Multiresolution Representation of Datasets with Material Interfaces

Benjamin F. Gregorski[1], David E. Sigeti[3], John Ambrosiano[3], Gerald Graham[3], Murray Wolinsky[3], Mark A. Duchaineau[2], and Bernd Hamann[1] Kenneth I. Joy[1]

[1] Center for Image Processing and Integrated Computing (CIPIC)
Department of Computer Science,
University of California,
Davis, CA 95616-8562, USA

[2] Center for Applied Scientific Computing (CASC) Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551, USA
Lawrence Livermore National Laboratory

[3] Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Abstract. We present a new method for constructing multiresolution representations of data sets that contain material interfaces. Material interfaces embedded in the meshes of computational data sets are often a source of error for simplification algorithms because they represent discontinuities in the scalar or vector field over mesh elements. By representing material interfaces explicitly, we are able to provide separate field representations for each material over a single cell. Multiresolution representations utilizing separate field representations can accurately approximate datasets that contain discontinuities without placing a large percentage of cells around the discontinuous regions. Our algorithm uses a multiresolution tetrahedral mesh supporting fast coarsening and refinement capabilities; error bounds for feature preservation; explicit representation of discontinuities within cells; and separate field representations for each material within a cell.

1 Introduction

Computational physics simulations are generating larger and larger amounts of data. They operate on a wide variety of input meshes, for example rectilinear meshes, adaptively refined meshes for Eulerian hydrodynamics, unstructured meshes for Lagrangian hydrodynamics and arbitrary Lagrange-Eulerian meshes. Often, these data sets contain special physical features such as material interfaces, physical boundaries, or thin slices of material that must be preserved when the field is simplified. In order to ensure that these features are preserved, the simplified version of the data set needs to be constructed using strict L^∞ error bounds that prevent small yet important features from being eliminated.

Data sets of this type require a simplification algorithm to approximate data sets with respect to several simplification criteria. The cells in the approximation must satisfy error bounds with respect to the dependent field variables over each mesh cell, and to the representation of the discontinuities within each cell. In addition, the simplification algorithm must be able to deal with a wide range of possible input meshes.

We present an algorithm for generating an approximation of a computational data set that can be used in place of the original high-resolution data set generated by the simulation. Our approximation is a resampling of the original data set that preserves user-specified as well as characteristic features in the data set and approximates the dependent field values to within a specified tolerance.

