IEEE/ EG Symposium on Volume and Point-Based Graphics (2008)
H.- C. Hege, D. Laidlaw, R. Pajarola, O. Staadt (Editors)

Multiresolution Interval Volume Meshes

Kenneth Weiss[1] and Leila De Floriani[2]

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

Abstract

Interval volumes are a generalization of isosurfaces that represent the set of points between two surfaces. In this paper, we present a compact multi-resolution representation for interval volume meshes extracted from regularly sampled volume data sets. The multi-resolution model is a hierarchical tetrahedral mesh whose updates are based on the longest edge bisection (LEB) subdivision strategy for tetrahedra. Although our representation is decoupled from the scalar field, it maintains the implicit structure of the LEB model to efficiently encode mesh updates. Our representation efficiently supports selective refinement queries and requires significantly less storage than an encoding of the interval volume mesh at full resolution.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Hierarchy and geometric transformations I.3.6 [Computer Graphics]: Graphics data structures and data types

1. Introduction

An interval volume is a volumetric mesh representation that generalizes the notion of isosurfaces. Rather than representing a single surface within a scalar field, an interval volume represents the region enclosed between two isosurfaces. Thus, it is a hybrid between direct and indirect volume rendering approaches since it is able to exploit the spatial coherence inherent in the field. Since these meshes span an interval of values, they are less sensitive to small fluctuations in the scalar field, such as those due to noise or sampling artifacts and are especially relevant in medical applications, where object boundaries can cover multiple values. However, since it is a volumetric approach, the sizes of extracted interval volume meshes can quickly exceed the processing capabilities of commodity computers. Thus, a multiresolution model can help in locating regions relevant to specific tasks such as rendering and mesh extraction. Multiresolution models have been developed in the literature for representing 3D scalar fields for computational and storage efficiency, mostly based on nested hexahedral or tetrahedral meshes. The multi-resolution model efficiently supports the generation of simplified representations of the field (as isosurfaces or as meshes for direct volume rendering) at both uniform and variable resolution according to a user-selected accuracy criterion. Our purpose here is not only to generate adaptive representations of an interval volume, but also to produce a multi-resolution model of the interval volume for which we propose a compact and efficient encoding. This enables efficient inspection and manipulation of an interval volume at different resolutions without the need to maintain the entire multi-resolution model of the scalar field. We consider as a representation of the underlying field a multiresolution model based on a nested tetrahedral mesh, that we call a *Hierarchy of Diamonds* (Section 4). The hierarchy of diamond representation is a variant of longest edge bisection (LEB) subdivision whose modeling primitive, the *diamond*, encodes the collection of tetrahedra sharing a common longest edge rather than individual tetrahedra. We propose an efficient way to represent a diamond in terms of the coordinates of the midpoint of its longest edge that we will use also for encoding the multi-resolution model of the scalar field (Section 5).

The main contribution of this paper is a novel multiresolution model of an interval volume, that we call a *multiresolution interval volume mesh* (Section 6), that is decoupled from the values of the scalar field. This model assumes a representation of the scalar field as a hierarchy of diamonds, and exploits this representation to produce a compact and efficient multi-resolution description of the irregular volume mesh subdividing the interval volume. A multi-resolution interval volume mesh is obtained by encoding only those diamonds in the hierarchy of diamonds of the underlying field

whose associated scalar field values contribute to the interval volume. It also encodes the interval volume patches associated with such diamonds efficiently. The multi-resolution structure of the model, which describes the dependencies among diamonds, is implicit (see Sections 4 and 5). The resulting multi-resolution interval mesh is thus a significantly more compact representation when compared to the encoding of the interval volume mesh at full resolution. It provides a very efficient support for selective refinement queries (Section 7). Since it reduces both storage costs and preprocessing times, it a very relevant tool for transmission and/or offline inspection and rendering.

2. Background Notions

A 3D scalar field is given as a discrete set of points V in a domain Ω in the three-dimensional space, where one or more field values are associated with each point of V. V plus such field values form a *volume data set*. A volume data set is modeled as a mesh formed by polyhedral cells having their vertices at the data points along with an interpolating function defined over the cells of the mesh. We consider models based on tetrahedral meshes, in which linear interpolation is used over the tetrahedra forming the mesh, and which are also *conforming*. i.e. 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 (and, thus, discontinuities in extracted meshes) in correspondence to the boundary of adjacent cells of the underlying mesh.

One way of visualizing volume data sets is through isosurfaces. The isosurface of *isovalue* κ passes through all tetrahedral cells having at least one vertex whose associated field value is greater than κ, and at least one vertex whose value is less than κ.

