Isodiamond Hierarchies: An Efficient Multiresolution Representation for Isosurfaces and Interval Volumes

Kenneth Weiss *Student Member, IEEE* and Leila De Floriani *Member, IEEE*

Abstract—Efficient multiresolution representations for isosurfaces and interval volumes are becoming increasingly important as the gap between volume data sizes and processing speed continues to widen. Our multiresolution scalar field model is a hierarchy of tetrahedral clusters generated by longest edge bisection, that we call a *hierarchy of diamonds*. We propose two multiresolution models for representing isosurfaces, or interval volumes, extracted from a hierarchy of diamonds which exploit its regular structure. These models are defined by subsets of diamonds in the hierarchy, that we call *isodiamonds*, which are enhanced with geometric and topological information for encoding the relation between the isosurface, or interval volume, and the diamond itself. The first multiresolution model, called a *relevant isodiamond hierarchy*, encodes the isodiamonds intersected by the isosurface, or interval volume, as well as their non-intersected ancestors, while the second model, called a *minimal isodiamond hierarchy*, encodes only the intersected isodiamonds. Since both models operate directly on the extracted isosurface or interval volume, they require significantly less memory and support faster selective refinement queries than the original multiresolution scalar field, but do not support dynamic isovalue modifications. Moreover, since a minimal isodiamond hierarchy only encodes intersected isodiamonds, its extracted meshes require significantly less memory than those extracted from a relevant isodiamond hierarchy. We demonstrate the compactness of isodiamond hierarchies by comparing them to an indexed representation of the mesh at full resolution.

Index Terms—Isosurfaces, interval volumes, multiresolution models, longest edge bisection, diamond hierarchies.

1 Introduction

The availability of very large meshes describing free-form shapes, terrains, and volume data sets, has led to the investigation of hierarchical methods to control and adjust the *level of detail (LOD)* in the representation of a given data set. Multiresolution models provide a great deal of flexibility since they compactly encode a large number of different mesh-based representations of a shape, or of a field, and enable the efficient extraction of a variety of different representations at uniform or variable resolutions.

Numerous multiresolution models have been proposed for regular volume datasets where the regularity of the sampling can be exploited to yield an efficient representation. Models based on nested cubes, such as *octrees*, are popular since their cells are aligned with the data samples. However, models based on nested tetrahedra enable extraction of meshes that are adapt better to a user defined extraction criterion [1].

Volumetric datasets are often analyzed through their surfaces of constant field value, known as *isosurfaces*, or through the subvolumes enclosed by two isosurfaces, known as *interval volumes*. These structures are based on a limited range of the field values and typically occupy a sparse, but spatially widespread, subset of the dataset. Multiresolution representations are useful to analyze and visualize isosurfaces and interval volumes, since at full resolution both can contain from millions to billions of triangles or tetrahedra. Furthermore, in scientific and medical applications, details at the highest available resolution are required on demand, and thus, simplified approximations of these datasets are not sufficient.

To this aim, we introduce two multiresolution models for encoding isosurfaces or interval volumes extracted from regularly sampled volume data. The basic ingredients of a mesh-based multiresolution model of a shape are a coarse representation of the shape, a collection of refinement modifications to the coarse mesh, and a dependency relation among such modifications which allow extracting correct simplified shape representations. The multiresolution models for isosurfaces and interval volumes that we propose are based on a nested tetrahedral decomposition of the cubic domain containing the data points, called a *hierarchy of diamonds*, which forms a multiresolution model of the 3D scalar field defined by the sampled volume data. The basic idea here is to exploit the regular nested decomposition of the domain to produce compact hierarchical representations of a single isosurface, or a single interval volume. Since the desired isosurface, or interval volume is known, these representations are able to decouple the implicit hierarchical and geometric relationships in the hierarchy of diamond from the field values of the volume dataset. Thus, such models enable a compact encoding of modifications to the irregular isosurface, or interval volume, in terms of regular modifications to the corresponding nodes of the hierarchy of diamonds.

The first multiresolution model that we propose for an isosurface, or an interval volume, which we call a *Relevant Isodiamond (RI) hierarchy*, encodes a collection of *active* modifications to the coarse representation, corresponding to the cells in the hierarchy of diamonds intersected by the isosurface, or interval volume, respectively. An RI hierarchy also encodes all *relevant* modifications, which are ancestors of active modifications that are not intersected by the isosurface, or the interval volume. Relevant modifications enable spatial access to the active modifications and guarantee that all the meshes extracted from the

- *K. Weiss is with the Department of Computer Science, University of Maryland, College Park, 4406 A.V. Williams Building, College Park, MD 20742. E-mail: kweiss@cs.umd.edu.*
- *L. De Floriani is with the Dipartimento di Informatica e Scienze dell´Informazione, Università di Genova, Via Dodecaneso, 35, 16146 Genova, Italy. E-mail: deflo@disi.unige.it.*

RI hierarchy are free of cracks.

The second multiresolution model that we propose, which we call a *Minimal Isodiamond (MI) hierarchy*, encodes the active modifications as well as a small subset of the relevant modifications corresponding to the creation of new isosurface, or interval volume, components. As a consequence, this model requires a careful analysis to guarantee that the extracted meshes do not self-intersect. In addition to the reduced storage requirements, we show that the same uniform- and variable-resolution representation of the encoded isosurface, or interval volume, can be extracted from the MI hierarchy as from the RI model, but in less time, and using less memory.

Isodiamond hierarchies extend our previous work on multiresolution interval volume meshes [2], in which the multiresolution model was equivalent to the relevant isodiamond hierarchy encoding an interval volume. Here, we first unify the mesh-based extraction framework and generalize the method to isosurfaces as well as interval volumes. Moreover, we introduce the smaller and faster minimal isodiamond model which only represents modifications that increase the number of cells in the extracted mesh. Both representations support efficient general selective refinement operations to extract crack-free meshes at different resolutions from the model using application-dependent extraction criteria.

The remainder of this paper is organized as follows. Section 2 provides some background concepts that we will use throughout the paper. In Section 3, we review related work. The properties of diamond hierarchies are discussed in Section 4, which forms the basis for our multiresolution isodiamond model introduced in Section 5. In Sections 6 and 7, we discuss the relevant isodiamond and minimal isodiamond hierarchies, respectively. We present our experimental results in Section 8, and draw some concluding remarks in Section 9.

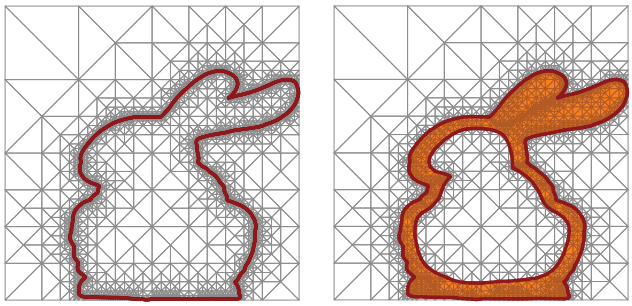
2 Background Notions

In this Section, we introduce some background notions on isosurfaces and interval volumes, on nested meshes generated through longest edge bisection, and on multiresolution mesh-based models, that we will use in the remainder of the paper.

2.1 Isosurfaces and interval volumes

A three-dimensional scalar field is given as a discrete set of points V in a domain $\Omega \subset \mathbb{R}^3$, where a scalar value $F(\mathbf{v})$ is associated with each point $\mathbf{v} \in V$. The points of V, along with their associated field values, form a *volume data set*, which is modeled as a polyhedral mesh. The vertices of the mesh are located at the data points and an interpolating function is defined over the cells of the mesh. Here, we consider models of a volume data set based on tetrahedral meshes defined over regularly sampled rectilinear grids, in which linear interpolation is used over the tetrahedra forming the mesh. We use *conforming* tetrahedral meshes, in which the intersection of any two tetrahedra is either empty or it is a triangle, edge or vertex belonging to the boundary of both of them. A model of a volume data set defined on a non-conforming mesh may contain cracks in correspondence to the boundary of adjacent cells of the mesh.

An isosurface of *isovalue* κ intersects all cells that have at least one vertex whose field value is greater than κ, and at least one vertex whose field value is less than κ. Such cells are considered to be *active* with respect to κ. Figure 1a shows an example of an isosurface extracted from a nested triangle mesh.



(a) Isosurface (b) Interval Volume

Fig. 1. Isosurface (a) and interval volume (b) extracted from a nested triangle mesh.

An interval volume of *isorange* $K := [\alpha, \beta]$ is bounded by two surfaces: the *lower surface*, corresponding to the isosurface of isovalue α, and the *upper surface*, corresponding to the isosurface of isovalue β. Thus, the interval volume passes through cells that intersect one, or both, isosurfaces, or that lie between the two surfaces. Such cells are considered to be *active* with respect to isorange K. Cells that are not active with respect to the chosen isovalue, or isorange, are considered to be *inactive*, or *empty*. Figure 1b shows an example of an interval volume extracted from a nested triangle mesh.

Each isovalue κ implicitly defines a binary-valued *sign field* B_κ on the vertices $\mathbf{v} \in V$ whose bits are set when corresponding scalar values of F are greater than κ, namely:

$$B_\kappa(\mathbf{v}) = \begin{cases} 1, & \text{if } \kappa > F(\mathbf{v}), \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, each isorange $K := [\alpha, \beta]$ implicitly defines a ternary-valued *sign field* R_K on the vertices $\mathbf{v} \in V$ whose values are determined by the relative values of F and K, namely:

$$R_K(\mathbf{v}) = \begin{cases} -1, & \text{if } F(\mathbf{v}) < \alpha, \\ 0, & \text{if } \alpha \leq F(\mathbf{v}) \leq \beta, \\ 1, & \text{if } \beta < F(\mathbf{v}). \end{cases}$$

Within each tetrahedron $\mathbf{t}$, the sign field, along with an appropriately defined lookup table, can be used to unambiguously triangulate the intersection of the isosurface (or interval volume) with $\mathbf{t}$. We call this intersection the isosurface (or interval volume) *patch* embedded within $\mathbf{t}$. Note that, when using linear interpolation on tetrahedral cells, the vertices of the patch are only generated along the *active edges* of $\mathbf{t}$.

2.2 Longest edge bisection meshes

A tetrahedral mesh, in which the tetrahedra are defined by the uniform subdivision of a tetrahedron into scaled copies, is called a *nested tetrahedral mesh*. A special class of nested tetrahedral meshes are those generated by bisecting tetrahedra along their unique longest edge, that we call *Longest-Edge-Bisection (LEB) meshes*. The *bisection rule* for a tetrahedron $\mathbf{t}$ consists of replacing $\mathbf{t}$ with the two tetrahedra obtained by bisecting $\mathbf{t}$ along the plane passing through the middle point of its longest edge $\mathbf{e}$ and the edge of $\mathbf{t}$ opposite to $\mathbf{e}$ (see Figure 2). This rule, when applied recursively to an initial decomposition of a cubic domain into six tetrahedra sharing a diagonal of

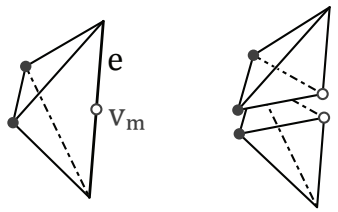


Fig. 2. A tetrahedron with longest edge **e** (left) is bisected (right) along the plane containing the midpoint $\mathbf{v}_m$ of **e** (hollow vertex) and the two vertices not adjacent to **e** (dark vertices).

the cube, generates a nested tetrahedral mesh composed of three similarity classes of tetrahedra that we call *0-tetrahedra*, *1-tetrahedra* and *2-tetrahedra*, respectively [3]. The unique longest edge of a 0-tetrahedron is aligned with the diagonal of an axis-aligned cube, that of a 1-tetrahedron is aligned with the diagonal of a face of an axis-aligned cube, and that of a 2-tetrahedron is aligned with an edge of an axis-aligned cube (see Figure 3 and compare to Figure 4).