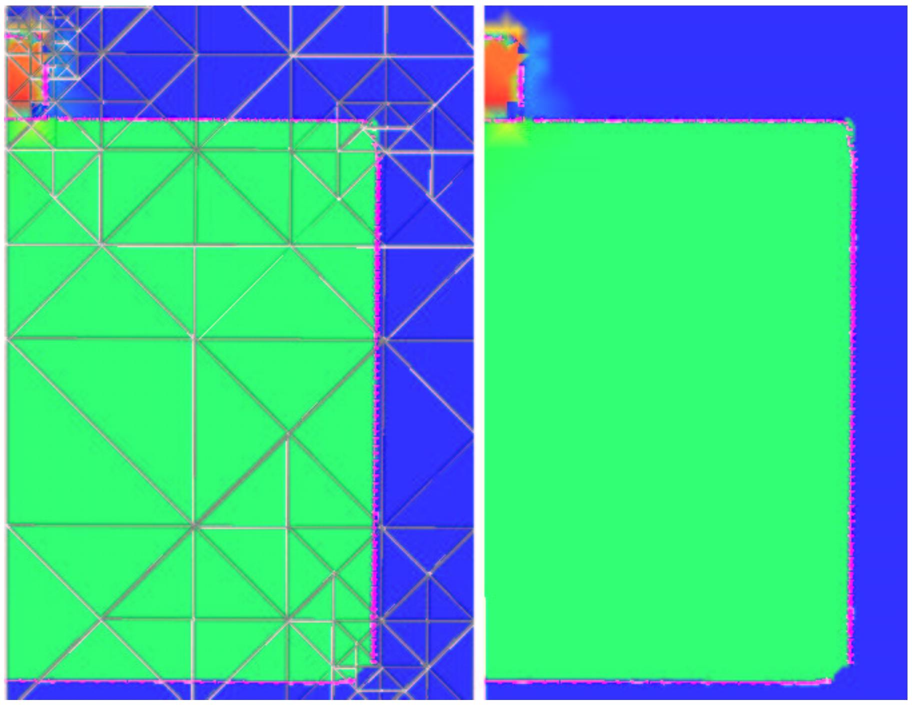

Fig. 1. Cross section of a density field approximated using explicit interface representations and separate field representations. The left picture shows the field along with the approximating tetrahedral mesh. (interface error = 0.15).

2 Related Work

Hierarchical approximation techniques for triangle meshes, scattered data, and tetrahedral meshes have matured substantially over recent years. The approximation of material interfaces is similar to the approximation or simplification of large polygonal meshes. The approximating mesh represents the large mesh to within a certain error tolerance using a substantially smaller number of triangles. Mesh approximation and simplification algorithms can be divided into decimation techniques and remeshing techniques.

Decimation based techniques attempt to simply the existing geometry by removing vertices, edges or faces and evaluating an error function that determines the fidelity of the new mesh. a large amount of research has been done in the field of surface simplification. In [9] an iterative triangle mesh decimation method is introduced. Triangles in nearly linear regions are identified and collapsed to a point that is comoputed in a locally optimal fashion. In [11], Hoppe describes a progressive mesh simplification method for triangle meshes. An arbitrary mesh is simplified through a series of edge collapse operations to yield a very simply base mesh. Heckbert and Garland present a comprehensive survey of these techniques in [10]. In more recent work, Heckbert and Garland [7] use a quadric error metric for surface simplification. They use vertex pair collapse operations to simplify triangle meshes and they use *quadric matrices* that define a quadric object at each vertex to control the error of the simplified surfaces. In [6], they use the same technique to simplify meshes that have associated colors and textures. Hoppe [12] uses a modified quadric metric for simplifying polygonal meshes while preserving surface attributes such as normal vectors, creases, and colors. Lindstrom [16] developed out-of-core techniques for simplifying large meshes that do not fit in main memory.

Decimation based techniques start with the existing high resolution geometry and remove features until a specified error tolerance has been reached. Another approach to mesh approximation is to generate a completely new mesh through a remeshing or fitting process. The remeshing process starts with a coarse mesh and adds in geometry until the surface approximates the original model to within a specified error tolerance. Lee et al. [15] use a mesh parameterization method for surface remeshing. A simplified base mesh is constructed and a subdivision surface is fitted to the original model. Kobbelt et al. [14] use a shrink wrapping approach to remesh polygonal meshes with subdivison surfaces. Multiresolution methods for reconstruction and simplification have also been explored using subdivision techniques and wavelets [5] and [4].

Our approximation of material interfaces falls into the category of remeshing techniques. As described in Section 5, the material interfaces are given as triangle meshes. Within each of our cells we construct a piecewise linear approximation of the material interfaces to within a specified error tolerance based on the distance between the original mesh and our approximation.

Simplification of tetrahedral meshes has been discussed in [20], [2], [19], [17], and [18]. Zhou et al. [20] present the multiresolution tetrahedral framework that is the basis of our simplification algorithm. This structure is also used by Gerstner and Pajarola [8] for multiresolution iso-surface visualization that preserves topological genus. (This is further discussed in Section

