Simplex and Diamond Hierarchies: Models and Applications

K. Weiss[†1] and L. De Floriani[‡2]

[1]University of Maryland, College Park, USA
[2]University of Genova, Italy

Abstract

Hierarchical spatial decompositions are a basic modeling tool in a variety of application domains. Several papers on this subject deal with hierarchical simplicial decompositions generated through simplex bisection. Such decompositions, originally developed for finite elements, are extensively used as the basis for multiresolution models of scalar fields, such as terrains, and static or time-varying volume data. They have also been used as an alternative to quadtrees and octrees as spatial access structures and in other applications. In this state of the art report, we distinguish between approaches that focus on a specific dimension and those that apply to all dimensions. The primary distinction among all such approaches is whether they treat the simplex or clusters of simplexes, called diamonds, as the modeling primitive. This leads to two classes of data structures and to different query approaches. We present the hierarchical models in a dimension–independent manner, and organize the description of the various applications, primarily interactive terrain rendering and isosurface extraction, according to the dimension of the domain.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Hierarchy and geometric transformations I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types

1. Introduction

One of the fundamental problems in computer graphics, scientific visualization, geographic data processing, and shape analysis and understanding is how to deal with the huge amount of data that describe the objects of interest. Let us consider, as an example, terrain modeling and visualization, where we deal with millions to billions of elevation samples, or modeling a volume data set for visualization and analysis, where we need to deal with huge point data sets or very large meshes describing isosurfaces.

One common solution to deal with such large quantities of information in the form of point data or of meshes is to use hierarchical spatial decomposition techniques. The use of such techniques is quite diverse. For example, we have representations like octrees, kd-trees, or R-trees which are used as spatial indexes on points, or on 3D shapes, and we also have multiresolution representations for terrains, and volumetric 3D or higher dimensional scalar fields which are based on a hierarchical spatial decomposition of the domain [Sam06]. In this very broad context, we focus on hierarchical spatial decompositions of square, cubic or hypercubic domains that are sampled at regular intervals on each of the dimensions. A typical example is given by a regular grid of points in two, three or higher dimensions at which one or more scalar field values are given.

Although *irregularly* sampled domains provide a great deal of flexibility in the sampling locations, and can thus represent the features of a problem domain using fewer elements than a regularly sampled domain, they require explicit encoding for the positions and connectivity of their vertices (see Figure 2a). In contrast, regular sampling permits implicit encodings of position and connectivity, but it requires the domain to be uniformly sampled.

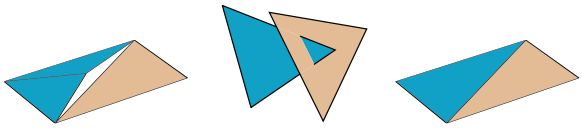
A common compromise is to adaptively sample the domain at the vertices of a regular grid using a d-dimensional octree. In dimension d, each hypercubic subdomain of side length ℓ is subdivided into 2^d hypercubes of side length $\ell/2$. Each such refinement introduces an exponential number of

† kweiss@cs.umd.edu
‡ deflo@disi.unige.it

© The Eurographics Association 2010.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

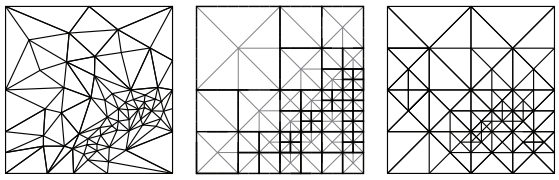
new vertices (in the dimension d of the domain). Additionally, such refinements can introduce cracks along neighboring cells of the domain decomposition, and consequently, discontinuities in functions defined on the cells. This can be remedied by using *restricted octrees* [VHB87, SS92], in which neighboring nodes can differ in size by at most a factor of two, which have been suitably triangulated (see Figure 2b).



(a) Crack (b) Intersect (c) Conforming

Figure 1: A decomposition is conforming (c) if it is crack-free (a) and its cells do not intersect (b).

The *simplex bisection* operator over regular grids breaks up each octree subdivision into d steps [Pas02]. Thus, its extracted meshes are significantly more adaptable to features defined on the domain [DFM02], while still enabling an implicit encoding of the underlying geometry [WD09a] (see Figure 2c). Conforming subdivisions apply to a cluster of simplexes with a local support and double the number of simplexes in the cluster, while only adding a single new vertex.



(a) Irregular (b) Quadtree (c) RSB Mesh

Figure 2: (a) Irregular sampling (b) triangulated restricted quadtree (c) Regular Simplex Bisection mesh.

In this state of the art report, we consider approaches that decompose a regularly sampled domain using *Regular Simplex Bisection (RSB)*. We consider such approaches in 2D, 3D and in higher dimensions and classify them on the basis of the dimension of the domain subdivided, which can be two-, three- or higher dimensional, and on the choice of the basic building blocks of the representation, which can be simplexes generated by the RSB rule, or *diamonds* generated by collections of such simplexes sharing a common bisection edge. In addition, we consider the method by which the various representations are queried, and the applications of such representations, mainly focusing on terrain modeling and visualization and analysis of 3D and higher dimensional scalar fields, which have been the major application areas of such hierarchical simplicial structures. On the other hand, simplicial hierarchies based on regular simplex bisection are a valid alternative to quadtree and octrees as spatial access structures [CGG*03a, CGG*04, AM04]. These hierarchies have been used in finite element analysis and actually originate from that field [Riv91, Mau95, Heb94]. Other applications include image reconstruction [Heb98], subdivision surfaces [VZ01], direct volume rendering [MDM04], surface reconstruction [MVT03] and segmentation of volumetric datasets [KTY*04].

To highlight the RSB framework, we separate the models for hierarchical domain decomposition, which we treat in a dimension–independent manner, from the applications of such decompositions, where we can focus on the dimension–specific optimizations that have been proposed in the literature. We feel that this abstraction enables one to focus on the properties of these decomposition and to understand the suitability of this approach for new application domains.

The reminder of this state of the art report is organized as follows. We present some preliminary definitions and related work in Section 2. In Section 3, we focus on the differences in modeling primitives for describing meshes from a dimension–independent perspective. Using this distinction, in Sections 4 and 5, we describe the hierarchies of meshes defined by the dependency relations among the primitives and the means of querying such hierarchies, respectively. Next, we survey the applications of RSB meshes in 2D (Section 6), 3D (Section 7) and higher dimensions (Section 8). We conclude in Section 9 with suggestions for when to use simplex hierarchies or diamond hierarchies.

2. Preliminaries

In the following, we assume we have a domain $\Omega \in \mathbb{R}^d$, and a set of samples V, which can be *regularly* or *irregularly* distributed within Ω. Furthermore, Ω is decomposed by a set of polyhedral cells whose vertices are defined on a subset of V. We are primarily concerned with conforming simplicial meshes defined over a regularly sampled domain.

2.1. Simplicial complexes

A k-simplex σ is the convex hull of $k+1$ affinely independent points in a subspace of $\mathbb{R}^d$, where k is called the *order* of the simplex. A *simplicial mesh* Σ is a finite collection of simplexes such that if σ is a simplex in Σ, then all of the simplexes bounding it (called the *faces* of σ) also belong to Σ, and the interiors of all simplexes in Σ are disjoint.

The *dimension* or the *order* of a simplicial mesh is the maximum of the orders of the simplexes forming it. Simplexes that are not on the boundary of any other simplex are called *top simplexes*. In a simplicial mesh of order d with a manifold domain, as we will consider here, all top simplexes have order d.

If the intersection of any two simplexes σ_1, σ_2 in a

© The Eurographics Association 2010.

simplicial mesh Σ is a lower dimensional simplex on the boundary of σ_1 and σ_2, then Σ is *conforming*. A conforming simplicial mesh is also referred to as a *simplicial complex*. Figure 1 illustrates examples of conforming and non-conforming meshes in 2D.

A simplicial mesh whose cells are defined by the uniform subdivision of a small set of cells into scaled copies is called a *nested mesh*.

2.2. Nested mesh refinement

A *refinement scheme* consists of rules for replacing a set of cells Γ_1 with a larger set of cells Γ_2 covering the same domain. When Γ_1 and Γ_2 are both simplicial meshes, the refinement defined by replacing Γ_1 from a simplicial mesh Σ with Γ_2 does not invalidate Σ. Furthermore, when Γ_1 and Γ_2 share the same combinatorial boundary, the refinement does not introduce cracks into Σ, i.e, it is conforming.

Recall that two cells τ_1 and τ_2 are *similar* if there is an affine mapping $\mathcal{A}$ (defined by translations, rotations, reflections and uniform scaling) between them, i.e. $\tau_1 = \mathcal{A} \cdot \tau_2$.

The number of similarity classes for the cells generated by successive refinements is an important characteristic of a refinement scheme since it enables the analysis of properties of all generated cells. In particular, it is important in many applications, such as finite element analysis, for the vertex angles to be bounded. Such a scheme is referred to as *stable* [Bey00].

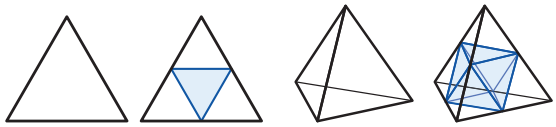
The two primary categories of nested refinement schemes for regularly sampled domains are those built on *regular refinement* (see Section 2.2.1) and on *simplex bisection* (see Section 2.2.1). For an example of a nested mesh refinement scheme over irregularly sampled domains, see [DP95].

2.2.1. Regular refinement

The *regular refinement* of a d-dimensional cell τ is defined by adding vertices at all edge midpoints of τ and decomposing τ into 2^d disjoint cells covering τ [BSW83].

Regular refinement on (hyper)-cubic cells, generates quadtree and octree decompositions as well as their higher dimensional analogues, in which all 2^d children are similar and share the midpoint of their parent's domain as a vertex. The recent book by Samet [Sam06] provides a detailed overview of quadtree-like decompositions.

Regular refinement of a triangle σ generates four triangles that are similar to σ, of which three triangles are adjacent to the vertices of σ, while the fourth triangle is defined by the edge midpoints of σ (see Figure 3a). However, on simplicial meshes of *order* $d > 2$, regular refinement is not uniquely defined, and the children are not guaranteed to be similar to their parent. For example, when refining a tetrahedron σ, the four children tetrahedra adjacent to the vertices of σ are similar to σ, while the remaining four tetrahedra which are obtained by subdividing an octahedral domain $\mathcal{O}$ defined by the edge midpoints of σ, are not (see Figure 3b).



(a) Triangle Refinement (b) Tetrahedral Refinement

Figure 3: Regular refinementof a simplex. (a) A triangle is decomposed into four similar triangles. (b) A tetrahedron is decomposed into four similar tetrahedra and four non-similar tetrahedra covering an octahedral domain $\mathcal{O}$ (blue).

The decomposition of $\mathcal{O}$ into four tetrahedra is not uniquely defined. Zhang [Zha95] introduces a *geometry-based* rule for regular tetrahedral refinement in which $\mathcal{O}$ is decomposed into four tetrahedra along its *shortest* diagonal. Zhang proves that this generates a stable refinement when the angles of the initial mesh are non-obtuse, whereas other geometric choices, such as refinement along the octahedron's longest diagonal, do not yield stable refinements.

Bey [Bey95] proposes a *typographical* refinement rule based on maintaining a specific vertex ordering during refinement. In contrast to geometric refinements, typographical refinements require the cells to be initially ordered, but are invariant under affine transformations.

An interesting modification is the Tetrahedral/Octahedral regular refinement scheme [GG98, GG00] in which the octahedra generated by regular tetrahedral refinement are treated as primitives. In this scheme, each tetrahedron is refined into four similar tetrahedra and one octahedron, while each octahedron is decomposed into six similar octahedra (incident to its vertices), and eight similar tetrahedra (defined by the edge centers of the octahedron and its midpoint).

Bey shows [Bey00] that his tetrahedral refinement scheme [Bey95] as well as the 2D scheme of Banks et al. [BSW83] are instances of a d-dimensional typographical scheme introduced by Freudenthal [Fre42], and proves a tight bound of $d!/2$ similarity classes generated for each simplex class in the initial mesh. Moore [MW95] provides a consistent labeling and simplex enumeration algorithm for the 2^d children generated by regular refinement.

Regular refinement does not generate conforming adaptive refinements of a domain, i.e. where the cells can be at different levels of resolution. Banks et al. [BSW83] introduce the *red/green refinement* scheme in 2D, in which the regular (*red*) refinement rules are augmented by a set of irregular (*green*) refinement rules, also called *closure* refinements, to patch cracks between regular cells at different resolutions. An additional *regularity constraint* restricts the degree of decomposition between neighboring cells, thereby

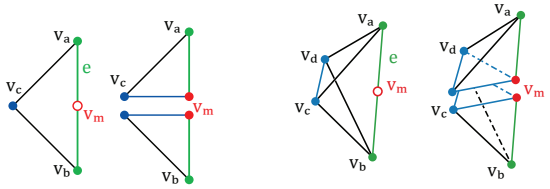
© The Eurographics Association 2010.

reducing the number of green refinement rules that need to be considered. Conforming variable-resolution meshes are extracted by first applying regular (red) refinement to all cells that fail to satisfy an *acceptance criterion* and then applying closure (green) refinements to all cells containing at least one refined edge. It is important to note that the simplices generated by green refinements are never refined. This scheme has been extended to tetrahedral meshes [Bey95].

The recent *RGB subdivision* scheme for triangle meshes [PP09] introduces *blue* updates to transition between triangles generated by regular (red) and irregular (green) refinements.

2.2.2. Simplex bisection schemes

The second class of nested meshes is defined by the *simplex bisection* operation over simplicial meshes. This operation bisects a d-simplex σ along the hyperplane defined by the midpoint $\mathbf{v}_m$ of some edge $\mathbf{e}$ and the $(d-1)$ vertices of σ not incident to $\mathbf{e}$. We call $\mathbf{e}$ the *bisection edge* of σ. In contrast to regular refinement, which generates 2^d cells, simplex bisection generates only two simplexes with disjoint interiors covering its domain. Figure 4 illustrates the simplex bisection rule in 2D and 3D.



(a) Triangle Bisection (b) Tetrahedron Bisection

Figure 4: A simplex is bisected along the hyperplane defined by the midpoint $\mathbf{v}_m$ of an edge $\mathbf{e} := [\mathbf{v}_a, \mathbf{v}_b]$ and all vertices not adjacent to $\mathbf{e}$.

Similarly to regular refinement, the decomposition induced by simplex bisection is not uniquely defined. In particular, it does not specify which edge to bisect. Thus, researchers have proposed schemes to implicitly determine the bisection edge via geometric [Riv84] or typographical [Mit91] schemes. As with regular refinement, typographical approaches require an initialization process to ensure consistent labeling, but they are affine invariant, while geometric approaches are not affine invariant but do not require preprocessing.

Rivara's *longest edge bisection* scheme over triangle [Riv84] and tetrahedral meshes [Riv91] uses a geometric property to determine the bisection edge. Specifically, it chooses the longest edge of the simplex as its bisection edge.

