Volume xx (200y), Number z, pp. 1–27

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 regular 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. The primary distinction among all such approaches is whether they treat the simplex or clusters of simplices, 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.*

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 volumetric datasets for visualization and analysis, where we need to deal with huge point data sets or very large meshes describing isosurfaces.

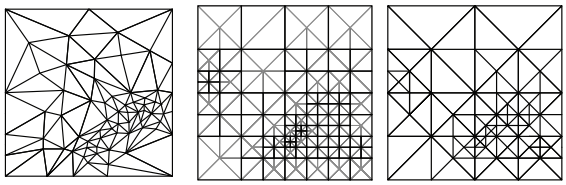
One common solution to deal with such large quantities of information in the form of point data or of meshes is to use hierarchical spatial decomposition techniques. The use of such techniques is quite diverse. For example, we have representations like octrees, kd-trees, or R-trees, which are used as spatial indexes on points, or on 3D shapes, and we also have multiresolution representations for terrains, volumetric datasets or higher dimensional scalar fields which are based on a hierarchical spatial decomposition of the domain [Sam06].

In this very broad context, we focus here 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 1a). In contrast, regular sampling permits implicit encodings of position and connectivity, but it requires the domain to be uniformly sampled.

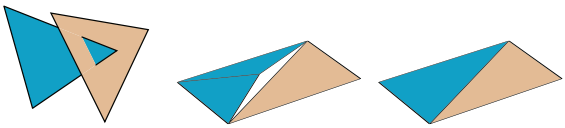
A compromise is to adaptively sample the domain at the vertices of a regular grid using a d-dimensional octree, in which each hypercubic subdomain Ω can be decomposed into 2^d hypercubic cells covering Ω. However, such *refinements* introduce an exponential number of new cells into the representation and can introduce cracks along neighboring cells of the domain decomposition, leading to discontinuities in functions defined on the cells (see Figure 2b). The discontinuity problem can be remedied by applying a suitable set of triangulation rules [VHB87, SS92, GB99, WD10a] to the cells of a *balanced* octree (also called a *restricted* octree), i.e. an octree in which neighboring nodes can differ in size by at most a factor of two (see Figure 1b).

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



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

Figure 1: (a) Irregular mesh (b) Triangulated balanced quadtree (c) Regular Simplex Bisection (RSB) mesh.



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

Figure 2: A decomposition is *conforming* (c) if its cells do not intersect (a) and it is free of cracks (b).

The *Regular Simplex Bisection* scheme enables more flexible decompositions over the same adaptive domain. By breaking up each octree refinement into d steps [Pas02], extracted meshes can be significantly more adaptable to features defined on the domain [DM02], while still supporting implicit encodings of the underlying d-dimensional geometry (see Figure 1c) [AM07, WD09a]. Crack-free refinements in this scheme apply to local clusters of simplices, doubling the number of simplices in the cluster while only adding a single new vertex.

Here, we consider approaches that decompose a regularly sampled domain using *Regular Simplex Bisection (RSB)*. We analyze such approaches in 2D, 3D and in higher dimensions and classify them on the basis of the dimension of the domain subdivided and on the choice of the basic building blocks of the representation, which can be simplices generated by the RSB rule, or *diamonds*, which are collections of such simplices 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 the modeling, visualization and analysis of 2D, 3D and higher dimensional scalar fields, which have been the major application areas of such hierarchical simplicial structures. Simplicial hierarchies based on regular simplex bisection are also a valid alternative to quadtrees and octrees as access structures to irregularly sampled spatial data [CGG*03a, CGG*04, AM07]. These hierarchies have been used in finite element analysis and actually originate from that field [Riv91, Mau95, Heb94, AG03]. Other applications include image reconstruction [HK95, 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 decompositions and to understand their suitability for new application domains.

The reminder of this paper 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, we describe the hierarchies of meshes defined by the dependency relations among the primitives (Section 4), and efficient means of encoding (Section 5) and querying (Section 6) such hierarchies. We then survey the applications of RSB meshes in 2D (Section 7), 3D (Section 8) and higher dimensions (Section 9). We conclude in Section 10 with suggestions for when to use simplex hierarchies or diamond hierarchies.

2. Preliminaries

We consider a domain $\Omega \subset \mathbb{R}^d$, and a set of samples V, which can be *regularly* or *irregularly* distributed within Ω. Furthermore, Ω is decomposed into a set of cells whose vertices are defined on a subset of V. We are primarily concerned with *conforming* simplicial meshes (see Section 2.1) generated through a nested refinement scheme (see Section 2.2) of a hypercubic domain. We discuss simplicial decompositions of a hypercube in Section 2.3.

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 simplices such that (a) if σ is a simplex in Σ, then all the simplices bounding it (called the *faces* of σ) also belong to Σ, and (b) the interiors of all simplices in Σ are disjoint.

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

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

2.2. Nested mesh refinement

A *nested refinement scheme* consists of rules for replacing a set of cells Γ_1 in a mesh Σ with a larger set of cells Γ_2 covering the same domain. When Γ_1 and Γ_2 share the same combinatorial boundary, the refinement does not introduce cracks into the decomposition, i.e, it is conforming.

Recall that two cells τ_1 and τ_2 are *similar* if there is an affine map $\mathcal{A}$ defined by translations, rotations, reflections and uniform scaling between them, i.e. $\tau_1 = \mathcal{A} \cdot \tau_2$. An equivalence class of similar cells is referred to as a *similarity class* of cells. The number of similarity classes 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, that the angles at the vertices are 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 *bisection refinement* (see Section 2.2.2). For an example of a nested mesh refinement scheme over irregularly sampled domains, see [DP95].

2.2.1. Regular refinement

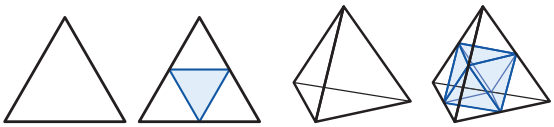
The *regular refinement* of a d-dimensional cell τ is defined by adding vertices at all edge midpoints of τ and decomposing τ into 2^d disjoint cells covering τ [BSW83].

Regular refinement on (hyper)-cubic cells generates quadtree and octree decompositions as well as their higher dimensional analogues, in which all 2^d generated cells are similar and share the midpoint of the refined domain as a common 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 incident 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 can generate multiple similarity classes of simplices. For example, when refining a tetrahedron σ, the four tetrahedra incident to the vertices of σ are similar to σ, while the remaining four tetrahedra obtained by subdividing the octahedral domain $\mathcal{O}$ defined by the edge midpoints of σ, are not similar to σ (see Figure 3b).

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

Bey [Bey95] proposes a *typographical* refinement rule



(a) Triangle Refinement (b) Tetrahedral Refinement

Figure 3: Regular refinement of 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).

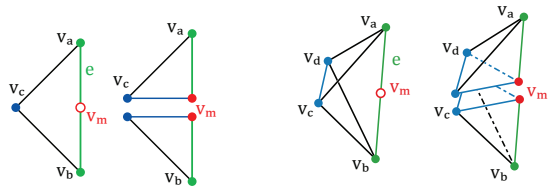
based on maintaining a specific vertex ordering during refinement. In contrast to geometric refinements, typographical refinements require the cells to be initially ordered, but are invariant under affine transformations.

An interesting modification is the *tetrahedral/octahedral* regular refinement scheme [GG98, GG00, CQ06] in which the octahedra generated by regular tetrahedral refinement are also treated as primitives. In this scheme, each tetrahedron is refined into four similar tetrahedra and a single octahedron (see Figure 3b), while each octahedron is decomposed into six similar octahedra incident to its vertices, and eight similar tetrahedra corresponding to its truncated triangular faces.

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 upper bound of $(d!/2)$ similarity classes generated for each class of simplex in the initial mesh. Moore [MW95] provides a consistent labeling and simplex enumeration algorithm for the 2^d simplices generated by regular refinement of a simplex.

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 refinement rules (*red*) are augmented by a set of irregular *closure* refinement rules (*green*) to patch cracks between regular cells at different resolutions. An additional *balancing* constraint restricts the degree of decomposition between edge-adjacent cells, thereby reducing the number of green refinement rules that need to be considered. This scheme has been extended to tetrahedral meshes in [Bey95].

The recent *RGB subdivision* scheme for triangle meshes [PP09] introduces *blue* refinement rules, to transition between triangles generated by regular (red) and irregular (green) refinements. The addition of the blue refinements rules enables red and green operations to be applied in arbitrary order.



(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 incident to $\mathbf{e}$.

2.2.2. Bisection refinement

The second class of nested refinement schemes is defined by *bisection refinement*, in which a cell is bisected along a hyperplane into two cells. When the cells are axis-aligned hyper-rectangles which are bisected by axis-aligned hyperplanes, this generates kd-trees [Ben75].

Alternatively, the *simplex bisection* 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 simplices with disjoint interiors covering its domain. Figure 4 illustrates the simplex bisection rule in 2D and 3D.

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, Riv91, PC00] or *typographical* [Mit91, Bän91, Mau95, Tra97] bisection 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 any 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 the pioneering work of Sewell [Sew72, Sew79]) introduces the *newest vertex bisection* algorithm for irregular triangle meshes, in which the bisection 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 create a vertex $\mathbf{v}_m$ at the midpoint of edge $[\mathbf{v}_a, \mathbf{v}_b]$, and the generated triangles 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, Kos94, 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 $(d-i)^{th}$ vertex. A similar d-dimensional scheme was independently proposed by Traxler [Tra97]. After initialization, simplex bisection generates at most $(2^{d-2}d!)$ similarity classes of simplices for each class of simplex in the initial mesh [Bey00].

Simplex bisection is not a conforming refinement. Specifically, it introduces cracks between the neighbors of a bisected simplex σ that are incident to its bisection edge, and the two simplices generated during σ's bisection (see Figure 2b). This can be remedied by ensuring that all simplices incident to the bisection edge $\mathbf{e}$ have $\mathbf{e}$ as their bisection edge and are refined concurrently. When the first property is not satisfied for a bisecting simplex σ, a *closure* operation can be applied (recursively) on σ's neighboring simplices, triggering additional bisection operations, and thereby enforcing conforming refinements (as we describe in Section 6).