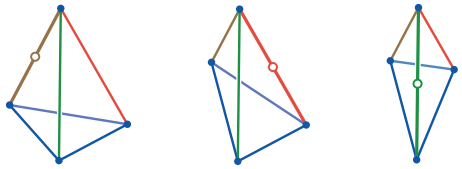


Fig. 3. The three similarity classes of tetrahedra generated by applying the longest edge bisection rule. The unique longest edge is aligned with (a) a cube diagonal (brown), (b) a diagonal of a face (red), or (c) an edge of an axis aligned cube (green).

2.3 Mesh-based multiresolution models

Multiresolution models based on the decomposition of a shape into a simplicial mesh compactly encode a large number of different mesh-based representations of the shape. In [1], [4], a general dimension-independent framework for mesh-based multiresolution representations has been proposed that has been shown to encompass the vast majority of multiresolution models developed in the literature. The three components of a mesh-based multiresolution model of a shape are:

- a coarse mesh Γ_b, that we call the *base mesh*,
- a set of *modifications* U, and
- a *dependency relation* R defined on the modifications in U.

Each modification specifies a local change to a mesh. It consists of replacing a subset Γ of its cells with another set of cells Γ' and is denoted as $u = (\Gamma, \Gamma')$. Each cell γ in Γ must either be part of the base mesh, or be created by exactly one modification in U. We are interested in modifications such that both Γ and Γ' are conforming meshes and the combinatorial boundary of Γ consists of the same set of cells as that of Γ'. Recall that, in a conforming mesh, the intersection between two cells is either empty or a lower dimensional cell on the boundary of both cells.

A *direct dependency* relation R is defined over the set U of all modifications as follows: a modification $u_1 = (\Gamma_1, \Gamma_1')$ directly *precedes* a modification $u_2 = (\Gamma_2, \Gamma_2')$ if and only if some cell of Γ_1' is also in Γ_2, i.e., if u_2 removes some cell inserted by u_1. The transitive closure of the dependency relation can be proven to be a partial order over set U [1]. The direct dependency relation can thus be described as a Directed Acyclic Graph (DAG), in which the nodes represent modifications and the arcs represent direct dependency links. We call this graph the *dependency graph* of the model.

Thus, a *multiresolution model* $M = (\Gamma_b, U, R)$ provides a compact way of encoding all conforming meshes that can be obtained by recursively applying the modifications in U to the base mesh Γ_b. Each such mesh is in one-to-one correspondence with a subset of U that is closed with respect to relation R. A subset U' of the modifications in U is called *closed* if, for each modification u in U', all predecessors of u with respect to the transitive closure of R belong to U'. The collection of all closed sets of modifications in a multiresolution model M defines the complete set of meshes that can be extracted from M. By applying all the modifications in U to the base mesh, we obtain the *mesh at full resolution* described by M.

Selective refinement is the basic query operation on a multiresolution model in terms of which all application-dependent queries can be expressed [5]. Selective refinement is the operation of extracting the smallest-size mesh from a multiresolution model based on a user specified selection criterion. This is equivalent to computing the closed set of modifications from U necessary to extract a mesh Σ of minimal size satisfying the specified criterion. The selection criterion is usually based on approximation error, field values, spatial location and/or distance to objects of interest such as the viewpoint.

3 Related Work

In this Section, we discuss methods related to isosurface and interval volume extraction with a specific focus on those operating on regularly sampled volume data sets. We also discuss hierarchical and multiresolution representations for isosurfaces and interval volumes extracted from volume datasets.

The original marching cubes (MC) algorithm [6] is a case-based divide-and-conquer approach over cubic cells, where isosurface vertices are positioned along *active* edges of the cubic grid, and isosurface *patches* are triangulated based on the configuration of the sign values of the scalar field at the vertices of each cell. Ambiguous configurations can be resolved through more extensive lookup tables [7], or by using tetrahedral cells [8]. An alternative set of intersection cases can be created for surfaces dual to those of the marching cubes, where a single vertex is generated within each active cell, and a single quad is generated for each intersected edge [9], [10].

Interval volumes were introduced concurrently by Fujishiro et al. [11] and Guo [12] as a bridge between direct and indirect volume rendering techniques. Fujishiro et al. [11], [13] extend the MC algorithm to extract interval volumes from cubic cells. Nielson and Sung [14] provide a globally consistent interval volume triangulation of tetrahedral cells, through a unique lexicographic ordering of the patch vertices within a cell. Alternatively, Bhaniramka et al. [15], [16] create interval volume intersection cases for convex cells by first extracting an isosurface from higher dimensional hyperprisms and then projecting the isosurface onto an interval volume in the cell. Zhang et al. [17] develop a dual interval volume technique where cells intersecting a surface are extracted using an adaptation of the dual contouring method [10] while internal cells are tetrahedralized by inserting Steiner points at their center.

Due to the size of extracted isosurfaces, there has been much research on accelerating the extraction process and on simplifying the resulting meshes. The Min-Max octree [18]

provides hierarchical access to active cells by aggregating field values within nested regions. Adaptive marching cubes algorithms [19], [20], [21] enable the extraction of variable-resolution isosurfaces, where the sizes of active cells are not required to be uniform. A specific concern when extracting an isosurface, or an interval volume, from a multiresolution scalar field is the extraction of conforming (crack-free) meshes. This is typically (although not always [21]) accomplished through restrictions on the decomposition of neighboring cells. Certain classes of volume data sets, such as distance fields, admit an efficient adaptive representation of the regularly sampled scalar field [22].

Conforming tetrahedral meshes do not suffer from the cracking problem but such meshes contain more cells than hexahedral meshes. Some authors have been using hierarchies of tetrahedra that encode the nested structure of LEB meshes as a multiresolution representation of a 3D scalar field [23], [24], [25], [26]. Gerstner et al. [24], [25] propose a top-down isosurface extraction algorithm from a hierarchy of tetrahedra, in which the extracted mesh is maintained conforming through the use of a conservative error saturation criterion. For the same problem, Lee et al. [26] propose an $O(1)$ neighbor-finding algorithm which allows tetrahedra adjacent through their longest edge to be processed together. An alternative approach to ensure the extraction of conforming meshes is to consider a multiresolution model of an LEB mesh in which the modifications are clusters of tetrahedra, called *diamonds*, sharing their longest edge [1], [27]. Gregorski et al. [27] propose an efficient data structure for such a model and extend the dual-queue refinement and coarsening approach of Duchaineau et al. [28] to enable interruptible extraction with frame to frame coherence. A data structure for clustering diamonds to represent a sparse scalar field is introduced in [29]. Cases describing the topology of isosurface patches within diamonds are introduced in [25].

Hierarchical representations have also been developed for variable-resolution meshes extracted from a volume data set. The key distinction of these representations is that they do not store the field values but, instead, they represent the sign field at each vertex of the hierarchical grid. Consequently, they only represent a single (variable- or multi-resolution) mesh and do not support scalar field operations such as distance computations and Boolean operations. Mello et al. [30] extract an approximate variable-resolution representation of an isosurface from a 3D point cloud by first computing the sign field of the vertices of a nested tetrahedral mesh. The Adaptive Dual Contouring method proposed in [10] represents the isosurface as an octree in which the leaves contain the isosurface, and the connectivity between nodes is implicitly determined through the face neighbors in the octree. This method enables a bottom-up simplification process, but does not guarantee that the extracted meshes are conforming. Later approaches [31], [32] ensure that the extracted surface is manifold, but do not guarantee that is is free of self-intersections. Conversely, the approach of Ju and Udeshi [33] ensures that the surface does not intersect, but does not guarantee that it is manifold. Zhang et al. [17] extend the adaptive dual contouring method for variable-resolution interval volumes. All these approaches support a (bottom-up) simplification process, rather than the more general (top-down) selective refinement query, i.e., it is not possible to extract variable-resolution meshes of minimal size satisfying an error criterion. Alternatively, a multiresolution representation of the extracted mesh enables the generation of variable-resolution meshes satisfying an application-dependent error criterion at runtime.

Pascucci and Bajaj [34] present a progressive isosurfacing algorithm for LEB hierarchies in 2D and 3D in terms of a small set of mesh update primitives. They define a one-to-one correspondence between isosurface modifications and conforming modifications that generate them. Additionally, they prove that for a hierarchy with n nodes, the extraction of an isosurface of size k requires $O(k \log n)$ size and time. However, since the modifications are defined in terms of basic mesh operations, it becomes more difficult to define the different intersection cases and to guarantee correctness as the dimension and complexity of the cases increases (e.g. for conforming modifications for interval volumes). Also, they do not discuss a data structure for the extracted meshes or how the structure might be queried. Lewiner et al. [35] provide a multiresolution isosurface encoding where only modifications in the hierarchical *tubular neighborhood* of the surface are encoded. However, their model defines a total order for the modifications, rather than a partial order, limiting the number of encoded meshes. In [2], we present a model for multiresolution interval volume meshes generated from an LEB hierarchy where the modifications are compactly encoded clusters of tetrahedra, and the dependency relation is implicitly determined from that of the LEB hierarchy.

The current work improves on previous approaches by abstracting the mesh extraction method of [2] through the use of isodiamonds. Thus, multiresolution isosurfaces and interval volumes can be treated in the same framework. Additionally, we introduce minimal isodiamond hierarchies as a means of reducing the storage and computational costs associated with extracting variable resolution isosurfaces and interval volumes. Isodiamond hierarchies efficiently support general selective refinement queries to extract conforming meshes of minimal size satisfying application-dependent selection criteria.

4 A Hierarchy of Diamonds

The modification to a tetrahedral mesh induced by the longest edge bisection rule does not generate a conforming mesh. Thus, we consider the cluster of tetrahedra that need to be subdivided together, i.e. the set of tetrahedra sharing their longest edge [1], [27]. Such clusters of tetrahedra are usually called *diamonds*, and we call the shared longest edge the *spine* of the diamond. Since all tetrahedra in a diamond share their spine, there are three similarity classes of diamonds, corresponding to the three classes of tetrahedra in an LEB mesh (see Section 2.2). Specifically, 0-*diamonds* have spines aligned with the diagonal of an axis-aligned cube (see Figure 4a), 1-*diamonds* have spines aligned with a face-diagonal (see Figure 4b) and 2-*diamonds* have spines aligned with an edge of such a cube (see Figure 4c). An i-diamond for $i \in \{0, 1, 2\}$ is formed by $\{6, 4, 8\}$ tetrahedra and is bounded by $\{8, 6, 10\}$ vertices, respectively.

A diamond δ is subdivided by bisecting all the tetrahedra forming δ according to the longest edge bisection rule. The local effect of the subdivision of diamond δ is the removal of its spine, the insertion of a vertex $\mathbf{v}_c$ at the center of δ, and of new edges from $\mathbf{v}_c$ to each of the vertices of δ (see Figure 5). We call such edges the *subdivision edges* of δ, and typically denote the *subdivided diamond* associated with δ as δ_s. A diamond can be identified by its spine, or alternatively, by the midpoint of the spine, called its *central vertex*. Since the central vertex and the subdivision edges lie entirely within a single diamond, they can be uniquely associated with that

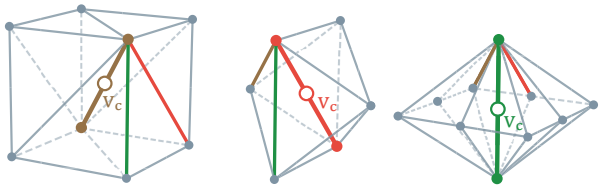


Fig. 4. The three classes of diamonds. The *central vertex* $\mathbf{v}_c$ of a diamond is located at the midpoint of its *spine*.

diamond. Note that during a subdivision, all changes are local and occur within the interior of the diamond. As such, the vertices, edges and faces on the boundary of δ are not affected by its subdivision.