Mitchell [Mit91] (following Sewell [Sew72, Sew79]) introduces the *newest vertex bisection* algorithm for irregular triangle meshes, where the bisected edge is always opposite the most recently introduced vertex. This edge can be implicitly determined through a consistent ordering of the vertices, where, e.g., the newest vertex is always in the final position. Then, to bisect a triangle with vertices $[\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c]$ we generate a vertex $\mathbf{v}_m$ at the midpoint of edge $[\mathbf{v}_a, \mathbf{v}_b]$, and the two children are $[\mathbf{v}_a, \mathbf{v}_c, \mathbf{v}_m]$ and $[\mathbf{v}_b, \mathbf{v}_c, \mathbf{v}_m]$ (see Figure 4a). This scheme has been generalized to tetrahedral meshes as well [Bän91, LJ95, AMP00].

Maubach [Mau95] generalizes these approaches to d-dimensional domains through an implicit typographical scheme (different from, but equivalent to [Mit91] in $2D$) that cycles between d rules. In this scheme, the bisection edge of a simplex generated using rule i, for $0 \leq i < d$ is defined by the first vertex and the i^{th} vertex. A similar d-dimensional scheme was introduced by Traxler [Tra97].

When applied to an arbitrary simplicial mesh, this scheme generates $2^{d-2}d!$ similarity classes of simplices for each simplex class in the initial mesh [Bey00]. We discuss the case of bisection applied to simplicial meshes decomposing an integer lattice in Section 3.2.

Simplex bisection is not a conforming refinement. Specifically, it introduces cracks between the neighbors of a bisecting simplex σ that are adjacent to the bisection edge, and the two simplices generated during σ's bisection (see Figure 1a). This can be remedied by ensuring that all simplices incident to the bisection edge (a) share the same bisection edge and (b) are subdivided concurrently. When the first property is not satisfied for a bisecting simplex σ, a *closure* operation can be applied (recursively) on σ's neighboring simplexes, triggering additional bisection operations, and thereby enforcing conforming refinements (as we describe in Section 5).

2.2.3. Comparison

Over an arbitrary simplicial mesh, regular refinement generates fewer similarity classes of simplices than bisection (i.e. $d!/2$ vs. $2^{d-2}d!$). Furthermore, these similarity classes are a subset of those generated by bisection [Bey00]. However, when applied to a regularly sampled hypercubic domains (see Section 3.1), the number of similarity classes generated by bisection has been shown to be only d [Mau95].

Although red/green schemes have two types of refinement rules (i.e. red and green), theoretical considerations regarding the number of similarity classes are typically only applied to simplices generated by red refinements, while ignoring the simplices generated by green refinements. For example, in 3D, the number of tetrahedra generated by green refinements has been shown to be greater than 150 [Bas94, Mau96]. In contrast, since the closure operation for bisection is defined recursively in terms of simplex bisection operations, it generates only one family of simplex similarity classes. This simplifies analysis of the mesh as well as data structures and refinement algorithms, especially as the dimensionality of the domain increases.

© The Eurographics Association 2010.

Meshes generated through bisection are significantly more adaptive than those generated by regular refinement as each bisection only generates two new elements while each regular refinement generates 2^d new elements. Thus, the refinement rate for bisection can be viewed as being independent of the dimensionality of the mesh [Pas02].

2.3. Multiresolution models based on irregular meshes

There are, in the literature, several proposals for multiresolution models based on irregular triangle meshes, in particular capable of supporting *selective refinement*, i.e., of extracting meshes at a uniform or variable resolution. This capability derives from organizing updates to a coarse mesh according to a partial order based on a dependency relation, so that a virtually *continuous* set of representations, in which the resolution may vary in different parts of the domain, can be extracted. Dependencies between updates may be represented directly through a Directed Acyclic Graph (DAG) [DPM97, GTLH98] (or similar structures [XESV97]), or through forests of binary trees of vertices [Hop97, LE97], sometimes enriched with vertex labeling [ESV99].

In [DPM97], a *continuous* multiresolution model for representing k-dimensional spatial entities through simplicial complexes, called a *Multi-Tessellation (MT)*, has been defined which is independent of the dimension of the complex and of the specific strategy through which the model is built. The fundamental idea is that the basic structure of any multiresolution model is a coarse complex at the lowest resolution plus a collection of small complexes arranged according to a partial order. In [CDM*04], a multiresolution data structure for irregular tetrahedral meshes, called an *Edge-based Multi-Tessellation (MT)*, has been developed to efficiently encode an MT describing a three-dimensional scalar field built through a simplification process based on edge collapse.

3. Regular Simplex Bisection (RSB)

In this Section, we review the components of the *Regular Simplex Bisection (RSB)* scheme which guides the generation of adaptive simplicial complexes from a regularly sampled domain. The two components of this scheme are:

1. an initial triangulation of a (hyper)-cubic domain Ω, called *Kuhn's subdivision* (see Section 3.1), and
2. a *typographical* simplex bisection operation that is equivalent to that of Maubach [Mau95] (see Section 2.2.2).

We refer to any simplex generated by repeated applications of Maubach's bisection scheme to a simplex from a Kuhn-subdivided cube as an *RSB simplex*, and to a mesh composed entirely of RSB simplices as an *RSB mesh*. We review properties of RSB meshes in Section 3.2.

We note that, for $d < 4$, the regular simplex bisection scheme is equivalent to the longest edge bisection scheme [Riv91] defined on the same initial mesh. Since the latter decomposition is more intuitive and easier to describe, it is the more commonly used term for $d = 2$ and $d = 3$. However, when $d \geq 4$, the bisection edge of an RSB simplex is no longer guaranteed to be its (geometrically) longest edge, and the decompositions are no longer equivalent.

3.1. Simplicial decomposition of a hypercube

We consider the canonical subdivision of a hypercube into $d!$ simplexes along a diagonal. This decomposition was initially proposed by Freudenthal [Fre42] and was popularized by Kuhn [Kuh60] in the context of fixed point calculations.

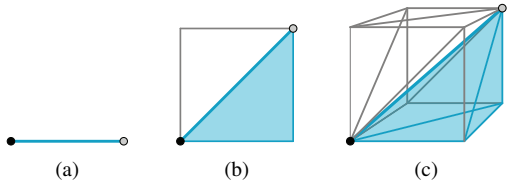


Figure 5: Kuhn-subdivided hypercubes in (a) 1D (b) 2D (c) 3D. One of the $d!$ simplexes is highlighted in blue. All edges are aligned with the diagonal of an axis-aligned hypercube.

The Kuhn-subdivision of a d-dimensional unit cube h, which we denote as $\mathcal{K}(h)$, is a simplicial complex [AG79] defined by $d!$ simplexes of order d, sharing a common diagonal of h. Figure 5 illustrates Kuhn-subdivided d-cubes for $d = 1,2,3$, and highlights one of the $d!$ simplexes.

An interesting property of $\mathcal{K}(h)$ is that its lower dimensional faces are Kuhn-subdivided as well [DM82, WD09a]. Furthermore, opposite faces of a Kuhn-subdivided cube are compatibly decomposed, thus, a regularly sampled domain can be tiled by Kuhn-subdivided cubes [Kuh60, Tod76].

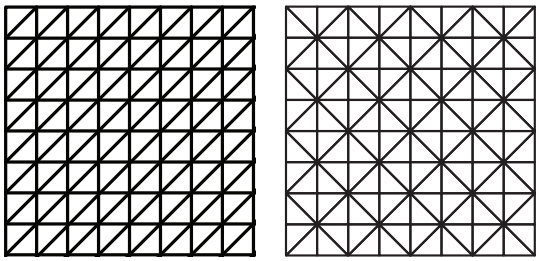
When a hyper-rectangular domain is tiled by Kuhn cubes using only translation, this tiling is referred to as *Freudenthal's triangulation* [Tod76] and typically denoted as K_1 (see Figure 6a for an example in 2D).

Alternatively, the *Tucker-Whitney* triangulation [Tuc45, Whi57], typically denoted as J_1, is obtained over the same grid by reflecting adjacent Kuhn cubes (see Figure 6b for an example in 2D). Due to the reflectional symmetry, J_1 can be viewed as a tiling of an integer lattice by clusters of 2^d oriented Kuhn-cubes, which we refer to as *fully subdivided*-cubes [WD09a], and denote as $\mathcal{F}$ (see Figure 8c for $d = 2$ and Figure 16a for $d = 3$). A fully-subdivided d-cube $\mathcal{F}$ is thus defined by $2^d \cdot d!$ simplices of order d. This notation can be simplified using the *double-factorial* [Mes48] identity

$$(2d)!! \equiv 2^d \cdot d! \tag{1}$$

where $n!! = 1$, if $n \in \{0,1\}$, and $n!! = n*(n-2)!!$, otherwise. The first few values of $(2d)!!$, for $d = [1,2,3,4]$ are $[2,8,48,384]$.

© The Eurographics Association 2010.



(a) K_1 (b) J_1

Figure 6: Tiling the plane with Kuhn-squares. (a) Translating adjacent tiles leads to *Freudenthal's Triangulation* K_1. (b) Reflecting adjacent tiles leads to the *Tucker-Whitney Triangulation* J_1.

Given a Kuhn-subdivision $\mathcal{K}(\Omega)$ of a hypercubic domain Ω, Freudenthal's algorithm [Fre42, Bey00], generates K_1 by applying *regular refinement* to its simplices (upper path of Figure 7). Alternatively, this decomposition can be obtained by applying the Kuhn-subdivision to the leaves of a complete quadtree (lower path of Figure 7) and has been referred to as a *simplicial quadtree* [MW95, LS00]. K_1 is therefore the canonical decomposition for *complete* red/green refinement meshes.

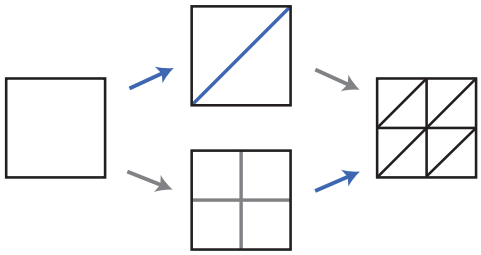
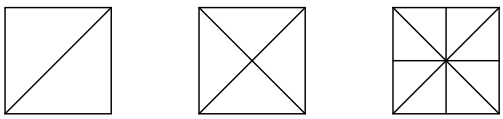


Figure 7: K_1 can be generated by applying regular refinement to a Kuhn subdivided cube (upper path) or by Kuhn-subdividing the leaves of a complete quadtree (lower path).

In contrast, the Tucker-Whitney triangulation J_1 can be obtained by applying successive RSB refinements to all elements of $\mathcal{K}(\Omega)$ [Mau96]. Thus, J_1 is the canonical decomposition for *complete* RSB meshes (see Figure 8).



(a) $\mathcal{K}(h)$ (b) (c) $\mathcal{F}(h)$

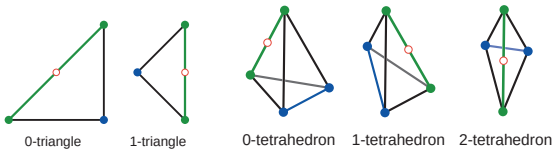
Figure 8: Bisecting the triangles in $\mathcal{K}(h)$ twice generates $\mathcal{F}(h)$. In general, d bisections are required.

Todd [Tod76] evaluates the efficiency of K_1 and J_1 (and their adaptive counterparts) with respect to the number of simplices intersecting an arbitrary line segment. While both decompositions achieve the same asymptotic complexity, K_1 can be more efficient when the direction of the line is known in advance, while J_1 is less sensitive to the line's direction.

3.2. Regular Simplex Bisection simplexes

Maubach's bisection scheme, when applied to a Kuhn-subdivided hypercubic domain, generates d *similarity classes* of RSB simplexes [Mau95].

The *class* of an RSB simplex cycles with every d refinements, i.e. after m bisections, the class of a simplex σ is $i := m \bmod d$. Furthermore, the *bisection edge* of a *class* i RSB simplex is the diagonal of an axis aligned $(d-i)$-cube (i.e. a hypercube of dimension $(d-i)$). Thus, the $d!$ simplexes in $\mathcal{K}(h)$ all belong to class 0, and the two simplexes generated by bisecting a simplex of class i belong to class $(i+1) \bmod d$. Figure 9 illustrates the two classes of RSB triangles and the three classes of RSB tetrahedra.



(a) RSB triangles (b) RSB tetrahedra

Figure 9: The two *classes* of RSB triangles in 2D (a) and the three *classes* of RSB tetrahedra in 3D (b). In general, the bisection edge (green) of a class i RSB simplex is aligned with the diagonal of an axis aligned $(d-i)$-cube.

After d bisections, the size of an RSB simplex is a factor of two smaller than its d-fold ancestor [Mau95].

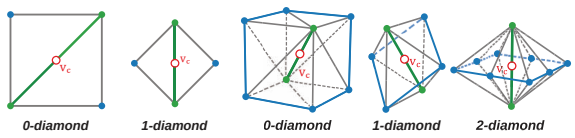
3.3. Diamonds

We are often interested in generating conforming meshes using the regular simplex bisection scheme. As mentioned above (Section 2.2.2), simplex bisection (and thus regular simplex bisection) is not a conforming refinement. In general, it introduces cracks into an RSB mesh between the bisected simplex and all *neighboring* d-simplexes adjacent to the bisection edge. In other words, not all nested RSB meshes are conforming.

Conforming RSB meshes are maintained through the simultaneous bisection of all simplexes in the mesh that share the same bisection edge. The cluster of such simplexes is typically called a *diamond* [DWS*97, GDL*02, Pas02, WD09a], and we refer to the bisection edge as the *spine* of

© The Eurographics Association 2010.

the diamond. Thus, all RSB simplexes within a diamond belong to the same similarity class, and there are d similarity classes of diamonds, in correspondence to the d similarity classes of RSB simplexes. Figure 10 illustrates the two classes of diamonds in 2D and the three classes of diamonds in 3D.

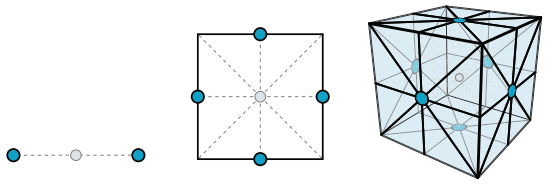


(a) 2D diamonds (b) 3D diamonds

Figure 10: The two classes of diamonds in 2D (a) and the three classes of diamonds in 3D (b). In general, the *spine* (green edge) of an i-diamond is aligned with the diagonal of am axis-aligned $(d-i)$-cube.

In 2D, diamonds are defined by two isosceles right triangles that are adjacent along their hypotenuse (see Figure 10a), and thus lie in a square domain defined by 4 vertices. The three classes of diamonds in 3D are formed by 6, 4 and 8 tetrahedra and contain 8, 6 and 10 vertices, respectively [ZCK97,GDL*02] (see Figure 10b). Four dimensional diamonds are defined by 24, 12, 16 and 48 *pentatopes* (i.e. 4-simplexes), respectively [LDS04].

Weiss and De Floriani [WD09a] examine the combinatorial structure of diamonds in arbitrary dimension. They define the *boundary of a fully subdivided cube*, which we denote as $\mathcal{B}_F$, as the simplicial complex obtained by removing all interior simplices from a fully-subdivided cube $\mathcal{F}$, i.e. remove all simplexes incident to the vertex at the midpoint of $\mathcal{F}$. The boundary of a fully subdivided cube $\mathcal{B}_F$ is thus, a simplicial complex defined by $(2d)!!$ simplexes of order $(d-1)$. Figure 11 illustrates $\mathcal{B}_F$ in 1D, 2D and 3D.