An alternative method of visualizing regions of a volume dataset is through an *interval volume*, which is the set of points enclosed between two isosurfaces. Let $K := [\alpha, \beta]$ be defined as the interval between isovalues α and β, where $\alpha \leq \beta$. Then the interval volume Σ of *isorange* K within F, denoted as Σ_K or $\Sigma_{[\alpha,\beta]}$, is defined as

$$\Sigma_{[\alpha,\beta]} = F^{-1}([\alpha,\beta]) = \{x \in \Omega(F) | \alpha \leq F(x) \leq \beta\}.$$

Once a particular isorange K is chosen, it implicitly defines a ternary-valued scalar field R_K on each vertex $\mathbf{v} \in V$ whose values are defined by the relative values of F and K, namely:

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

Consequently, an interval volume mesh is bounded by two surfaces: the *lower surface* corresponds to the isosurface of isovalue α while the *upper surface* corresponds to the isosurface of isovalue β.

A tetrahedral mesh in which the tetrahedra are defined by the uniform subdivision of a tetrahedron into scaled copies of it is called a *nested tetrahedral mesh*. A special class of nested tetrahedral meshes are those generated by bisecting tetrahedra along their 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 splitting $\mathbf{t}$ along the plane defined by the middle point of its longest edge $\mathbf{e}$ and the two vertices of $\mathbf{t}$ which are not endpoints of $\mathbf{e}$. When this rule is applied recursively to an initial decomposition of a cubic domain into six tetrahedra sharing a diagonal of the cube it generates three congruent classes of tetrahedra, each with a single longest edge. We denote the *class* of tetrahedra congruent to those sharing a diagonal of the base cube as 0-*tetrahedra*, and the tetrahedra congruent to the ones obtained by splitting an i-tetrahedron as $(i+1)$-*tetrahedra* for $i \in \{0,1\}$. The tetrahedra obtained by splitting 2-tetrahedra are congruent to 0-tetrahedra. Observe that any edge $\mathbf{e}$ of a tetrahedron $\mathbf{t}$ in these meshes is aligned with either (a) the diagonal of an axis aligned cube; (b) the diagonal of a face of such a cube, or (c) an edge of such a cube.

3. Related Work

Interval volumes were introduced concurrently by Fujishiro et al. [FMS95] and by Guo [Guo95]. Guo [Guo95] introduced *interval sets* as a bridge between direct volume rendering (DVR) and indirect volume rendering (e.g. isosurface extraction) techniques. Fujishiro et al. [FMS95, FMST96] extend the Marching Cubes algorithm [LC87] to extract interval volumes from cubic cells. Nielson and Sung [NS97] offer a case-based lookup table for interval volumes over tetrahedral meshes. They reduce the configuration space to 15 unique cases and provide a consistent global triangulation of the polyhedral interval volume patches. Bhaniramka et al. [BWC00] create interval volume cases for cubes by first extracting an isosurface from a four dimensional hypercube and then projecting the isosurface into an interval volume in the cube. Zhang et al. [ZBS03] introduce a technique for extracting adaptive interval volume meshes from volume data. Octree cells containing a boundary of the interval are triangulated using a modified dual contouring technique, while cells completely within the interval are triangulated by inserting Steiner points. Since a pointer-based octree is used, this technique is likely to have problems scaling to larger datasets.

A very large class of multi-resolution models for 3D scalar fields is provided by nested tetrahedral meshes. LEB meshes were introduced for domain decomposition in finite element analysis [Mau95], and have been applied in several applications, including scientific visualization [ZCK97, GDL*02]. The containment hierarchy among the tetrahe-

dra in an LEB mesh induces a natural tree representation, in which the nodes are tetrahedra and the two children of a tetrahedron **t** are the tetrahedra generated by bisecting **t** [ZCK97, GP00, GDL*02]. As an LEB mesh can be non-conforming, the issue here is to extract conforming meshes. Since tetrahedra in this model are subdivided along a longest edge, the cracking problem can be avoided by simultaneously subdividing all tetrahedra sharing that edge. Gerstner et al. [GP00] pre-compute a saturated error for each vertex splitting a tetrahedron in an LEB mesh. The saturation condition implicitly forces all longest-edge neighbors to split, thus ensuring a conforming mesh. This scheme does not enable general navigation on the extracted meshes but it is suitable for parallel implementation and front-to-back traversal. Cases describing the topology of isosurfaces within diamonds were also introduced in [GP00].

An alternative approach is to directly encode the set of all tetrahedra sharing a longest edge. Gregorski et al. [GDL*02] denote such a primitive as a *diamond*, and associate each diamond with the central vertex of its longest edge. This method was extended to time-varying datasets in [GSDJ04], where the temporal coherence is exploited for generating approximations to isosurfaces between consecutive time steps and in compressing the dataset. A diamond-based scheme was also used by Takahashi et al. [TTNF04] to accelerate construction of contour trees.