2.2.3. Comparison

Over an arbitrary simplicial mesh, regular refinement generates fewer similarity classes of simplices than bisection (i.e. $2^{-1}d!$ 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 domain (see Section 2.3), the number of similarity classes generated by bisection has been shown to be at most d [Mau95].

Interestingly, for $d > 2$, it is unknown if all simplicial complexes admit a typographical simplex bisection scheme [NSV09]. Kossaczký [Kos94] proposes an initialization refinement algorithm to ensure that arbitrary tetrahedral meshes are admissible, although this increases the number of simplex classes. Stevenson [Ste08] generalizes this approach to higher dimensions.

Although red/green schemes have two families 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 simplex bisection is defined recursively in terms of bisections, only one family of simplex similarity classes is generated. This simplifies the analysis of the scheme as well as data structures and refinement algorithms, especially as the dimensionality of the domain increases.

Meshes generated through bisection are significantly

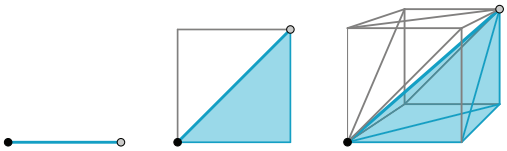


Figure 5: Kuhn-subdivided hypercubes in (a) 1D (b) 2D and (c) 3D, defined by one edge, two triangles and six tetrahedra, respectively. One of the $d!$ simplices is highlighted in blue. All edges are aligned with the diagonal of an axis-aligned hypercube.

more adaptive than those generated by regular refinement since 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 domain [Pas02].

2.3. Simplicial decomposition of a hypercube

In this subsection, we consider the canonical decomposition of a hypercube into $d!$ simplices along a diagonal (although smaller decompositions are possible [Mar76, Zon05]). This decomposition was initially proposed by Freudenthal [Fre42] and was popularized by Kuhn [Kuh60] in the context of fixed point calculations.

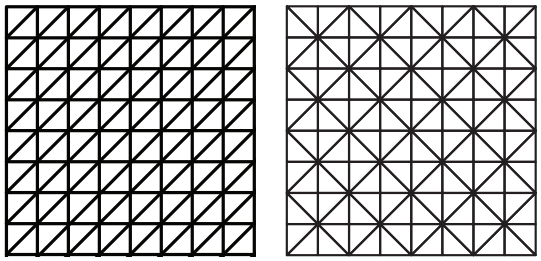
The *Kuhn-subdivision* of a d-dimensional unit cube h, which we denote as $\mathcal{K}(h)$, is a simplicial complex [AG79] defined by $d!$ simplices of order d, sharing a common diagonal of h. Figure 5 illustrates Kuhn-subdivided d-cubes for $d = 1,2,3$, and highlights one of the $d!$ simplices.

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 is 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 Figures 8c and 16a for examples in 2D and 3D). 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* identity

$$(2d)!! \equiv 2^d \cdot d! \qquad (1)$$



(a) K_1 triangulation (b) J_1 triangulation

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.

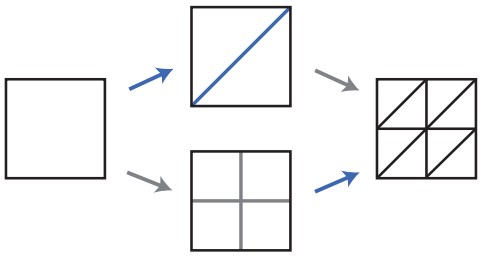


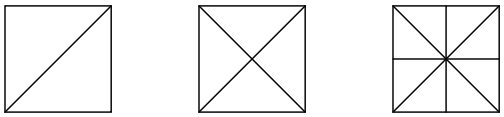
Figure 7: K_1 can be generated by applying regular refinement to a Kuhn-subdivided cube (upper path) or by Kuhn-subdividing the leaves of a complete quadtree (lower path).

where $n!! = 1$, if $n \in \{0,1\}$, and $n!! = n \cdot (n-2)!!$, otherwise [Mes48]. The first few values of $(2d)!!$, for $d = [1,2,3,4]$ are $[2,8,48,384]$.

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 (see upper path of Figure 7). Alternatively, this decomposition can be obtained by applying the Kuhn-subdivision to the leaves of a complete quadtree (see 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.

In contrast, the Tucker-Whitney triangulation J_1 can be obtained by applying successive bisection refinements to all elements of $\mathcal{K}(\Omega)$ [Mau96]. Thus, J_1 is the canonical decomposition for *complete* regular simplex bisection meshes (as defined in Section 3, see Figure 8).

Adaptive variations of the J_1 triangulation can be defined by applying bisection refinements to a balanced octree mesh. When all edge-adjacent hypercubes (i.e. top hypercubes sharing at least one edge in the mesh) are within one level of refinement of each other, this leads to an RSB mesh triangulating the 2D [VHB87, SS92, RHSS98], 3D [CNS*06]



(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.

or nD [WD10a] octree. The family of RSB meshes obtained by triangulating an edge-balanced octree is a subset of the family of conforming RSB meshes, i.e., there are conforming RSB meshes that cannot be obtained in this manner (compare Figures 1b and 1c) [DM02, WD10a]. More generally, a balanced octree in which face-adjacent hypercubes are within one level of refinement can be triangulated using the J_1^a triangulation [CNS*06]. In this scheme, a vertex is inserted at the center of every hypercube and the simplices within a hypercube h are defined by the edges connecting this vertex to the vertices on the boundary of h. This scheme is stable and generates an RSB mesh only when the input octree is edge-balanced.

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

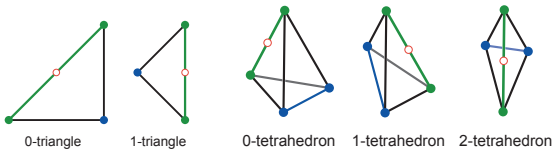
3. Regular Simplex Bisection (RSB)

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 *Kuhn-subdivision* of a (hyper)-cubic domain Ω (see Section 2.3).
2. a *typographical* simplex bisection operation, 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 hypercube as an *RSB simplex*, and to a mesh composed entirely of RSB simplices as an *RSB mesh*.

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.



(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.

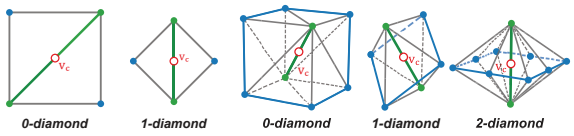
In dimension d, the RSB scheme generates d *similarity classes* of RSB simplices, where the *class* of a simplex cycles with every d refinements [Mau95]. Thus, the $d!$ simplices in $\mathcal{K}(h)$ all belong to class 0, and the two simplices generated by bisecting a simplex of class i belong to class $(i+1) \bmod d$. After m bisections, the class of a simplex σ is $(m \bmod d)$.

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 $(d-i)$). Furthermore, after d bisections, the (hyper)-volume of an RSB simplex is half that of its d-fold predecessor [Mau95]. Figure 9 illustrates the two classes of RSB triangles and the three classes of RSB tetrahedra.

We are often interested in generating conforming meshes using the RSB 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-simplices incident to the bisection edge. In other words, not all nested RSB meshes are conforming.

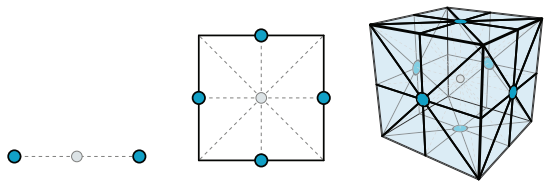
Conforming refinements along an edge $\mathbf{e}$ are carried out by simultaneously bisecting all incident simplices. Since the RSB scheme imposes the bisection edge of each RSB simplex, this requires all incident simplices to define $\mathbf{e}$ as their bisection edge. The cluster of such simplices is typically called a *diamond* [DWS*97, GDL*02, Pas02, WD09a], and we refer to the bisection edge as the *spine* of the diamond. Thus, all RSB simplices of order d within a d-dimensional diamond are similar, and there are d similarity classes of diamonds, in correspondence to the d similarity classes of RSB simplices. Figure 10 illustrates the two classes of diamonds in 2D and the three classes of diamonds in 3D.

In 2D, diamonds are defined by two isosceles right triangles that are adjacent along their hypotenuse (see Figure 10a), and, thus, they lie in a square domain defined by four 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, GP00, GDL*02] (see Figure 10b). Four-dimensional diamonds are defined by



(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 an axis-aligned $(d-i)$-cube.

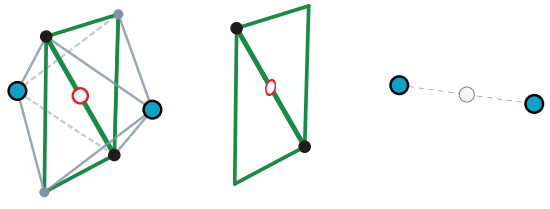


(a) (b) (c)

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 simplices of order $(d-1)$. The dashed lines and gray vertex at the midpoint of the cube belong to $\mathcal{F}$ but not $\mathcal{B}_F$.

24, 12, 16 and 48 *pentatopes* (i.e. 4-simplices), 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 simplices incident to the vertex at the midpoint of $\mathcal{F}$. $\mathcal{B}_F$ is therefore defined by $(2d)!!$ simplices of order $(d-1)$. Figure 11 illustrates $\mathcal{B}_F$ in 1D, 2D and 3D.



(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.

	0	1	2	3	i
1	1				
2	2	2			
3	6	4	8		
4	24	12	16	48	
d	$d!$	$2(d-1)!$	$8(d-2)!$	$48(d-3)!$	$(2i)!!(d-i)!$

Table 1: A d-dimensional i-diamond has $(2i)!!(d-i)!$ RSB simplices. The $(d-i)!$ factor comes from the Kuhn-subdivided component $\mathcal{K}(h_k)$ (rows), while the $(2i)!!$ factor comes from the fully-subdivided i-cube boundary component $\mathcal{B}_F(h_i)$ (columns).