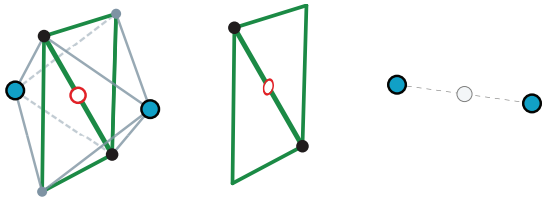


(a) (b) (c)

Figure 11: $\mathcal{B}_F$, the boundary of a fully-subdivided d-cube in (a) 1D (b) 2D and (c) 3D, is defined by $(2d)!! \equiv 2^d \cdot d!$ RSB simplexes of order $(d-1)$. The dashed lines and gray vertex at the midpoint of the cube belong to $\mathcal{F}$ but not $\mathcal{B}_F$.

A d-dimensional diamond of class i can be decomposed as a cross product of two mutually orthogonal hypercubes: a Kuhn-subdivided $(d-i)$-cube and the boundary of a fully subdivided i-cube. For example, a three dimensional 1-diamond (Figure 12a) can be decomposed into a Kuhn-subdivided 2-cube (Figure 12b) and the boundary of a fully subdivided 1-cube (Figure 12c).

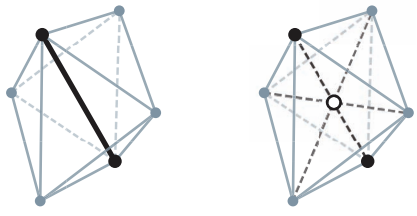
Consequently, an i-diamond of dimension d is defined by $(d-i)! \cdot (2i)!!$ simplexes all sharing their spine, which is the diagonal of an axis-aligned $(d-i)$-cube and contains $(2^{d-i}) + (3^i - 1)$ vertices. The number of simplexes in a d-dimensional diamond is therefore factorial in the dimension, while the number of vertices is exponential in the dimension.



(a) (b) (c)

Figure 12: A 1-diamond in 3D (a) can be decomposed into a Kuhn-subdivided 2-cube (b) and the boundary of a fully subdivided 1-cube (c). In general, an i-diamond in d dimensions can be decomposed into a Kuhn-subdivided $(d-i)$-cube and the boundary of a fully subdivided i-cube.

A diamond δ is refined by bisecting all of its simplexes using the regular simplex bisection operation. The local effect of such a refinement on an RSB mesh Σ is to remove the spine of the diamond, to insert a vertex at the midpoint of its spine, which we refer to as the *central vertex* of the diamond and denote as v_c, and finally to insert an edge from v_c to all vertices of δ (see Figure 13 for an example in 3D). Diamond refinement is an instance of *stellar subdivision* and is a conforming refinement in arbitrary dimension [Ale30, New31, Lic99]. The one to one correspondence between a diamond and its spine enables diamonds to be uniquely identified by their spine, or alternatively, by their central vertex [GDL*02].



(a) δ (b) δ_s

Figure 13: Refinement of a three dimensional 1-diamond δ removes one edge (the *spine*), adds a single vertex (the *central vertex*), and adds an edge from that vertex to each of the original vertices of δ.

© The Eurographics Association 2010.

Pascucci's *Slow Growing Subdivision (SGS)* scheme [Pas02] generalizes the diamond subdivision paradigm to cell complexes of arbitrary dimension. In the SGS scheme, a d-dimensional cell of class i is subdivided by inserting a vertex v at its center and replacing it with pyramid-shaped cells with apex at v and bases defined by its facets. All pyramids sharing the same $(d-(i+1))$-dimensional face of a class 0 cell are then merged into a new cell of class $(i+1) \mod d$.

4. Hierarchies of nested RSB meshes

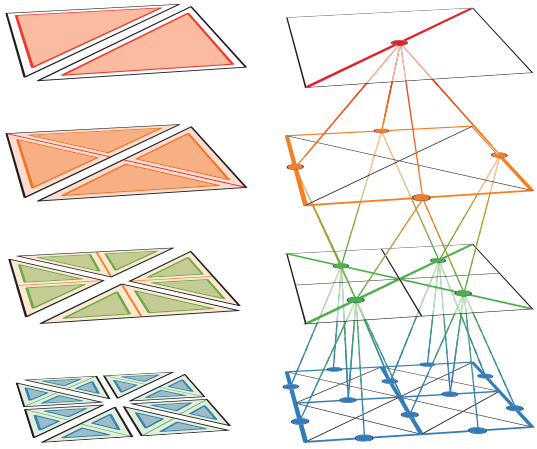
In this Section, we review and analyze representations for hierarchies of nested RSB meshes, based on the containment relations among top simplexes or on the parent-child relation among diamonds. To simplify the discussion, we will refer generally to hierarchies of nested RSB meshes as *RSB hierarchies*.

The basic classification for representations of RSB meshes is between *simplex-based representations* and *diamond-based representations*. The former consider the simplex as the basic primitive, while the latter consider the diamond as primitive. Note that a simplex-based representation implicitly encodes all the nested RSB meshes i.e. meshes generated from the initial Kuhn-subdivided domain through regular simplex bisection, while a diamond-based one encodes only the conforming nested RSB meshes with the same domain and set of vertices.

A simplex-based representation is described by a *hierarchy of simplexes* which encodes the containment relation between top simplexes, while a diamond-based representation is described by a DAG of diamonds, called a *hierarchy of diamonds*, where the parent-child relation between diamonds is induced by the containment relation among all of a diamond's top simplexes. Both hierarchies can be encoded through the use of pointers, but the regularity of the domain decomposition admits an implicit formulation of these spatial and hierarchical relationships. Thus, we can distinguish in both cases between explicit and implicit representations.

4.1. Simplex hierarchies

The containment relation between simplexes in a nested RSB mesh defines a hierarchical relationship, where the two simplexes created during the bisection of a *parent* simplex σ are referred to as the *children* of σ. This relationship can be captured as a binary tree, often referred to as a *bintree* [VH89, DWS*97, LRC*02], whose root is a simplex of the Kuhn-subdivided hypercubic domain Ω. Thus, the hierarchy of nested RSB meshes can be modeled as a forest of $d!$ binary trees, which we call a *hierarchy of simplexes*. When the domain is specific to 2D, or 3D, we refer to the model as a *hierarchy of (right) triangles*. or a *hierarchy of tetrahedra*, respectively.



(a) Hierarchy of Triangles (b) Hierarchy of Diamonds

Figure 14: Comparison of the two hierarchical models for nested RSB meshes (in 2D). (a) A *hierarchy of simplexes* encodes the simplex bisection containment relation as a pair of binary trees. (b) A *hierarchy of diamonds* encodes the dependency relation among the diamonds as a rooted DAG.

The *depth* of a simplex in a hierarchy of simplexes is defined recursively as 0 for the bintree roots, and as one greater than the depth of its parent simplex otherwise. All root simplexes belong to $\mathcal{K}(\Omega)$, so they are all RSB simplexes of class 0. Since regular simplex bisection is used to generate the simplexes at successive depths, all simplexes at the same bintree depth belong to the same class of simplexes. Furthermore, since the d classes of simplexes in a hierarchy of simplexes, repeat cyclically, the class of a simplex at depth m is $i := (m \mod d)$. The simplexes at d successive depths define a *level* of the hierarchy. The *level* ℓ of a simplex at depth m is then $\ell := \lfloor m/d \rfloor$.

Figure 14a illustrates the hierarchical relationship between triangles in a hierarchy of triangles at four successive depths (two levels). Note that the hierarchy of triangles contains two trees, and that each triangle has two children covering its domain, thus forming a nested decomposition of the square domain.

When a nested RSB mesh is encoded as a forest of simplex trees, each simplex can be indexed through a unique *location code* [Sam06]. Assuming that the two children of a simplex are labeled as child 0 and child 1, a location code of a simplex σ at depth m is encoded using the index of the bintree containing σ and an m-bit binary string indicating the traversal path from the root of the bintree to σ. The vertices of a simplex can be determined from those of its bintree root by descending the tree.

© The Eurographics Association 2010.

4.2. Diamond hierarchies

The containment relation among the simplexes of a nested RSB mesh induces a hierarchical *dependency relation* on the diamonds within the hierarchy. Specifically, a diamond δ_c is a *child* of a diamond δ_p if δ_c contains at least one simplex generated during the bisection of δ_p's simplexes. In contrast to the containment relation induced by the bisection operation, the domain of a diamond's children is not nested within its own domain and diamonds can have multiple parents and children.

Lindstrom et al. [LKR*96] were the first to consider the *simplification dependency relation* among the vertices of a nested RSB mesh in the context of terrain simplification. They observed that each vertex corresponds to triangles in two different branches of the hierarchy of triangles and that cracks are introduced into the mesh if only one of those branches is fused. They propose a conforming *triangle fusion* operation that replaces both pairs of sibling triangles with their parent triangles. Pajarola [Paj98] considers the *refinement dependency relation* for triangle bisection over a nested RSB mesh by reversing the simplification dependency relation arcs of Lindstrom et al. [LKR*96]. Puppo [Pup98] describes how the diamond dependency relation can be described as a special case of their Multi-Tessellation (MT) framework.

Thus, the dependency relation can be modeled as a Directed Acyclic Graph (DAG) of diamonds, whose single root is the 0-diamond defined by $\mathcal{K}(\Omega)$. Figure 14b illustrates a 2D hierarchy of diamonds at four successive depths (two levels). Each diamond is indicated by its spine (a colored edge) and its central vertex (a colored circle). Due to the one-to-one correspondence between diamonds and their central vertices, each arc of the dependency graph connects the central vertex of a parent diamond to the central vertex of one of its children.

In 2D, diamonds have two parents and four children [LKR*96, Paj98]. Gregorski et al. [GDL*02] establish the number of parents and children of the three classes of diamonds as $\{3,2,4\}$ and $\{6,4,8\}$, respectively.

Weiss and De Floriani [WD09a] use their diamond decomposition (see Section 3.3) to find the number of parents and children of a diamond of arbitrary dimension. They demonstrate that the central vertices of a diamond's children coincide with the midpoints of the facets of its Kuhn-subdivided component, while those of its parents coincide with the facet midpoints of its fully-subdivided component. Thus, an i-diamond of dimension d has $2\cdot(d-i)$ children and $2\cdot i$ parents. However, when $i=d-1$, it has 2^d children, and when $i=0$, it has d parents. Thus, diamonds always have $O(d)$ parents, and have $O(d)$ children in all but the final class (where it is $O(2^d)$). As a concrete example, 4D diamonds have $\{8,6,4,16\}$ children and $\{4,2,4,6\}$ parents, respectively.

4.3. Dependency domains

Due to the 1-1 correspondence between diamonds and their central vertices, the dependency relation among diamonds is often studied by considering the set of vertices on which it depends [LKR*96, BLV03]. For refinements, a vertex depends on vertices introduced at shallower depths of the hierarchy, while, for simplifications, a vertex depends on vertices introduced deeper in the hierarchy.

Balmelli et al. [BLV03] consider the *splitting domain* and *merging domain* of each vertex in a 2D nested RSB. They define the splitting domain of a vertex $\mathbf{v}$ to be the domain covered by all triangles that must be refined concurrently to maintain a conforming triangulation upon the insertion of $\mathbf{v}$ to a nested RSB mesh. Similarly, the merging domain of a vertex $\mathbf{v}$ is the domain covered by all triangles that must be merged concurrently to maintain a conforming triangulation upon the removal of $\mathbf{v}$ from a nested RSB mesh. The splitting domain of a vertex is thus covered by all ancestor triangles up to the root(s) of the hierarchy while the merging domain of a triangle is covered by all descendant triangles down to the hierarchy's leaf nodes. They observe that both triangles in a diamond share the same merging domain, but have different splitting domains. This is due to the fact that simplexes of a diamond are generated during the refinement of a parent diamond, and thus some simplexes of a diamond can be absent from a conforming RSB mesh. However, when that diamond is refined (and thus, its central vertex added to the mesh) all of its simplexes are required to be in the mesh.

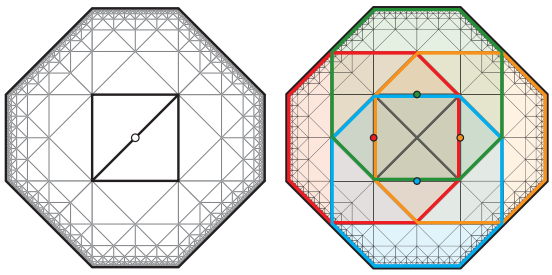
Gerstner [Ger03b] further develops this relationship by considering the limit shape of a diamond's merging domain. The set of *hierarchical descendants* of a 2D diamond covers a region of the domain whose limit shape is octagonal (see Figure 15a). If the side length of a triangle in the diamond is s and its hypotenuse is $s\sqrt{2}$, then its corresponding bounding octagon has edge lengths that alternate between s and $s\sqrt{2}$. The octagon is centered at the diamond's central vertex $\mathbf{v}_c$ and extends a distance of $1.5\cdot s$ from $\mathbf{v}_c$ towards the four edges of the diamond, and a distance of $s\sqrt{2}$ from $\mathbf{v}_c$ towards the diamond's four corner vertices.

The octagons corresponding to diamond merging domains form a containment hierarchy i.e. the merging domain of a diamond's four children are nested within its own merging domain (see Figure 15b). Gerstner shows that this nesting relationship forms the tightest possible bounding hierarchy for diamonds within an RSB hierarchy.

4.4. RSB hierarchies and fully-subdivided cubes

Since both types of RSB hierarchies are defined over the underlying Tucker-Whitney triangulation J_1 (see Section 3.1), which is tiled at each level of resolution by the highly symmetric fully subdivided cubes $\mathcal{F}$, researchers have analyzed the relationship between RSB hierarchies and $\mathcal{F}$. Observe that a *complete* RSB mesh of maximum level of resolution

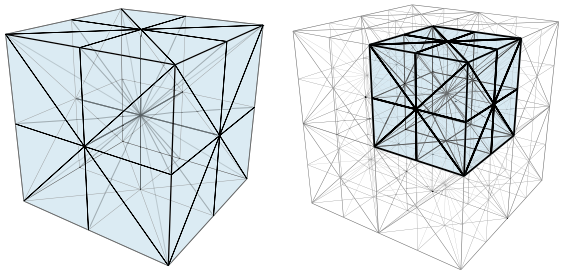
© The Eurographics Association 2010.



(a) Merging domain (b) Nested octagons

Figure 15: (a) The limit shape of a diamond's *merging domain* is an octagon with the edge lengths of its triangles. (b) The octagon shaped merging domains of its four children (with colored central vertices) are nested within its octagonal merging domain.

N is defined by a grid of $(2^N+1)^d$ vertices and is tiled by $(2^{N-1})^d$ fully subdivided cubes. Figure 16 illustrates a cubic domain Ω tiled by fully-subdivided cubes at two consecutive levels of resolution.



(a) J_1 at level 1 (b) J_1 at level 2

Figure 16: Each level of resolution within an RSB hierarchy is tiled by scaled fully-subdivided cubes. Two consecutive levels of resolution covering the same 3D domain are shown, containing 1, and 8 fully-subdivided cubes.