4. A Hierarchy of Diamonds

In an LEB mesh Σ, all tetrahedra sharing a common longest edge are called a *cluster* or a *diamond* [DM02]. We call the longest edge the *spine* of the diamond, and the midpoint $\mathbf{v}_c$ of the spine the *central vertex* of the diamond. Thus, diamonds can be uniquely identified by their spine or, equivalently, by their central vertex. We distinguish between the two vertices of a diamond's spine, which are common to all tetrahedra in the diamond and its remaining vertices, which we denote as the diamond's *belt* vertices. Figure 1 shows examples of diamonds: vertex $\mathbf{v}_c$ denotes the central vertex, the spine vertices are the extreme vertices of the spine (colored brown, red and green, respectively), and the remaining vertices (light blue) are the belt vertices.

Since each tetrahedron in a diamond mesh has a single longest edge, the tetrahedra in a diamond all belong to the same class of tetrahedra (see Section 2). As a consequence, there are three different congruence *classes* of diamonds: those with spines aligned with cube diagonals (*0-diamonds*, see Figure 1(a)), with face diagonals of a cube (*1-diamonds*, see Figure 1(b)) or with cube sides (*2-diamonds*, see Figure 1(c)). An i-diamond, for $i \in \{0,1,2\}$ is formed by $\{6,4,8\}$ tetrahedra, and contains $\{8,6,10\}$ vertices, respectively.

Given a regular volume data set, let us consider the collection T of all the tetrahedra generated from the initial subdivision of the cubic domain through the longest edge bisection

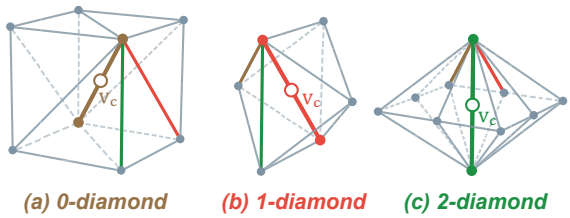


Figure 1: *The three classes of diamonds. Central vertices, (hollow circles labeled $\mathbf{v}_c$) are located at the midpoint of spines whose extreme vertices are the spine vertices. All other vertices (light blue) are belt vertices. (a) The spine of a 0-diamond (brown edge) is a cube diagonal, (b) the spine of a 1-diamond (red edge) is a square diagonal and (c) the spine of a 2-diamond (green edge) is a cube edge.*

rule. If we consider the collection Δ of all the diamonds associated with the longest edges of the tetrahedra in T, the containment relation between the tetrahedra in T induces a parent-child dependency relation over set Δ. A diamond δ' is a *parent* of another diamond δ if and only if some tetrahedra in δ are created by the splitting of at least one tetrahedron in δ'. If δ' is a parent of δ then δ is called a *child* of δ'.

The set Δ with the parent-child dependency relation defined over Δ can be easily shown to define a partial order relation and thus it can be described as a Directed Acyclic Graph (DAG) (see [DM02] for a proof). The DAG describing the dependencies between diamonds has a fixed structure. With the exception of diamonds whose spines lie on the domain boundary, 0-diamonds always have 3 parents and 6 children, while 1-diamonds have 2 parents and 4 children and 2-diamonds have 4 parents and 8 children.

In the literature, a hierarchy of diamonds is built on a grid of dimensions $(2^N+1)^3$, where N is the maximum level, LEVEL_{Max}, of the hierarchy Δ. Since the coordinates of diamonds can be inferred from their locations within the grid, they do not need to be explicitly encoded, and it is sufficient to encode the scalar and error values at each vertex of the grid. For each vertex $\mathbf{v}$, the error associated with $\mathbf{v}$ is the error for approximating the domain of the original scalar field by the diamond δ whose central vertex is $\mathbf{v}$. Specifically, in a pre-processing step, we assign to $\mathbf{v}$ the maximum of the interpolation errors calculated at all points in the domain of its associated diamond δ. For efficiency in extracting meshes, the minimum and maximum field values within each diamond δ are also maintained.

5. Representing Diamonds

We describe here a novel interpretation of the binary representation of a diamond's central vertex, $\mathbf{v}_c$, which provides all information required to compute the local mesh topology of a diamond, including the location of its ver-

tices, and the central vertices of its parents and children in terms of scaled offsets from $\mathbf{v}_c$. Thus, as opposed to previous schemes [GDL*02] which required explicit storage of a diamond's type and level, the only geometric overhead associated with a diamond is its central vertex (used to index the diamond) and a few bits of bookkeeping that we use for selective refinement queries.