3). Cignoni at al. [2] describe a multiresolution tetrahedral mesh simplification technique built on scattered vertices obtained from the initial dataset. Their algorithm supports interactive level-of-detail selection for rendering purposes. Trotts et al. [19] simplify tetrahedral meshes through edge-collapse operations. They start with an initial high-resolution mesh that defines a linear spline function and simplify it until a specified tolerance is reached. Staadt and Gross [18] describe progressive tetrahedralizations as an extension of Hoppe's work. They simplify a tetrahedral mesh through a sequence of edge collapse operations. They also describe error measurements and cost functions for preserving consistency with respect to volume and gradient calculations and techniques for ensuring that the simplification process does not introduce artifacts such as intersecting tetrahedra. Different error metrics for measuring the accuracy of simplified tetrahedral meshes have been proposed. Lein et al. [13] present a simplification algorithm for triangle meshes using the Hausdorff distance as an error measurement. They develop a symmetric adaption of the Hausdorff distance that is an intuitive measure for surface accuracy.

3 Multiresolution Tetrahedral Mesh

The first basis for our simplification algorithm is the subdivision of a tetrahedral mesh as presented by Zhou et al. [20]. This subdivision scheme has an important advantage over other multiresolution spatial data structures such as an octree as it makes it easy to avoid introducing spurious discontinuities into representations of fields. The way we perform the binary subdivision ensures that the tetrahedral mesh will always be a conformant mesh, i.e., a mesh where all edges end at the endpoints of other edges and not in the interior of edges. The simplest representation for a field within a tetrahedral cell is given by the unique linear function that interpolates field values specified at the cell's vertices. In the case of a conformant mesh, this natural field representation will be continuous across cell boundaries, resulting in a globally C^0 representation.

We have generalized the implementation presented by Zhou et al. by removing the restriction that the input data needs to be given on a regular rectilinear mesh consisting of (2^N+1)x(2^N+1)x(2^N+1) cells. A variety of input meshes can be supported by interpolating field values to the vertices of the multiresolution tetrahedral mesh. In general, any interpolation procedure may be used. In some cases, the procedure may be deduced from the physics models underlying the simulation that produced the data set. In other cases, a general-purpose interpolation algorithm will be appropriate.

We construct our data structure as a binary tree in a top-down fashion. Data from the input data set, including grid points and interface polygons, are assigned to child cells when their parent is split.

The second basis for our algorithm is the ROAM system, described in [3]. ROAM uses priority queue-driven split and merge operations to provide optimal real-time display of triangle meshes for terrain rendering applications. The tetrahedral mesh structure used in our framework can be regarded as an extension to tetrahedral meshes of the original ROAM data structure for triangle meshes.

Since our data structure is defined recursively as a binary tree, a representation of the original data can be pre-computed. We can utilize the methods developed in ROAM to efficiently select a representation that satisfies an error bound or a desired cell count. This makes the framework ideal for interactive display. Given a data set and polygonal representations for the material interfaces, our algorithm constructs an approximation as follows:

1. Our algorithm starts with a base mesh of six tetrahedra and associates with each one the interface polygons that intersect it.
2. The initial tetrahedral mesh is first subdivided so that the polygonal surface meshes describing the material interfaces are approximated within a certain tolerance. At each subdivision, the material interface polygons lying partially or entirely in a cell are associated with the cell's children; approximations for the polygons in each child cell are constructed, and interface approximation errors are computed for the two new child cells.
3. The mesh is further refined to approximate the field of interest, for example density or pressure, within a specified tolerance.

For the cells containing material interfaces, our algorithm computes a field representation for each material. This is done by extrapolating *ghost* values for each material at the vertices of the tetrahedron where the material does not exist. A ghost value is an educated guess of a field value at a point where the field does not exist. When material interfaces are present, field values for a given material do not exist at all of the tetrahedron's vertices. Since tri-linear approximation over a tetrahedron requires four field values at the vertices, extra field values are needed to perform the interpolation. Thus for a given field and material, the ghost values and the existing values are used to form the tri-linear approximation within the tetrahedron.