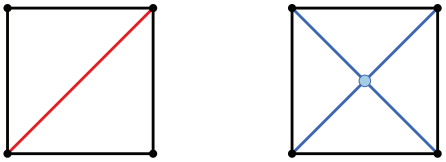


Fig. 5. Subdivision of a diamond δ (in 2D). The pair $u = (\delta, \delta_s)$ defines a modification in a hierarchy of diamonds.

Thus, we define a *Hierarchy of Diamonds (HD)* as a multiresolution model, that we denote as Δ, in which the base mesh is defined by the diamond whose spine is a diagonal of the cubic domain Ω. A modification in Δ is a pair defined by an (unsubdivided) diamond δ and by the subdivided diamond δ_s associated with it, and is denoted as $u = (\delta, \delta_s)$ (see Figure 5). The dependency relation is defined as in Section 2.3 and thus Δ is described by a dependency graph which is a DAG.

Let $u_p = (\delta_p, \delta_{p_s})$ be a modification that directly precedes $u_c = (\delta_c, \delta_{c_s})$ in the dependency graph of Δ (we say that modification u_p is a parent of u_c, and u_c is a child of u_p). Then, there is at least one tetrahedron in the subdivided diamond δ_{p_s} that is also in the unsubdivided diamond δ_c. We call δ_p a *parent* diamond of δ_c and δ_c a *child* diamond of δ_p in the hierarchy. The dependency graph in a hierarchy of diamonds Δ has a fixed structure which is induced by the regularity of the vertex distribution and by the subdivision pattern. With the exception of diamonds whose spines lie on the domain boundary, 0-diamonds have three parents and six children, 1-diamonds have two parents and four children, and 2-diamonds have four parents and eight children. This also holds for the corresponding modifications in Δ.

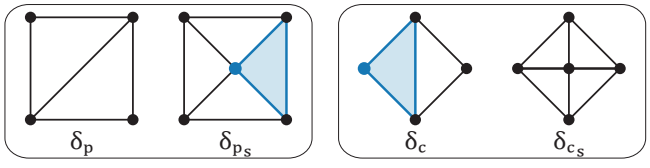


Fig. 6. Modification u_p is a parent of modification u_c in Δ since subdivided diamond δ_{p_s} and unsubdivided diamond δ_c have a triangle in common (light blue).

Figure 6 shows the direct dependency relation between two modifications $u_p = (\delta_p, \delta_{p_s})$ and $u_c = (\delta_c, \delta_{c_s})$ in 2D. u_p is a parent of u_c since δ_{p_s} and δ_c have a triangle in common (light blue). In the 3D case, we observe that a subdivided parent diamond δ_{p_s} and its (unsubdivided) child δ_c always have two tetrahedra in common. We call this pair of tetrahedra a *parent-child duet* or, simply, a *duet*. A duet between δ_{p_s} and δ_c consists of two face-adjacent tetrahedra whose shared face is defined by the two spine vertices of δ_c and by the central vertex of δ_{p_s} (see Figure 7). Duets define the unique contribution of tetrahedra from a parent diamond to one of its children, and thus, they are in one-to-one correspondence with the arcs of the dependency graph of Δ.

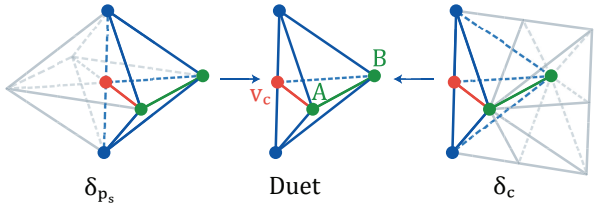


Fig. 7. A *parent-child duet* maps the pair of tetrahedra common to a subdivided diamond δ_{p_s} and a child δ_c. The shared face in the duet contains the central vertex $\mathbf{v}_c$ of δ_{p_s} (red vertex) and the two spine vertices **A** and **B** of δ_c (green vertices).

The regularity of a hierarchy of diamonds can be exploited to yield an implicit encoding for all geometric and hierarchical *components* of a diamond. We have developed an implicit encoding, which has been generalized in [36], to encode a hierarchy of diamonds in arbitrary dimensions. In the following, we assume Δ covers a regular cubic domain Ω of size $(2^N+1)^3$, where N is the maximum level of resolution and vertices of diamonds in Δ have integer coordinates in the range $[0, 2^N]$. We consider the binary representation of the coordinates v_x, v_y and v_z of the central vertex $\mathbf{v}_c$ of a diamond δ:

$$\mathbf{v}_c = \begin{bmatrix} v_x &= x_1 x_2 \dots x_n & d_{x_1}\, d_{x_2} & \underbrace{00\dots0}_{\sigma} \\ v_y &= y_1 y_2 \dots y_n & d_{y_1}\, d_{y_2} & 00\dots0 \\ v_z &= z_1 z_2 \dots z_n & \underbrace{d_{z_1}\, d_{z_2}}_{\tau} & \end{bmatrix} \quad (1)$$

The location of the vertices of diamond δ as well as the central vertices of its parents and children can be derived in terms of scaled offsets from $\mathbf{v}_c$. The *level* ℓ of an i-diamond δ is equal to the number of i-diamond ancestors of δ in the dependency graph (i.e., its depth in the graph modulo 3). We define the *scale* σ of δ in terms of its level as $\sigma(\delta) = N - \ell(\delta)$. The scale is encoded as the minimum of the number of trailing zeros among the coordinates v_x, v_y and v_z of $\mathbf{v}_c$ (see σ in Eq. 1). Consequently, the rightmost σ bits of all coordinates of $\mathbf{v}_c$ are all zero, but at least one of the bits at location $\sigma+1$ (i.e. d_{x_2}, d_{y_2} or d_{z_2}) is nonzero. The bits at positions $\sigma+1$ and $\sigma+2$ in the coordinates of $\mathbf{v}_c$ (i.e. those containing the letter d in Eq. 1) uniquely define the *type* τ of δ. The similarity class of δ is encoded within τ by the number of zeros at position $\sigma+1$ of $\mathbf{v}_c$.

The scaled offsets that allow computing the vertices of δ and the central vertices of its parents and children can be derived directly from the *type* τ and the *scale* σ of δ or can be accessed from a precomputed lookup table based on τ [36]. Thus, the parent-child dependency relation of Δ is implicitly encoded.

A hierarchy of diamonds is the basis for a multiresolution model of a scalar field. In this case, a scalar value is associated with the central vertex of each diamond as is the diamond's approximation error, which is often precomputed to accelerate selective refinement queries. The error ϵ associated with a diamond δ is the maximum interpolation error for all points within its domain, i.e., $\epsilon(\delta) = \underset{\mathbf{p}\in\delta}{Max}\,(\epsilon(\mathbf{p}))$, where $\epsilon(\mathbf{p})$ is the absolute difference between the scalar value at point $\mathbf{p}$ and the value obtained by interpolating the scalar values at the vertices of δ. Often, the range of scalar values within the domain of δ is also encoded [18] so that irrelevant regions can be culled during selective refinement. All such information is associated with a diamond, or its central vertex, and thus a hierarchy of diamonds built on a regular volume data set can be efficiently encoded as a three-dimensional array.

5 Multiresolution Isodiamonds

Due to the size of isosurfaces, or interval volumes, extracted from volume data sets, simplified representations for such structures need to be used. These simplified representations are usually obtained by applying a local mesh coarsening operator, such as an edge collapse, to the full resolution mesh describing the isosurface, or interval volume.

Moreover, it is often desirable to have a multiresolution representation of a specific isosurface, or interval volume, from which simplified adaptive representations can be efficiently extracted on demand. When a multiresolution isosurface, or interval volume, is extracted from a multiresolution model of the underlying 3D scalar field, its structure should be coherent with the model of the scalar field. In other words, a natural multiresolution representation for an isosurface, or an interval volume, is defined by the intersection of the former with the atomic modifications in the multiresolution model of the field. The resulting multiresolution model clearly inherits the dependency relation from the dependency graph of the multiresolution field model.

We consider here the problem of defining multiresolution models of isosurfaces, and interval volumes, when the underlying multiresolution field model is defined by a hierarchy of diamonds. As shown in Section 4, the regularity of the vertex distribution of a hierarchy of diamonds enables a very compact encoding which we exploit to produce effective and compact multiresolution models for isosurfaces and interval volumes. Specifically, each local modification to the isosurface, or interval volume, intersecting a specific diamond δ has a one-to-one correspondence with the modification associated with δ. Since diamonds are defined on a regular grid, they are much simpler to encode than the modifications to the general triangle or tetrahedral mesh representing the isosurface, or interval volume.

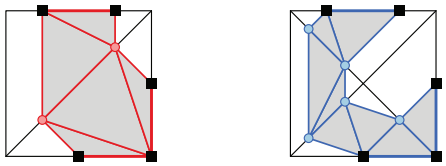
The basic idea in defining a multiresolution model for an isosurface S, or interval volume I, extracted from a hierarchy of diamonds Δ consists of considering only a subset of the diamonds in Δ and possibly the intersection of S or I with such diamonds. We call the diamonds in this multiresolution model *isodiamonds*, and the intersection of an isodiamond with S or I an isosurface or interval volume *patch*, respectively. Each patch is triangulated based on the values of the sign field associated with the corresponding diamond in Δ. We call the ordered set of such sign values the *bit pattern* of the diamond.

Since we are interested in the ability to reconstruct the isosurface S, or interval volume I, at intermediate uniform or variable-resolutions, we need to include in the model all isodiamonds with non-empty patches, that we call *active isodiamonds*. However, since S or I depends on the range of the scalar field rather than its domain, we require a spatial index on the active isodiamonds. The latter is obtained by considering the *relevant isodiamonds*, which are the ancestors of active isodiamonds that have empty patches. An important subset of the relevant isodiamonds are the *creation isodiamonds*, which create a new topological component of S or I upon subdivision. The remaining isodiamonds are *inactive* with respect to S or I.

More formally,

- an *active isodiamond* δ is an isodiamond such that both δ and the associated subdivided isodiamond δ_s contain at least one active tetrahedron; thus, both δ and δ_s are intersected by the isosurface or interval volume (see Figure 9).
- a *creation isodiamond* is an isodiamond δ which does not intersect the isosurface S or the interval volume I, but the tetrahedra of the associated subdivided isodiamond δ_s are all active (see Figure 10).
- a *relevant isodiamond* is an isodiamond δ which does not intersect the isosurface S or interval volume I but at least one of its descendants intersects either S or I.