The *level* of an i-diamond δ is equal to the number of i-diamond ancestors of δ in the DAG, i.e. its depth in the DAG modulo three. The scale σ of a diamond δ is defined in terms of its level as $\sigma_\delta = \text{LEVEL}_{Max} - \text{LEVEL}_\delta$, and is encoded as the minimum of the number of trailing zeros among all coordinates of $\mathbf{v}_c$ (see σ in eq. 1). Consequently, the rightmost σ bits of $\mathbf{v}_c$ are all zero, but at least one of the bits at location $\sigma+1$ is nonzero.

$$\mathbf{v}_c = \begin{bmatrix} v_x &= x_1x_2\ldots x_n & d_{x_1}d_{x_2} & \underbrace{00\ldots0}_{\sigma} \\ v_y &= y_1y_2\ldots y_n & d_{y_1}d_{y_2} & 00\ldots0 \\ v_z &= z_1z_2\ldots z_n & \underbrace{d_{z_1}d_{z_2}}_{\tau} & 00\ldots0 \end{bmatrix} \quad (1)$$

We use the two bits at positions $\sigma+1$ and $\sigma+2$ to uniquely associate a *type* with each diamond (i.e. τ in Eq. 1). Since diamonds in Δ have three coordinates, and τ has two bits for each coordinate, there are $(2^2)^3 = 64$ possible diamond types. However, the restriction that at least one of the rightmost bits of τ is nonzero reduces this number by eight, yielding 56 unique diamond types. The *class* of δ is encoded within τ by the number of zeros at position $\sigma+1$ of $\mathbf{v}_c$. The remaining $n = \text{LEVEL}_{Max} - (\sigma+2)$ bits in each component of $\mathbf{v}_c$ are used only to distinguish diamonds.

A diamond's type τ and its scale σ are used in conjunction with a table of unscaled offsets from its central vertex to find the vertices of δ as well as its parents and children at runtime. These offsets vectors all have components $f_i \in \{-1,0,1\}$. Since all vertices of the diamond mesh have integer coordinates, and there are only 56 types of diamonds, we precalculate these offsets and access them via a lookup table. This table is generated by using an LEB tetrahedral subdivision scheme (such as the one given in [Mau95]). When subdividing tetrahedra, we keep track the central vertices of parent and children tetrahedra, and use Equation (1) to find the unscaled offsets between them. The explicit location of vertex $\mathbf{p}$ of diamond δ at scale σ and at unscaled offset $\vec{f}$ is computed at runtime as:

$$\mathbf{p} = \mathbf{v}_c + 2^\sigma * \vec{f}. \quad (2)$$

Example Consider the diamond δ (in 2D) whose central vertex $\mathbf{v}_c$ has coordinates $(72,20)$. From Equation 1, we determine is scale, σ, and type, τ, as follows:

$$\mathbf{v}_c = \begin{bmatrix} 72 \\ 20 \end{bmatrix} = \begin{bmatrix} 100\ 10\ 00_2 \\ 001\ 01\ 00_2 \end{bmatrix} = \begin{bmatrix} 100 & 10 & 00 \\ 001 & \underbrace{01}_{\tau} & \underbrace{00}_{\sigma} \end{bmatrix}.$$

Thus, $\sigma = 2$ and $\tau = (10_2, 01_2) = (2,1)$. Since there is a single 0-bit at position $\sigma+1$, δ is a 1-diamond. We can use Equation (2) to find the central vertex of δ's parent diamond δ_p located at unscaled offset $\vec{f} = (-1,0)$ (see Figure 2). We first scale $\vec{f}$ by 2^σ and then add the result to $\mathbf{v}_c$, so

$$\mathbf{v}_c(\delta_p) = (72,20) + 2^2 \times (-1,0) = (68,20).$$

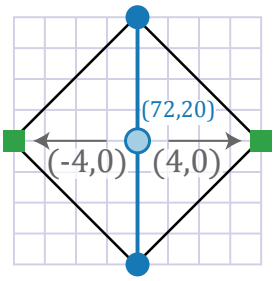


Figure 2: *The vertices, parents and children of a diamond δ are located at scaled offsets from its central vertex. These offsets are accessed from a table and indexed by the type τ of δ. Here, the parents of a diamond with central vertex (72,20) are located four units away on the x-axis.*

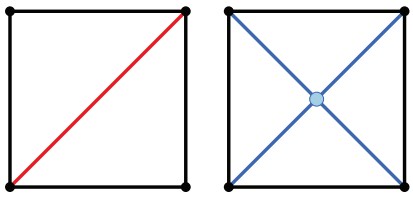
6. A Compact Representation for Multi-resolution Interval Volume Meshes