This is illustrated in Figure 2 for a field sampled over a triangular domain containing two materials. For the field sampled at V_0 and V_1, a ghost value at vertex V_2 is needed to compute a linear approximation of the field over the triangle. The approximation is used only for those sample points that belong to this material. Given a function sampled over a particular domain, the ghost value computation extrapolates a function value at a point outside of this domain. When the field approximation error for the cell is computed, the separate field representations, built using these ghost values, are used to calculate an error for each distinct material in the cell. The decomposition process of a cell that contains multiple materials consists of these steps:

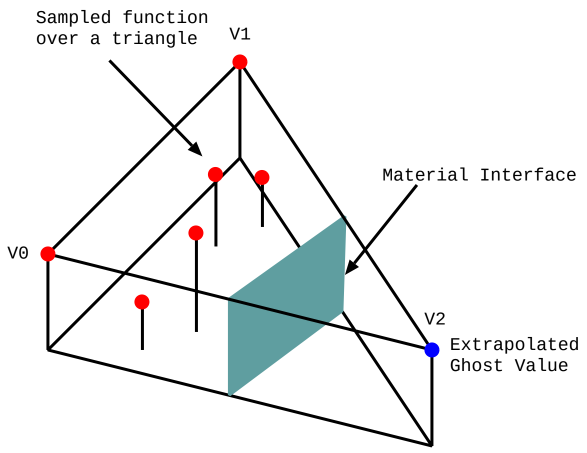

Fig. 2. 2D Ghost Value Computation

1. The signed distance values and ghost values for the new vertex are computed when the vertex is created during a split or subdivision operation. This is done by examining those cells that share the edge being split.
2. The interface representations, i.e., triangle meshes, are associated with the child cells, and the approximating representations of interfaces and their associated errors are computed.
3. The field error for each of the materials is computed, and the maximum value of these errors is defined as the overall error associated with a cell containing multiple materials.

4 General Multiresolution Framework

A multiresolution framework for approximating numerically simulated data needs to be a robust and extensible. The framework must be capable of supporting the wide range of possible input structures used in the simulations and the wide range of output data generated by these simulations. The following properties and operations are desirable for such a framework:

1. **Interactive transition between levels of detail**. The ability to quickly move between different levels of detail allows a user to select a desired approximation at which to perform a calculation (for a visualization application).
2. **Strict L^∞ error bounds**. Strict error bounds prevent small yet important features from being averaged or smoothed out by the approximation process.
3. **Local and adaptive mesh refinement and local error computations**. Local mesh refinement allows the representation to be refined only

in the areas of interest while keeping areas of little interest at relatively lower resolutions. This is essential for maintaining interactivity and strict cell count on computers with limited resources. The error calculations for datasets consisting of millions or billions of cells should not involve a large amount of original data.

4. **Accommodating different mesh types**. Computational simulations are done on a large variety of mesh structures. and it is cumbersome to write a multiresolution algorithm for each specific structure. In order for a framework to be useful it should be easily adaptable to a broad class of input meshes.
5. **Explicit representation of field and/or material discontinuities**. Discontinuities are very important in scientific datasets and very often need to be preserved when the datasets are approximated. A multiresolution framework should support the explicit representation and approximation of these discontinuities.

Our multiresolution recursive tetrahedral framework satisfies these design criteria. Tetrahedral cells allow us to use linear basis functions to approximate the material interfaces and the dependent field variables in a cell. A representation of the original data can be computed in a pre-processing step, and we can utilize methods developed for the ROAM system [3] to efficiently select a representation that satisfies an error bound or a desired cell count. This makes the framework ideal for interactive display. Strict L^∞ error bounds are incorporated into the refinement process.