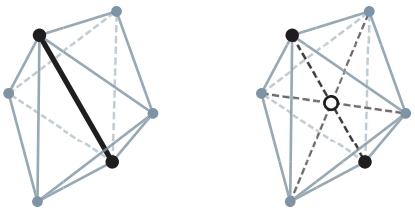
	0	1	2	3	i
1	2				
2	4	4			
3	8	6	10		
4	16	10	12	28	
d	2^d	$2^{d-1}+2$	$2^{d-2}+8$	$2^{d-3}+26$	$2^{d-i}+(3^i-1)$

Table 2: A d-dimensional i-diamond has $2^{d-i}+(3^i-1)$ vertices. The 2^{d-i} comes from the Kuhn-subdivided component $\mathcal{K}(h_k)$ (rows), while the (3^i-1) comes from the fully-subdivided i-cube boundary component $\mathcal{B}_F(h_i)$ (columns).

Using these definitions, a d-dimensional diamond of class i is the simplicial complex defined by the pairwise *simplicial join* [Lic99] of two orthogonal simplicially decomposed hypercubes sharing the same center: 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 $(2i)!! \cdot (d-i)!$ simplices all sharing their spine, which is the diagonal of an axis-aligned $(d-i)$-cube and contains $(2^{d-i})+(3^i-1)$ vertices. Tables 1 and 2 list the number of simplices and vertices in an i-diamond of dimension d. Observe that the number of simplices in a d-dimensional diamond is factorial in the dimension, while the number of vertices is exponential in the dimension.

A diamond δ is refined by bisecting all of its simplices using the regular simplex bisection operation, thus, doubling its simplices. 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



(a) (b)

Figure 13: Diamond refinement removes its *spine* (bold edge in (a)), adds its *central vertex* (hollow vertex in (b)), and adds edges from that vertex to each of the original vertices.

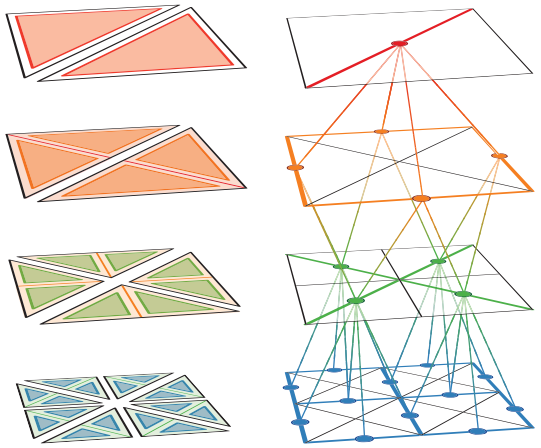
enables diamonds to be uniquely identified by their spine, or alternatively, by their central vertex [GDL*02].

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 $(d-i)$-faces. 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, which we refer to generally as *RSB hierarchies*. The basic classification for representations of nested RSB meshes is between *simplex-based* and *diamond-based* representations. The former consider the simplex as the basic primitive, while the latter consider the diamond as primitive. A simplex-based representation for an RSB hierarchy implicitly encodes all the nested RSB meshes i.e., the meshes generated from the initial Kuhn-subdivided domain through regular simplex bisection, while a diamond-based representation 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 simplices*, which encodes the containment relation between RSB simplices induced by regular simplex bisection. A diamond-based representation is described by a Directed Acyclic Graph (DAG) of diamonds, called a *hierarchy of diamonds*, where the arcs between diamonds are induced by the containment relation among all RSB simplices within a diamond. 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.



(a) Hierarchy of Simplices (b) Hierarchy of Diamonds

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

4.1. Simplex hierarchies

The containment relation between simplices in a nested RSB mesh defines a hierarchical relationship, where the two simplices created during the bisection of a *parent* simplex σ are referred to as the *children* of σ. This relationship can be captured using a binary tree, often referred to as a *bintree* [VH89, DWS*97, LRC*02], whose root is a simplex from 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 simplices*. When the domain is specific to 2D, or 3D, we refer to the model as a *hierarchy of (right) triangles*. or as a *hierarchy of tetrahedra*, respectively.

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

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 cov-

ering its domain, thus forming a nested decomposition of the square domain.

4.2. Diamond hierarchies

The containment relation among the simplices of a nested RSB mesh also 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 simplices. 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.

Due to the one-to-one 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 diamond's central vertex depends on vertices introduced at shallower depths of the hierarchy, while, for simplifications, it depends on vertices introduced deeper in the hierarchy.

Lindstrom et al. [LKR*96] were the first to consider the *simplification dependency relation* among the vertices of a two-dimensional nested RSB mesh. They observe 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 simplified. 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 the Multi-Tessellation (MT) framework [DPM97], a dimension-independent, strategy-agnostic multiresolution framework over irregular cell complexes (see also [DMP99, DM02]).

Thus, the dependency relation can be modeled as a 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 by its central vertex (a colored circle at the midpoint of its spine). 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 three-dimensional diamonds as $\{3,2,4\}$ and $\{6,4,8\}$, respectively.

Weiss and De Floriani [WD09a] use their diamond decomposition (see Section 3) to find the number of parents and children of a diamond in arbitrary dimension. They demonstrate that the central vertices of a diamond's parents coincide with the facet midpoints of its fully-subdivided component, while those of its children coincide with the facet midpoints of its Kuhn-subdivided component. In general, an i-diamond of dimension d has $2i$ parents and $2(d-i)$ children. However, when $i = 0$, it has d parents, and when $i = (d-1)$, it has 2^d children. Thus, diamonds always have $O(d)$ parents, and have $O(d)$ children in all but the final class, where they have $O(2^d)$ children. Using these equations, we can determine that the four classes of diamonds in 4D have $\{4,2,4,6\}$ parents and $\{8,6,4,16\}$ children, respectively.

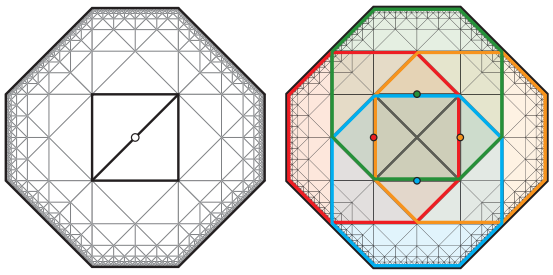
4.3. Dependency domains

Since conforming RSB refinements do not define a containment hierarchy, some researchers have considered the shape of the domain covered by subsets of a node's ancestors or descendants. In particular, the limit shape covered by all possible descendants can be useful, since regions outside this domain are unaffected by conforming refinements within it. This was first considered by Tanaka [Tan95], in the context of parallel triangle mesh refinement.

Balmelli et al. [BAV98, BLV03] consider the *ancestor domain* and *descendant domain* of each vertex in terms of its insertion or removal from a nested mesh. They define the ancestor domain of a vertex $\mathbf{v}$ (which they refer to as its *splitting domain*) as the domain covered by all triangles that must be refined concurrently to maintain a conforming triangulation upon the insertion of $\mathbf{v}$ into a nested RSB mesh. Similarly, they define the descendant domain of a vertex $\mathbf{v}$ (which they refer to as its *merging domain*) as 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. Thus, the ancestor domain of a vertex is the region covered by all ancestors up to the root(s) of the hierarchy while its descendant domain is covered by all descendants down to the hierarchy's leaf nodes.

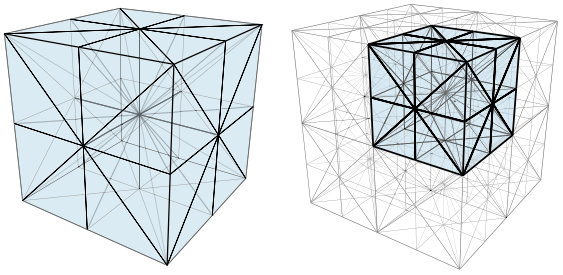
Gerstner [Ger03b] further develops this relationship by considering the limit shape of a vertex's descendant domain in a 2D RSB hierarchy. If the side length of a triangle in the diamond is s and its hypotenuse is $s\sqrt{2}$, then its descendant domain covers an octagonal region whose edge lengths alternate between s and $s\sqrt{2}$ (see Figure 15a). 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 octagonal descendant domains form a containment hierarchy. That is, the descendant domains of a diamond's four children are nested within its own descendant domain (see Figure 15b).

Tanaka et al. [TTW03] propose the *rhombicuboctahedron* as a generalization of the descendant domain to 3D.



(a) Descendant domain (b) Nested domains

Figure 15: (a) The limit shape of a diamond's 2D *descendant domain* is an octagon with the edge lengths of its triangles. (b) Descendant domains form a containment hierarchy.



(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.

However, this shape does not generate a containment hierarchy [WD10c]. Weiss and De Floriani [WD10c] demonstrate that the descendant domains of 3D diamonds have fractal boundaries, and propose two simpler families of nested polyhedra to conservatively approximate the descendant domain. An analysis of the properties of dependency domains in higher dimensions remains an open problem.

5. Encoding RSB hierarchies

In this section, we discuss efficient representations that have been proposed for encoding hierarchies of simplices and diamonds. Recall that *explicit* representations encode pointers to the vertices, parents, children and neighbors of an RSB simplex or diamond. *Implicit* representations exploit the regularity of the domain decomposition and vertex distribution, to yield significant savings in storage space. An API for an explicit hierarchy of triangles based on the half-edge data structure is described in [Vel01].

Since both types of RSB hierarchies are defined over the underlying Tucker-Whitney triangulation J_1 (see Section 2.3), which is tiled at each level of resolution by the highly symmetric *fully subdivided cubes* $\mathcal{F}$, several implicit representations exploit the relationship between RSB hierarchies and their corresponding fully subdivided cubes. Observe that a *complete* RSB mesh of maximum level of resolution 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.

5.1. Encoding simplex hierarchies

Implicit encodings for the simplices within the hierarchy have been inspired by *location codes* on linear quadtrees and octrees [Gar82,Sch92]. Recall that regular refinement of a d-dimensional hypercube generates 2^d hypercubes sharing its midpoint $\mathbf{v}_m$ (see Section 2.2.1). Each of the generated hypercubes can be indexed by a d-bit integer $i_r \in [0,2^d)$ based on the *orthant* (higher-dimensional generalization of *quadrant* and *octant*) in which it lies with respect to $\mathbf{v}_m$.

