell may lay in any position across the domain. This would require to use adaptive re-sampling strategies, which produce meshes at different resolutions also not easy to manage with standard regular multiresolution techniques. A second problem arises when we consider that most of the irregular datasets have non-convex domains and the regular grid should properly represent the original data domain. In this case the field has good chances to present sharp discontinuities across the boundary of such domain since it is unknown outside the boundary itself.

6 Conclusions

A synthetic overview of methodologies for the efficient multiresolution management of large volume datasets has been presented. We have introduced the main contributions currently present in literature, subdividing them into regular or irregular refinement techniques, and analyzed strength and weakness of both the approaches. We have seen that irregular techniques, at the expense of a more complex structure, allow a better approximation and tighter refinement of any kind of dataset, while regular techniques mainly allows for efficient refinement of regularly gridded datasets. On their side regular techniques deals better with all the issues arising from the necessity of working out-of-core. The adoption of regular subdivision pattern allows for efficient data organization to improve memory occupancy and efficient access policy to secondary memory, which is becoming a must in the visualization of very large datasets. Visualization results from both techniques are comparable, even if irregular techniques perform better on adaptive refinement while regular techniques deals better with dynamic changes of critical refinement constraint. Both approaches are best suited for specific kinds of problems and it is still not possible to formulate a unique judgement of quality that could prefer one approach instead of another.

References

1. J. Bey, *Tetrahedral grid refinement*, Computing **55** (1995), 355–378.
2. Y. Chiang, C. T. Silva, and W.J. Schroeder, *Interactive out-of-core isosurface extraction*, IEEE Visualization '98 Proceedings, IEEE Press, 1998, pp. 167–175.
3. P. Cignoni, D. Costanza, C. Montani, C. Rocchini, and R. Scopigno, *Simplification of tetrahedral volume with accurate error evaluation*, Proceedings IEEE Visualization'00, IEEE Press, 2000, pp. 85–92.
4. P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno, *Multiresolution modeling and rendering of volume data based on simplicial complexes*, Proc.of 1994 Symp. on Volume Visualization, October 17-18 1994, pp. 19–26.
5. P. Cignoni, Paola Magillo, Leila De Floriani, Enrico Puppo, and R.Scopigno, *Memory-efficient selective refinement on unstructured tetrahedral meshes for volume visualization*, IEEE TVCG **9** (2003), To appear.
6. P. Cignoni, C. Montani, E. Puppo, and R. Scopigno, *Multiresolution modeling and visualization of volume data*, IEEE TVCG **3** (1997), no. 4, 352–369.
7. P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno, *External memory management and simplification of huge meshes*, IEEE Transactions on Visualization and Computer Graphics **8**.
8. L. De Floriani, P. Magillo, and E. Puppo, *Building and traversing a surface at variable resolution*, Proc. IEEE Visualization 97, October 1997, pp. 103–110.
9. L. De Floriani and E. Puppo, *Hierarchical triangulation for multiresolution surface description*, ACM Transactions on Graphics **14** (1995), no. 4, 363–411.
10. L. De Floriani, E. Puppo, and P. Magillo, *A formal approach to multiresolution modeling*, Theory and Practice of Geometric Modeling, Springer-Velrag, 1997.

11. M. Duchaineau, M. Wolinshy, D.E. Sigeti, M.C. Miller, C.Aldrich, and M. B. Mineev-Weinstein, *ROAMing terrain: Real-time optimally adapting meshes*, Proc. IEEE Visualization 1997, pp. 81–88.
12. William Evans, David Kirkpatrick, and Gregg Townsend, *Right triangulated irregular networks*, Algorithmica **30** (2001), no. 2, 264–286.
13. R.J. Fowler and J.J. Little, *Automatic extraction of irregular network digital terrain models*, Siggraph '79 Proc., no. 3, 199–207.
14. M. Garland, *Multiresolution modeling: Survey & future opportunities*, EUROGRAPHICS'99, State of the Art Report (STAR), 1999.
15. M. Garland and P.S. Heckbert, *Surface simplification using quadric error metrics*, SIGGRAPH 97 Conference Proc., pp. 209–216.
16. B. Gregorski, M. Duchaineau, P. Lindstrom, V. Pascucci, and K.I. Joy, *Interactive view-dependent rendering of large IsoSurfaces*, Proc. IEEE Visualization 2002, pp. 475–484.
17. R. Grosso, C. Lürig, and T. Ertl, *The multilevel finite element method for adaptive mesh optimization and visualization of volume data*, IEEE Visualization 97, pp. 387–394.
18. S. Gumhold, S. Guthe, and W. Straßer, *Tetrahedral mesh compression with the cut-border machine*, Proceedings IEEE Visualization'99, IEEE, 1999, pp. 51–58.
19. H. Hoppe, *Progressive meshes*, Proceedings of SIGGRAPH '96 (1996), 99–108.
20. V. Pascucci, *Slow growing subdivision (sgs) in any dimension: towards removing the curse of dimensionality*, Computer Graphics Forum (Proc. EUROGRAPHICS '02), no. 3, 451 – 460.
21. V. Pascucci and C. L. Bajaj, *Time critical isosurface refinement and smoothing*, Proc. IEEE Symp. on Volume visualization 2000.
22. A. Plaza and G.F. Carey, *About local refinement of tetrahedral grids based on local bisection*, 5th International Meshing Roundtable (1996), 123–136.
23. J. Popovic and H. Hoppe, *Progressive simplicial complexes*, ACM Computer Graphics Proc., Annual Conference Series, (SIGGRAPH 97), pp. 217–224.
24. E. Puppo, *Variable resolution terrain surfaces*, Proceedings Eight Canadian Conference on Computational Geometry 1996, Ottawa, Canada, pp. 202–210.
25. V. Pascucci R. Borgo, *Distributed oriented massive data management: Progressive algorithms and data structures*, CODATA'02, pp. 387–394.
26. María-Cecilia Rivara, *Mesh refinement processes based on the generalized bisection of simplices*, SIAM Journal on Numerical Analysis (1984), no. 3, 604–613.
27. O.G. Staadt and M.H. Gross, *Progressive tetrahedralizations*, Proc. IEEE Visualization '98, IEEE, 1998, pp. 397–402.
28. I.J. Trotts, B. Hamann, and K.I. Joy, *Simplification of tetrahedral meshes with error bounds*, IEEE TVCG **5** (1999), no. 3, 224–237.
29. R.E. Bank A.H. Sherman A. Weiser, *Refinement algorithms and data structures for regular local mesh refinement*, Scientific Computing, IMACS Trans. on Scientific Computation, vol. 1, 1983.
30. P.L. Williams, N.L. Max, and C.M. Stein, *A high accuracy volume renderer for unstructured data*, IEEE TVCG **4** (1998), no. 1.
31. Y. Zhou, B. Chen, and A. Kaufman, *Multiresolution tetrahedral framework for visualizing regular volume data*, Proc. IEEE Visualization '97, pp. 135–142.