Hebert [Heb94] considers the subtree in a hierarchy of tetrahedra defined by the tetrahedra in a fully-subdivided cube $\mathcal{F}$ and their descendants at the same refinement level (which Atalay and Mount [AM04, AM07] generalize to arbitrary dimensions). Recall that a fully-subdivided d-cube $\mathcal{F}$ is defined by the 2^d Kuhn-subdivided cubes generated by reflecting a unit Kuhn cube across the d axis-aligned planes. Then each RSB simplex σ of class 0 in $\mathcal{F}$ can be indexed by a *reflection number* $i_r \in [0,2^d)$, encoding the index of the Kuhn-cube containing σ and a *permutation number* $i_p \in [0,d!)$ encoding σ's index within that Kuhn-cube. The *descendant number* $i_d \in [0,2^d-1)$ can be used to find the path from σ to any of its (2^d-1) descendants within this subtree. Thus each subtree indexes $(2^d-1)\cdot 2^d \cdot d!$ distinct RSB simplexes.

Since each fully subdivided cube covers a hypercubic domain, its corresponding subtree can be indexed by considering the octree-like *location code* from the hypercubic node covering the decomposition domain Ω. Since a d-dimensional octree node has 2^d children, each branch of this location code can be encoded using d bits. The location code for an RSB simplex σ at depth m based on this index can be encoded using $(d\cdot(\lfloor m/d\rfloor+2)+\log_2(d!))$ bits. An advantage of this encoding over the more straightforward one (described in Section 4.1) which encodes the path from the root of the bintree to the simplex is that the vertices of a simplex can be directly computed without requiring a tree traversal [Heb94, AM07].

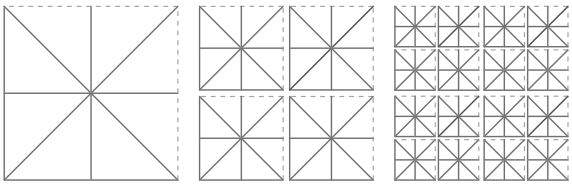
Alternatively, since diamond hierarchies are defined in terms of the subdivision of diamonds along their spines, the structure of the edges within a fully-subdivided cube determines all possible *types* of diamonds (up to scale).

Gregorski et al. [GDL*02] consider the oriented edge directions from the center of a three-dimensional $\mathcal{F}$ to one of its vertices to define 26 oriented diamond types within a hierarchy of diamonds. They introduce a pointerless encoding for the vertices, parents and children of each type of diamond in terms of scaled offsets from its central vertex. These offsets are precomputed and stored in a lookup table indexed by diamond orientation. A generalization of this scheme to dimension d would yield (3^d-1) oriented diamond types.

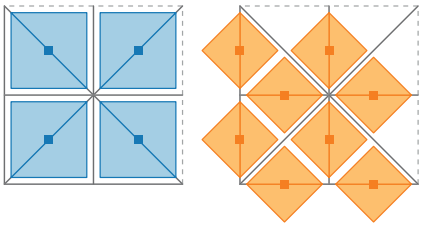
Weiss and De Floriani [WD08c, WD09b] consider each edge within a fully subdivided cube, which they call a *supercube*, to correspond to a distinct diamond type τ. To ensure that each edge in an RSB hierarchy is associated with only a single supercube, they adopt the *half-open interval* convention [Sam06] that internal edges of a supercube $\mathbf{s}$ as well as those on its lower boundaries are indexed by $\mathbf{s}$ while edges on its upper boundaries belong to a neighboring supercube. Figure 17a illustrates supercubes at three levels of resolution covering a two-dimensional domain, where solid lines indicate edges that belong to their containing supercube and dashed lines indicate edges belonging to a neighbor.

Based on this interpretation, they generalize the pointerless diamond encoding of [GDL*02] and their diamond decomposition [WD09a] to implicitly derive all geometric and hierarchical relationships of a diamond δ of arbitrary dimension directly from the binary representation of the coordinates $(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_d)$ of its central vertex $\mathbf{v}_c$. Consider a hierarchy of diamonds with maximum level of resolution N.

© The Eurographics Association 2010.



(a) Supercubes tile each level of resolution



(b) Diamonds in a 2D supercube

Figure 17: Supercubes are structured sets of edges that tile each level of a nested RSB mesh. (a) Three consecutive levels of resolution covering the same domain are shown. (b) A supercube in 2D contains four 0-diamonds (blue) and eight 1-diamonds (orange).

Let

$$\mathbf{v}_c = \begin{bmatrix} \mathbf{x}_1 = x_1^1\, x_1^2 \ldots x_1^n\ \ \tau_1^1\, \tau_1^2\ \ 00\ldots0 \\ \mathbf{x}_2 = x_2^1\, x_2^2 \ldots x_2^n\ \ \tau_2^1\, \tau_2^2\ \ 00\ldots0 \\ \vdots \\ \mathbf{x}_d = \underbrace{x_d^1\, x_d^2 \ldots x_d^n}_{\mathbf{s}}\ \ \underbrace{\tau_d^1\, \tau_d^2}_{\tau}\ \ \underbrace{00\ldots0}_{\gamma} \end{bmatrix}^T \tag{2}$$

be the binary representation of $\mathbf{v}_c$. This encoding depends on three quantities which can be efficiently extracted from the binary representation of the central vertex of a diamond through bit shifting operations: the *scale* γ, the *type* τ and the supercube origin $\boldsymbol{s}$ of δ.

Let $\textsc{Trailing}(\mathbf{x}_i)$ denote the number of trailing zeros in the binary representation of a coordinate $\mathbf{x}_i$ of $\mathbf{v}_c$. Then, the minimum of the number of trailing zeros among each of the d coordinates of $\mathbf{v}_c$ encodes the *scale* γ of δ, A diamond's level of resolution ℓ is related to its scale through the maximum level of resolution N as $\ell = N - \gamma$.

The two bits at position $\gamma+1$ and $\gamma+2$ of each coordinate $\mathbf{x}_i$, which we denote as τ_i^2 and τ_i^1, respectively, uniquely encode the *type* τ of δ. Since each vertex has d coordinates, and τ has two bits for each coordinate, there are $(2^2)^d = 4^d$ possible values for τ. However, the definition of γ precludes the 2^d cases where all bits of τ^2 are zero. Thus, there are $4^d - 2^d$ valid diamond types. Figure 17b shows the four 0-diamonds and eight 1-diamonds in a two-dimensional supercube.

The similarity *class* i of δ is encoded within τ as the number of zeros in position τ^2, e.g. an i-diamond has $(d-i)$ nonzero bits at this position. Furthermore, when considering the decomposition of δ into two hypercubes, the orientation of δ's Kuhn-subdivided component is determined by the $(d-i)$ nonzero bits in τ^2 while that of its fully-subdivided component is determined by the remaining i bits of τ^2.

The final $n = N - (\gamma+2)$ bits in each of the coordinates of $\mathbf{v}_c$ encode the *origin* of the supercube $\mathbf{s}$ containing δ, this corresponds to the lower left corner of its corresponding $\mathcal{F}$ and is related to the quadtree-like location code used in the simplex location code above.

5. Querying an RSB hierarchy

In this Section, we review queries for extracting nested RSB meshes from simplex and diamond hierarchies. Such queries are implemented by evaluating an application–dependent predicate μ, which we refer to as the *acceptance criterion*, at each node (i.e. simplex or diamond) to determine whether its should be refined or coarsened. Any cell that does not satisfy the predicate, must be refined.

These predicates can be defined locally between a node and its children, or hierarchically, by considering properties of its descendants. The acceptance criterion can be based on a combination of factors such as the node's location, e.g. for *region of interest* queries, its distance to an object of interest, such as the viewpoint, or the degree to which it approximates some underlying geometry. The latter case is referred to as the *approximation error* of the node. We discuss properties of error metrics in Section 5.1 and some notable application–specific acceptance criteria in Sections 6, 7 and 8.

Queries on a multiresolution model can be broadly classified into two categories [Mag99, DFM02];

Level of Detail (LOD) queries, which extract a mesh of minimum size covering the entire domain and satisfying a given acceptance criterion.

Spatial selection queries, which extract a mesh of minimum size covering only the portion of the domain interfering with the query and satisfying the acceptance criterion. This includes *point location* queries, as well as *windowing* and *range* queries.

Problems in computer graphics and visualization, such as rendering, are mostly concerned with queries of the former type, while those in finite element analysis and the field of GIS are concerned with the latter query type, where such queries are used, for example, to compute geometric overlays and in visibility queries [DM03]. Since most of the relevant literature is of the former type, the focus of our discussion will be on LOD queries. Interestingly, the set of simplices returned by spatial selection queries is not a subset of those satisfying LOD queries for the same acceptance criterion [Mag99].

The result of an LOD query is a *closed* set of refinements

© The Eurographics Association 2010.

necessary to extract a nested RSB mesh Σ, which we call the *current mesh*, from the RSB hierarchy. The status of the query is described by a cut of the hierarchy's dependency graph, called the *active front*, separating the refinements that have been applied from those that have not. Such methods can begin with a coarse approximation of the domain in which nodes of the dependency graph are refined in a *top-down* manner. Alternatively, the full resolution dataset can be coarsened in a *bottom-up* manner through a series of simplifications. *Incremental* methods apply a series of refinements and simplifications to a previously extracted mesh, and are thus interruptible [DWS*97, DL04].

For conforming refinements, all neighbors sharing bisection edge $\mathbf{e}$ of a simplex σ or diamond δ in an RSB mesh Σ must also have $\mathbf{e}$ as a bisection edge. Any such neighbor that is at a shallower depth of the hierarchy is forced to refine. Thus, conforming refinements can propagate up to the root(s) of the hierarchy. Alternatively, simplifications typically (although, not always [BLV03]) do not force deeper nodes to simplify. Thus, simplifications only occur when all simplexes created during a conforming bisection are present in Σ.

We distinguish between two forms of LOD queries in the literature:

Adaptive refinement queries, in which the extracted set of refinements is *closed* with respect to regular simplex bisection.

Selective refinement queries, in which the extracted set of refinements is *closed* with respect to diamond refinement.

Selective refinement queries correspond to a traversal of the dependency graph of a hierarchy of diamonds, which defines a partial order relation [DFM02]. Any set of diamonds which is closed with respect to the partial order defines a conforming mesh.

Adaptive refinement queries correspond to a traversal of the containment relation described by a hierarchy of simplexes. In general, this is not sufficient to guarantee that extracted meshes are conforming. Conforming refinements in a simplex-based representation can be accomplished implicitly, via a *saturated* error metric, or explicitly through a *neighbor-finding* operation.

Following a discussion of error metrics in Section 5.1, we review queries to a hierarchy of diamonds in Section 5.2. and to a hierarchy of simplexes in Section 5.3 We compare the two approaches in terms of their spatial and time complexity in Section 5.4.

5.1. Error evaluations

We now consider the error for a simplex σ or diamond δ with spine $\mathbf{e} := (\mathbf{v}_a, \mathbf{v}_b)$ in a nested RSB mesh Σ. Here we classify the choices in determining an error metric as *local*, when only the vertices of the spine are used, and as *total* when all samples within the domain of σ or δ are used. Additionally, a monotonic error metric is *saturated* if it ensures that the error of a node is greater than that of its descendants.

When scalar values are associated with the nodes of the hierarchy, the *approximation error* of a node (simplex or diamond) describes how well the current node approximates the scalar field. Since most applications of nested RSB meshes are defined on a scalar field, we provide examples of approximation error metrics that are local, total and saturated below. In such cases, there is a scalar value $F(\mathbf{v})$ associated with each vertex $\mathbf{v}$ in the domain. The value of a sample within a cell can be approximated by linear interpolation on the vertices of the cell, although other models are possible. Hierarchical error metrics are considered by [OR97, OR99, GRW00], and a comprehensive survey of LOD error metrics is given by [LRC*02].

Local error metrics. Since the only new vertex introduced during a refinement is the central vertex, which is located at the midpoint of the bisection edge, a simple approximation to the node's error is obtained by evaluating the error introduced into the mesh due to its omission. For example, when using linear interpolation, the local approximation error is

$$\varepsilon(\sigma) = |F(\mathbf{v}_c) - F_{\mathbf{e}}(\mathbf{v}_c)|, \tag{3}$$

where $F_{\mathbf{e}}(\mathbf{v}_c) = (F(\mathbf{v}_a) + F(\mathbf{v}_b))/2$ is the average of the values at the vertices of the bisection edge.

Local error metrics can be efficiently computed on the fly, and thus, do not require a preprocessing stage or storage space. They are independent of the dimension of the domain and of the primitive used (i.e. simplex or diamond).

Total error metrics. A metric with greater accuracy is based on all samples within the domain of the cell.

For example, when using linear interpolation, a common definition for the total error of a simplex σ is

$$\varepsilon(\sigma) = \max_{p \in \sigma}(|F(\mathbf{p}) - F_{\sigma}(\mathbf{p})|) \tag{4}$$

where $F_{\sigma}(\mathbf{p})$ is the value obtained through barycentric interpolation of the field values at the vertices of σ.

Diamond-based schemes typically utilize linear interpolation over the vertices of the diamond's simplices, and thus, the error of a diamond is the maximum of the errors due to its simplices [GDL*02, WD08c]

$$\varepsilon(\delta) = \max_{\sigma \in \delta}\{\varepsilon(\sigma)\} \tag{5}$$

An alternative could be to evaluate each sample's error with respect to the entire diamond, i.e. using barycentric interpolation over all vertices of the diamond.

Total error metrics often require an efficient enumeration scheme for the points within an RSB simplex or diamond. Marchesin et al. [MDM04] exploit the alignment of RSB tetrahedra to the coordinate axes to create an efficient enumeration algorithm for the points within a tetrahedron that

© The Eurographics Association 2010.

is similar to a raster scan-conversion algorithm. The algorithm first enumerates the axis aligned planar slices of a tetrahedron. Within each triangular slice, it enumerates all axis aligned lines. Finally, it enumerates the points within each line.

Hierarchical error metrics. An error metric is hierarchical, if the (hyper)-volume of the cell is taken into account. The scale of the cell can be used as a quick approximation of its volume.

For example, if the scale of a simplex or diamond is γ, then,

$$\varepsilon'(\sigma) = 2^{-\gamma} \cdot \varepsilon(\sigma) \tag{6}$$

is a possible hierarchical error metric.

Saturated error metrics. An monotonic error metric is saturated if the error of a cell is greater than that of its descendants, and all simplexes sharing the same bisection edge have the same error.

A possible saturated error metric for a cell σ is

$$\varepsilon'(\sigma) = \max\left\{\varepsilon(\sigma), \max_{\sigma_0 \in \text{CHILDREN}(\sigma)} \{\varepsilon'(\sigma_0)\} + \varepsilon_0\right\} \tag{7}$$

where $\varepsilon(\sigma)$ is a local error metric. The ε_0 term ensures that the error is strictly greater than any of its descendants.

Saturated error metrics typically require a bottom-up evaluation either while the metric is being evaluated or as a post-process.

5.2. Querying a hierarchy of diamonds

In this subsection, we discuss algorithms for performing selective refinement on a hierarchy of diamonds. Such algorithms are based on traversals of the DAG defining the hierarchy's dependency graph.

By definition, each simplex in a diamond is generated during the refinement of one of its parents. Thus, a diamond δ in an RSB mesh Σ will contain all of its simplexes only after all of its parents have refined. We refer to a diamond that contains all of its simplexes as *complete*.