The framework supports various input meshes by resampling them at the vertices of the tetrahedral mesh. The resampling error of the tetrahedral mesh is a user specified variable. This error defines the error between our field approximation using the tetrahedral mesh and the field defined by the input dataset and its interpolation method. The resampled mesh can be refined to approximate the underlying field to within a specified tolerance or until the mesh contains a specific number of tetrahedra. The resampled field is a linear approximation of the input field. The boundaries of the input mesh are represented in the same manner as the surfaces of discontinuity. The volume of space outside of the mesh boundary is considered as a separate material with constant field values. This region of *empty* space is easy to evaluate and approximate; no ghost values and no field approximations need to be computed. This allows a non-rectilinear input mesh to be embeded into the larger rectilinear mesh generated by the refinement of the tetrahedral grid. Discontinuities are supported at the cell level allowing local refinement of the representations of surfaces of discontinuity in geometrically complex areas. The convergence of the approximation depends upon the complexity of the input field and the complexity of the input mesh. For meshes with complex geometry and interfaces but simple fields, the majority of the work is done approximating the input geometry and material interfaces. For meshes with simple geometry and interfaces but complex fields, the majority of work is done approximating the field values.

The framework has several advantages over other multiresolution spatial data structures such as an octree. The refinement method ensures that the tetrahedral mesh will always be free of cracks and *T-intersections.* This makes it easy to guarantee that representations of fields and surfaces of discontinuity are continuous across cell boundaries.

Our resampling and error bounding algorithms require that an original data set allow the extraction of the values of the field variables at any point and, for a given field, the maximum difference between the representation over one of our cells and the representation in the original dataset over the same volume.

5 Material Interfaces

A material interface defines the boundary between two distinct materials. Figure 3 shows an example of two triangles crossed by an single interface (smooth curve). This interface specifies where the different materials exist within a cell. Field representations across a material interface are often discontinuous. Thus, an interface can introduce a large amount of error to cells that cross it. Instead of refining an approximation substantially in the neighborhood of an interface, the discontinuity in the field is better represented by explicitly representing the surface of discontinuity in each cell. Once the discontinuity is represented, two separate functions are used to describe the dependent field variables on either side of the discontinuity. By representing the surface of discontinuity exactly, our simplification algorithm does not need to refine regions in the spatial domain with a large number of tetrahedra.

5.1 Extraction and Approximation

In the class of input datasets with which we are working, material interfaces are represented as triangle meshes. In the case that these triangle meshes are not known, they are extracted from volume fraction data by a material interface reconstruction technique, see [1]. (The volume fractions resulting from numerical simulations indicate what percentages of which materials are present in each cell.) Such an interface reconstruction technique produces a set of crack-free triangle meshes and normal vector information that can be used to determine on which side and in which material a point lies.

Within one of our tetrahedra, an approximate material interface is represented as the zero set of a signed distance function. Each vertex of a tetrahedron is assigned a signed distance value for each of the material interfaces in the tetrahedron. The signed distance from a vertex $\mathbf{V}$ to an interface mesh $\mathbf{I}$ is determined by first finding a triangle mesh vertex $\mathbf{V}_i$ in the triangle mesh describing **I** that has minimal distance to **V**. The sign of the distance is determined by considering the normal vector $\mathbf{N}_i$ at $\mathbf{V}_i$. If $\mathbf{N}_i$ points towards **V**, then **V** is considered to be on the *positive side* of the interface; otherwise it is considered to be on the *negative side* of the interface. The complexity of this computation is proportional to the complexity of the material interfaces within a particular tetrahedra. In general a coarse cell in the mesh will contain a large number of interface polygons, and a fine cell in the mesh will contain a small number of interface polygons. The signed distance values are computed as the mesh is subdivided. When a new vertex is introduced via the mesh refinement, the computation of the signed distance for that vertex only needs to look at the interfaces that exist in the tetrahedra around the split edge. If those tetrahedra do not contain any interfaces, no signed distance value needs to be computed.

In Figure 3, the true material interface is given by the smooth curve and its approximation is given by the piecewise linear curve. The minimum distances from the vertices of the triangles to the interface are shown as dotted lines. The distances for vertices on one side of the interface (say, above the interface) are assigned positive values and those on the other side are assigned negative values. These signed distance values at the vertices determine linear functions in each of the triangles, and the approximated interface will be the zero set of these linear functions. Because the mesh is conformant, the linear functions in the two triangles will agree on their common side, and the zero set is continuous across the boundary. The situation in three dimensions is analogous.