In this Section, we introduce the multi-resolution model for an interval volume mesh and a compact representation for such models based on the diamond encoding introduced in the previous section. Recall that an interval volume mesh Σ_K with isorange $K := [\alpha,\beta]$ extracted from the scalar field F encodes the set of points from the domain Ω between isosurfaces α and β. A multi-resolution representation for Σ_K consists of a coarse representation of Σ_K, a set of refinement modifications, and a dependency relation among the modifications. When the scalar field is modeled by a hierarchy of diamonds Δ, the coarse mesh is the interval volume patch associated with the root diamond, the modifications correspond to the subdivision of diamonds from the hierarchy that are intersected by the interval volume, and the dependency relation is induced by the parent-child relation in the diamonds of Δ. As such, a multi-resolution interval volume can be described by a DAG of diamonds, which is a subgraph of the DAG describing Δ. The resulting multi-resolution model can be encoded as a Multi-Tessellation, in which the DAG is implicitly encoded and the modifications are stored as the collection of tetrahedra in the interval volume generated by diamond splitting [Mag00].

In our compact representation presented here, we exploit the fact that the modifications are generated from a hierarchy of diamonds and that only a subset of the original diamonds are involved. Specifically, we encode only those diamonds that have a non-empty intersection with the interval volume at any level, plus those that have at least one non-empty de-

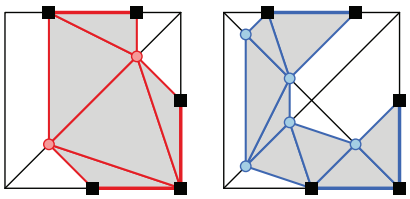
scendant. Such diamonds are considered *relevant* to the isorange K. Since all boundary vertices of Σ_K lie on edges of Δ, they can be encoded using interpolation coefficients. Since internal vertices of Σ_K coincide with vertices of Δ they do not require explicit storage. Thus, our multi-resolution representation of Σ_K consists of the set of relevant diamonds as well as an array of interpolation coefficients corresponding to the boundary vertices of Σ_K, i.e. those lying on the upper or lower surface of Σ_K.

To define a compact description of the modifications in a multi-resolution interval volume mesh we need to analyze how an interval volume mesh is locally updated in correspondence to a diamond split. When subdividing a diamond δ, its spine is removed, a new vertex is inserted at its central vertex $\mathbf{v}_c$ and edges connecting $\mathbf{v}_c$ to all vertices of δ are inserted (see Figure 3). Since these new edges are entirely within the domain of a single diamond, they can be uniquely associated with that diamond. Let us denote these new edges as the *subdivision edges* of δ. We observe that the vertices, edges and faces on the boundary of δ are unaffected by the subdivision.



(a) Before Subdivision (b) After Subdivision

Figure 3: *Subdivision of a diamond (in 2D). When a diamond δ subdivides, its spine (red edge in (a)) is removed, a vertex (light blue) is inserted at its center and edges (blue) are inserted from the center to all vertices of δ.*



(a) Before Subdivision (b) After Subdivision

Figure 4: *Updating an interval volume patch (in 2D). Diamond-based updates to the interval volume remove vertices lying on the diamond's spine (red circles in (a)) and add vertices lying on the subdivision edges of the diamond (blue circles in (b)).*

We denote the intersection of a diamond δ and an interval volume mesh Σ_K as an *interval volume patch* and observe that the edges of δ that are intersected by Σ_K (and hence the vertices of the local patch) are entirely determined by R_K, that is, the relative values of the field F at the vertices of δ with respect to isorange K (see Section 2). Given the interval volume patch associated with δ, a local update requires only the removal of the patch vertices lying on δ's spine, the insertion of patch vertices lying on the subdivision edges of δ and a local re-triangulation of the patch (see Figure 4). Thus, given the interval volume patch associated with an unsubdivided diamond δ with central vertex $\mathbf{v}_c$, an update requires only the value of R_K at $\mathbf{v}_c$ as well as the interpolation coefficients of the boundary vertices of Σ_K along the subdivision edges of δ. These coefficients are stored in a global array to ensure that updates have a fixed size. Since the number of intersected subdivision edges of δ is uniquely determined by the values of R_K, each update requires only a single pointer into this array.

An update u_δ associated with a diamond $\delta \in \Delta$, is indexed using the coordinates of its central vertex $\mathbf{v}_c$ which incurs a six byte overhead per update. Also, to support selective refinement queries, the approximation error of u_δ is encoded as well. The approximation error of u_δ is the maximum interpolation error over all points of V within the domain of δ (as described in Section 4) and is quantized to 14 bits per update. The relative value R_K of $\mathbf{v}_c$ requires 2 bits, and the pointer into the interpolation array requires 4 bytes. Finally, the interpolation coefficients are quantized to 8 bits each. Thus, our multi-resolution interval volume representation requires a single byte per vertex and 12 bytes for each diamond-based update.