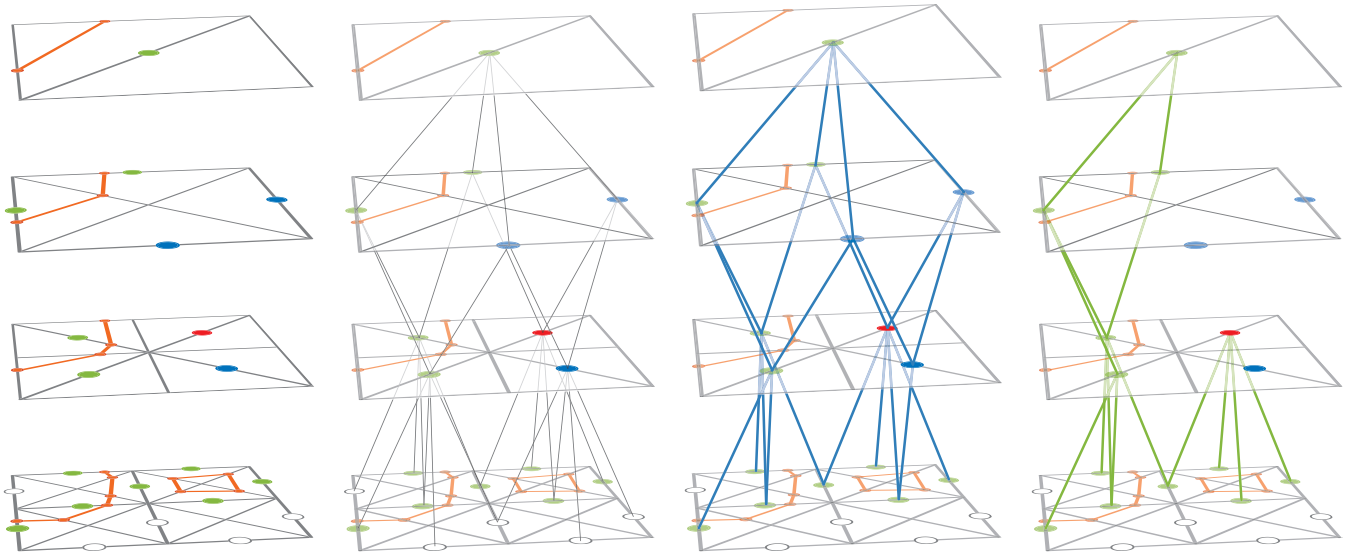
Figure 8a illustrates the various isodiamond types on a small isodiamond hierarchy whose underlying hierarchy of diamonds Δ covers a square 2D domain and has the dependency graph depicted in Figure 8b. For simplicity, we only show the unsubdivided diamonds. The central vertices of active, relevant, creation and inactive isodiamonds are indicated by green, blue and red and white vertices at their spine centers, respectively. The top row describes the single (unsubdivided) isodiamond corresponding to the root of Δ. Since it intersects the isosurface, it is an active isodiamond (indicated with a green central vertex). The next row describes the four children of the root diamond in Δ. Two of these are active isodiamonds (green), and the other two are relevant isodiamonds (blue). The third row shows the four children of those at level two, of which, two are active isodiamonds, one is a relevant isodiamond and one is a creation isodiamond. As illustrated in the final row, the patches generated by subdividing the creation isodiamond form a new isosurface component. Figure 11 illustrates the isodiamond types of the 2D Bonsai tree dataset ($\kappa = 58$), by coloring the central vertices as above. The hierarchy is encoded implicitly using a 2D array as described in Section 4.



(a) δ (b) δ_s

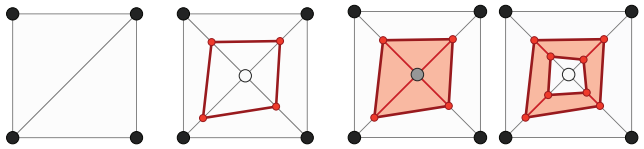
Fig. 9. A modification associated with an active isodiamond for an interval volume (in 2D). Compare to Figure 5.

We have developed two multiresolution models for isosurfaces and interval volumes extracted from a diamond hierarchy. The *Relevant Isodiamond (RI)* hierarchy (described in Section 6) closely follows the hierarchy of diamonds encoding the underlying field since it represents both the active isodiamonds and the relevant isodiamonds. Thus, a modification in



(a) Isodiamonds (b) HD dependency graph (c) RI dependency graph (d) MI dependency graph

Fig. 8. Isodiamonds (a) and associated dependency graphs (b-d) for a small 2D isodiamond hierarchy. *Active isodiamonds* (green central vertices) intersect the isosurface (orange lines). *Relevant isodiamonds* (blue central vertices) are empty ancestors of the active isodiamonds. *Creation isodiamonds* (red central vertices) are relevant isodiamonds that create a new topological component after subdivision. All other isodiamonds are inactive (white central vertices). The dependency graph of the MI hierarchy (d) is a subgraph of the RI hierarchy's dependency graph (c), which, in turn, is a subgraph of the HD's dependency graph (b).



(a) (b) (c)

Fig. 10. All possible creation isodiamond modifications. The unsubdivided isodiamond δ (a) and the subdivided isodiamond δ_s for an isosurface (b) or for an interval volume (c, two cases).

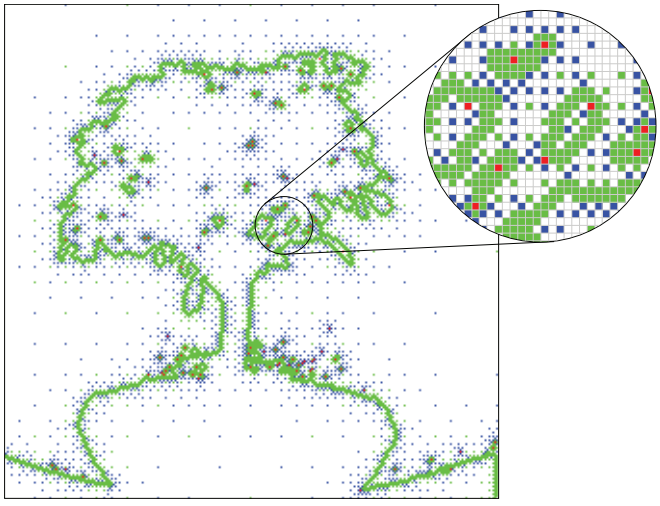


Fig. 11. Isodiamond types for 2D bonsai tree dataset ($\kappa = 58$). The color of the central vertex of a diamond indicates its corresponding isodiamond type. Of the 257^2 diamonds, there are 5,242 *active* isodiamonds (green), 123 *creation* isodiamonds (red) and 2,877 *relevant* isodiamonds (blue).

an RI hierarchy corresponds to a diamond δ in the associated diamond hierarchy Δ such that the range of the field values within δ contains the isovalue κ defining the isosurface, or the isorange K defining the interval volume. In contrast, the *Minimal Isodiamond (MI)* hierarchy (described in Section 7) just contains the modifications that intersect the isosurface S, or interval volume I. Since the relevant isodiamonds do not intersect S or I, and mainly serve as a spatial access structure for the active and creation isodiamonds, they are not strictly necessary to reconstruct S or I. The key novelty in the MI model is its ability to extract simplified conforming meshes describing the isosurface, or interval volume, without storing the relevant isodiamonds.

5.1 Encoding Isodiamonds

Due to the one-to-one correspondence between isodiamonds and diamonds in the hierarchy of diamonds, each patch in the RI or MI hierarchy is encoded (using a marching cells rule) through: (a) the bit pattern of the corresponding diamond and (b) the intersection vertices of the corresponding patch, which we call *isovertices*. Each patch can then be triangulated via a lookup table on the bit pattern, by using the isovertices.

Since diamond modifications are local and only affect the interior vertices and edges of the diamond (see Figure 5), we may assume that the bit pattern and isovertices on the boundary of an isodiamond have already been encoded in ancestor isodiamonds. Thus, the patch corresponding to a subdivided isodiamond δ_s can be reconstructed from the patch corresponding to δ using only (a) the sign value of the central vertex of δ_s and (b) the intersection vertices for each active subdivision edge of δ_s. Figure 9 illustrates a modification to an interval volume patch (in 2D) corresponding to the subdivision of an active isodiamond δ. Observe that the modification removes the isovertices on the spine of δ (red vertices in Figure 9a) and

inserts new isovertices along its active subdivision edges (blue vertices in Figure 9b). The patch is then retriangulated using the new isovertices. Patch vertices and edges intersecting the diamond's boundary are unaffected by the modification.

Recall that, under linear interpolation, all patch intersection vertices lie along edges of tetrahedra. Thus, rather than encoding each intersection vertex using its (x, y, z) position, we use an interpolation coefficient along the unique subdivision edge containing the vertex. For compactness, we quantize each intersection vertex to a single byte. This is an appropriate compromise between storage space and accuracy, since the lengths of the diamond edges shrink as their level increases.

Based on the bit pattern associated with the isodiamond, a lookup table can be used to determine the number of isovertices introduced by the modification as well as an (ordered) list of indices for the active subdivision edges. To maintain constant-sized modifications, the isovertices are stored in a global *isovertex array* and only the index of the first isovertex is encoded with each modification.

Note that the encoding of an isodiamond does not include the scalar values of the original volume data set, but only the sign field of the vertices of the data set. The efficiency of the resulting data structure derives from exploiting the regular spatial decomposition and the implicit dependency relation induced by the hierarchy of diamonds representation, while only encoding the modifications that relate to the specific isosurface or interval volume.

6 Relevant Isodiamond hierarchies

In this Section, we present the Relevant Isodiamond (RI) hierarchy for an isosurface S, or an interval volume I. We describe first the data structure encoding it (Section 6.1). Next, we review the algorithm for generating an RI hierarchy from a hierarchy of diamonds (Section 6.2). Finally, we discuss how variable-resolution isosurfaces, or interval volumes, can be extracted from an RI hierarchy through selective refinement (Section 6.3).

A *Relevant Isodiamond (RI)* hierarchy, that we denote as Δ_R, is defined by the subset of modifications of the corresponding hierarchy of diamonds Δ that are *active* or *relevant* with respect to the isosurface S, or interval volume I. The dependency relation in Δ_R is defined as the restriction of Δ's dependency relation to the active and relevant isodiamonds. Thus, every relevant and active isodiamond will have in Δ_R the same parents as the corresponding diamond in Δ but some of the children isodiamonds might be missing in Δ_R (as compared to Δ), if they are neither active nor relevant. Figure 8c shows the dependency graph of the RI hierarchy Δ_R associated with the multiresolution isosurface representation of Figure 8a. The graph describing Δ_R contains a subset of the arcs of Δ's dependency graph (see Figure 8b), where the missing arcs have endpoints corresponding to *inactive* isodiamonds (white central vertices).

It can be easily seen that the dependency graph describing Δ_R is a connected subgraph of the one describing Δ and has the same root, which can be an active or a relevant isodiamond. Although the relevant isodiamonds do not intersect S or I, and thus their associated patches are empty, they are used here to guarantee the transitive closure of the dependency relation (as discussed in Section 2.3).

6.1 Data structure

The patch of S or I corresponding to the base mesh is encoded through the sign field of the eight corner vertices of the cubic field domain and through interpolation coefficients for all the intersection vertices along the active edges of the root diamond.

All isovertices introduced during a modification are located along the active subdivision edges of its associated subdivided isodiamond δ_s, and are stored in the isovertex array as interpolation coefficients. For interval volumes, each subdivision edge can be active with respect to the lower surface and/or the upper surface, and thus can generate at most two isovertices. Since isovertices between the two surfaces of I coincide with the vertex of a diamond they are implicitly encoded. Thus, only the isovertices of an interval volume's upper and lower surfaces need explicit encoding. Since the dependency relation in Δ_R is inherited from the dependency relation of Δ, we can compute all parents and children from the coordinates of the central vertex of the isodiamonds.

With each modification $u = (\delta, \delta_s)$ in an RI hierarchy, we encode the following information, thus using 12 bytes per modification:

- central vertex $\mathbf{v}_c$ of the isodiamond δ (encoded using six bytes as three `unsigned short`s);
- sign value of $\mathbf{v}_c$ (encoded using one bit for isosurfaces or two bits for interval volumes);
- index of the first intersection vertex associated with the modification in the isovertex array (encoded as a four-byte `unsigned int`);
- approximation error associated with the isodiamond to accelerate selective refinement queries (quantized to fourteen bits in the range $[0, 1]$).