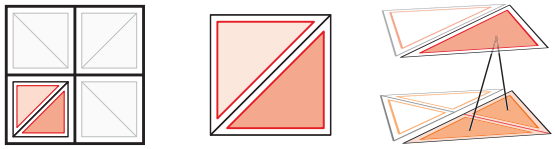
A hypercube h at refinement level ℓ can be uniquely indexed by its *location code*, a string of d-bit integers (of length ℓ) describing a traversal of the decomposition path from the root of the d-dimensional octree to h. To simplify the encoding, the level ℓ of h is typically encoded as well.

Generalizations of octree-like location codes to simplex hierarchies have led to two primary variants, which we refer to as *bintree location codes* and *subtree location codes*.

In the former case, each simplex is encoded in terms of a bintree traversal from one of the $d!$ roots of the simplex hierarchy [EKT01,LDS01,LDS04]. Thus, assuming that the two children of a simplex are labeled as child `0` and child `1`, the *bintree location code* of a simplex σ at depth m is encoded as the index $i_p \in [0,d!)$ of the bintree containing σ followed by an m-bit binary string indicating the traversal path from the root of bintree i_p to σ. Each such simplex can be indexed using $(m+\lceil \lg(d!) \rceil)$ bits. Note that, in this encoding, the vertices of a simplex are not stored, and can be determined from those of its bintree root in $O(m)$ time by descending the tree.

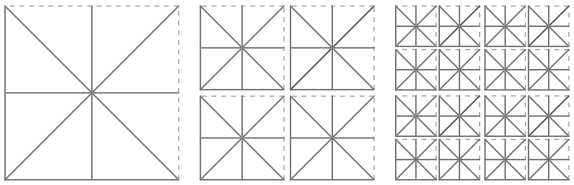
Alternatively, each simplex can be uniquely indexed with respect to a local *subtree* of simplices rooted at a single fully subdivided cube within the hierarchy (see Figure 17) [HK95, Heb94, Pas00, AM07]. Since each fully-subdivided cube corresponds to a hypercube generated during regular refinement, the *subtree location code* of a simplex combines its local index within the subtree with the octree-like location code [Gar82] of its fully subdivided hypercube.

Recall that a fully-subdivided d-cube $\mathcal{F}$ is defined by the 2^d Kuhn-subdivided cubes generated by reflecting a Kuhn

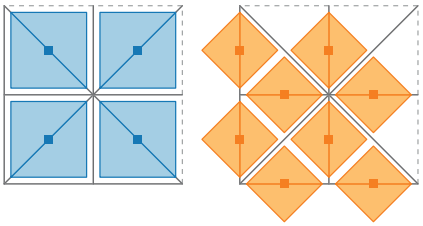


(a) Reflection i_r (b) Permutation i_p (c) Descendant i_d

Figure 17: Each RSB simplex σ within a *subtree* (shown in 2D) is uniquely indexed by (a) the *reflection number* i_r of its containing Kuhn cube $\mathcal{K}_\sigma$, (b) the *permutation number* i_p of its containing class 0 simplex σ_0 within $\mathcal{K}_\sigma$ and (c) the *descendant number* i_d of σ from σ_0.



(a) Supercubes tile each level of resolution



(b) Diamonds in a 2D supercube

Figure 18: 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 indexes four 0-diamonds (blue) and eight 1-diamonds (orange).

cube across the d axis-aligned planes. Then, each RSB simplex σ in the subtree corresponding to a fully subdivided cube $\mathcal{F}$ can be indexed in terms of three integers representing its containing Kuhn cube $\mathcal{K}_\sigma$ (see Figure 17a), its class-0 ancestor σ_0 within $\mathcal{K}_\sigma$ (see Figure 17b) and a traversal of the local bintree from σ_0 to σ (see Figure 17c). We refer to these three indexes as its *reflection number* $i_r \in [0, 2^d)$, its *permutation number* $i_p \in [0, d!)$ and its *descendant number* $i_d \in [0, 2^d - 1)$, respectively. Thus, each subtree uniquely indexes $2^d \cdot (2^d - 1) \cdot d!$ distinct RSB simplices. The *subtree location code* for an RSB simplex σ at level ℓ can be encoded using $(d \cdot \ell + d \cdot 2 + \lceil \lg(d!) \rceil)$ bits, where the first term encodes its octree-like location code.

By substituting the level ℓ and class i for the depth m of the simplex, i.e. $m = d \cdot \ell + i$, we observe that both types of location codes have the same complexity $O(d \cdot \ell + \lceil \lg(d!) \rceil)$. Furthermore, by substituting the maximum level of resolution N for ℓ, and using the identity $\lg(n!) = \Theta(n \lg n)$ [CLRS01], we can determine the space complexity of simplex-based location codes to be $O(dN + d \lg d)$ bits.

An advantage of *subtree location codes* over *bintree location codes* is that the vertices of a simplex can be directly computed without requiring a tree traversal [Heb94, AM07].

5.2. Encoding diamond hierarchies

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 within the hierarchy (up to scale).

Gregorski et al. [GDL*02] consider the oriented edge directions from the center of a fully subdivided 3-cube to one of its boundary vertices to define 26 oriented diamond types within a three-dimensional 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 spine 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 $\boldsymbol{\tau}$. To ensure that each edge in an RSB hierarchy is associated with only a single supercube $\mathbf{s}$, 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 18a 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 encodings of [Paj98, GDL*02] and the diamond decomposition of [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. Let

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

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* $\boldsymbol{\tau}$ and the supercube origin $\mathbf{s}$ of δ.

Let $\text{TRAILING}(\mathbf{x}_i)$ 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 γ 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* $\boldsymbol{\tau}$ of δ. Since each vertex has d coordinates, and $\boldsymbol{\tau}$ has two bits for each coordinate, there are $(2^2)^d = 4^d$ possible values for $\boldsymbol{\tau}$. 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 18b shows the four 0-diamonds and eight 1-diamonds indexed by a two-dimensional supercube.

The similarity *class* i of δ is encoded within $\boldsymbol{\tau}$ 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 [WD09a], 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.

The final $n = N - (\gamma+2)$ bits in each of the coordinates of $\mathbf{v}_c$ encode the *origin* of the supercube s containing δ. This corresponds to the lower corner (in all coordinates) of its corresponding fully subdivided cube $\mathcal{F}$ and is related to the octree-like location code [Gar82].

6. 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 *selection criterion*, at each node (i.e. simplex or diamond) to determine whether it should be refined or simplified. Typically, refinements are mandatory, while simplifications are optional. Thus, cells that do not satisfy μ *must* be refined, while cells that satisfy μ are merely *eligible* for simplification.

These predicates can be defined locally between a node and its children, or hierarchically, by considering properties of its descendants. The selection 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 6.1, and some notable application–specific selection criteria in Sections 7, 8 and 9.

Queries on a multiresolution model can be broadly classified into two categories [Mag99, DM02]:

Level of Detail (LOD) queries, which extract a mesh of minimum size covering the entire domain and satisfying a given selection 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 selection 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 in geographic data processing are concerned with the latter query type, where such queries are used, for example, to compute geometric overlays and in visibility [DM03] and range [Paj98] queries. Interestingly, the set of simplices returned by spatial selection queries is not a subset of those satisfying LOD queries for the same selection criterion [Mag99].

The result of an LOD query is a *closed* set of refinements 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 [Paj98]. Alternatively, the full resolution dataset can be coarsened in a *bottom-up* manner through a series of simplifications [LKR*96, ZCK97]. *Incremental* methods apply a series of refinements and simplifications to a previously extracted mesh [DWS*97, DL04].

For conforming refinements, all neighbors sharing the 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 simplices 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 simplex bisections.

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 [DM02]. Any set of diamonds

which is closed with respect to the partial order defines a conforming RSB mesh.

Adaptive refinement queries correspond to a traversal of the containment relation described by a hierarchy of simplices. In general, this is not sufficient to guarantee that extracted meshes are conforming. However, when applied to a hierarchy of simplices that is monotonic with respect to the given selection criterion μ (i.e. if a node fails μ, then all of its ancestors in the vertex dependency relation also fail μ), adaptive refinement will implicitly extract a conforming mesh.

Following a discussion of error metrics in Section 6.1, we review queries to a hierarchy of diamonds in Section 6.2. and to a hierarchy of simplices in Section 6.3 We compare the two approaches in terms of their spatial and computational complexity in Section 6.4.

6.1. 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. 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, an error metric is *saturated* if it is monotonic with respect to the diamond dependency relation, i.e. the error of a node is greater than those of its descendants [OR97, OR99, Paj98, GRW00]. A comprehensive survey of LOD error metrics can be found in [LRC*02].

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 underlying 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 [MV10].

Local error metrics. Since the only new vertex introduced during a refinement is the central vertex $\mathbf{v}_c$, which is located at the midpoint of the bisection edge $(\mathbf{v}_a + \mathbf{v}_b)/2$, 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

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

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 computed efficiently on the fly, and thus, do not require preprocessing or additional storage. 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, WD09b]

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

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 [ABJ05].

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

Saturated error metrics. An error metric (local or total) can be *saturated* by ensuring that the error associated with each cell is greater than or equal to those of its descendants, and that all simplices sharing the same bisection edge have the same error. Thus, saturated error metrics are monotonic with respect to the diamond dependency relation.

A possible saturated error metric for a simplex σ is

$$\varepsilon'(\sigma) = \max\left\{\varepsilon(\sigma), \max_{\sigma \in \delta}\left\{\varepsilon'(\textsc{Child}_0(\sigma)), \varepsilon'(\textsc{Child}_1(\sigma))\right\}\right\} \tag{6}$$

where $\varepsilon(\sigma)$ is a local error metric, and $\textsc{Child}_i(\sigma)$ are the two children of σ. Saturated error metrics typically require a bottom-up evaluation either while the metric is being evaluated or as a pre-process.

6.2. Querying a hierarchy of diamonds

Here, 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 non-root simplex in a diamond δ is generated during the refinement of one of δ's parents. Thus,

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 a selection 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$_\delta(\delta_p,$ **true**$)$
6: // *Bisect all simplices 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$_\delta(\delta_c,$ **false**$)$

a diamond δ in an RSB mesh Σ will contain all of its simplices only after all of its parents have refined. We refer to a diamond that contains all of its simplices as *complete*.

Since conforming refinements to Σ require all simplices 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 selection criterion μ. However, forced refinements short-circuit the selection criterion using the boolean `ForceRefine` (Line 1).