Since conforming refinements to Σ require all simplexes in a refining diamond δ to bisect at the same time, δ must be complete before it can be refined. This completion process is carried out by (recursively) forcing all parents of δ to refine, thereby satisfying the *transitive closure* of the dependency graph.

In Algorithm 1, we outline a top-down selective refinement query for a hierarchy of diamonds, which is initialized using the root diamond of the hierarchy. Most diamonds are checked against the acceptance criterion μ. However, forced refinements short-circuit the acceptance criterion using the boolean `ForceRefine` (Line 1).

Algorithm 1 SELECTIVEREFINE$_\delta(\delta, \texttt{ForceRefine})$

Require: δ is a diamond in a nested RSB mesh Σ
Require: `ForceRefine` is a boolean
Require: μ is an acceptance criterion
1: **if** `ForceRefine` is true **or** $\mu(\delta)$ fails **then**
2: // *Ensure diamond is complete*
3: **for all** $\delta_p \in$ PARENTS(δ) **do**
4: **if** δ_p is not refined **then**
5: SELECTIVEREFINE$_\delta(\delta_p, \textbf{true})$
6: // *Bisect all simplexes of* δ
7: REFINEDIAMOND(δ)
8: // *Check all children*
9: **if** `ForceRefine` is false **then**
10: **for all** $\delta_c \in$ CHILDREN(δ) **do**
11: SELECTIVEREFINE$_\delta(\delta_c, \textbf{false})$

Refinement is carried out in REFINEDIAMOND (Line 7). Since Σ is a diamond-based RSB mesh, this can be efficiently achieved by marking δ as refined, inserting all children of δ in Σ, and indicating in each child that the simplexes due to δ belong to Σ. Since the $O(d)$ parents in a diamond δ generate its $O(d!)$ simplexes, all simplexes due to a parent δ_p can be tracked using a single bit. This operation is $O(d)$ when the class of *delta* is not $(d-1)$, and $O(2^d)$, otherwise.

Finally, the children of diamonds that are not forcibly refined are added (Lines 8–11).

Since a diamond δ has $O(d)$ parents, and each diamond is refined only once, diamond refinements in SELECTIVEREFINE require an amortized $O(d)$ time and spatial accesses. Additionally, when d is reasonably low, the status of each parent's subdivision can be cached (using at most $2 \cdot d$ bits), which can reduce the number of required spatial accesses [WD09a].

Selective refinement can converge faster through the use of a priority queue to order the refinements according to the approximation error. Gregorski et al. [GDL*02] describe an incremental selective refinement query for 3D diamonds based on a split and merge queue [DWS*97].

5.3. Querying a hierarchy of simplexes

Here, we discuss the two approaches for extracting nested RSB meshes from a simplex hierarchy.

Adaptive refinement corresponds to a traversal of the bintrees describing the containment relation among simplexes in a hierarchy of simplexes (Algorithm 2), and is initialized with the $d!$ bintree roots. When a saturated error metric is employed, the extracted mesh is implicitly guaranteed to be conforming [OR97, GRW00]. Otherwise, the extracted mesh is not likely to be conforming.

The second approach uses a selective refinement algo-

© The Eurographics Association 2010.

Algorithm 2 AdaptiveRefine(σ)

Require: σ is an RSB simplex in a nested RSB mesh Σ
Require: μ is an acceptance criterion
1: **if** $\mu(\sigma)$ fails **then**
2: BisectSimplex(σ)
3: AdaptiveRefine(Child$_0(\sigma)$)
4: AdaptiveRefine(Child$_1(\sigma)$)

rithm on the hierarchy of simplexes, and ensures conforming refinements by refining all bisection edge neighbors concurrently. This is carried out through a series of *neighbor-finding* operations that traverse all simplexes in an RSB mesh that are incident to the bisection edge of a refining simplex.

A useful property of conforming RSB meshes is that neighboring simplices can differ in depths by at most one [EKT01]. Specifically, the neighbors of a simplex σ in a conforming RSB mesh Σ sharing the bisection edge of σ can either be at the same depth, in which case, they belong to the same diamond, or their depth can be shallower by one bisection. Alternatively, neighbors not sharing a bisection edge of σ are at the same depth, or are deeper by one bisection.

If the depth of a neighboring simplex σ' along the bisection edge $\mathbf{e}$ is less than that of σ, σ' must be recursively bisected until the same-depth neighbor of σ along $\mathbf{e}$ belongs to Σ. This corresponds to the transitive closure of the diamond dependency relation used in diamond-based selective refinement (Algorithm 1) and is guaranteed to terminate (possibly at the root of a bintree).

We outline a top-down simplex-based selective refinement algorithm in Algorithm 3, which is initialized with the $d!$ bintree roots. As in Algorithm 1, forced refinements are carried out through the boolean `ForceRefine`. The neighbor–finding operation is encapsulated by the BisectionNeighbors function in Line 3.

Pointer-based selective refinement algorithms for a hierarchy of triangles, that are top-down and incremental can be found in [LRC*02] and [DWS*97], respectively. However, explicit encodings require up to $(2*d+5)$ pointers for each simplex since they must encode pointers to the $(d+1)$ vertices, and to the $(d+1)$ neighboring simplices as well as two pointers to the children simplices and one pointer to the parent simplex. Due to the regularity of the decomposition rules, efficient pointerless neighbor-finding schemes based on location codes have been developed. We refer to location codes based on tree traversal (see Section 4.1) as *bintree location codes*, and those based on the index within a fully-subdivided cube (see Section 4.4) as *subtree location codes*. An incremental selective refinement algorithm based on neighbor–finding is presented in [DL04].

Maubach extends his original pointer-based algorithm [Mau95] with an algebraic pointerless neighbor-

Algorithm 3 SelectiveRefine$_\sigma$(σ, `ForceRefine`)

Require: σ is an RSB simplex in a nested RSB mesh Σ
Require: `ForceRefine` is a boolean
Require: μ is an acceptance criterion
1: **if** `ForceRefine` is true **or** $\mu(\sigma)$ fails **then**
2: // *Ensure all neighbors of* σ *are in* Σ
3: **for all** $\sigma' \in$ BisectionNeighbors(σ) **do**
4: **if** $\sigma' \notin \Sigma$ **then**
5: SelectiveRefine$_\sigma$(Parent(σ'), **true**)
6: BisectSimplex(σ')
7: // *Apply RSB operation to* σ
8: BisectSimplex(σ)
9: // *Check both children*
10: **if** `ForceRefine` is false **then**
11: SelectiveRefine$_\sigma$(Child$_0(\sigma')$, **false**)
12: SelectiveRefine$_\sigma$(Child$_1(\sigma')$, **false**)

finding algorithm in arbitrary dimension that runs in time proportional to the depth of the simplex [Mau96].

Hebert [Heb94] introduces a symbolic neighbor-finding operation for tetrahedra based on subtree location codes. In this method, codes for the three neighbors within the same subtree can be found in constant time, but the neighbor outside this cube can be found in time proportional to the nodes depth.

A pointerless constant time neighbor finding algorithm for hierarchies of triangles based on bintree location codes is presented by Evans et al. [EKT01]. In this scheme, each neighboring triangle's location code can be found in constant time through hardware bit manipulation operations.

Lee et al. present a constant time neighbor-finding algorithm based on bintree location codes for hierarchies of tetrahedra [LDS01] and for hierarchies of pentatopes [LDS04]. The algorithm depends on updating a *neighbor bitmask* to determine which neighbors share the bisection edge, as well as a lookup table to determine the labels of the children simplices. These algorithms are specific to 3D and 4D, respectively.

Atalay et al. [AM04, AM07] combine the symbolic approach of Hebert [Heb94] with the efficient hardware-based neighbor-finding approach of Lee et al. [LDS01, LDS04] to simplex hierarchies of arbitrary dimension using a subtree location code. In this scheme, each neighbor–finding operation requires $O(\log d)$ time.

5.4. Comparison

Since a d-dimensional diamond has $O(d!)$ simplexes, and neighbor-finding must be applied to all simplexes before each refinement, each conforming refinement to a hierarchy of simplexes requires $O(d!)$ time. More importantly, since

© The Eurographics Association 2010.

it is not practical to cache the status of neighbor refinements, $O(d!)$ spatial accesses to the pointerless data structure are required. In contrast, conforming refinements to diamond-based meshes require $O(d)$ time and spatial accesses [WD09a].

Saturated error metrics can simplify the extraction algorithm, since they guarantee conforming meshes without requiring refinements to propagate to shallower depths. Since there are no dependencies in the adaptive refinement algorithm, conforming meshes can be extracted in parallel through the use of a saturated error metric [GR99, BPSC04, CGG*04, Mau05].

De Floriani and Lee [DL04, Lee06] compare meshes extracted using error saturation to those extracted using neighbor finding, in the case where a per-simplex error is defined. They determine that smaller meshes can be extracted using a neighbor-finding approach (approximately 5% smaller in 3D [DMPS02] and 1% smaller in 4D [Lee06]), in the same time as a saturated approach. However, their hierarchies have a higher storage overhead due to the use of a per-simplex error (i.e. there is a 6x overhead in 3D and a 24x overhead in 4D).

6. Applications in two dimensions

Much of the early research on efficient representations and queries for nested RSB meshes has been for interactive rendering of terrain datasets [LKR*96, DWS*97, LP02]. Recent research has shifted towards batched refinements [HDJ05, GMC*06] to make better use of the graphics hardware and towards efficient representations for encoding sparse subsets of a regular grid [Ger03a, WD08c].

We review applications of triangle hierarchies to terrain in Section 6.1 and other applications in Section 6.2.

6.1. Interactive terrain rendering

Terrains are often presented as a height field where an elevation is associated with each 2D point on a regular grid. Since such datasets can be quite large, with many datasets containing billions of samples [CGG*03b], there has been a lot of research on optimizing interactive terrain rendering based on nested RSB meshes

In particular, since the viewpoint is typically located above the terrain, many samples at a great distance are visible in the viewframe. Thus, view-dependent queries are important for achieving interactivity.

In this Subsection, we briefly review some of the issues in interactive terrain rendering and how they are resolved through the use of nested RSB meshes. Early approaches to interactive terrain rendering using nested RSB meshes are reviewed in detail in [LRC*02]. The recent survey of Pajarola and Gobbetti [PG07] presents a comprehensive overview of approaches for interactive rendering of regularly sampled terrain datasets, including those based on RSB hierarchies.

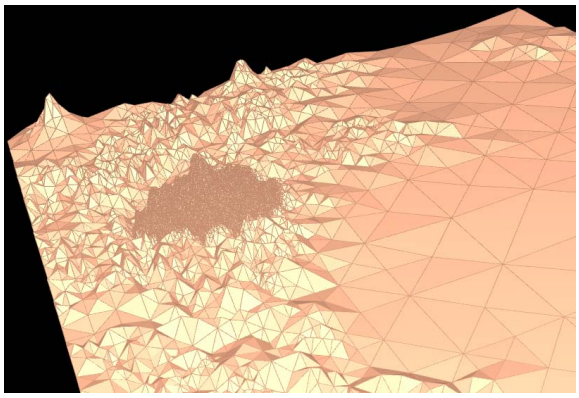


Figure 18: An RSB mesh extracted from the 1025^2 sample Puget Sound dataset using a circular ROI query with an error $\varepsilon \leq 10\%$ outside the circle and no error within the circle.

6.1.1. Hierarchy representation

Due to the regularity of the datasets, most approaches encode the dataset's elevations as a linear array of $(2^N+1)^2$ samples. This array is typically indexed implicitly e.g. using row-major ordering. For extremely large datasets, the samples can be stored out-of-core in square tiles [LKR*96, Paj98], or using hierarchical space-filling curves with on-demand paging [LP02]. The latter have been shown to perform an order of magnitude faster than row-major indexing, which performs an order of magnitude faster than tile-based approaches [LP02].

Since one is often interested in approximating the full dataset using fewer triangles, the approximation error due to each node is often precomputed and associated with each triangle or vertex of the hierarchy. In the former case, such errors can be encoded as a pair of binary trees and accessed using a triangle's location code [EKT01]. This approach can guarantee that extracted meshes are within a given tolerance, but requires approximately four times as much storage as a vertex-based error [EKT01]. Consequently, most simplex-based approaches utilize vertex-based errors (e.g. [DWS*97, Ger03a]) as do all diamond-based approaches (e.g. [LKR*96, LP02, HDJ05, WD08c]). The errors are typically stored in an array and indexed by the vertex coordinates.

Efficient representations have also been proposed for *incomplete* terrain datasets built on a sparse subset of a *full* $(2^N+1)^2$ grid, where it is no longer efficient to encode the elevation and error values using an array. This can be useful in cases where the field is oversampled or if values for portions of the domain are unavailable.

Although both simplex-based and diamond-based approaches that represent the dependency relation explicitly,

© The Eurographics Association 2010.

e.g. with pointers [CGG*03a, HDJ05], can be modified to handle an incomplete hierarchy, there is a great deal of spatial and hierarchical coherence among the samples that can be exploited to yield significantly more effient representations.

Gerstner's triangle-based approach [Ger03a], encodes two binary trees as linear arrays. To account for the missing data of pruned branches, each node encodes a variable sized-pointer indicating the number of bytes to skip if the node is not refined. This imposes an amortized overhead of three bytes per retained node, but does not enable efficient random access to the dataset's elevation values.

In contrast, Weiss and De Floriani's diamond-based approach [WD08c] utilizes a two-level approach, where samples are clustered into supercubes (Section 4.4), while the supercubes at each level of resolution are indexed using a hash table. Due to the coherence among the retained samples in typical terrain datasets, their encoding requires less than one byte of overhead per sample and provides efficient random access to the samples. Both approaches support the embedding of higher resolution regions of data into an existing hierarchy.

6.1.2. Querying a terrain

The definition of an appropriate acceptance criterion is critical for achieving interactivity during runtime. Such metrics have often involved view-dependent terms involving the screen space or distance-based error.

Many of the earlier approaches are based on a local error metric (see Equation 3) [SS92, LKR*96, EKT01], By generating a monotonic error, these simple evaluations are able to maintain global (but not necessarily tight) bounds on the error [DWS*97, Paj98, LP02, Ger03a].

Blow [Blo00] proposes a nested hierarchy of spheres defined in the terrain's object space rather than its projected screen space. Since this metric relates a triangle's position to the viewpoint, only the triangles on the isosurface defined by these nested spheres need to be checked for visibility, enabling more efficient view-dependent extraction queries. A drawback of this approach is that the bounding sphere hierarchy is independent of the RSB hierarchy, and performance depends on the quality of the bounding volume generated during a preprocess.

Lindstrom and Pascucci [LP02] combine the two hierarchies by using the spine vertices of a diamond to define the nested hierarchy of spheres. The view-dependent error metric is then defined in terms of (a) a saturated local approximation error, (b) the distance to the viewpoint, and (c) the sphere radius.

Gerstner [Ger03b] uses a diamond's octagonal merging domain to define an optimally-tight nested hierarchy of thickened octagonal prisms. He also shows how independent saturated error metrics can be combined at runtime, which enables dynamic modifications to the acceptance criterion.