Assuming that the vertices of the base mesh of the hierarchy of diamonds Δ are located at offset zero in the vertex array, the cost of encoding the base mesh is just that of encoding its sign field. Since the root diamond is a 0-diamond with eight vertices, its sign field requires a single byte for isosurfaces or two bytes for interval volumes. Since each isovertex is encoded as an interpolation coefficient using eight bits, storing the $|\mathbf{v}|$ intersection vertices requires $|\mathbf{v}|$ bytes. Finally, since the dependency relation is implicitly derived, the cost of the data structure for encoding an RI hierarchy is

$$\underbrace{2}_{\text{base mesh}} + \underbrace{12 * (|m_a| + |m_r|)}_{\text{modifications}} + \underbrace{|\mathbf{v}|}_{\text{vertices}} \text{ bytes,}$$

where $|m_a|$ is the number of active isodiamonds and $|m_r|$ is the number of relevant isodiamonds.

6.2 Generating an RI hierarchy

Given a hierarchy of diamonds Δ and an isovalue κ for isosurfaces (or an isorange K for interval volumes), an RI hierarchy Δ_R is generated from Δ by performing a top-down traversal of the dependency graph describing Δ. At each node, the algorithm checks whether the min/max field values associated with the corresponding modification contain the isovalue (or the isorange). Modifications that do not contain the isovalue (or isorange) are irrelevant and neither they nor their descendants are stored in the RI hierarchy. All modifications that contain the isovalue (or isorange) will be copied in Δ_R as pointed out before, but only some of them (i.e. the active ones) intersect the isosurface (or interval volume). A closure operation must then be applied to ensure that all relevant ancestors are retained.

For all diamonds δ which intersect the isosurface, or the interval volume, the field values associated with the vertices of δ are used to compute the interpolation coefficients of each new intersection vertex along the active subdivision edges. Specifically, for each active subdivision edge $\mathbf{e} := \{\mathbf{v}, \mathbf{v}_c\}$ of δ, where $\mathbf{v}_c$ is the central vertex of δ, the interpolation coefficient γ is computed as $\gamma = (\kappa - F(\mathbf{v}_c))/(F(\mathbf{v}) - F(\mathbf{v}_c))$. It is then quantized to eight bits as the integer value $\lfloor \gamma * 2^8 + 0.5 \rfloor$.

6.3 Selective refinement on an RI hierarchy

Simplified isosurfaces, or interval volumes, can be extracted from the relevant isodiamond hierarchy through selective refinement. The selective refinement query is defined through a selection criterion based on the approximation error, which can vary in different parts of the isosurface (or interval volume).

The selective refinement algorithm operates as a top-down traversal of the dependency graph describing the RI hierarchy Δ_R. It is initialized with the modification corresponding to the root of Δ_R and at each step extracts a closed set of modifications defining a cut in the dependency graph of Δ_R, called the *active front*.

Since each modification in an RI hierarchy Δ_R is a modification in the corresponding hierarchy of diamonds Δ there are two meshes associated with a cut C defining an active front. The *current diamond mesh* Σ_δ is formed by tetrahedra of Δ belonging to the base mesh or generated by the modifications in C or in their ancestors. The *current extracted mesh* Σ is formed by triangles of the isosurface (or tetrahedra of the interval volume) intersecting the tetrahedra in the current diamond mesh. The latter contains the triangles (or tetrahedra) intersecting the base mesh or generated by the active or creation modifications in C or in their ancestors. In our implementation, we keep track of the active front and encode Σ's patches by storing the bit patterns and isovertices of the unsubdivided isodiamonds corresponding to diamonds in Σ_δ.

The selective refinement process extracts the current extracted mesh satisfying the specified selection criterion by moving the active front down from the root of the dependency graph. Modifications are performed if an isodiamond does not satisfy the selection criterion, or they are forced in order to satisfy the transitive closure of the dependency relation (thus, they may be applied even to isodiamonds which satisfy the error criterion). In either case, a modification cannot be applied until all of its ancestor modifications have been applied. Since relevant isodiamonds are also stored in an RI hierarchy, all ancestors of a given isodiamond are guaranteed to be available.

All meshes extracted from an RI hierarchy are conforming since the patch within each diamond is conforming, and a careful generation rule for the intersection cases ensures that adjacent patches are conforming (e.g. edges of neighboring patches are aligned). This is not an issue for isosurfaces, but for interval volumes this is guaranteed through the use of a unique lexicographic ordering on the vertices [14].

Figure 14a illustrates the result of a selective refinement query on the RI hierarchy defined by isovalue $\kappa = 58$ on the 2D Bonsai tree dataset (see Figure 11). The *current diamond mesh* (gray, blue and green triangles) covers the entire domain, and the *current extracted mesh* (blue line segments) approximates the isosurface at full resolution.

7 Minimal Isodiamond Hierarchies

In this Section, we introduce the Minimal Isodiamond (MI) hierarchy for an isosurface S, or an interval volume I. A *Minimal Isodiamond (MI)* hierarchy, that we denote as Δ_M, is defined by the subset of modifications of a relevant isodiamond hierarchy Δ_R that increase the number of simplices in the extracted isosurface, or interval volume. Thus, only creation and active isodiamonds are included. The remaining relevant isodiamonds are excluded from Δ_M, and their absence must be accounted for in the model as well as during selective refinement.

The base mesh of Δ_M is formed by the (unsubdivided) creation isodiamonds and by the unsubdivided diamond associated with the root of Δ_R, if the latter corresponds to an active isodiamond. The modifications in Δ_M correspond to the subdivision of active and creation isodiamonds. Finally, Δ_M's dependency relation is a subset of Δ_R's dependency relation restricted to the active and creation isodiamonds. Clearly, it can also be viewed as the restriction of the dependency relation of Δ to the modifications in Δ_M. Thus, the dependency graph of Δ_M is a (possibly disconnected) subgraph of the dependency graph describing Δ_R, whose roots are the creation isodiamonds as well as the root of Δ_R, if it is an active isodiamond. Figure 8d shows the dependency graph for the MI hierarchy Δ_M associated with the multiresolution isosurface representation of Figure 8a. The single creation isodiamond on the third row is a root of Δ_M's dependency graph. Additionally, since the root of Δ_R is an active isodiamond, it is also a root of Δ_M's dependency graph. Compared to the dependency graph of Δ_R in Figure 8c, we observe that all arcs leading from active or creation isodiamonds to active isodiamonds are retained, while those containing a relevant isodiamond or leading to a creation isodiamond are not retained by Δ_M.

We can get a better understanding of the structure of the dependency graph by analyzing the arcs between its modifications. Recall that the parent-child duets have a one-to-one correspondence with the arcs of the dependency graph of Δ (see Section 4). We consider a parent-child duet to be active if at least one of its tetrahedra are active, i.e. intersected by the isosurface or interval volume. We now show that the arcs of the dependency graph of Δ_M correspond to the active parent-child duets of the dependency graph of Δ_R.

Theorem 7.1: Let d_{pc} be an active parent-child duet between a parent diamond δ_p and its child diamond δ_c. Then, δ_p is either an active isodiamond, or a creation isodiamond.

Proof: Since d_{pc} is active, it has at least one vertex $\mathbf{v}$ with a different sign value than some other vertices in d_{pc}. If $\mathbf{v}$ is not the central vertex of δ_p, then through the duet correspondence between vertices of the parent and child diamond, δ_p has two vertices with different bit values and is therefore an active isodiamond. Otherwise, assume that $\mathbf{v}$ is the central vertex of δ_p and that δ_p is not an active isodiamond. Then, since $\mathbf{v}$ has a distinct bit value from the other vertices of δ_p, it is a creation isodiamond. □

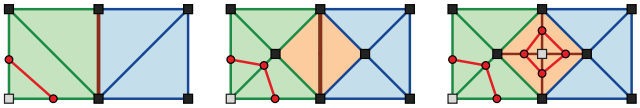
Since non-active duets do not carry information related to the isosurface S, or the interval volume I, the arcs of the dependency graph of Δ_M retained from the ones of Δ_R are exactly those between parents and children corresponding to active parent-child duets. However, recall that the dependency graph is not explicitly represented in the model.

7.1 Properties of an MI hierarchy

We report here some properties of creation isodiamonds δ_c and active isodiamonds δ_a in the MI hierarchy and in the corresponding RI hierarchy. These properties are important for extracting conforming representations from an MI hierarchy.

The parents of a creation isodiamond in Δ_R can be either active or relevant isodiamonds, as depicted in Figure 12, where the parents of a creation isodiamond (red) are an active isodiamond (green) and a relevant isodiamond (blue), respectively. Since creation isodiamonds are roots of Δ_M, none of these arcs are retained in the dependency graph of Δ_M.

The children of a creation isodiamond in Δ_R are all active isodiamonds. Since the duet d_{ca} between a creation isodiamond δ_c and a child δ_a is active, the corresponding arcs are all retained in the dependency graph of Δ_M.



(a) (b) (c)

Fig. 12. (a) A parent of a creation isodiamond in the hierarchy of diamonds (red diamond in (b)) can be either an active isodiamond (green) or a relevant isodiamond (blue). (b) Since a creation isodiamond (red) is a local root in the dependency graph of Δ_M, its ancestors in the hierarchy of diamonds are not retained. (c) All children of the creation isodiamond (red triangles) are active.

An active isodiamond δ_a, by definition, has at least one active duet such that δ_a is the parent (unless δ_a is a leaf in the dependency graph), and at least one active duet such that δ_a is the child (unless δ_a is the root of Δ_R). The parents of an active isodiamond δ_a in Δ_R can be either active, creation or relevant isodiamonds:

- All creation isodiamond parents corresponds to active duets and the corresponding arcs are retained in the dependency graph of Δ_M.
- An active isodiamond parent is retained in the dependency graph of Δ_M only if the corresponding duet is active.
- The relevant isodiamond parents in Δ_R do not correspond to active duets and are not retained in Δ_M.

Similarly, the children of an active isodiamond δ_a in Δ_R can be either active, creation, relevant or inactive isodiamonds. The only arcs retained in the dependency graph of Δ_M are those corresponding to an active child diamond δ and to an active duet between δ_a and δ.

Recall that, by definition, the spine of an active diamond δ_a is part of all duets containing δ_a as child. If the sign values of the spine differ, all duets containing δ_a as child are active, and their corresponding arcs are retained in Δ_M. This corresponds to a diamond whose spine intersects the isosurface or one (or both) of the bounding surfaces of the interval volume. In addition, when representing interval volumes, all duets are active if the signs of the spine vertices are both zero, corresponding to a diamond whose spine is located between the two bounding surfaces. Otherwise, some of the duets could be inactive. Only the arcs between δ_a and a parent δ corresponding to active duets are retained by Δ_M. Table 1 summarizes the status of Δ_M's dependency graph arcs based on the isodiamond types of the parent δ_p and the child δ_c of a duet d_{pc}.

7.2 Data structure

The MI hierarchy is represented in the same way as the RI hierarchy (see Section 6.1), where each modification is encoded

TABLE 1
Status of arc in dependency graph of Δ_M based on isodiamond types of parent δ_p (rows) and child δ_c (columns) of a parent-child duet d_{pc}. Types $\{R, C, A, I\}$ denote, respectively, Relevant, Creation, Active, Inactive.