Refinement is carried out in REFINEDIAMOND (Line 7). Assuming that Σ is a diamond-based RSB mesh, this can be efficiently achieved by marking δ as refined, inserting all children of δ into Σ, and indicating in each child that the simplices due to δ are now present in Σ. Since the $O(d)$ parents of a diamond δ generate its $O(d!)$ simplices, all simplices due to a parent δ_p can be tracked using a single bit. REFINEDIAMOND requires $O(d)$ time when the class of δ is not $(d-1)$, and $O(2^d)$, otherwise. Finally, the children of diamonds that are not forcibly refined are checked for refinement (Lines 8–11).

Figure 19a illustrates the results of a selective refinement query on a 2D hierarchy of diamonds (left) as well as its corresponding (conforming) *current mesh* Σ_δ (right) after refining the root diamond (red), its north child (orange diamond farthest into the page), and its northeast child (green diamond in the upper right corner). All other refinements were *forced* to satisfy the transitive closure of the dependency relation. In the figure, nodes with outgoing arcs (left) have been refined, and their central vertices have been added to Σ_δ (right). Nodes that only have incoming arcs belong to the *active front* of the query, and therefore contribute triangles to Σ_δ.

Algorithm 2 ADAPTIVEREFINE(σ)

Require: σ is an RSB simplex in a nested RSB mesh Σ
Require: μ is a selection criterion
1: **if** $\mu(\sigma)$ fails **then**
2: BISECTSIMPLEX(σ)
3: ADAPTIVEREFINE$($CHILD$_0(\sigma))$
4: ADAPTIVEREFINE$($CHILD$_1(\sigma))$

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]. Gregorski et al. [GDL*02] describe an incremental selective refinement query for 3D diamonds based on a split and merge queue [DWS*97].

6.3. Querying a hierarchy of simplices

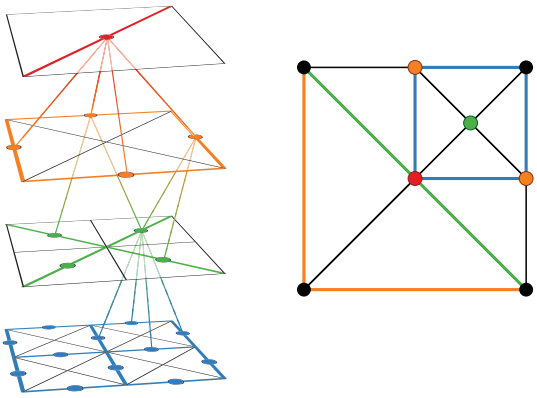
Here, we discuss the two approaches for extracting nested RSB meshes from a simplex hierarchy.

Adaptive refinement corresponds to a traversal of the bintrees describing the containment relation among simplices in a hierarchy of simplices (Algorithm 2), and is initialized with the $d!$ bintree roots. When a saturated error metric, is employed, the extracted mesh is guaranteed to be conforming [OR97, GRW00]. Otherwise, the extracted mesh is unlikely to be conforming.

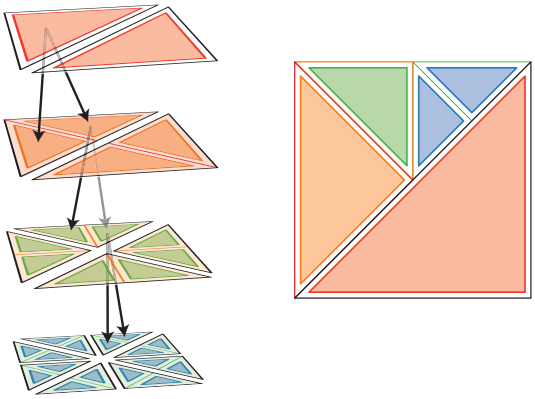
Figure 19b illustrates the results of an adaptive refinement query on a 2D hierarchy of triangles (left) as well as its corresponding *current mesh* Σ_{σ_a} (right), after three (unsaturated) bisections have been applied. Note that without forced refinements, Σ_{σ_a} is not conforming.

The second approach uses a selective refinement algorithm on the hierarchy of simplices, and ensures conforming refinements by concurrently refining all bisection-edge neighbors. This is carried out through a series of *neighbor-finding* operations that traverse all simplices in an RSB mesh that are incident to the bisection edge of a refining simplex.

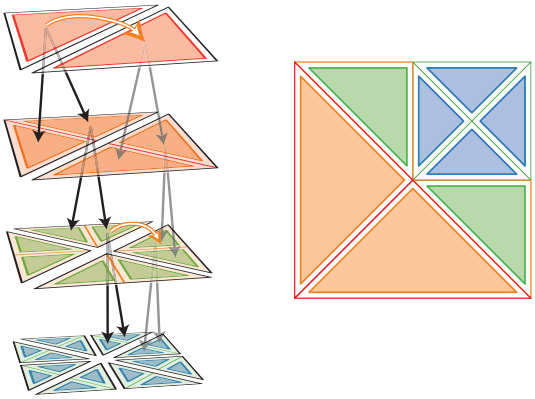
A useful property of conforming RSB meshes is that neighboring simplices can differ in depths by at most one [EKT01]. Specifically, the neighbors of a simplex σ in a conforming RSB mesh Σ sharing the bisection edge of σ can either be at the same depth, in which case, they belong to the same diamond,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



(a) Selective refinement on 2D hierarchy of diamonds



(b) Adaptive refinement on hierarchy of triangles



(c) Selective refinement on hierarchy of triangles

Figure 19: Status of queries on 2D RSB hierarchies (left) and their corresponding *current RSB meshes* (right). Nodes with outgoing arcs have been refined, while those with only incoming arcs belong to the *active front*. Selective refinement on a hierarchy of diamonds (a) and on a hierarchy of triangles (c) generates conforming RSB meshes, while adaptive refinement on a hierarchy of triangles (b) typically generates non-conforming meshes. Compare to Figure 14.

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 a selection criterion
1: **if** `ForceRefine` is true **or** $\mu(\sigma)$ fails **then**
2: // *Ensure all bisection 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$_0$(σ'), **false**)
12: SELECTIVEREFINE$_\sigma$(CHILD$_1$(σ'), **false**)

Thus, if the depth of a neighboring simplex σ' along the bisection edge $\mathbf{e}$ is less than that of σ, then it must be the parent of a same-depth neighbor of σ, and σ' must be recursively bisected using SELECTIVEREFINE. 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 a bintree root).

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.

Figure 19c illustrates the results of a selective refinement query on a 2D hierarchy of triangles (left) as well as its corresponding *current mesh* Σ_{σ_s} (right), after the same three bisections as Figure 19c have been applied. Since this is selective refinement, neighbor-finding operations were applied (orange arrows), and Σ_{σ_s} is conforming.

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. An incremental selective refinement algorithm based on implicit neighbor–finding is presented in [DL04].

Maubach extends his original pointer-based algorithm [Mau95] with an algebraic pointerless neighbor-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 (two or three) neighbors within the same subtree can be found in constant time, but the neighbors outside this cube can be found in time proportional to the node's 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 (similar to the method of Schrack for quadtrees and octrees [Sch92]). These algorithms are specific to 3D and 4D, respectively.

Atalay and Mount [AM07] combine the symbolic approach of Hebert [Heb94] with the efficient hardware-based neighbor-finding approach of Lee et al. [LDS01, LDS04] for simplex hierarchies of arbitrary dimension using a subtree location code. A lookup table is used to find neighbor simplices in adjacent subtrees, which enables each neighbor–finding operation to run in $O(\lg d)$ time.

6.4. Comparison

Since a d-dimensional RSB simplex has $O(d!)$ bisection edge neighbors, which must be checked before refinement, conforming refinements to a hierarchy of simplices requires $O(d!)$ time. More importantly, since 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 can be achieved using $O(d)$ time and spatial accesses [WD09a].

Saturated selection criteria can simplify the extraction algorithm by implicitly incorporating the diamond dependency relations. Since no dependencies need to be checked in the adaptive refinement algorithm (Algorithm 2), conforming meshes can be extracted in parallel through the use of a top-down saturated selection criterion [GR99, BPSC04, CGG*04, Mau05, GMBP10].

Interestingly, the standard generation approach for saturated error metrics (Equation 6) requires $O(d!)$ computation per diamond since the bisection edge errors of both children of $O(d!)$ simplices must be compared. However, the diamond decomposition of [WD09a] implies that when using a local error metric (e.g. Equation 3), only the $O(d)$ bisection edges corresponding to the diamond's children's spines need to be evaluated in the general case (and $O(2^d)$ evaluations are required when $i = (d-1)$).

Finally, we note that the saturation condition restricts the available types of selection criterion that can be applied since these must either be precomputed and explicitly stored for the entire dataset, or they apply to restricted subdomains of the dataset [Ger03b]. Furthermore, such saturated selection criteria typically generate larger meshes than their unsaturated counterparts, and thus require more bisections. De Floriani and Lee [DL04, Lee06] compare meshes extracted using saturated adaptive refinement queries against those extracted using unsaturated selective refinement queries with bintree-based pointerless neighbor-finding and a per-simplex total error. 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 amount of time as a saturated approach.

7. Applications in two dimensions

Much of the early research on efficient representations and queries for nested RSB meshes focused on 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 the regular grid [Ger03a, WD08c]. We review applications of triangle hierarchies to terrain in Section 7.1 and other applications in Section 7.2.

7.1. Interactive terrain rendering

Terrains are often presented as height fields where elevation values are 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 utilizing RSB hierarchies to optimize interactive rendering. In particular, since the viewpoint is typically located above the terrain, many samples at a great distance are visible but can be incredibly small. 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 by Pajarola and Gobbetti [PG07] presents a comprehensive overview of approaches for interactive rendering of regularly sampled terrain datasets, including those based on RSB hierarchies.