7. Querying Multi-Resolution Interval Volumes

The basic query on a multi-resolution model is selective refinement, which consists of extracting the mesh of smallest size that satisfies an application-dependent predicate called the Level-Of-Detail (LOD) criterion. An LOD criterion ϕ defines whether a given update is needed to achieve the required mesh resolution and can depend on factors such as approximation error, scalar value, spatial location, distance to the view point and projected screen error of an update. Selective refinement is performed through a top-down traversal of the DAG describing the multi-resolution model, and ensures that all ancestors of a diamond are subdivided before subdividing the diamond. Numerous variations of selective refinement exist, including those that allow incremental and priority-based refinement of meshes. In the case of multi-resolution interval volume meshes we need to show how we triangulate the interval volume patch associated with a diamond δ and how this triangulation is updated after the subdivision of δ.

The triangulation of an interval volume patch associated with a diamond δ is determined by the relative values R_K of the vertices of δ, which, when taken together, we denote

as the *bitpattern* of δ. We have adapted the tetrahedra-based cases of Nielson and Sung [NS97] to triangulate the interval volume patches within diamonds. This technique guarantees a globally consistent tetrahedralization of the polyhedral patch through a lexicographic ordering of the patch vertices. Consequently, for each diamond δ in an approximation Σ'_K of Σ_K, a globally consistent triangulation of Σ'_K is achieved through table lookups on the bitpattern of δ.

As indicated in the previous section, updating an interval volume patch associated with a diamond δ with central vertex $\mathbf{v}_c$ requires the generation of patch vertices on the intersected subdivision edges of δ as well as a re-triangulation of the patch. Given the update u_δ associated with the subdivision of diamond δ, we first augment the bitpattern of δ with the relative value R_K of $\mathbf{v}_c$. Next, we use the pointer into the interpolation array to find the set of interpolation coefficients required to generate the new vertices on the subdivision edges of δ. The augmented bitpattern is sufficient to determine the number of new vertices as well their corresponding edges. Finally, for each intersected subdivision edge with vertices $\mathbf{v}$ and $\mathbf{v}_c$ and interpolation coefficient γ, a new vertex $\mathbf{p}$ is generated as $\mathbf{p} = \gamma \cdot \mathbf{v} + (1-\gamma) \cdot \mathbf{v}_c$.

To simplify the representation, we only allow unsubdivided diamonds to contribute to the approximated interval volume mesh Σ'_K. Thus, upon subdivision of a diamond δ, portions of the new interval volume patch are distributed among the children of δ. This is accomplished via a mapping between the edges and vertices of parent and children diamonds that we denote as *duets* (see Figure 5). For a diamond δ_c with k parents, we can split the domain of δ_c into k pairs of tetrahedra, the duets, such that both tetrahedra in each duet were generated during the splitting of a single parent diamond of δ_c. A duet between non-boundary diamonds δ_p and δ_c consists of two adjacent tetrahedra whose shared face includes the two vertices of δ_c's spine and the central vertex of δ_p and contains 5 vertices and 9 edges (see Figure 5). Boundary duets have only a single tetrahedron. Bit

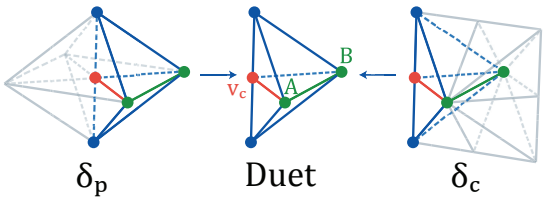


Figure 5: *A* parent-child duet *maps the common edges and vertices of a subdivided diamond δ_p and its child diamond δ_c. The common face of the duet contains the central vertex v_c of δ_p (red vertex) and the two spine vertices of δ_c (green vertices). The map contains 5 vertices and 9 edges.*

patterns corresponding to the vertex mappings of a duet are propagated to children diamonds using a few efficient bit operations, and surface vertices are passed to children diamonds using the edge mappings of duets.

8. Experimental Results

In this Section, we present experimental results demonstrating the compactness and efficiency of our multi-resolution interval volume representation. First we compare our representation to the interval volume tetrahedral mesh at full resolution described as an indexed data structure. In an indexed tetrahedral mesh, vertices are represented by their three floating point coordinates and tetrahedra by the indices of their four vertices. Thus, each vertex requires 12 bytes and each tetrahedron 16 bytes. Recall that our compact multi-resolution interval volume representation requires one byte per vertex and 12 bytes per update/diamond. We experimented with intervals of varying sizes on several different types of datasets, including distance fields (Bunny dataset), as well as single byte (engine, neghip) and double byte volume datasets (tooth, buckyball). Table 1 shows the dimensions of each tested dataset as well as the isorange K and the sizes of the two representations. It is evident from Table 1 that our multi-resolution representation requires about $1/8^{th}$ the space of the mesh at full resolution. Interestingly, this ratio is consistent even when the interval is narrow (as in the *Bunny4* dataset), or it only contains a single surface (as in *Bunny3* and *Bunny5* datasets). Since we often desire meshes where the surface detail is refined, but the internal cells are as large as possible, we used an LOD criterion ϕ that subdivides the diamonds containing boundary isovalues of the interval K (i.e. α or β) rather than those containing only internal values of K for most of our experiments. However, the *Bunny5* example (row 5 of Table 1) uses an LOD criterion with uniform error over the entire interval, and demonstrates the same characteristic benefits of our method.