$\delta_p \setminus \delta_c$	R	C	A	I
R	x	x	x	x
C	–	–	✓	–
A	x	x	*	x
I	–	–	–	x

✓ Always retained
* Retained if active duet
x Never retained
– Not possible

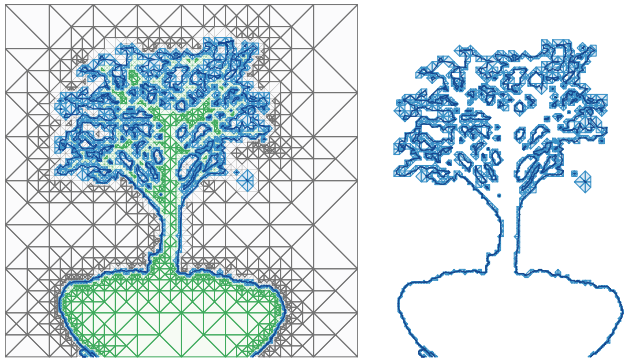
using 12 bytes, and each vertex requires a single byte. Consequently, a model with $|m_c|$ creation isodiamonds, $|m_a|$ active isodiamonds and $|\mathbf{v}|$ isovertices requires

$$\underbrace{2}_{\text{base mesh}} + \underbrace{12 * (|m_a| + |m_c|)}_{\text{modifications}} + \underbrace{|\mathbf{v}|}_{\text{vertices}} \text{ bytes.}$$

Since the creation isodiamonds are required to initialize the base mesh, for efficiency, we store the creation isodiamonds separately from the active isodiamonds.

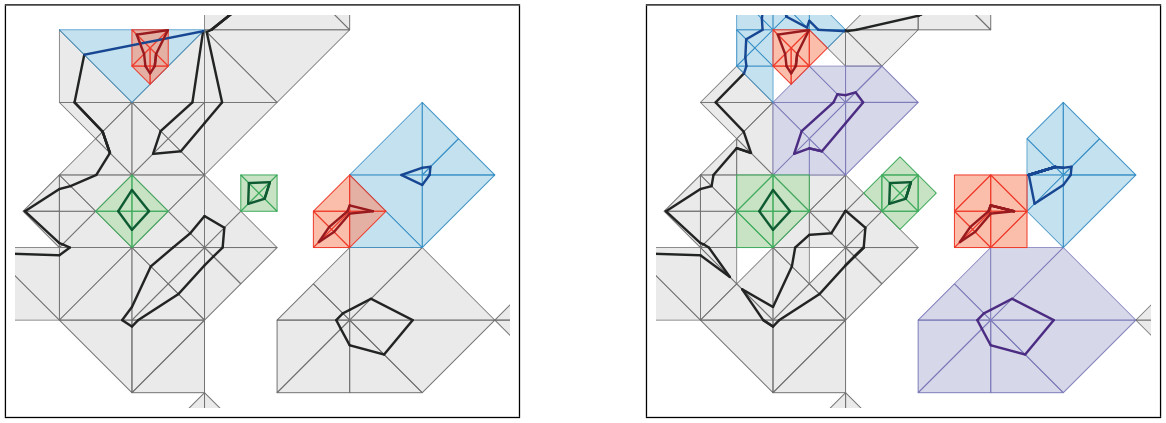
Note that, since active and creation isodiamonds are the only modifications that contain patches of the output isosurface or interval volume, the MI hierarchy represents exactly the same set of isovertices (and thus patches) as the RI hierarchy, despite encoding significantly fewer modifications.

The Minimal Isodiamond hierarchy is generated from a hierarchy of diamonds in a similar manner as the RI hierarchy (as detailed in Section 6.2). The only difference is that when adding isodiamonds to the model, we must additionally test for the type of isodiamond. These tests are efficiently carried out by applying bitmasks to the bit pattern of each diamond. If the bit pattern of a diamond δ is not homogeneous, then δ is an active isodiamond. If only the sign of the central vertex differs from those of the other vertices, then δ is a creation isodiamond. Otherwise, the diamond is discarded. In all cases, if the min/max range of a diamond contains the isovalue or isorange, we recursively test all of its descendants.



(a) Extracted from RI (b) Extracted from MI

Fig. 14. Diamond mesh Σ_δ (gray, green and blue triangles) and current mesh Σ (dark blue line segments) of the 2D bonsai tree dataset ($\kappa = 58$) extracted from (a) an RI hierarchy Δ_R and (b) an MI hierarchy Δ_M. Observe that the mesh extracted from Δ_R covers the entire domain while the mesh extracted from Δ_M only covers the isosurface patches.



(a) Before closure operator (b) After closure operator

Fig. 13. Subdivision of a creation isodiamond δ_c can cause the new patches to intersect other patches of the current mesh as in the overlap between red and blue triangles in (a). However, applying the closure operator to all children of δ_c forces the active front to include δ_c (red and green triangles in b). The extracted isosurface in (b) is conforming even though its diamond-based LEB mesh is not (interface between the red and purple triangles in (b)).

7.3 Selective refinement on an MI hierarchy

Uniform– and variable-resolution isosurfaces, or interval volumes, can be extracted from an MI hierarchy through a similar selective refinement process as the RI hierarchy (as discussed in Section 6.3). There are, however, two key differences due to the fact that the relevant isodiamonds are not encoded and due to the changes to the dependency relation in the MI hierarchy.

The first difference concerns the bit pattern of a diamond. Recall that selective refinement requires a closure operation, that is, all parent modifications of a diamond δ must be applied before δ can be applied. Moreover, our isodiamond encoding scheme (see Section 5.1) assumes that the entire bit pattern of a diamond can be recovered from its ancestors. But, in an MI hierarchy Δ_M, the parents of an active isodiamond δ are a subset of the parents of its corresponding isodiamond in the RI hierarchy Δ_R. As a consequence, the sign values of the vertices from an RI parent δ_r, that is not an MI parent of δ, will not be available to complete the bit pattern associated with δ. This situation, however, only occurs in those active isodiamonds in which the two spine vertices have the same sign value. Since the duet corresponding to the RI parent δ_r is not active, all of its vertices, including the two spine vertices of δ, must have the same sign value. Thus, the missing vertices can be safely assigned the same sign value as the spine vertices.

The other issue is related to the active front when extracting a mesh from an MI hierarchy Δ_M. As in selective refinement from an RI hierarchy (see Section 6.3), we consider an active front C on the dependency graph of Δ_M and two meshes: the *current diamond mesh* Σ_δ, i.e., the tetrahedral mesh associated with front C and extracted from diamonds in Δ, and the *current extracted mesh* Σ, formed by the triangles (or tetrahedra) of the currently extracted isosurface (or interval volume) and defined by the patches intersected by the diamonds in Σ_δ. Since the relevant isodiamonds are not encoded in Δ_M, the domain Ω of the dataset is no longer covered by the tetrahedra in Σ_δ (see Figure 14b, where Σ_δ consists of blue triangles and Σ consists of dark blue line segments). Further, since Δ_M is not necessarily described by a connected dependency graph, C does not need to be connected. Consequently, the current diamond mesh can be disconnected and is no longer guaranteed to be conforming (see the interface between the red and purple triangles in Figure 13b).

Nevertheless, since isosurface, or interval volume patches are only embedded within active isodiamonds, the current extracted mesh Σ is guaranteed to be conforming as long as (a) the interiors of active tetrahedra do not overlap and (b) face-adjacent tetrahedra are conforming along their shared face if that face intersects the isosurface or interval volume. Property (b) is ensured by the patch triangulation cases for each isodiamond as well as by the closure operation in selective refinement (as long as property (a) is not violated).

Since (unsubdivided) creation isodiamonds do not contain active tetrahedra, property (a) is not violated if their tetrahedra overlap other tetrahedra in Σ_δ. However, creation isodiamonds require a modified subdivision rule to ensure that the above two properties remain satisfied after their subdivision. Applying the modification associated with a creation isodiamond δ adds a new disconnected node to the active front C. Since δ is a root in the dependency graph of Δ_M, it has no ancestors and thus, applying the closure operator to δ does not connect it to its ancestors in the DAG describing the hierarchy of diamonds Δ. As a consequence, the diamonds in Σ_δ overlapping the domain of δ will not subdivide. If any of these diamonds are active (in violation of property (a)), their patches can overlap (see the red and blue regions of Figure 13a).

We solve this by applying the closure operator to each child of δ, i.e. applying all modifications on which the children depend. The closure operator cascades up on the dependency graph until (a) the existing front is reached (b) an arc corresponding to a non-active duet is reached or (c) another creation node (i.e. a local root of the DAG) is reached. This joins the fronts of δ and the diamonds overlapping its domain (see Figure 13b), and thus Σ satisfies both the above properties, guaranteeing that the extracted meshes (isosurfaces or interval volumes) are conforming.

8 Results

We demonstrate the compactness and efficiency of the two multiresolution isodiamond models introduced here. We have run experiments on a testbed of over 20 medical, scientific

TABLE 2
Comparison between the size and number of elements in the isodiamond hierarchies (intersection vertices v_i; relevant m_r, creation m_c and active m_a isodiamonds) and those of the mesh at full resolution (vertices v and triangles or tetrahedra t). Data structures sizes are listed in megabytes (1 MB = 1024^2 B).

			Isodiamond Hierarchies				Indexed		Data Structure Sizes (MB)			Comparison
Dataset		Iso	$\|v_i\|$	$\|m_r\|$	$\|m_c\|$	$\|m_a\|$	$\|v\|$	$\|t\|$	MI	RI	Indexed	RI / MI
Isosurface	Neghip.64	59.1	56.3 k	14.9 k	17	24.5 k	43.1 k	86.1 k	0.33	0.51	1.48	1.51 x
	Hydrogen.128	24	75.8 k	29.5 k	1	33.6 k	56.9 k	114 k	0.46	0.79	1.95	1.74 x
	Head.256	58	1.17 m	258 k	623	490 k	884 k	1.77 m	6.73	9.68	30.4	1.44 x
	Engine.256	100	1.39 m	336 k	219	601 k	1.06 m	2.11 m	8.21	12.1	36.2	1.47 x
	Aneurism.512	650	2.26 m	758 k	3.75 k	971 k	1.74 m	3.48 m	13.3	21.9	59.7	1.65 x
	Armadillo.512	0	1.16 m	343 k	8	513 k	870 k	1.74 m	6.97	10.9	29.9	1.56 x
	XMasTree.512	868	2.55 m	760 k	15.6 k	984 k	2.09 m	4.17 m	13.9	22.4	71.7	1.61 x
Interval Volume	Fuel.64	(7.2,11.4)	45.8 k	5.45 k	2	10.8 k	35.3 k	109 k	0.17	0.23	2.07	1.37 x
	Hydrogen.128	(24,48)	101 k	18.3 k	1	56.9 k	75.9 k	400 k	0.75	0.96	6.97	1.28 x
	Tooth.256	(440,1290)	350 k	68.2 k	1.27 k	182 k	277 k	1.32 m	4.41	5.41	23.3	1.31 x
	Aneurism.256	(118,128)	557 k	144 k	1.34 k	123 k	444 k	1.33 m	4.39	5.48	25.3	1.84 x
	Bunny 201	(0,30)	850 k	127 k	10	488 k	639 k	3.45 m	3.38	4.19	59.9	1.23 x
	Head.256	(42,72)	2.48 m	199 k	429	1.20 m	1.89 m	9.28 m	10.4	12.7	163	1.14 x
	Engine.256	(55,175)	1.70 m	196 k	130	809 k	1.29 m	6.17 m	16.1	18.3	109	1.21 x
	Bonsai.256	(45.5,86.5)	2.58 m	438 k	3.52 k	864 k	2.03 m	7.08 m	10.9	13.1	131	1.40 x

and synthetic regular volume data sets of dimensions up to 512^3 and summarize the aggregate information and trends. All experiments were performed on a 2 GHz Pentium Core 2 Duo laptop with 4 GB RAM.