Duchaineau et al. [DWS*97] introduce the first incremental selective refinement query for a hierarchy of triangles and note that any nested RSB mesh can be obtained from any other one via a series of merges and splits. Their query maintains separate priority queues for merging and splitting triangles. Although incremental selective refinement can be difficult to implement, it exploits frame-to-frame coherence by initializing the next frame from the previously extracted frame

6.1.3. Clustered updates

Recent work on interactive terrain rendering has focused on reducing the CPU and memory transfer bottlenecks associated with fine-grained refinements through the use of coarse-grained clustered updates. This is motivated by the observation that current graphics hardware permits pixel-sized triangles to be rendered interactively [HDJ05].

In RUSTiC [Pom00] and CABTT [Lev02], each update corresponds to the insertion of triangles from several bintree depths lower in the hierarchy. Despite increasing the number of triangles required to satisfy a given error, these methods reduce the processing overhead.

The Batched Dynamic Adaptive Meshing (BDAM) method of Cignoni et al. [CGG*03a, CGG*03b] follows up on the clustering idea of RUSTiC and CABTT by encoding a set of irregular updates with each bintree triangle. Thus, they use the hierarchy as an efficient spatial access structure to irregular triangulation data. This enables efficient transmission of terrain data to the GPU in batches. A corresponding octree-based texture hierarchy, enables colors to be mapped onto the triangles, and thus generates more realistic images using fewer triangles.

Hwa et al. [HDJ05] combine the spatial access of BDAM, with the regular clustering of [Pom00]. Furthermore, their textures are encoded as a diamond hierarchy, thus following more closely with the hierarchy defined by the height values. Interestingly, although they propose an implicit indexing scheme for the vertices, parents and children of a diamond, they report better empirical performance using an explicit pointer-based representation for diamonds.

Gobbetti et al. [GMC*06] compress diamond-based clustered updates using a wavelet lifting scheme to achieve an adaptive RSB mesh with the same compression rates achieved by nested regular grid approaches [LH04].

6.2. Other applications

Another popular application of 2D RSB hierarchies has been for encoding images. For example, a wavelet-based image coding based on a hierarchy of triangles is presented by Hebert and Kim [Heb98] (based on [HK95]).

© The Eurographics Association 2010.

Von Herzen [VH89] uses a hierarchy of triangles to convert colored texture maps into polygonal meshes. The subdivision in this approach is driven by the difference in vertex color values. After extracting a conforming RSB mesh, triangles of constant shading are merged to create a polygonal mesh with C_0 connectivity.

Tanaka [Tan95] generates adaptive decompositions of colored images and range-scanned shapes defined over a rectangular or cylindrical parameter domain. This method utilizes a diamond's octagonal merging domain (which are called the *region of reference* to a diamond's *region of influence*) to define a curvature-driven, parallel refinement algorithm over a coarse grid of Kuhn subdivided squares.

A novel use of a hierarchy of diamonds is proposed by Weber et al. [WBP07], in which a nested RSB mesh is used as an embedding space for a graph representing scalar field morphology. This provides a visualization of properties of the graph, such as the persistence of critical points [ELZ02], using well-understood metaphors from terrain analysis to aid in the user's understanding of the field.

7. Applications in three dimensions

In this Section, we review approaches for nested RSB meshes in 3D with a focus on those methods that have been used to model multiresolution volume datasets (Section 7.1) and to enable efficient extraction of variable resolution isosurfaces (Section 7.2). We also discuss the use of nested RSB meshes as a spatial access structure and other applications (Section 7.3).

7.1. Volume visualization

The structure of a scalar field is often understood by analyzing its surfaces of constant field value, known as *isosurfaces* or by the subvolumes enclosed between two such surfaces, known as *interval volumes* [FMS95, Guo95]. However, due to the increasing size of volumetric datasets, it is difficult to interactively analyze such extracted meshes at full resolution.

RSB hierarchies have been used as the basis for a multiresolution model of a volume dataset, to accelerate isosurface extraction [ZCK97, GR99, GDL*02, Pas04], direct volume rendering [MDM04] and the analysis of field morphology [TTNF04]. In the former case, conforming updates to the scalar field model along with an appropriate set of triangulation cases within each tetrahedron guarantee that extracted isosurfaces are free of self-intersections.

As with terrain datasets, volume datasets are typically defined over a regularly sampled domain of resolution $(2^N + 1)^3$, and queries are accelerated through the use of a precomputed error metric. Isosurface extraction queries can be accelerated by storing the range of field values contained in the domain of each cell [WVG92, GDL*02]. This enables culling the portions of the domain that do not intersect the desired isosurface.

Zhou et al. [ZCK97] first proposed the use of nested tetrahedral RSB meshes to extract simplified conforming isosurfaces from a volume dataset. An interesting feature of their approach is that they incorporate a topological criterion into their acceptance criterion to ensure that the topology of the simplified surface matches that of the surface at full resolution. This is accomplished by disallowing fusion of tetrahedra whose bisection edge vertices lie on the same side of the isosurface, but whose central vertex lies on the opposite side.

Gerstner and Pajarola [GP00] note that the topology preserving approach of Zhou et al. [ZCK97] is too conservative. Although their proposed solution is based on a hierarchy of tetrahedra, they consider changes to an isosurface's topology at the diamond level. To do so, they define isosurface cases within each diamond, based on the relative values of a diamond's vertices and the given isovalue. Their saturated topology-based error metric encodes the range of isovalues in which the topology of the extracted mesh changes.

However, topology preservation limits the adaptability of the approach since all nodes in which the topology changes must be present in all extracted meshes. Consequently, surfaces with complicated topology can never be simplified beyond a certain point. To resolve this, they introduce a *topology control* mechanism which weights the topology metric at each node. They use this metric to clean up a noisy isosurface by reducing the genus of the extracted mesh.

Gerstner and Rumpf [GR99] accelerate the extraction of isosurfaces from a hierarchy of tetrahedra using a parallel adaptive refinement algorithm. Additionally, they introduce a mechanism to cull the isosurface's backfaces through the use of a saturated view-dependent curvature-based error criterion, reducing the size of extracted isosurfaces by a factor of two. They note that the gradient of a vertex is necessary for smooth shading, but requires three times as much space to encode as the scalar values. Thus, rather than encoding the gradient, they compute it on the fly and cache the values in a hash table. In follow-up work [Ger02], Gerstner describes a hierarchical scheme to compute the gradient of a tetrahedron from that of its parent. When used in conjunction with a sorting scheme for tetrahedra based on the bisection splitting plane, this enables back-to-front isosurface extraction, which can be used to extract multiple transparent isosurfaces during a single traversal of the hierarchy.

Pascucci [Pas04] introduces a hardware accelerated top-down view-dependent isosurface extraction approach and provides details of his marching tetrahedra [PT90] vertex program. He introduces a tetrahedral stripping operation, which generalizes the 2D Sierpinski space-filling curve used for triangle stripping on terrain [Paj98, LP02].

Gregorski et al.'s diamond-based scheme utilizes ROAM's [DWS*97] incremental selective refinement

© The Eurographics Association 2010.

algorithm to exploit frame-to-frame coherence between extracted meshes during view-dependent isosurface extraction, and during small adjustments to the isovalue. They compress the scalar values, field gradient and min-max ranges within each diamond from 19 bytes to 4 bytes. Additionally, they avoid dealing with boundary cases by operating on a toroidal domain of size $(2^N)^3$. Finally, they rearrange the data hierarchically [PF01] and use the operating system's virtual memory paging for cache-coherent out-of-core memory management [LP02].

Recently, Gregorski et al. [GSDJ09] proposed a method to further accelerate view-dependent isosurface extraction through the use of hardware compression and occlusion culling. They use the mesh extracted during the previous frame to estimate the occluded regions of the dataset.

Linsen et al. [LGP*04] introduce the $\sqrt[3]{2}$ subdivision as a volumetric subdivision basis for meshes with diamond connectivity. They use trilinear B-spine wavelets to downsample the dataset, which generates higher quality images compared to subsampled datasets. In an adaptive setting [LHJ07] this generates approximations equivalent to those generated on subsampled datasets using 10-15% fewer tetrahedra.

Marchesin et al. [MDM04] use a hierarchy of tetrahedra for view-dependent Direct Volume Rendering (DVR). They examine the implications of applying non-conforming bisections, which can extract meshes with fewer elements in less time, but can introduce a significant amount of noise into the visualization. In some cases through, the resultant image is determined to be of sufficient quality.

Weiss and De Floriani [WD09b] extend their supercube representation [WD08c] to associate information with coherent subsets of vertices, edges, tetrahedra and diamonds of a 3D hierarchy of diamonds. They suggest the use of supercube-based diamond hierarchies when the number of retained samples is *sparse* with respect to the the complete dataset while the average clustering *concentration* of each supercube is high. Empirically, they found many common datasets to be oversampled by a factor of three or more with respect to a total approximation error of Section 5.1). They also propose the use of supercubes to retain only the set of diamonds required to extract one or more isosurfaces. Due to the spatial and hierarchical coherence, such representations can achieve a 20-100 fold savings compared to the original dataset, while still enabling general selective refinement queries. However, due to the reduced sampling, the quality of isosurfaces away from the retained isovalues is significantly reduced.

7.2. Multiresolution isosurfaces

The previous schemes operate by extracting a nested RSB mesh Σ from an RSB hierarchy that satisfies some extraction criterion and then extracting an isosurface from Σ. An alternate approach, when a desired isosurface can be determined ahead of time is to extract a multiresolution model for a specific isosurface from the hierarchy. This model can then be queried offline. Since such methods encode only a single isosurface, they cannot dynamically change the isovalue, but can require significantly less space than the original scalar field. When the modifications follow the diamond subdivisions, they ensure that the extracted surface is manifold and free of self-intersections [PB00].

Pascucci and Bajaj [PB00] propose a multiresolution model whose modifications are based on a small set of local update primitives to the extracted mesh, corresponding to the change in isosurface after each refinement. Once an isosurface is extracted, a set of subdivision masks enable the mesh to be smoothed. They do not, however, describe an encoding for their proposed multiresolution model, nor do they describe how this model can be queried.

Borgo et al. [BPSC04] describe a top-down progressive isosurface extraction method where the isosurface for each successive depth of a hierarchy of diamonds is extracted from the previous one. They establish an explicit correspondence between the pairs of tetrahedra in successive depths of the hierarchy to pass isovertices (i.e. isosurface vertices) from one depth to the next. This scheme has an overhead of 70 bytes per encoded diamond and achieves a reported 3x speedup in isosurface extraction on datasets of resolution 65^3 and 129^3.

Lewiner et al. [LVLM04] use the spatial decomposition of a hierarchy of tetrahedra to compress and progressively encode extracted isosurfaces. They encode the relative sign value of each RSB vertex in the desired isosurface's *tubular neighborhood*, that is, the set of tetrahedra intersected by the isosurface. Since adjacent tetrahedra can differ by at most a single bintree depth, they track the depth changes using two bits per tetrahedron (although a single bit would suffice [EKT01]). This method provides a total order to the modifications, rather than a partial order, thus limiting the extraction capabilities of the technique [LRC*02].

Weiss and De Floriani [WD08b] introduce a multiresolution model for interval volumes extracted from a hierarchy of diamonds, where refinements to an irregular interval volume are encoded in terms of refinements to the diamonds that contain them. Their implicit scheme encodes refinements corresponding to diamonds intersecting the interval volume as well as all their hierarchical ancestors, and requires 14 bytes per refinement. The non-intersected ancestors are used as a spatial access structure to the intersected nodes, and aid in the extraction of conforming interval volume meshes.

Recently, this technique has been generalized into an *Isodiamond Hierarchy* framework [WDF10], which encompasses multiresolution isosurfaces as well as interval volumes. They introduce a new *Minimal Isodiamond Hierarchy* that enables the extraction of conforming isosurfaces and interval volumes without encoding the (non-intersected) hier-

© The Eurographics Association 2010.

archical ancestor diamonds. By analyzing the hierarchical connectivity of the embedded mesh, they ensure that the extracted isosurfaces or interval volumes are conforming even though the corresponding RSB mesh might not be conforming or cover the cubic domain.

7.3. Other applications

Although volume visualization has been the predominant application of nested tetrahedral RSB meshes, other researchers have focused on their use as an adaptive partitioning of a spatial domain and as a spatial indexing structure.

Roxborough and Nielson [RN00] utilize a nested tetrahedral RSB mesh as a spatial indexing structure to reconstruct a variable resolution scalar field from a collection of intensity samples. Each sample is represented by its barycentric coordinates within its containing tetrahedron, which simplifies redistribution of the samples after a tetrahedral bisection. They generate intensity values at the vertices of the mesh by minimizing a trivariate scattered data approximation functional and use a least-squares error metric to guide the refinement. To avoid excessive computation, they utilize a heuristic of checking only the 5% of tetrahedra with the worst error. They apply this method to generate scalar fields from freehand ultrasound images but suggest its application to other scattered data problems as well.

A similar approach is used by Mello et al. [MVT03] on the surface reconstruction problem, where a densely sampled surface is reconstructed from a 3D point cloud by first computing the *In/Out* function of the vertices of a nested RSB mesh with respect to the samples. That is, they determine on which side of the surface each vertex of the RSB mesh lies. Their algorithms applies principle component analysis to the samples within each tetrahedron to determine the best fitting oriented plane for the points, in the least squares sense. Tetrahedra with a poor plane–fitting error or with too many points are bisected to yield. The signed distance of each vertex of the RSB mesh can then be determined by its distance from the fitting planes of its tetrahedra. Since this method yields adaptive conforming spatial decompositions, Mello et al. suggest this scheme as an effective representation for adaptive distance fields [FPRJ00].

Kimura et al. [KTY*04] consider parallel segmentation of a volume dataset, based on their earlier work [TTW03]. They introduce a *split-and-merge* algorithm that evaluates a *homogeneity criterion* on each tetrahedron. Nodes that fail this test are bisected and processed separately. The results are then combined and passed back up the hierarchy.

The adaptive tetrapuzzles approach of Cignoni et al. [CGG*04] generalizes the two dimensional BDAM [CGG*03a]. It uses a hierarchy of tetrahedra as a spatial index over a set of triangles in an irregular triangle mesh for efficient out of core construction and rendering of extremely large meshes. During construction, they build an adaptive nested RSB mesh Σ in a top-down manner by distributing the triangles of the mesh T into the leaf nodes of Σ. The multiresolution triangle mesh is generated in parallel, in a bottom-up manner, following the parent-child dependency relation of the hierarchy of tetrahedra. Thus, each simplification step corresponds to a fusion operation on the pair of tetrahedra covering them. The triangles associated with each tetrahedron are then converted to a triangle strip for efficient batched updates at runtime. During rendering, updates to the underlying triangle mesh T follow conforming refinements to Σ and are thus conforming as well. A saturated view-dependent error metric enables efficient adaptive refinement queries.

Gerstner et al. [GMC*02] present a hybrid visualization system that overlays volumetric weather data representing atmospheric rainfall on a terrain dataset to aid in the spatial analysis of the data. They treat these two hierarchies separately, and render the rainfall data as a series of transparent isosurfaces [Ger02] on top of the terrain data. The error metrics for the two hierarchies are combined by a single user-controlled global error metric.

8. Applications in higher dimensions

In this Section, we review approaches to nested RSB meshes in 4D and higher. One of the original motivations for nested RSB meshes was to compute fixed points of functions in a high dimensional parameter space [Tod76]. Such fixed points correspond to point location queries and typically do not decompose the entire domain.