7.1.1. Hierarchy representation

Due to the regularity of the datasets, most approaches encode the elevation values as a linear array of $(2^N+1)^2$ samples, which can be 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 by using hierarchical space-filling curves with on-demand paging [LP02]. In this context, the latter have been shown to perform an order of magnitude faster than row-major indexing, which

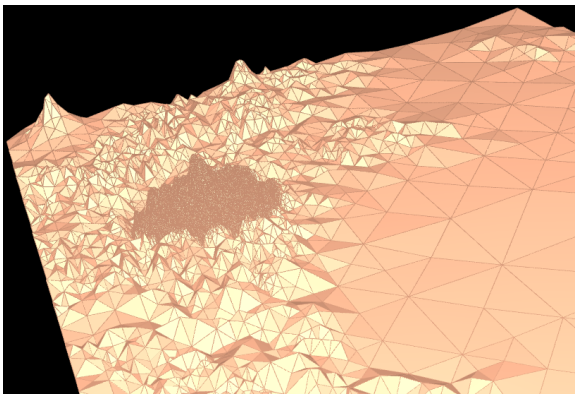


Figure 20: An RSB mesh extracted from the 1025^2 sample Puget Sound dataset using a circular *region of interest* query.

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 location codes [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, e.g. with pointers [PAL02, CGG*03a, HDJ05], can be modified to handle an incomplete hierarchy, there is a great deal of spatial and hierarchical coherence among the retained samples that can be exploited to yield significantly more efficient representations. Thus, these techniques are able to scale well when encoding the full hierarchy.

Gerstner's triangle-based approach [Ger03a] represents each of the hierarchy's binary trees as a linear array. 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 first clustered into their corresponding supercube (Section 5), and 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 [Ger03a, WD08c] support the local embedding of higher-resolution regions of data into an existing hierarchy.

7.1.2. Querying a terrain

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

Many of the earlier approaches utilize local error metrics (see Equation 3) [SS92, LKR*96, EKT01]. By saturating these local errors, global (but not necessarily tight) bounds on the approximation error can be obtained [DWS*97, Paj98, LP02, Ger03a].

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 refining and coarsening triangles. Although incremental selective refinement can be more difficult to implement, it exploits frame-to-frame coherence by initializing the mesh in subsequent frames from the one extracted in the previously frame.

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 volumes generated during a preprocess.

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

Gerstner [Ger03b] uses a diamond's octagonal descendant domain to define an optimally-tight nested bounding hierarchy of thickened octagonal prisms. He also shows how the

various independent saturated error metrics can be combined at runtime, to achieve dynamic refinements to the saturated selection criterion.

7.1.3. Clustered refinements

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]. This implies that for visualization purposes, bandwidth is more important than precise control over the output mesh.

In RUSTiC [Pom00] and CABTT [Lev02], each such 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. A similar approach can be found in [LPT03], which applies batched RSB updates to nodes in a balanced quadtree.

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 highly optimized (irregular) triangle patch with each bintree triangle in the hierarchy. 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 quadtree-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 [CGG*03a], with the regular clustering of RUSTiC [Pom00]. Additionally, they encode their textures as diamond hierarchies (rather than quadtrees), to better align the texture hierarchy with the elevation hierarchy. 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.

Gobbetti et al. [GMC*06] demonstrate that batch-updated diamond-based approaches can be competitive with the non-conforming state of the art grid-based approaches [LH04] in terms of compression rates and runtime performance. Lindstrom and Cohen [LC10] propose a batched update variable-rate GPU compression compression scheme for RSB meshes in which the batches have Freudenthal's triangulation. The generated mesh is conforming as long as boundary edges of adjacent triangles have the same number of vertices. The recent work by Goswami et al. [GMBP10] utilizes a batch-based hierarchy for parallel rendering of large datasets across multiple machines and displays.

7.2. Other applications

Another popular application of 2D RSB hierarchies has been for image processing. For example, Hebert and Kim [HK95, Heb98] present a wavelet-based image encoding based on a hierarchy of triangles.

Von Herzen [VH89] uses a hierarchy of triangles to compress colored texture maps. The refinement in this approach is driven by the difference in vertex color values. After extracting a conforming RSB mesh, neighboring triangles of constant color are merged (arbitrarily) to create a conforming polygonal mesh.

Tanaka [Tan95] generates adaptive decompositions of colored images as well as range-scanned shapes defined over a rectangular or cylindrical parameter domain. This method utilizes the octagonal descendant domains to define a curvature-driven parallel refinement algorithm over a coarse rectangular grid of Kuhn-subdivided squares. Similarly, Pajarola et al. [PSM04] utilize a 2D diamond hierarchy to interactively blend multiple depth images to generate a 3D surface.

Alternatively, it is possible to utilize the connectivity of an RSB mesh while modifying its geometry [VG00, PAL02]. This is the basis for the highly adaptive subdivision surface scheme of Velho and Zorin [VZ01], which guarantees the extraction of conforming meshes with greater smoothness than existing subdivision schemes (such as the Catmull-Clark subdivision [CC78]) while only doubling (rather than quadrupling) the number of cells involved in each refinement. This notion is formalized in [MV10], where other possible interpolation schemes are also discussed.

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.

8. 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 8.1) and to enable efficient extraction of variable resolution isosurfaces (Section 8.2). We also discuss other applications of nested RSB meshes including their use as spatial access structures (Section 8.3).

8.1. 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 refinements to the underlying RSB mesh along with an appropriate set of isosurface or interval volume triangulation cases within each tetrahedron [PT90, NS97] guarantee that extracted isosurfaces and interval volumes 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. Additionally, isosurfacing queries can be accelerated by maintaining the range of field values contained in the domain of each cell [WVG92, GDL*02]. This enables hierarchical culling of 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 simplify conforming isosurfaces. An interesting feature of their approach is that they incorporate topology preservation into their selection 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 simplex-based, they consider changes to isosurface 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 desired isovalue. Their saturated topology-based error metric encodes the range of all possible isovalues in which the topology of the extracted mesh could change.

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 noisy isosurfaces by reducing the genus of the extracted mesh. They propose a less conservative metric in follow-up work [PG05], where diamonds can encode multiple non-overlapping intervals in which the topology can change. This reduces the size of its extracted meshes but increases the storage requirements of the hierarchy.

Gerstner and Rumpf [GR99] accelerate the extraction of isosurfaces from a hierarchy of tetrahedra using a parallel adaptive refinement algorithm. They cull the backfaces of the isosurface through the use of a saturated view-dependent curvature-based error criterion, thereby 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 isosurfacing vertex program. He introduces a tetrahedral stripping operation, which generalizes the 2D Sierpinski space-filling curve used for triangle stripping on terrain datasets [Paj98, LP02].

Gregorski et al.'s diamond-based approach [GDL*02] utilizes ROAM's [DWS*97] incremental selective refinement 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 isovalue ranges within each diamond from 19 bytes to 4 bytes. Additionally, they avoid dealing with the domain boundaries by modeling the domain as a 3-torus of resolution $(2^N)^3$. 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]. In follow-up work, Gregorski et al. [GSDJ09] further accelerate their view-dependent isosurface extraction through the use of hardware compression and occlusion culling. They use the RSB 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 approximations 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 2D 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 the total approximation error of Section 6.1.

Alternatively, if only a subset of the field's range will be required, for example, in a client/server context, they propose a supercube-based *incomplete* representation of the hierarchy to retain only the closed set of samples whose diamonds intersect such isovalues. 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.

8.2. Multiresolution isosurfaces

The previous schemes operate by first extracting a variable-resolution RSB mesh Σ satisfying some selection criterion from an RSB hierarchy and then extracting an isosurface from Σ. An alternative approach, when a desired isosurface can be determined ahead of time, is to directly extract a multiresolution model of that isosurface from the hierarchy. This model can then be queried offline to extract variable-resolution isosurfaces satisfying a new selection criterion. Since such methods encode only a single isosurface, they cannot dynamically modify the isovalue, but can require significantly less space than the original scalar field.

Pascucci and Bajaj [PB00] propose a multiresolution isosurface model whose refinements are based on a small set of local primitives for updating the extracted isosurface mesh, which correspond to the changes to the isosurface during each conforming refinement to the RSB mesh. Once an approximated isosurface is extracted from the multiresolution isosurface, it can be smoothed using a set of subdivision masks. However, details of how the proposed multiresolution model is encoded or queried are not provided.

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 that of the previous one. This is done through an explicit correspondence between pairs of overlapping tetrahedra in successive depths of the hierarchy. 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,LLVM06] use the spatial decomposition of a 3D and a 2D hierarchy of simplices, respectively, 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 refinements, 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 compactly encoded in terms of refinements to their embedding diamonds. This implicit scheme encodes refinements corresponding to diamonds intersecting the interval volume as well as the non-intersected diamonds in their *ancestor domains* (see Section 4.3), requiring 14 bytes per retained diamond. 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 the *Isodiamond Hierarchy* framework [WD10b], which encompasses multiresolution isosurfaces as well as interval volumes. In addition to the above scheme which is dubbed the *Relevant Isodiamond Hierarchy*, a new *Minimal Isodiamond Hierarchy* is introduced that enables the extraction of conforming isosurfaces and interval volumes without encoding the (non-intersected) diamonds in their ancestor domains. This is accomplished through an analysis of the hierarchical connectivity of the embedded mesh, that ensures that extracted isosurfaces or interval volumes are conforming even if the underlying RSB mesh (i.e. the tubular neighborhood) is not conforming.

8.3. Other applications

Although volume visualization has been the predominant application of nested tetrahedral RSB meshes, other researchers have focused on their use as adaptive partitions of a volumetric 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 bisection operations. They generate intensity values at the vertices of the adaptive RSB 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 algorithm applies principal 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 an adaptive decomposition. 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] for view-dependent rendering of extremely large triangle meshes. It uses a hierarchy of tetrahedra as a spatial index over patches of (irregular) triangles from an underlying surface. To construct the hierarchy, it first builds 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 explicit (i.e. pointer-based) multiresolution triangle mesh is then 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 its triangles. 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 adaptive volumetric weather data representing atmospheric rainfall on an adaptive 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.

9. Applications in higher dimensions

In this section, we review approaches to nested RSB meshes in four and higher dimensions. 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 can be found using point location queries and thus, a conforming decomposition of the entire domain is not typically required.

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].

Weigle and Banks [WB96, WB98] propose an RSB triangulation for uniform hypercubic grids. In this scheme, each hypercube is recursively decomposed into $(2^{d-1} \cdot d!)$ RSB simplices of class $(d-1)$ through a recursive procedure that adds vertices at the midpoint of all faces, and connects edges from the higher-dimensional face midpoints to the lower-dimensional ones. These simplicial complexes are then used to visualize contours on the complex plane as well as time-varying isosurfaces.