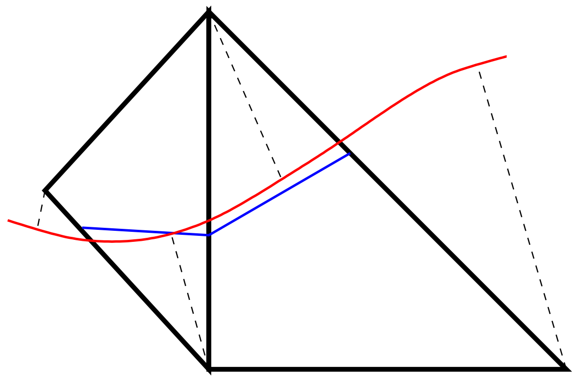

Fig. 3. True and approximated interfaces.

Figure 4 shows a two-dimensional example of a triangle with several material interfaces and their approximations. In this figure, the thin, jagged lines are the original boundaries and the thick, straight lines are the approximations derived from using the signed distance values. For the interface between materials A and B, the thin, dashed lines from vertices A, B, and C indicate the points on the interface used to compute the signed distance values. The dashed line L demonstrates that the projection of a point onto an approximation does not always lie inside the cell. The signed distance function is assumed to vary linearly in the cell, i.e., a tetrahedron. The distance function is a linear function $f(x, y, z) = Ax + By + Cz + D$. The coefficients for the linear function defining a boundary representation are found by solving a 4x4 system of equations, considering the requirement that the signed distance function over the tetrahedron must interpolate the signed distance values at the four vertices.

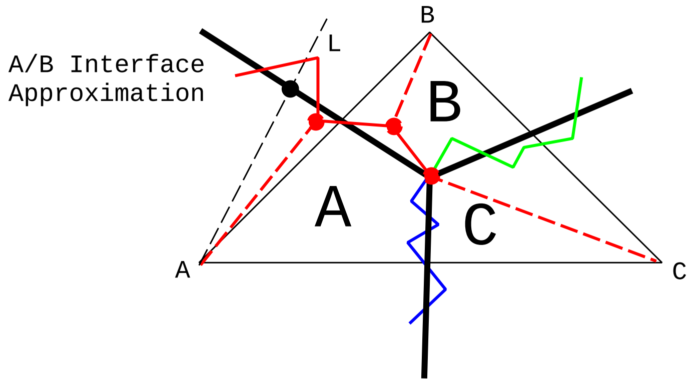

Fig. 4. Triangle with three materials (A, B, and C) and three interfaces.

The three-dimensional example in Figure 5 shows a tetrahedron, a material interface approximation, and the signed distance values d_i for each vertex V_i. The approximation is shown as a plane cutting through the tetrahedron. The normal vector N indicates the positive side of the material boundary approximation. Thus, the distance to V_0 is positive and the distances for V_1, V_2, and V_3 are negative.

We note that a vertex has at most one signed distance value for each interface. This ensures that the interface representation is continuous across cell boundaries. If a cell does not contain a particular interface, the signed distance value for that interface is meaningless for that cell. Given a point **P** in an interface polygon and its associated approximation B_r, the error associated with **P** is the absolute value of the distance between **P** and B_r. The material interface approximation error associated with a cell is the maximum of these distances, considering all the interfaces within the cell.