Table 2 compares selective refinement query times from the hierarchy of diamonds Δ to those of our compressed representation Σ where the LOD criterion ϕ is a percentage of the the maximum error. Our representation, on average is noticeably quicker, and typically requires only 70-80% the time as the hierarchy of diamonds Δ. This is most likely due to the fact that all pertinent information can be derived from a single access to the diamond rather than having to access the scalar values associated with the diamond's vertices. Furthermore, the compact representation typically requires only a fraction of the diamonds in Δ, and since it implicitly encodes all relevant diamonds, does not require an array of minimum and maximum values for efficiency. On the other hand, our proposed representation is less appropriate for tasks that require frequent changes to the isorange K, such as the data exploration stage. Since relevant diamonds at all scales of Δ must be accessed during creation, creation times for the multiresolution representation are of the same order as extraction times for the 0 error mesh from the orig-

		Isorange	Multires			Max Res			
Dataset	Dims	[α,β]	$\|V\|$	$\|\delta\|$	MB	$\|V\|$	$\|T\|$	MB	Savings
Bunny1	201^3	[10,10]	1.1 M	808 K	10.3	1.2 M	4.6 M	84	8.13 x
Bunny2		[0,30]	850 K	616 K	7.9	938 K	3.4 M	63	8.06 x
Bunny3		[0,∞]	671 K	486 K	6.2	735 K	2.7 M	49	7.96 x
Bunny4 (Thin Shell)		[-0.5,0.5]	1.3 M	548 K	7.6	1.1 M	3.2 M	61	8.1 x
Bunny5 (Uniform LOD)		[0,∞]	671 K	2.3 M	27	2.6 M	14 M	238	8.81 x
Engine1	256^3	[175,256]	331 K	205 K	2.7	325 K	1.1 M	21	7.87 x
Engine2		[55,175]	1.7 M	1 M	13.1	1.7 M	6.2 M	114	8.67 x
Tooth	256^2x161	[440,1290]	350 K	252 K	3.2	364 K	1.3 M	24	7.57 x
Neghip	64^3	[59.1,124.1]	90 K	45 K	.6	81 K	262 K	4.9	8.27 x
Buckyball	128^3	[128,188]	595 K	316 K	4.2	553 K	2 M	37	8.82 x

Table 1: *Comparison between the compact multiresolution representation and an indexed tetrahedral mesh representation.*

	30%		3%		.3%		.03%		.01%		0%		
	Δ	Σ	Δ	Σ	Δ	Σ	Δ	Σ	Δ	Σ	Δ	Σ	Avg. Savings
Bunny	.1	.1	.43	.4	1	.74	1.2	.78	1.1	.8	1.2	.82	77%
Engine	.05	.04	1.4	1.8	5.6	5	6.3	4.9	6.8	4.9	6.1	4.9	87%
Tooth	.17	.13	1.7	1.6	8.9	5.7	12	6.6	12	7.3	16	7.6	66%
Neghip	.12	.09	.27	.25	3.7	4.3	24	18	34	23	37	25	82%
Bucky	3.5	1.5	23	15	27	20	26	22	25	24	28	30	78%

Table 2: *Comparison times (in seconds) between extracting interval volume meshes with uniform relative error from the hierarchy of diamonds representation (Δ) and from the decoupled multiresolution interval volume representation (Σ).*

inal scalar field. All timing experiments were performed on a 2 GhZ Pentium Core 2 Duo processor with 4 GB RAM.

Figure 6 shows a DVR rendering of an interval volume mesh extracted from the engine dataset with a 1% uniform error around the boundaries of the interval [55,175]. Figure 7 illustrates a variable resolution interval volume mesh extacted from the bunny dataset using a region of interest (ROI) query around the bunny's head. Diamonds within the ROI have an error of 0 while those outside the region can have arbitrary error.