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 batched refinement approach within each diamond to reduce the granularity of each refinement and to accelerate hardware rendering of isosurfaces (similar to RUSTiC [Pom00] and related approaches in 2D, see Section 7.1.3).

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.

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].

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 are achieved by applying a lazy neighbor-finding algorithm to locally patch cracks [Ata04]. 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.

10. Concluding remarks

The popularity of RSB-based approaches stems from the need for high quality adaptive simplicial decompositions of regularly sampled domains coupled with the flexibility of its refinement algorithms. Compared to models based on regular refinement over a regularly sampled domain, and to (triangulated) balanced d-dimensional octrees, 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.

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 simplices 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 a diamond-based representation. These benefits increase with the dimensionality of the domain since the complexity of many operations, such as selective refinement, is reduced from $O(d!)$ to $O(d)$ when using diamond hierarchies.

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

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 without the runtime costs associated with enforcing closure refinements.

The on-demand conforming refinement algorithm of [Ata04] 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 storage overhead and computation times. An interesting question relates to the overhead associated with converting a nonconforming RSB mesh into a conforming one. In 2D, 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.

Of similar interest, is the relationship between the family of RSB meshes extractable from a diamond hierarchy (i.e. conforming RSB meshes) and the family of RSB meshes generated by triangulating balanced d-dimensional octrees [VHB87, SS92, GB99, WD10a]. De Floriani and Magillo [DM02] demonstrate that RSB hierarchies have a higher representational power than triangulated octrees, i.e. there are conforming RSB meshes that cannot be generated from triangulated d-dimensional octrees. Empirically, diamond meshes have been found to be more adaptive than triangulated octrees, by a factor of 3.5 in 2D [VH89] and a factor of 2.5 in 3D [WD10a], but theoretical bounds remain an open question as well.

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

[ABJ05] ANDERSON J., BENNETT J., JOY K.: Marching diamonds for unstructured meshes. In *Proceedings IEEE Visualization* (Minneapolis, MN., Oct. 2005), IEEE, pp. 423–429. 13

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

[AG03] ALLGOWER E., GEORG K.: *Introduction to numerical continuation methods*, vol. 45 of *Classics in applied mathematics*. Society for Industrial and Applied Mathematics, 2003. 2

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

[AM06] ATALAY F., MOUNT D.: The cost of compatible refinement of simplex decomposition trees. In *Proc. International Meshing Roundtable* (Berlin Heidelberg, 2006), Pébay P. P., (Ed.), Springer, pp. 57–69. 22

[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. 2, 10, 11, 16, 21, 22

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

[Ata04] ATALAY F.: *Spatial Decompositions for Geometric Interpolation and Efficient Rendering*. PhD thesis, University of Maryland, College Park, 2004. 21, 22

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

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

[BAV98] BALMELLI L., AYER S., VETTERLI M.: Efficient algorithms for embedded rendering of terrain models. In *Proceeding International Conference on Image Processing (ICIP)* (Chicago, IL , USA, Oct. 1998), vol. 2, IEEE, pp. 914–918. 9

[Ben75] BENTLEY J. L.: Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 9 (1975), 509–517. 4

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

[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, 5

[Blo00] BLOW J.: Terrain rendering at high levels of detail. In *Proceedings of the Game Developers Conference* (San Jose, CA, 2000). 17

[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. 16, 20

[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

[CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (1978), 350–355. 18

[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, 17, 18, 21

[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. 16, 18

[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, 16, 21

[CLRS01] CORMEN T., LEISERSON C., RIVEST R., STEIN C.: *Introduction to algorithms*. The MIT press, 2001. 11

[CNS*06] CASTELO A., NONATO L., SIQUEIRA M., MINGHIM R., TAVARES G.: The J_1^a triangulation: An adaptive triangulation in any dimension. *Computers & Graphics 30*, 5 (2006), 737–753. 6

[CQ06] CHANG Y., QIN H.: A unified subdivision approach for multi-dimensional non-manifold modeling. *Computer Aided Design 38*, 7 (2006), 770–785. 3

[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.), Mathematics and Visualization. Springer Verlag, Berlin Heidelberg, 2004, pp. 329–344. 12, 15, 16

[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

[DM02] DE FLORIANI L., MAGILLO P.: Multiresolution mesh representation: Models and data structures. In *Principles of Multi-resolution Geometric Modeling*, Floater M., Iske A., Quak E., (Eds.), Lecture Notes in Mathematics. Springer Verlag, Berlin, 2002, pp. 364–418. 2, 6, 9, 12, 22

[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. 12

[DMP99] DE FLORIANI L., MAGILLO P., PUPPO E.: Multiresolution representation of shapes based on cell complexes. In *Discrete Geometry for Computer Imagery*, Bertrand G., Couprie M., Perroton L., (Eds.), vol. 1568 of *Lecture Notes in Computer Science*. Springer Verlag, New York, 1999, pp. 3–18. 9

[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. 16

[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, Berlin Heidelberg, 1997, pp. 302–323. 9

[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* (Los Alamitos, CA, USA, October 1997), Yagel R., Hagen H., (Eds.), VIS '97, IEEE Computer Society, pp. 81–88. 6, 8, 12, 14, 15, 16, 17, 19

[EKT01] EVANS W., KIRKPATRICK D., TOWNSEND G.: Right-triangulated irregular networks. *Algorithmica 30*, 2 (2001), 264–286. 10, 14, 16, 17, 20

[ELZ02] EDELSBRUNNER H., LETSCHER D., ZOMORODIAN A.: Topological persistence and simplification. *Discrete and Computational Geometry 28*, 4 (2002), 511–533. 18

[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), VIS '95, IEEE Computer Society, pp. 151–158. 19

[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. 21

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

[Gar82] GARGANTINI I.: An effective way to represent quadtrees. *Communications of the ACM 25*, 12 (December 1982), 905–910. 10, 12

[GB99] GREAVES D. M., BORTHWICK A. G. L.: Hierarchical tree-based finite element mesh generation. *Int'l Journal for Numerical Methods in Engineering 45*, 4 (1999), 447–471. 1, 5, 22

[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), VIS '02, IEEE Computer Society Washington, DC, USA, pp. 475–484. 6, 8, 9, 11, 13, 14, 19, 21, 22

[Ger02] GERSTNER T.: Multiresolution extraction and rendering of transparent isosurfaces. *Computers & Graphics 26*, 2 (2002), 219–228. 19, 21

[Ger03a] GERSTNER T.: Multi-resolution visualization and compression of global topographic data. *GeoInformatica 7*, 1 (2003), 7–32. 16, 17

[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, 17

[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*, 6 (2000), 357–369. 3

[GMBP10] GOSWAMI P., MAKHINYA M., BÖSCH J., PAJAROLA R.: Scalable parallel out-of-core terrain rendering. In *Proceedings Eurographics Symposium on Parallel Graphics and Visualization* (2010), pp. 63–71. 16, 18

[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* (October 2002), Pfister H., Bailey M., (Eds.), VIS '02, IEEE Computer Society Washington, DC, USA, IEEE Computer Society Press, pp. 533–536. 21

[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. 16, 18

[GP00] GERSTNER T., PAJAROLA R.: Topology-preserving and controlled topology simplifying multi-resolution isosurface extraction. In *Proceedings IEEE Visualization* (2000), VIS '00, IEEE, pp. 259–266. 6, 19

[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. 16, 19

[GRW00] GERSTNER T., RUMPF M., WEIKARD U.: Error indicators for multilevel visualization and computing on nested grids. *Computers & Graphics 24*, 3 (2000), 363–373. 13, 14

[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. 21

[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*, Farin G., Hege H.-C., Hoffman D., Johnson C., Polthier K., (Eds.), Mathematics and Visualization. Springer, Berlin Heidelberg, 2009, pp. 303–323. 19

[Guo95] GUO B.: Interval set: A volume rendering technique generalizing isosurface extraction. In *Proceedings IEEE Visualization* (1995), VIS '95, IEEE Computer Society Washington, DC, USA, pp. 3–10. 19

[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. 16, 17, 18

[Heb94] HEBERT D.: Symbolic local refinement of tetrahedral grids. *Journal of Symbolic Computation 17*, 5 (May 1994), 457–472. 2, 10, 11, 15, 16

[Heb98] HEBERT D.: Cyclic interlaced quadtree algorithms for quincunx multiresolution. *Journal of Algorithms 27*, 1 (1998), 97–128. 2, 18

[HK95] HEBERT D. J., KIM H.: Image encoding with triangulation wavelets. In *Proceedings SPIE* (1995), vol. 2569, pp. 381–392. 2, 10, 18

[Kos94] KOSSACZKÝ I.: A recursive approach to local mesh refinement in two and three dimensions. *Journal of Computational and Applied Mathematics 55*, 3 (1994), 275–288. 4

[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, 21

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

[LC10] LINDSTROM P., COHEN J. D.: On-the-fly decompression and rendering of multiresolution terrain. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), I3D '10, ACM, pp. 65–73. 18

[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. 10, 16

[LDS04] LEE M., DE FLORIANI L., SAMET H.: Constant-time navigation in four-dimensional nested simplicial meshes. In *Proceedings Shape Modeling International 2004* (Genova, Italy, June 2004), IEEE Computer Society, pp. 221–230. 7, 10, 16, 21

[Lee06] LEE M.: *Spatial Modeling using Triangular, Tetrahedral and Pentatopic Decompositions*. PhD thesis, The University of Maryland, College Park, 2006. 16, 21, 22

[Lev02] LEVENBERG J.: Fast view-dependent level-of-detail rendering using cached geometry. In *Proceedings IEEE Visualization* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 259–266. 18

[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. 19, 21

[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. 18

[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. 19

[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), SIGGRAPH '96, pp. 109–118. 9, 12, 16, 17

[LLVM06] LEWINER T., LOPES H., VELHO L., MELLO V.: Extraction and compression of hierarchical isocontours from image data. *Computerized Medical Imaging and Graphics 30*, 4 (2006), 231–242. Medical Imaging and Graphics in SIBGRAPI/SIACG. 20

[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. 16, 17, 19

[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. 21

[LPT03] LARIO R., PAJAROLA R., TIRADO F.: Hyperblock-QuadTIN: Hyper-block quadtree based triangulated irregular networks. In *Proceedings IASTED International Conference on Visualization, Imaging and Image Processing (VIIP)* (2003), pp. 733–738. 18