More recently, there has been increased interest in adaptive decompositions of time-varying volumetric datasets. The temporal dimension of such datasets can be treated as a set of values in the same 3D location [GSDJ04], or as a fourth spatial dimension [LPD*04, LDS04]. A recent survey of approaches to time-varying volume datasets can be found in [WD08a].

Gregorski et al. [GSDJ04], apply their diamond-based isosurface extraction framework [GDL*02] to time-varying volumetric datasets modeled as a stack of volumetric datasets covering the same volumetric domain. They exploit the temporal coherence of the dataset by initializing the isosurface extraction for each new time-step with the RSB mesh extracted during the previous time-step. Additionally, they propose a clustered approach for subtrees of tetrahedra within each diamond to reduce the granularity of each modification and to accelerate hardware rendering of isosurfaces (similar to RUSTiC [Pom00] and related approaches in 2D, see Section 6.1.3).

Linsen et al. generalize their $\sqrt[3]{2}$ approach [LGP*04] to 4D grids with the $\sqrt[n]{2}$ scheme [LPD*04]. This enables treating the temporal dimension in time-varying volume data as a spatial dimension. Volume rendering techniques are used to visualize the extracted hypervolume.

© The Eurographics Association 2010.

Lee [Lee06] also treats the temporal dimension as a fourth spatial dimension, where adaptive pentatopic RSB meshes are extracted from a pointerless hierarchy of pentatopes encoded using bintree location codes [LDS04] (see Section 4.1).

Atalay and Mount [AM07] use a hierarchy of pentatopes as a point-location structure to accelerate ray tracing of atmospheric effects. In this scheme, each ray is represented as a 4D point, and values for unrepresented points are interpolated based on the RSB decomposition. Compared to a non-adaptive approach, the hierarchy of pentatopes achieves a 6x savings in time. Further optimizations were achieved by applying a lazy neighbor-finding algorithm to patch cracks locally on demand. Thus, a non-conforming RSB mesh can be extracted from the hierarchy of pentatopes, and local regions can be made conforming on-demand. This yields a further 3x savings in time and 9.3x in space.

9. Concluding remarks

The popularity of RSB-based decompositions stems from the prevalence of datasets defined on regularly sampled grids and the relative ease in which RSB enables adaptive conforming domain decompositions to be extracted. Compared to models based on regular refinement over a regularly sampled domain, and to (triangulated) restricted quadtrees, RSB hierarchies are more adaptive, generate only a single family of similarity classes of primitives, and produce smaller extracted meshes. Furthermore, RSB hierarchies admit efficient pointerless representations for simplex-based [Lee06, AM07] as well as diamond-based [GDL*02, WD09a] representations.

In this STAR, we have reviewed hierarchical models based on the regular simplex bisection scheme in a dimension–independent manner and presented details of dimension–specific applications in two, three and higher dimensions. We have discussed the primary distinction among such approaches, i.e. whether they treat individual RSB simplexes or diamonds composed of RSB simplex clusters as the modeling primitive, and how this choice leads to different hierarchical dependency relations, and to different query algorithms.

Diamond-based representations are centered around conforming refinements. As such, most applications that require conforming domain decompositions should benefit from diamond-based representations. For higher dimensional applications, the complexity of selective refinement algorithms is reduced from $O(d!)$ to $O(d)$.

Furthermore, since $O(d!)$ simplexes within a diamond are generated concurrently during the refinement of each parent, diamond-based RSB meshes require significantly less space than their corresponding simplex-based meshes. As an example, in a three dimensional context (i.e. where $d! = 6$), simplex-based RSB meshes were found [WD09a] to have an average of 3.75 simplexes per diamond.

In contrast, since simplex-based representations enable the extraction on non-conforming nested RSB meshes, they have a higher expressive power than diamond-based representations. In particular, they should be more efficient in applications where conforming meshes are not required [MDM04], or where a containment hierarchy would be preferable, such as in spatial selection queries. Furthermore, in applications that utilize a saturated error metric, the adaptive refinement algorithm (Algorithm 2) admits parallel extraction of conforming meshes.

The on-demand conforming refinement algorithm of [AM04] is an interesting compromise in that it enables locally conforming regions to be extracted from nonconforming RSB meshes only when they are needed, thus reducing the overhead and computation times.

An interesting question relates to the overhead introduced by converting a nonconforming RSB mesh into a conforming one. For triangle meshes, Atalay and Mount [AM06] found that conforming meshes are at most fourteen times larger than non-conforming meshes. For higher dimensional domains, they found a conservative upper bound of $3^d \cdot d! \cdot 2^d - 1$, but a tight bound remains an open question. In practice, they found conforming two-, three- and four-dimensional meshes to be 5, 31, and 228 times larger than their non-conforming counterparts.

Acknowledgments

We would like to thank the anonymous reviewers for their many helpful suggestions. This work has been partially supported by the National Science Foundation under grant CCF-0541032.

References

[AG79] ALLGOWER E., GEORG K.: Generation of triangulations by reflection. *Utilitas Mathematica 16* (1979), 123–129. 5

[Ale30] ALEXANDER J.: The combinatorial theory of complexes. *The Annals of Mathematics 31*, 2 (1930), 292–320. 7

[AM04] ATALAY F., MOUNT D.: Pointerless implementation of hierarchical simplicial meshes and efficient neighbor finding in arbitrary dimensions. In *Proc. 13th International Meshing Roundtable* (2004), pp. 15–26. 2, 10, 14, 20

[AM06] ATALAY F., MOUNT D.: The cost of compatible refinement of simplex decomposition trees. In *Proc. International Meshing Roundtable* (2006), Springer. 20

[AM07] ATALAY F., MOUNT D.: Pointerless implementation of hierarchical simplicial meshes and efficient neighbor finding in arbitrary dimensions. *International Journal of Computational Geometry and Applications 17*, 6 (2007), 595–631. 10, 14, 20

[AMP00] ARNOLD D., MUKHERJEE A., POULY L.: Locally adapted tetrahedral meshes using bisection. *SIAM Journal on Scientific Computing 22*, 2 (2000), 431–448. 4

© The Eurographics Association 2010.

[Bän91] BÄNSCH E.: Local mesh refinement in 2 and 3 dimensions. *IMPACT Comput. Sci. Engineering 3* (1991), 181–191. 4

[Bas94] BASTIAN P.: *Parallele adaptive Mehrgitterverfahren*. PhD thesis, University of Heidelberg, Germany, 1994. 4

[Bey95] BEY J.: Tetrahedral mesh refinement. *Computing 55* (1995), 355–378. 3, 4

[Bey00] BEY J.: Simplicial grid refinement: on freudenthal's algorithm and the optimal number of congruence classes. *Numerische Mathematik 85*, 1 (2000), 1–29. 3, 4, 6

[Blo00] BLOW J.: Terrain rendering at high levels of detail. In *Proceedings of the Game Developers Conference* (2000). 16

[BLV03] BALMELLI L., LIEBLING T., VETTERLI M.: Computational analysis of mesh simplification using global error. *Computational Geometry Theory and Applications 25*, 3 (2003), 171–196. 9, 12

[BPSC04] BORGO R., PASCUCCI V., SCOPIGNO R., CIGNONI P.: A progressive subdivision paradigm (PSP). *Proceedings of SPIE 5295* (2004), 223. 15, 18

[BSW83] BANK R., SHERMAN A. H., WEISER A.: Refinement algorithms and data structures for regular local mesh refinement. In *Scientific Computing, IMACS*, Stepleman R., Carver M., Peskin R., Ames W. F., Vichnevetsky R., (Eds.), vol. 1. North-Holland, Amsterdam, 1983, pp. 3–17. 3

[CDM*04] CIGNONI P., DE FLORIANI L., MAGILLO P., PUPPO E., SCOPIGNO R.: Selective refinement queries for volume visualization of unstructured tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics 10*, 1 (January-February 2004), 29–45. 5

[CGG*03a] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: BDAM - Batched Dynamic Adaptive Meshes for high performance terrain visualization. *Computer Graphics Forum 22*, 3 (2003), 505–514. 2, 16, 19

[CGG*03b] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Planet-Sized Batched Dynamic Adaptive Meshes (P-BDAM). In *Proceedings IEEE Visualization* (2003), IEEE Computer Society Washington, DC, USA, pp. 147–154. 15, 16

[CGG*04] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Adaptive tetrapuzzles: efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. *ACM Transactions on Graphics 23*, 3 (2004), 796–803. 2, 15, 19

[DFM02] DE FLORIANI L., MAGILLO P.: Multiresolution mesh representation: models and data structures. In *Principles of Multi-resolution Geometric Modeling* (Berlin, 2002), Floater M., Iske A., Quak E., (Eds.), Lecture Notes in Mathematics, Springer Verlag, pp. 364–418. 2, 11, 12

[DL04] DE FLORIANI L., LEE M.: Selective refinement on nested tetrahedral meshes. In *Geometric Modeling for Scientific Visualization*, Brunett G., Hamann B., Mueller H., (Eds.). Springer Verlag, 2004. 12, 14, 15

[DM82] DAHMEN W. A., MICCHELLI C. A.: On the linear independence of multivariate b-splines, i. triangulations of simploids. *SIAM Journal on Numerical Analysis 19*, 5 (1982), 993–1012. 5

[DM03] DE FLORIANI L., MAGILLO P.: Algorithms for visibility computation on terrains: a survey. *Environment and Planning B - Planning and Design 30*, 5 (2003), 709–728. (invited paper). 11

[DMPS02] DE FLORIANI L., MAGILLO P., PUPPO E., SOBRERO D.: A multi-resolution topological representation for non-manifold meshes. In *Proceedings 7th ACM Symposium on Solid Modeling and Applications (SM02)* (June 2002), ACM Press. 15

[DP95] DE FLORIANI L., PUPPO E.: Hierarchical triangulation for multi-resolution surface description. *ACM Transactions on Graphics 14*, 4 (October 1995), 363–411. 3

[DPM97] DE FLORIANI L., PUPPO E., MAGILLO P.: A formal approach to multi-resolution modeling. In *Geometric Modeling: Theory and Practice*, Strasser W., Klein R., Rau R., (Eds.). Springer-Verlag, 1997, pp. 302–323. 5

[DWS*97] DUCHAINEAU M., WOLINSKY M., SIGETI D. E., MILLER M. C., ALDRICH C., MINEEV-WEINSTEIN M. B.: ROAMing terrain: real-time optimally adapting meshes. In *Proceedings IEEE Visualization* (Phoenix, AZ, October 1997), Yagel R., Hagen H., (Eds.), IEEE Computer Society, pp. 81–88. 6, 8, 12, 13, 14, 15, 16, 17

[EKT01] EVANS W., KIRKPATRICK D., TOWNSEND G.: Right-triangulated irregular networks. *Algorithmica 30*, 2 (2001), 264–286. 14, 15, 16, 18

[ELZ02] EDELSBRUNNER H., LETSCHER D., ZOMORODIAN A.: Topological persistence and simplification. *Discrete and Computational Geometry 28*, 4 (2002), 511–533. 17

[ESV99] EL-SANA J., VARSHNEY A.: Generalized view-dependent simplification. *Computer Graphics Forum 18*, 3 (1999), C83–C94. 5

[FMS95] FUJISHIRO I., MAEDA Y., SATO H.: Interval volume: a solid fitting technique for volumetric data display and analysis. In *Proceedings IEEE Visualization* (Los Alamitos, CA, USA, 1995), IEEE Computer Society, pp. 151–158. 17

[FPRJ00] FRISKEN S. F., PERRY R. N., ROCKWOOD A. P., JONES T. R.: Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings SIGGRAPH* (New Orleans, LA, July 2000), ACM Press, pp. 249–254. 19

[Fre42] FREUDENTHAL H.: Simplizialzerlegungen von beschrankter flachheit. *Annals of Mathematics 43*, 3 (1942), 580–582. 3, 5, 6

[GDL*02] GREGORSKI B., DUCHAINEAU M., LINDSTROM P., PASCUCCI V., JOY K.: Interactive view-dependent rendering of large isosurfaces. In *Proceedings IEEE Visualization* (October 2002), IEEE Computer Society Washington, DC, USA, pp. 475–484. 6, 7, 9, 10, 12, 13, 17, 19, 20

[Ger02] GERSTNER T.: Multiresolution extraction and rendering of transparent isosurfaces. *Computers & Graphics 26*, 2 (2002), 219–228. 17, 19

[Ger03a] GERSTNER T.: Multi-resolution visualization and compression of global topographic data. *GeoInformatica 7*, 1 (2003), 7–32. 15, 16

[Ger03b] GERSTNER T.: *Top-Down View-Dependent Terrain Triangulation using the Octagon Metric*. Tech. rep., Institut für Angewandte Mathematik, University of Bonn, 2003. 9, 16

[GG98] GROSSO R., GREINER G.: Hierarchical meshes for volume data. In *Proceedings Computer Graphics International* (1998), pp. 761–769. 3

[GG00] GREINER G., GROSSO R.: Hierarchical tetrahedral-octahedral subdivision for volume visualization. *The Visual Computer 16* (2000), 357–369. 3

[GMC*02] GERSTNER T., MEETSCHEN D., CREWELL S., GRIEBEL M., SIMMER C.: A case study on multiresolution visualization of local rainfall from weather radar measurements. In *Proceedings IEEE Visualization* (2002), Pfister H., Bailey M., (Eds.), IEEE Computer Society Press, pp. 533–536. 19

© The Eurographics Association 2010.