Recall that our isodiamond encoding requires twelve bytes per isodiamond and a single byte for each intersection vertex. Also, both the RI and the MI hierarchy encode all active isodiamonds, but the RI hierarchy encodes in addition all relevant isodiamonds, while the MI hierarchy stores only those relevant isodiamonds that are creation isodiamonds. The number of creation isodiamonds is typically less than 0.5% of the number of relevant isodiamonds (and often significantly less, as shown in columns 5 and 6 in Table 2). Further, since intersection vertices are only encoded in creation or in active isodiamonds, and both models encode these diamonds, the MI hierarchy encodes the same number of intersection vertices as the RI hierarchy. Our experiments on more than 20 datasets indicate that, when encoding isosurfaces, the MI hierarchy requires about 65% of the space of a corresponding RI hierarchy (with a standard deviation of less than 5%). For interval volumes, an MI hierarchy requires approximately 78% the space of the corresponding RI hierarchy (with a standard deviation of 8%).

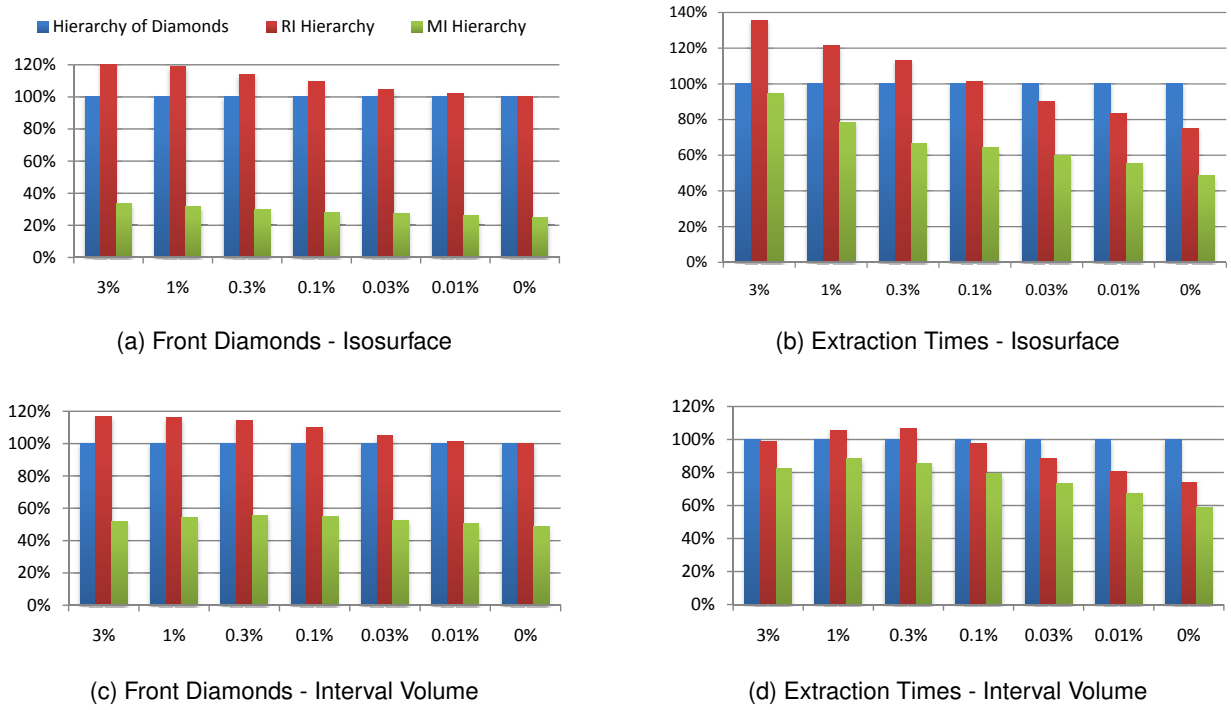
Although the principle goal of multiresolution hierarchies is not compression, we demonstrate the compactness of the two isodiamond hierarchies by comparing them to a simple *indexed* representation of the mesh describing the isosurface S or the interval volume I at full resolution (see Table 2). This representation encodes vertices as three floating-point coordinates, and triangles, or tetrahedra, through the indices of their three, or four vertices, respectively. Thus, encoding an isosurface S with $|v|$ vertices and $|t|$ triangles requires $(12\,|v| + 12\,|t|)$ bytes, and encoding an interval volume I with $|v|$ vertices and $|T|$ tetrahedra requires $(12\,|v| + 16\,|T|)$ bytes.

Compared to an indexed representation of the isosurface at full resolution, the RI hierarchy is approximately three times more compact and the MI hierarchy is about five times more compact. For interval volumes, the RI hierarchy is approximately eight times more compact (as reported in [2]), while the MI hierarchy is more than ten times more compact than the mesh at full resolution. In addition to the space savings, a mesh extracted from both isodiamond hierarchies also implicitly encodes the adjacencies between its triangles, or tetrahedra. This is typically required in downstream mesh processing applications and would require an extra 12 bytes per triangle, or 16 bytes per tetrahedron, since we would need to use for the output mesh an indexed representation enhanced with adjacency information among triangles or tetrahedra.

One of the key motivations for the isodiamond hierarchies is performing selective refinement queries efficiently on a multiresolution representation of a specific predetermined isosurface, or interval volume. These models enable a quicker extraction of uniform or variable-resolution output meshes with respect to extracting them from a hierarchy of diamonds. Moreover, the active front resulting from the MI hierarchy (i.e. the current diamond mesh Σ_δ) requires only a fraction of the diamonds to reconstruct an equivalent output mesh as either the original hierarchy of diamonds, or the corresponding RI hierarchy. Thus, queries on an MI hierarchy require less memory at runtime and the resultant output meshes can be post-processed more efficiently.

Isosurfaces extracted from an MI hierarchy have associated active fronts containing approximately 25% of the diamonds as in the active fronts extracted from a hierarchy of diamonds or from an RI hierarchy (see Figure 15a). Similarly, interval volumes extracted from an MI hierarchy have associated active fronts containing approximately 50% of the diamonds as the other two models (see Figure 15c). Note that in these queries, the selection criterion for the original hierarchy of diamonds Δ depends on the range of field values as well as the approximation error, while the selection criterion for the RI hierarchy Δ_R and the MI hierarchy Δ_M only depends on the approximation error. Since many isodiamonds near the root of the dependency graph of Δ_R are relevant, but their corresponding diamonds in Δ might not intersect the isovalue (or isorange), their modifications will be applied in Δ_R but not in Δ. Thus, when allowing a higher error tolerance, Δ_R can have a time and space overhead of around 10-20% compared to Δ. However, this effect is reduced as the error tolerance decreases, and the active front descends to lower levels of the dependency graph (e.g. compare the relative values of Δ_R and Δ on the horizontal axes of Figure 15 as the error tolerance decreases).

Both isodiamond hierarchies achieve significant reduction in extraction times during selective refinement, compared to the



(a) Front Diamonds - Isosurface

(b) Extraction Times - Isosurface

(c) Front Diamonds - Interval Volume

(d) Extraction Times - Interval Volume

Fig. 15. Comparison of relative number of diamonds in the active front (a,c) and extraction times (b,d) for isosurfaces and interval volumes extracted from the hierarchy of diamonds (blue), RI hierarchy (red) and MI hierarchy (green). The extraction has been performed at uniform resolution with a maximum error of $\epsilon \in \{3\%, 1\%, .3\%, .1\%, .03\%, .01\%, 0\%\}$ (horizontal axis). All values are relative to the hierarchy of diamonds (vertical axis). Values are averages across over 20 datasets of varying sizes and complexity.

diamond hierarchy, as the maximum allowed error decreases (see Figure 15 (b) and (d)). Specifically, as ϵ approaches zero, the RI hierarchy can extract an equivalent isosurface, or interval volume, in about 3/4 the time with respect to the hierarchy of diamonds, while the MI hierarchy can extract an equivalent mesh in about half the time with respect to the hierarchy of diamonds. Note that, when the error is high (e.g. the values toward the left side of each graph in Figure 15), the extracted meshes and times are relatively small, and, thus, relative differences are less significant, while for lower errors (e.g. toward the right side of each graph), the times and active front sizes increase. This suggests that the hierarchy of diamonds is ideally suited for extracting low resolution isosurfaces and interval volumes from the model during the exploratory phases of the analysis of a dataset. Isodiamond hierarchies are better suited for in-depth analysis once the desired isosurface or interval volume has been determined and higher resolution approximations are required for inspection and processing.

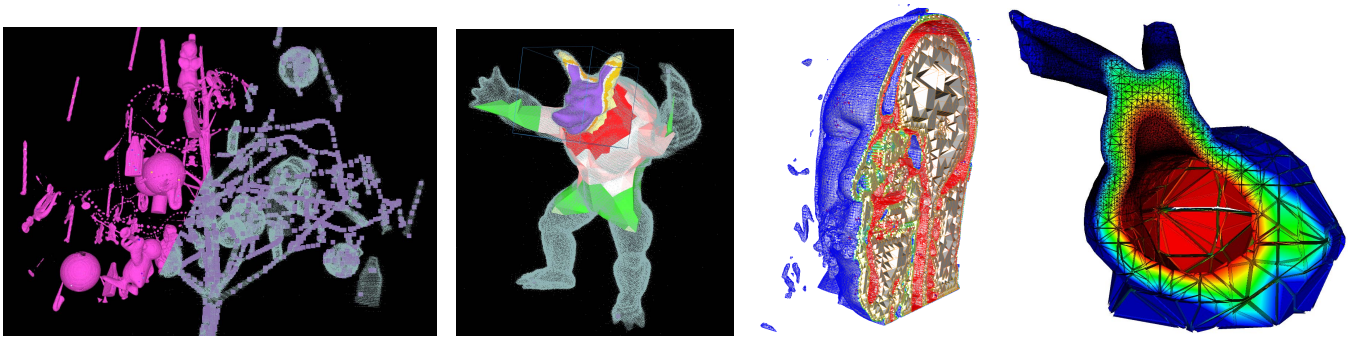
The left side of Figure 16a depicts a zero-error isosurface ($\kappa = 868$) containing approximately two million vertices and four million triangles extracted from the 512^3 Christmas Tree dataset. On the right half, the blue points depict the central vertices of active isodiamonds, and the purple squares coincide with the central vertices of creation isodiamonds. Figure 16c depicts an interval volume ($K = [42, 72]$) containing 879 K vertices and 3.3 million tetrahedra extracted from the 128×256^2 Visible Male Head dataset with uniform error less than 1%. The mesh has been clipped along the median plane to illustrate the sizes of tetrahedra.

The adaptability of the isodiamond models enables location dependent queries in addition to those defined by approximation errors. Figure 16b illustrates a variable-resolution isosurface ($\kappa = 0$) extracted from the Armadillo dataset and focuses the resolution in a region of interest around the head. The error of any triangle within the box is zero, and the error outside the box can be arbitrarily large. Colors indicate the resolution of the diamonds framing the triangles. The blue points coincide with the central vertices of active isodiamonds. Figure 16d illustrates a variable-resolution interval volume extracted from the 201^3 Bunny data set ($K = [0, 20]$), again with a region of interest around the head. The tetrahedra are shrunken slightly and the model is clipped along a plane in order to illustrate the cells of the interval volume.

9 Concluding Remarks

We have developed two efficient multiresolution mesh-based representations for individual isosurfaces and interval volumes which are extracted from a volume data set described as a hierarchy of diamonds. Both models exploit the one-to-one correspondence between diamonds in the hierarchical representation of the field and modifications in the multiresolution representation of the extracted isosurface or interval volume.