[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, 13, 15, 16, 20

[LS00] LEE M., SAMET H.: Navigating through triangle meshes implemented as linear quadtrees. *ACM Transactions on Graphics 19*, 2 (April 2000), 79–121. 5

[LVLM04] LEWINER T., VELHO L., LOPES H., MELLO V.: Simplicial isosurface compression. In *Vision, Modeling, and Visualization* (Stanford, CA, November 2004), pp. 299–306. 20

[Mag99] MAGILLO P.: *Spatial Operations on Multiresolution Cell Complexes*. PhD thesis, Dept. of Computer and Information Sciences, University of Genova (Italy), 1999. 12

[Mar76] MARA P.: Triangulations for the cube. *Journal of Combinatorial Theory, Ser. A 20*, 2 (1976), 170–177. 5

[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, 6, 15

[Mau96] MAUBACH J. M.: The efficient location of neighbors for locally refined n-simplicial grids. In *Proceedings International Meshing Roundable* (Oct. 1996), no. 5 in IMR '96, Sandia National Laboratories, pp. 137–156. 4, 5, 15

[Mau05] MAUBACH J.: Space-filling curves for 2-simplicial meshes created with bisections and reflections. *Applications of Mathematics 50*, 3 (2005), 309–321. 16

[MDM04] MARCHESIN S., DISCHLER J., MONGENET C.: 3D ROAM for scalable volume visualization. In *IEEE Symposium on Volume Visualization and Graphics* (Oct. 11–12 2004), VV '04, IEEE, pp. 79–86. 2, 13, 19, 22

[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

[MV10] MELLO V., VELHO L.: Simplicial diffeomorphisms. *Computer Aided Geometric Design 27*, 9 (2010), 734–745. 13, 18

[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), SMA '03, pp. 108–114. 2, 21

[MW95] MOORE D., WARREN J.: Adaptive simplicial mesh quadtrees. *Houston J. Math 21*, 3 (1995), 525–540. 3, 5

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

[NS97] NIELSON G., SUNG J.: Interval volume tetrahedralization. In *Proceedings IEEE Visualization* (1997), VIS '97, pp. 221–228. 19

[NSV09] NOCHETTO R., SIEBERT K., VEESER A.: Theory of adaptive finite element methods: An introduction. In *Multiscale, Nonlinear and Adaptive Approximation*, DeVore R., Kunoth A., (Eds.). Springer, Berlin Heidelberg, 2009, pp. 409–542. Dedicated to Wolfgang Dahmen on the Occasion of his 60th Birthday. 4

[OR97] OHLBERGER M., RUMPF M.: Hierarchical and adaptive visualization on nested grids. *Computing 56*, 4 (1997), 365–385. 13, 14

[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. 13

[Paj98] PAJAROLA R.: Large scale terrain visualization using the restricted quadtree triangulation. In *Proceedings IEEE Visualization* (Research Triangle Park, NC, Oct. 1998), Ebert D., Hagen H., Rushmeier H., (Eds.), VIS '98, IEEE Computer Society, pp. 19–26. 9, 11, 12, 13, 16, 17, 19

[PAL02] PAJAROLA R., ANTONIJUAN M., LARIO R.: QuadTIN: Quadtree based triangulated irregular networks. In *Proceedings IEEE Visualization* (2002), IEEE Computer Society, pp. 395–402. 17, 18

[Pas00] PASCUCCI V.: *Multi-dimensional and multi-resolution geometric data-structures for scientific visualization*. PhD thesis, Purdue University, West Lafayette, IN, USA, 2000. Major Professor-Bajaj, Chandrajit L. 10

[Pas02] PASCUCCI V.: Slow Growing Subdivision (SGS) in any dimension: Towards removing the curse of dimensionality. *Computer Graphics Forum 21*, 3 (September 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)* (May 2004), Deussen O., Hansen C., Keim D., Saupe D., (Eds.), VisSym '04, Eurographics Association, pp. 293–300. 19

[PB00] PASCUCCI V., BAJAJ C. L.: Time-critical isosurface refinement and smoothing. In *Proceedings IEEE Symposium on Volume Visualization* (Oct. 2000), VolViz '00, IEEE Computer Society, pp. 33–42. 20

[PC00] PLAZA A., CAREY G.: Local refinement of simplicial grids based on the skeleton. *Applied Numerical Mathematics 32*, 2 (2000), 195–218. 4

[PF01] PASCUCCI V., FRANK R. J.: Global static indexing for real-time exploration of very large regular grids. In *Proceedings ACM/IEEE Supercomputing* (2001), SC '01, pp. 45–45. 19

[PG05] PAJAROLA R., GERSTNER T.: *Topology Control in Multiresolution Isosurface Extraction*. Tech. Rep. UCI-ICS-05-01, Department of Computer Science,University of California Irvine, 2005. 19

[PG07] PAJAROLA R., GOBBETTI E.: Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer 23*, 8 (2007), 583–605. 16

[Pom00] POMERANZ A.: *ROAM using surface triangle clusters (RUSTiC)*. Master's thesis, U.C. Davis, 2000. 18, 21

[PP09] PUPPO E., PANOZZO D.: RGB subdivision. *IEEE Transactions on Visualization and Computer Graphics 15*, 2 (2009), 295–310. 3

[PSM04] PAJAROLA R., SAINZ M., MENG Y.: DMesh: Fast depth-image meshing and warping. *International Journal of Image and Graphics 4*, 4 (2004), 653–681. 18

[PT90] PAYNE B., TOGA A.: Surface mapping brain function on 3D models. *Computer Graphics and Applications, IEEE 10*, 5 (Sept. 1990), 33–41. 19

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

[RHSS98] RÖTTGER S., HEIDRICH W., SLUSALLEK P., SEIDEL H.: Real-time generation of continuous levels of detail for height fields. In *Proceedings Central Europe Winter School of Computer Graphics (WSCG)* (Feb. 1998), WSCG '98, pp. 315–322. 5

[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, 6

[RN00] ROXBOROUGH T., NIELSON G.: Tetrahedron-based, least-squares, progressive volume models with application to freehand ultrasound data. In *Proceedings IEEE Visualization* (Oct. 2000), VIS '00, IEEE Computer Society, pp. 93–100. 20

[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, 11

[Sch92] SCHRACK G.: Finding neighbors of equal size in linear quadtrees and octrees in constant time. *CVGIP: Image Understanding 55*, 3 (May 1992), 221–230. 10, 16

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

[Sew79] SEWELL E.: A finite element program with automatic user-controlled mesh grading. In *Advances in Computer Methods for Partial Differential Equations III*, Vichnevetsky R., Stepleman R., (Eds.). IMACS, June 1979, pp. 8–10. 4

[SS92] SIVAN R., SAMET H.: Algorithms for constructing quadtree surface maps. In *Proc. Symposium on Spatial Data Handling* (1992), pp. 361–370. 1, 5, 17, 22

[Ste08] STEVENSON R.: The completion of locally refined simplicial partitions created by bisection. *Mathematics of Computation 77*, 261 (2008), 227–242. 4

[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. 9, 18

[Tod76] TODD M.: *The computation of fixed points and applications*. No. 124 in Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, 1976. 5, 6, 21

[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), GMP '04, pp. 227–236. 19

[TTW03] TANAKA H., TAKAMA Y., WAKABAYASHI H.: Accuracy-based sampling and reconstruction with adaptive grid for parallel hierarchical tetrahedrization. In *Proceedings Volume Graphics* (2003), ACM Press, pp. 79–86. 9, 21

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

[Vel01] VELHO L.: Using semi-regular 4-8 meshes for subdivision surfaces. *Journal of Graphics Tools 5*, 3 (2001), 35–47. 10

[VG00] VELHO L., GOMES J.: Variable resolution 4-k meshes: concepts and applications. *Computer Graphics Forum 19*, 4 (2000), 195–212. 18

[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, 18, 22

[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. 1, 5, 22

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

[WB96] WEIGLE C., BANKS D.: Complex-valued contour meshing. In *Proceedings IEEE Visualization* (Oct. 1996), VIS '96, IEEE Computer Society, pp. 173–180. 21

[WB98] WEIGLE C., BANKS D.: Extracting iso-valued features in 4-dimensional scalar fields. In *Proceedings IEEE Visualization* (Oct. 1998), VolVis '98, IEEE Computer Society, pp. 103–110. 21

[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. 18

[WD08a] WEISS K., DE FLORIANI L.: Modeling and visualization approaches for time-varying volumetric data. In *Advances in Visual Computing*, vol. 5359 of *Lecture Notes in Computer Science*. Springer, 2008, pp. 1000–1010. 21

[WD08b] WEISS K., DE FLORIANI L.: Multiresolution interval volume meshes. In *IEEE/ EG Symposium on Volume and Point-Based Graphics* (2008), Eurographics Association, pp. 65–72. 20

[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. 11, 16, 17, 20

[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, 11, 12, 14, 16, 22

[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. 11, 13, 20

[WD10a] WEISS K., DE FLORIANI L.: Bisection-based triangulations of nested hypercubic meshes. In *Proceedings 19th International Meshing Roundtable* (October 3–6 2010), Shontz S., (Ed.), IMR '10, pp. 315–333. 1, 6, 22

[WD10b] WEISS K., DE FLORIANI L.: Isodiamond hierarchies: An efficient multiresolution representation for isosurfaces and interval volumes. *IEEE Transactions on Visualization and Computer Graphics 16*, 4 (July-Aug. 2010), 583 – 598. 20

[WD10c] WEISS K., DE FLORIANI L.: Nested refinement domains for tetrahedral and diamond hierarchies. In *IEEE Visualization 2010 Poster Compendium* (2010). 10

[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. 19

[ZCK97] ZHOU Y., CHEN B., KAUFMAN A.: Multiresolution tetrahedral framework for visualizing regular volume data. In *Proceedings IEEE Visualization* (Oct. 1997), Yagel R., Hagen H., (Eds.), VIS '97, IEEE Computer Society, pp. 135–142. 6, 12, 19

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

[Zon05] ZONG C.: What is known about unit cubes. *Bulletin of the American Mathematical Society 42*, 2 (2005), 181–211. 5