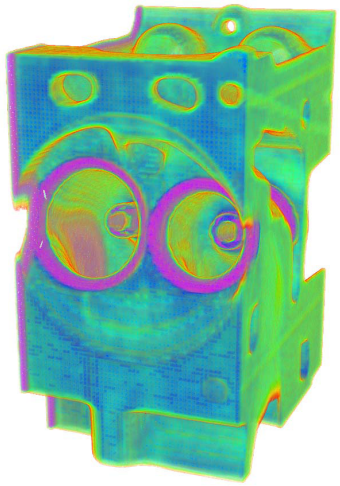


Figure 6: *Volume rendering of the Engine2 dataset with isorange $K = [55, 175]$ and uniform error less than 1%.*

9. Concluding Remarks

We have presented a multi-resolution representation for interval volume meshes defined on hierarchies of diamonds. This representation uses the implicit dependency relation of the hierarchy to yield compact meshes while enabling efficient selective refinement queries. Empirically, we demonstrated that these meshes are often more than eight times smaller than an indexed representation of the mesh at full resolution. We have also shown that selective refinement queries on the multi-resolution interval volume meshes are more efficient in terms of size and performance than queries on the original hierarchy of diamonds.

Although the decoupled representation significantly reduces the storage requirements of each modification we would like to further reduce the overhead associated with each diamond. In our experiments, we observed that the hierarchy requires storing a significant number of diamonds that do not intersect the interval volume but are necessary since they are ancestors of diamonds that contain pieces of the volume. We are looking into representations that avoid the need

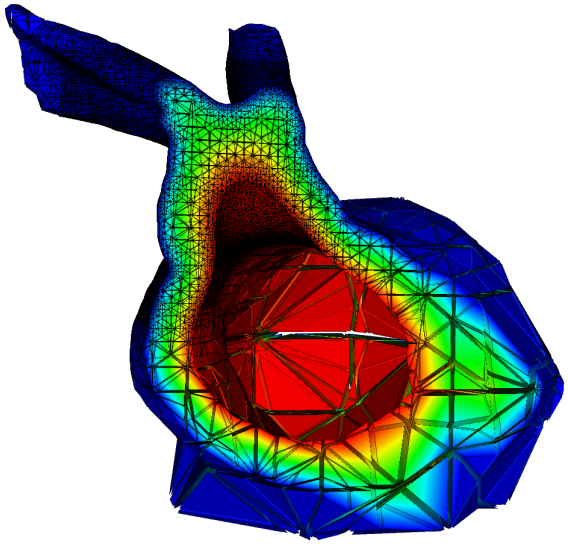


Figure 7: *Region based variable LOD on the bunny model (around the head) with isorange $K = [0, 20]$. The tetrahedra are shown shrunken and the model is clipped to show detail.*

to represent such diamonds and have achieved encouraging results toward this goal. We are also seeking to reduce the geometric overhead associated with each diamond, including the need to encode the central vertices of diamonds and the location of the interpolation coefficients of the vertices of volume patches.

Acknowledgments

We would like to thank the anonymous reviewers for their many helpful suggestions. This work has been partially supported by the National Science Foundation under grant CCF-0541032, by the MIUR-FIRB project SHALOM under contract number RBIN04HWR8 and by the MIUR-PRIN project on modeling of scalar fields and digital shapes. The Engine, Tooth and Neghip models are courtesy Stefan Roettger, the Buckyball model is courtesy AVS inc., and the Bunny model is courtesy of Ramani Duraiswami and Nail Gumerov.

References

[BWC00] BHANIRAMKA P., WENGER R., CRAWFIS R.: Isosurfacing in higher dimensions. In *Proceedings IEEE Visualization 2000* (October 2000), IEEE Computer Society, pp. 267–273.

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

[FMS95] FUJISHIRO I., MAEDA Y., SATO H.: Interval volume: a solid fitting technique for volumetric data display and analysis. *Proceedings IEEE Visualization'95* (1995), 151.

[FMST96] FUJISHIRO I., MAEDA Y., SATO H., TAKESHIMA Y.: Volumetric data exploration using interval volume. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (1996), 144–155.

[GDL*02] GREGORSKI B., DUCHAINEAU M., LINDSTROM P., PASCUCCI V., JOY K.: Interactive view-dependent rendering of large isosurfaces. In *Proceedings IEEE Visualization 2002* (San Diego, CA, October 2002), IEEE Computer Society.

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

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

[Guo95] GUO B.: Interval Set: A Volume Rendering Technique Generalizing Isosurface Extraction. *Proceedings of the 6th conference on Visualization'95* (1995).

[LC87] LORENSEN W., CLINE H.: Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), 163–169.

[Mag00] MAGILLO P.: *The MT (Multi-Tessellation) Package*. Department of Computer and Information Sciences (DISI), University of Genova, Genova, Italy, http://www.disi.unige.it/person/MagilloP/MT/index.html, January 2000.

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

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

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

[ZBS03] ZHANG Y., BAJAJ C., SOHN B.: Adaptive and quality 3D meshing from imaging data. *Proceedings of the eighth ACM symposium on Solid modeling and applications* (2003), 286–291.

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