The Relevant Isodiamond (RI) hierarchy encodes the set of active isodiamonds as well as their relevant ancestors. Extracted isosurfaces and interval volumes are guaranteed to be conforming due to the one-to-one correspondence between isodiamonds in the RI hierarchy and the diamonds of the diamond hierarchy. As demonstrated in Figure 15, current diamond meshes extracted from the RI hierarchy are generated by approximately the same number of diamonds as those extracted from the diamond hierarchy, in less time. The faster



(a) Xmas Tree Isosurface (b) Armadillo Isosurface (c) Head Interval Volume (d) Bunny Interval Volume

Fig. 16. (a) Full resolution isosurface extracted from 512^3 Christmas tree ($\kappa = 868$). Purple squares on right half indicate creation isodiamonds while blue points indicate active isodiamonds. (b) Isosurface extracted from 512^3 Armadillo ($\kappa = 0$) using a cubic ROI. Triangle colors indicate the DAG level of the isodiamonds and the blue points indicate active isodiamonds in the model. (c) Interval volume ($K = [42, 72]$) extracted from the Head dataset. The mesh is clipped along the median plane to show the internal tetrahedra. (d) Region-based LOD on the bunny model (around the head) with isorange $K = [0, 20]$.

extraction times are likely due to the pre-calculation of interpolation coefficients for the patch intersection vertices during the generation of the RI hierarchy. Since the scalar values of each diamond's vertices do not need to be retrieved, the modification corresponding to an isodiamond can be applied via a single lookup into the isovertex array.

The Minimal Isodiamond (MI) hierarchy encodes all active isodiamonds and only the subset of relevant isodiamonds that are creation isodiamonds. The dependency relation R of its modifications forms a forest of DAGs. An MI hierarchy is described by a dependency graph which is a subgraph of the dependency describing Δ. Since inactive isodiamonds are not created during mesh extraction, current diamond meshes that result from selective refinement applied to MI hierarchies are significantly smaller than those arising from a selective refinement applied to the RI hierarchy.

As indicated in Table 2, both models can be encoded quite compactly compared to an indexed representation of the extracted isosurface or interval volume at full resolution. However, the primary benefit of these models is that they efficiently support selective refinement queries, and thus enable the extraction of meshes satisfying an application-defined error criterion.

Since isodiamond hierarchies only encode the sign field and vertex interpolation coefficients, and not the higher dimensional simplices, this relative advantage increases significantly with the dimension of the encoded mesh. Thus, as long as an appropriate set of lookup tables is defined (as in [16]), the isodiamond representation should lead to significant savings for higher dimensional simplicial meshes. Furthermore, since an interval volume encodes the subvolume enclosed between two isosurfaces, and the storage requirements are related to the number of intersection vertices on its lower and upper surfaces, an interval volume with only one boundary surface can be used as an efficient multiresolution volumetric representation of an object.

Here, we have applied isosurface and interval volume cases to the tetrahedra within each extracted diamond. Given a consistent set of lookup tables defined on the vertex categories, our framework should work equally well for multiresolution non-manifold meshes [37], multi-material interface reconstruction [10], [38] or dual contouring on the tetrahedra or duets within each diamond [39]. We intend to explore these possibilities in our future work.

Since the retained isodiamonds in an RI or MI hierarchy correspond to a sparse but coherent subset of the diamonds in a hierarchy of diamonds, we plan to develop a clustered representation based on [29] to exploit this coherence. This should further reduce the storage costs of the representation and enable out-of-core representations for large multiresolution isosurfaces and interval volumes.

Another interesting direction is to incorporate topological considerations into the error criterion [23], [25]. For instance, in the MI hierarchy, we directly encode all creation isodiamonds. However, while the creation isodiamonds near the root of the dependency graph typically correspond to significant components of the extracted mesh, the creation isodiamonds near the leaves typically correspond to noise in the dataset. By filtering such noise, we might be able to achieve a more faithful representation of the embedded isosurface (or interval volume).

Finally, in our future work, we plan to extend the techniques presented here to the extraction and representation of multiresolution isosurfaces from time-varying volumetric datasets.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their many helpful comments and suggestions. We would also like to thank Michael Kazhdan for providing us with the software used to generate the Armadillo distance field, and Ramani Duraiswami and Nail Gumerov for the Bunny dataset. All other datasets are courtesy of volvis.org. This work has been partially supported by the National Science Foundation under grant CCF-0541032 and by the MIUR-FIRB project SHALOM under contract number RBIN04HWR8.

REFERENCES

[1] L. De Floriani and P. Magillo, "Multiresolution mesh representation: models and data structures," in *Principles of Multi-resolution Geometric Modeling*. Berlin: Springer Verlag, 2002, pp. 364–418.

[2] K. Weiss and L. De Floriani, "Multiresolution interval volume meshes," in *IEEE/ EG Symposium on Volume and Point-Based Graphics*. Los Angeles, California, USA: Eurographics Association, 2008, pp. 65–72.

[3] J. M. Maubach, "Local bisection refinement for n-simplicial grids generated by reflection," *SIAM Journal on Scientific Computing*, vol. 16, no. 1, pp. 210–227, January 1995.
[4] L. De Floriani, E. Puppo, and P. Magillo, "A formal approach to multi-resolution modeling," in *Geometric Modeling: Theory and Practice*. Springer-Verlag, 1997, pp. 302–323.
[5] P. Cignoni, L. De Floriani, P. Magillo, E. Puppo, and R. Scopigno, "Selective refinement queries for volume visualization of unstructured tetrahedral meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 1, pp. 29–45, 2004.
[6] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Proceedings SIGGRAPH Conference*. ACM Press New York, NY, USA, 1987, pp. 163–169.
[7] G. Nielson and B. Hamann, "The asymptotic decider: resolving the ambiguity in marching cubes," in *Proceedings IEEE Visualization*, 1991, pp. 83–91.
[8] B. Payne and A. Toga, "Surface mapping brain function on 3d models," *Computer Graphics and Applications, IEEE*, vol. 10, no. 5, pp. 33–41, Sept. 1990.
[9] S. Gibson, "Constrained elastic surface nets: generating smooth surfaces from binary segmented data," *MICCAI*, vol. 1496, pp. 888–898, 1998.
[10] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 339–346, 2002.
[11] I. Fujishiro, Y. Maeda, and H. Sato, "Interval volume: a solid fitting technique for volumetric data display and analysis," in *Proceedings IEEE Visualization*. Los Alamitos, CA, USA: IEEE Computer Society, 1995, pp. 151–158.
[12] B. Guo, "Interval set: A volume rendering technique generalizing isosurface extraction," in *Proceedings IEEE Visualization*. IEEE Computer Society Washington, DC, USA, 1995, pp. 3–10.
[13] I. Fujishiro, Y. Maeda, H. Sato, and Y. Takeshima, "Volumetric data exploration using interval volume," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 2, pp. 144–155, 1996.
[14] G. M. Nielson and J. Sung, "Interval volume tetrahedralization," in *Proceedings IEEE Visualization '97*, 1997, pp. 221–228.
[15] P. Bhaniramka, R. Wenger, and R. Crawfis, "Isosurfacing in higher dimensions," in *Proceedings IEEE Visualization*. IEEE Computer Society, October 2000, pp. 267–273.
[16] P. Bhaniramka, C. Zhang, D. Xue, R. Crawfis, and R. Wenger, "Volume interval segmentation and rendering," in *Proceedings Volume Visualization Symposium*, 2004.
[17] Y. Zhang, C. Bajaj, and B. Sohn, "Adaptive and quality 3d meshing from imaging data," in *Proceedings ACM Symposium on Solid Modeling and Applications*, 2003, pp. 286–291.
[18] J. Wilhelms and A. V. Gelder, "Octrees for faster isosurface generation," *ACM Trans. Graph.*, vol. 11, no. 3, pp. 201–227, 1992.
[19] R. Shekhar, E. Fayyad, R. Yagel, and J. Cornhill, "Octree-based decimation of marching cubes surfaces," in *Proceedings IEEE Visualization*. Los Alamitos, CA, USA: IEEE Computer Society, 1996, pp. 335–342.
[20] R. Westermann, L. Kobbelt, and T. Ertl, "Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces," *The Visual Computer*, vol. 15, no. 2, pp. 100–111, 1999.
[21] M. Kazhdan, A. Klein, K. Dalal, and H. Hoppe, "Unconstrained isosurface extraction on arbitrary octrees," in *Proceedings Eurographics Symposium on Geometry Processing*. Eurographics Association Aire-la-Ville, Switzerland, Switzerland, 2007, pp. 125–133.
[22] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: a general representation of shape for computer graphics," in *Proceedings SIGGRAPH'00 Conference*. New Orleans, LA: ACM Press, July 2000, pp. 249–254.
[23] Y. Zhou, B. Chen, and A. Kaufman, "Multi-resolution tetrahedral framework for visualizing regular volume data," in *Proceedings IEEE Visualization*, R. Yagel and H. Hagen, Eds. Phoenix, AZ: IEEE Computer Society, October 1997, pp. 135–142.
[24] T. Gerstner and M. Rumpf, "Multiresolutional parallel isosurface extraction based on tetrahedral bisection," in *Proceedings Symposium on Volume Visualization*. ACM Press, 1999, pp. 267–278.
[25] T. Gerstner and R. Pajarola, "Topology-preserving and controlled topology simplifying multi-resolution isosurface extraction," in *Proceedings IEEE Visualization*, 2000, pp. 259–266.
[26] M. Lee, L. De Floriani, and H. Samet, "Constant-time neighbor finding in hierarchical tetrahedral meshes," in *Proceedings International Conference on Shape Modeling*. Genova, Italy: IEEE Computer Society, May 2001, pp. 286–295.
[27] B. Gregorski, M. Duchaineau, P. Lindstrom, V. Pascucci, and K. Joy, "Interactive view-dependent rendering of large isosurfaces," in *Proceedings IEEE Visualization*. IEEE Computer Society Washington, DC, USA, October 2002, pp. 475–484.
[28] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein, "ROAMing terrain: real-time optimally adapting meshes," in *Proceedings IEEE Visualization*. Phoenix, AZ: IEEE Computer Society, October 1997, pp. 81–88.
[29] K. Weiss and L. De Floriani, "Supercubes: A high-level primitive for diamond hierarchies," *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2009)*, vol. 15, no. 6, November-December 2009.
[30] V. Mello, L. Velho, and G. Taubin, "Estimating the in/out function of a surface represented by points," in *Symposium on Solid Modeling and Applications*, 2003, pp. 108–114.
[31] A. Gress and R. Klein, "Efficient representation and extraction of 2-manifold isosurfaces using kd-trees," *Pacific Graphics*, vol. 00, p. 364, 2003.
[32] S. Schaefer, T. Ju, and J. Warren, "Manifold dual contouring," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 610–619, 2007.
[33] T. Ju and T. Udeshi, "Intersection-free contouring on an octree grid," in *Proceedings Pacific Graphics*, 2006.
[34] V. Pascucci and C. L. Bajaj, "Time-critical isosurface refinement and smoothing," in *Proceedings IEEE Symposium on Volume Visualization*. Salt Lake City, UT: IEEE Computer Society, October 2000, pp. 33–42.
[35] T. Lewiner, L. Velho, H. Lopes, and V. Mello, "Simplicial isosurface compression," in *Vision, Modeling, and Visualization*, Stanford, CA, November 2004, pp. 299–306.
[36] K. Weiss and L. De Floriani, "Diamond hierarchies of arbitrary dimension," *Computer Graphics Forum*, vol. 28, no. 5, pp. 1289–1300, 2009.
[37] H. Hege, M. Seeba, D. Stalling, and M. Zckler, "A generalized marching cubes algorithm," Konrad-Zuse-Zentrum für Informationstechnik Berlin, Tech. Rep., 1997. [Online]. Available: citeseer.ist.psu.edu/575196.html
[38] K. Bonnell, M. Duchaineau, D. Schikore, B. Hamann, and K. Joy, "Material interface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 4, pp. 500–511, 2003.
[39] G. M. Nielson, "Dual marching tetrahedra: Contouring in the tetrahedronal environment," in *Advances in Visual Computing*. Springer, 2008, pp. 183–194.