[GMC*06] GOBBETTI E., MARTON F., CIGNONI P., DI BENEDETTO M., GANOVELLI F.: C-BDAM Ű Compressed Batched Dynamic Adaptive Meshes for terrain rendering. *Computer Graphics Forum 25*, 3 (2006), 333–342. 15, 16

[GP00] GERSTNER T., PAJAROLA R.: Topology-preserving and controlled topology simplifying multi-resolution isosurface extraction. In *Proceedings IEEE Visualization* (2000), pp. 259–266. 17

[GR99] GERSTNER T., RUMPF M.: Multiresolutional parallel isosurface extraction based on tetrahedral bisection. In *Proceedings Symposium on Volume Visualization* (1999), ACM Press, pp. 267–278. 15, 17

[GRW00] GERSTNER T., RUMPF M., WEIKARD U.: Error indicators for multilevel visualization and computing on nested grids. *Computers & Graphics 24*, 3 (2000), 363–373. 12, 13

[GSDJ04] GREGORSKI B., SENECAL J., DUCHAINEAU M., JOY K.: Adaptive extraction of time-varying isosurfaces. *IEEE Transactions on Visualization and Computer Graphics 10*, 6 (2004), 683–694. 19

[GSDJ09] GREGORSKI B., SENECAL J., DUCHAINEAU M., JOY K. I.: Compression and occlusion culling for fast isosurface extraction from massive datasets. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, Mathematics and Visualization. Springer, 2009, pp. 303–323. 18

[GTLH98] GUEZIEC A., TAUBIN G., LAZARUS F., HORN W.: Converting sets of polygons to manifold surfaces by cutting and stitching. In *Conference abstracts and applications: SIGGRAPH 98* (1998), Computer Graphics, ACM Press, pp. 245–245. 5

[Guo95] GUO B.: Interval set: A volume rendering technique generalizing isosurface extraction. In *Proceedings IEEE Visualization* (1995), IEEE Computer Society Washington, DC, USA, pp. 3–10. 17

[HDJ05] HWA L., DUCHAINEAU M., JOY K.: Real-time optimal adaptation for planetary geometry and texture: 4-8 tile hierarchies. *IEEE Transactions on Visualization and Computer Graphics 11*, 4 (2005), 355–368. 15, 16

[Heb94] HEBERT D.: Symbolic local refinement of tetrahedral grids. *Journal of Symbolic Computation 17*, 5 (May 1994), 457–472. 2, 10, 14

[Heb98] HEBERT D.: Cyclic interlaced quadtree algorithms for quincunx multiresolution. *Journal of Algorithms 27*, 1 (1998), 97–128. 2, 16

[HK95] HEBERT D. J., KIM H.: Image encoding with triangulation wavelets. In *SPIE Conference Series* (1995), Laine A. F., Unser M. A., Wickerhauser M. V., (Eds.), vol. 2569, pp. 381–392. 16

[Hop97] HOPPE H.: View-dependent refinement of progressive meshes. In *ACM Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH'97)* (Los Angeles, August 1997), pp. 189–198. 5

[KTY*04] KIMURA A., TAKAMA Y., YAMAZOE Y., TANAKA S., TANAKA H.: Parallel volume segmentation with tetrahedral adaptive grid. *International Conference on Pattern Recognition 2* (2004), 281–286. 2, 19

[Kuh60] KUHN H.: Some combinatorial lemmas in topology. *IBM J. Res. Develop 4* (1960), 518–524. 5

[LDS01] LEE M., DE FLORIANI L., SAMET H.: Constant-time neighbor finding in hierarchical tetrahedral meshes. In *Proceedings International Conference on Shape Modeling* (Genova, Italy, May 2001), IEEE Computer Society, pp. 286–295. 14

[LDS04] LEE M., DE FLORIANI L., SAMET H.: Constant-time navigation in four-dimensional nested simplicial meshes. In *Proceedings Shape Modeling International 2004* (June 2004), IEEE Computer Society, pp. 221–230. 7, 14, 19, 20

[LE97] LUEBKE D., ERIKSON C.: View-dependent simplification of arbitrary polygonal environments. In *ACM Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '97)* (1997), ACM Press, pp. 199–207. 5

[Lee06] LEE M. T.: *Spatial Modeling using Triangular, Tetrahedral and Pentatopic Decompositions*. PhD thesis, The University of Maryland, College Park, 2006. 15, 20

[Lev02] LEVENBERG J.: Fast view-dependent level-of-detail rendering using cached geometry. In *Proceedings IEEE Visualization* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 259–266. 16

[LGP*04] LINSEN L., GRAY J., PASCUCCI V., DUCHAINEAU M. A., HAMANN B., JOY K.: Hierarchical large-scale volume representation with $\sqrt[3]{2}$ subdivision and trivariate b-spline wavelets. In *Geometric Modeling for Scientific Visualization*, Brunnett G., Hamann B., Mueller H., Linsen L., (Eds.), Mathematics + Visualization. Springer Verlag, Heidelberg, Germany, 2004, pp. 359–378. 18, 19

[LH04] LOSASSO F., HOPPE H.: Geometry clipmaps: terrain rendering using nested regular grids. In *Proceedings ACM Siggraph* (2004), ACM New York, NY, USA, pp. 769–776. 16

[LHJ07] LINSEN L., HAMANN B., JOY K.: Wavelets for adaptively refined "3rd-root-of-2" subdivision meshes. *International Journal of Computers & Applications 29*, 3 (2007), 223–231. 18

[Lic99] LICKORISH W.: Simplicial moves on complexes and manifolds. *Geometry and Topology Monographs 2*, 299-320 (1999), 314. 7

[LJ95] LIU A., JOE B.: Quality local refinement of tetrahedral meshes based on bisection. *SIAM Journal on Scientific Computing 16*, 6 (1995), 1269–1291. 4

[LKR*96] LINDSTROM P., KOLLER D., RIBARSKY W., HODGES L. F., FAUST N., TURNER G. A.: Real-time continuous level of detail rendering of height fields. In *Proceedings ACM SIGGRAPH* (August 1996), pp. 109–118. 9, 15, 16

[LP02] LINDSTROM P., PASCUCCI V.: Terrain simplification simplified: a general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 239–254. 15, 16, 17, 18

[LPD*04] LINSEN L., PASCUCCI V., DUCHAINEAU M., HAMANN B., JOY K.: Wavelet-based multiresolution with $\sqrt[n]{2}$ subdivision. *Journal on Computing, Special Edition: Dagstuhl Seminar on Geometric Modelling 72* (2004), 129–142. 19

[LRC*02] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3D Graphics*. Computer Graphics and Geometric Modeling. Morgan-Kaufmann, San Francisco, 2002. 8, 12, 14, 15, 18

[LS00] LEE M., SAMET H.: Navigating through triangle meshes implemented as linear quadtrees. *ACM Transactions on Graphics 19*, 2 (April 2000), 79–121. 6

[LVLM04] LEWINER T., VELHO L., LOPES H., MELLO V.: Simplicial isosurface compression. In *Vision, Modeling, and Visualization* (Stanford, CA, November 2004), pp. 299–306. 18

[Mag99] MAGILLO P.: *Spatial Operations on Multiresolution Cell Complexes*. PhD thesis, Dept. of Computer and Information Sciences, University of Genova (Italy), 1999. 11

[Mau95] MAUBACH J. M.: Local bisection refinement for n-simplicial grids generated by reflection. *SIAM Journal on Scientific Computing 16*, 1 (January 1995), 210–227. 2, 4, 5, 6, 14

© The Eurographics Association 2010.

[Mau96] MAUBACH J. M.: The efficient location of neighbors for locally refined n-simplicial grids. In *5th Int. Meshing Roundable* (1996). 4, 6, 14

[Mau05] MAUBACH J.: Space-filling curves for 2-simplicial meshes created with bisections and reflections. *Applications of Mathematics 50*, 3 (2005), 309–321. 15

[MDM04] MARCHESIN S., DISCHLER J., MONGENET C.: 3D ROAM for scalable volume visualization. In *IEEE Symposium on Volume Visualization and Graphics* (2004), pp. 79–86. 2, 12, 17, 18, 20

[Mes48] MESERVE B.: Double factorials. *The American Mathematical Monthly 55*, 7 (1948), 425–426. 5

[Mit91] MITCHELL W.: Adaptive refinement for arbitrary finite-element spaces with hierarchical bases. *Journal of computational and applied mathematics 36*, 1 (1991), 65–78. 4

[MVT03] MELLO V., VELHO L., TAUBIN G.: Estimating the in/out function of a surface represented by points. In *Symposium on Solid Modeling and Applications* (2003), pp. 108–114. 2, 19

[MW95] MOORE D., WARREN J.: Adaptive simplicial mesh quadtrees. *Houston J. Math 21*, 3 (1995), 525–540. 3, 6

[New31] NEWMAN M.: A theorem in combinatorial topology. *J. London Math. Soc s1–6*, 3 (1931), 186–192. 7

[OR97] OHLBERGER M., RUMPF M.: Hierarchical and adaptive visualization on nested grids. *Computing 56*, 4 (1997), 365–385. 12, 13

[OR99] OHLBERGER M., RUMPF M.: Adaptive projection operators in multi-resolution scientific visualization. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (1999), 74–93. 12

[Paj98] PAJAROLA R.: Large scale terrain visualization using the restricted quadtree triangulation. In *Proceedings IEEE Visualization* (Research Triangle Park, NC, October 1998), Ebert D., Hagen H., Rushmeier H., (Eds.), IEEE Computer Society, pp. 19–26. 9, 15, 16, 17

[Pas02] PASCUCCI V.: Slow Growing Subdivisions (SGS) in any dimension: towards removing the curse of dimensionality. *Computer Graphics Forum 21*, 3 (2002), 451–460. 2, 5, 6, 8

[Pas04] PASCUCCI V.: Isosurface computation made simple: Hardware acceleration, adaptive refinement and tetrahedral stripping. In *Eurographics/IEEE TVCG Symposium on Visualization (VisSym)* (2004), pp. 293–300. 17

[PB00] PASCUCCI V., BAJAJ C. L.: Time-critical isosurface refinement and smoothing. In *Proceedings IEEE Symposium on Volume Visualization* (Salt Lake City, UT, October 2000), IEEE Computer Society, pp. 33–42. 18

[PF01] PASCUCCI V., FRANK R. J.: Global static indexing for real-time exploration of very large regular grids. In *Proceedings ACM/IEEE Supercomputing* (2001), pp. 45–45. 18

[PG07] PAJAROLA R., GOBBETTI E.: Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer 23*, 8 (2007), 583–605. 15

[Pom00] POMERANZ A.: *ROAM using surface triangle clusters (RUSTiC)*. Master's thesis, U.C. Davis, 2000. 16, 19

[PP09] PUPPO E., PANOZZO D.: Rgb subdivision. *IEEE Transactions on Visualization and Computer Graphics 15*, 2 (2009), 295–310. 4

[PT90] PAYNE B., TOGA A.: Surface mapping brain function on 3d models. *Computer Graphics and Applications, IEEE 10*, 5 (Sept. 1990), 33–41. 17

[Pup98] PUPPO E.: Variable resolution triangulations. *Computational Geometry Theory and Applications 11*, 3-4 (December 1998), 219–238. 9

[Riv84] RIVARA M.: Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *International Journal for Numerical Methods in Engineering 20*, 4 (1984), 745–756. 4

[Riv91] RIVARA M.: Local modification of meshes for adaptive and/or multigrid finite-element methods. *Journal of Computational and Applied Mathematics 36*, 1 (1991), 79–89. 2, 4, 5

[RN00] ROXBOROUGH T., NIELSON G.: Tetrahedron-based, least-squares, progressive volume models with application to freehand ultrasound data. In *Proceedings IEEE Visualization* (October 2000), IEEE Computer Society, pp. 93–100. 19

[Sam06] SAMET H.: *Foundations of Multidimensional and Metric Data Structures*. The Morgan Kaufmann series in computer graphics and geometric modeling. Morgan Kaufmann, 2006. 1, 3, 8, 10

[Sew72] SEWELL E.: *Automatic generation of triangulations for piecewise polynomial approximation*. PhD thesis, Purdue University, 1972. 4

[Sew79] SEWELL G.: A finite element program with automatic user-controlled mesh grading. In *Proceedings International Symposium on Computer Methods for Partial Differential Equations* (1979), Vichnevetsky R., Stepleman R. S., (Eds.), IMACS, pp. 8–10. 4

[SS92] SIVAN R., SAMET H.: Algorithms for constructing quadtree surface maps. In *Proc. 5th Int. Symposium on Spatial Data Handling* (1992), pp. 361–370. 2, 16

[Tan95] TANAKA H.: Accuracy-based sampling and reconstruction with adaptive meshes for parallel hierarchical triangulation. *Computer Vision and Image Understanding 61*, 3 (1995), 335 – 350. 17

[Tod76] TODD M.: *The computation of fixed points and applications*. Springer-Verlag, 1976. 5, 6, 19

[Tra97] TRAXLER C. T.: An algorithm for adaptive mesh refinement in *n* dimensions. *Computing 59*, 2 (1997), 115–137. 4

[TTNF04] TAKAHASHI S., TAKESHIMA Y., NIELSON G., FUJISHIRO I.: Topological volume skeletonization using adaptive tetrahedralization. In *Proceedings Geometric Modeling and Processing* (2004), pp. 227–236. 17

[TTW03] TANAKA H., TAKAMA Y., WAKABAYASHI H.: Accuracy-based sampling and reconstruction with adaptive grid for parallel hierarchical tetrahedrization. In *Proceedings Volume Graphics* (New York, NY, USA, 2003), ACM Press, pp. 79–86. 19

[Tuc45] TUCKER A.: Some topological properties of disk and sphere. In *Proceedings First Canadian Math. Congress, Montreal* (1945), vol. 285–309. 5

[VH89] VON HERZEN B.: *Applications of surface networks to sampling problems in computer graphics*. PhD thesis, California Institute of Technology, Pasadena, CA, USA, 1989. 8, 17

[VHB87] VON HERZEN B., BARR A. H.: Accurate triangulations of deformed, intersecting surfaces. In *Proceedings ACM SIGGRAPH* (New York, NY, USA, 1987), ACM, pp. 103–110. 2

[VZ01] VELHO L., ZORIN D.: 4-8 Subdivision. *Computer Aided Geometric Design 18*, 5 (2001), 397–427. 2

[WBP07] WEBER G., BREMER P., PASCUCCI V.: Topological landscapes: A terrain metaphor for scientific data. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1416–1423. 17

© The Eurographics Association 2010.

[WD08a] WEISS K., DE FLORIANI L.: Modeling and visualization approaches for time-varying volumetric data. In *Advances in Visual Computing* (2008), Bebis G., Boyle R., Parvin B., Koracin D., Remagnino P., Porikli F., Peters J., Klosowski J., Arns L., Chun Y., Rhyne T., Monroe L., (Eds.), Springer, pp. 1000–1010. 19

[WD08b] WEISS K., DE FLORIANI L.: Multiresolution interval volume meshes. In *IEEE/ EG Symposium on Volume and Point-Based Graphics* (Los Angeles, California, USA, 2008), Hege H.-C., Laidlaw D., Pajarola R., Staadt O., (Eds.), Eurographics Association, pp. 65–72. 18

[WD08c] WEISS K., DE FLORIANI L.: Sparse terrain pyramids. In *Proceedings ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (New York, NY, USA, 2008), ACM, pp. 115–124. 10, 12, 15, 16, 18

[WD09a] WEISS K., DE FLORIANI L.: Diamond hierarchies of arbitrary dimension. *Computer Graphics Forum (Proceedings SGP 2009) 28*, 5 (2009), 1289–1300. 2, 5, 6, 7, 9, 10, 13, 15, 20

[WD09b] WEISS K., DE FLORIANI L.: Supercubes: A high-level primitive for diamond hierarchies. *IEEE Transactions on Visualization and Computer Graphics (Proceedings IEEE Visualization 2009) 15*, 6 (November-December 2009), 1603–1610. 10, 18

[WDF10] WEISS K., DE FLORIANI L.: Isodiamond hierarchies: An efficient multiresolution representation for isosurfaces and interval volumes. *IEEE Transactions on Visualization and Computer Graphics*, PrePrints (2010). 18

[Whi57] WHITNEY H.: *Geometric integration theory*. Princeton University Press, 1957. 5

[WVG92] WILHELMS J., VAN GELDER A.: Octrees for faster isosurface generation. *ACM Transactions on Graphics 11*, 3 (1992), 201–227. 17

[XESV97] XIA J. C., EL-SANA J., VARSHNEY A.: Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (1997), 171–183. 5

[ZCK97] ZHOU Y., CHEN B., KAUFMAN A.: Multi-resolution tetrahedral framework for visualizing regular volume data. In *Proceedings IEEE Visualization* (Phoenix, AZ, October 1997), Yagel R., Hagen H., (Eds.), IEEE Computer Society, pp. 135–142. 7, 17

[Zha95] ZHANG S.: Successive subdivision of tetrahedra and multigrid methods on tetrahedral meshes. *Houston Journal of Mathematics 21* (1995), 541–556. 3

© The Eurographics Association 2010.