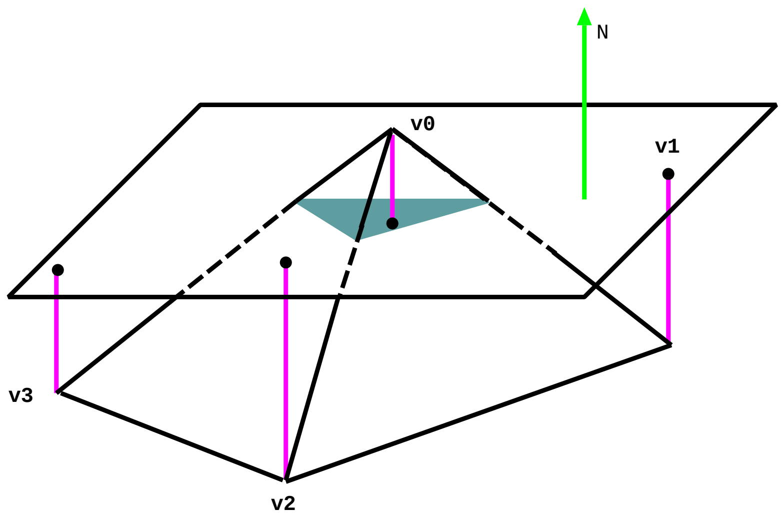

Fig. 5. Tetrahedron showing signed distance values and the corresponding boundary approximation.

6 Discontinuous Field Representations

Cells that contain material interfaces typically also have discontinuities in the fields defined over them. For example, the density field over a cell that contains both steel and nickel is discontinuous exactly where the two materials meet. In these situations, it is better to represent the density field over the cell as two separate fields, one field for the region containing only the first material and one for the second material. One way to accomplish field separation is to divide the cell into two distinct cells at the material interface. In Figure 6, the triangle would be divided into a quadrilateral for material A and a triangle for material B. The disadvantage of this method is that it introduces new cell types into the mesh which makes it harder to have continuous field representations across cells. Furthermore if new cell types are introduced, we loose the multiresolution structure and adaptive refinement capabilities of the tetrahedral mesh. Our algorithm represents the discontinuity by constructing a field representation for each material in the cell. Each of the vertices in a cell must have distinct field values for each material in the cell.

For a vertex **V** that does not reside in material **M**, we compute a ghost value for the field associated with material **M** at vertex **V**. This ghost value is an extrapolation of the field value for **M** at **V**. The process is illustrated in Figure 6. The known field values are indicated by the solid circles. A_0 and A_1 represent the known field values for material A, and B_0 represents the known field value for material B. Vertices A_0 and A_1 are in material A, and thus ghost values for material B must be calculated at their positions. Vertex

B_0 lies in material B, and thus a ghost value for material A must be calculated at its position. As described in Section 3, the ghost value computation is performed when the vertex is created during the tetrahedral refinement process.

6.1 Computation of Ghost Values

The ghost values for a vertex $\mathbf{V}$ are computed as follows:

1. For each material interface present in the cells that share the vertex, find a vertex V_{min} in a triangle mesh representing an interface with minimal distance to $\mathbf{V}$. (In Figure 6, these vertices are indicated by the dashed lines from A_0, A_1, and B_2 to the indicated points on the interface.)
2. Evaluate the data set on the far side of the interface at V_{min} and use this as the ghost value at $\mathbf{V}$ for the material on the opposite side of the interface.

Only one ghost value exists for a given vertex, field and material. This ensures that the field representations are C^0 continuous across cell boundaries. For example, consider vertex V_0 of the triangle in Figure 4. The vertex V_0 lies in material A, and therefore we must compute ghost values for materials B and C at vertex A_0. The algorithm will examine the three material boundaries and determine the points from materials B and C that are closest to A_0. The fields for materials B and C are evaluated at these points, and these values are used as the ghost values for A_0. These points are exactly the points that were used to determine the distance map that defines the approximation to the interface. This computation assumes that the field remains constant on the other side of the interface. Alternatively, a linear or higher order function can be used to approximate the existing data points within the cell and to extrapolate the ghost value.

7 Error Metrics

The error metrics employed in our framework are similar to the nested error bounds used in the ROAM system. Each cell has two associated error values: a field error and a material interface error. In order to calculate the field errors for a leaf cell in our tetrahedral mesh hierarchy, we assume that the original data set can be divided into *native data elements*. Each of these is presumed to have a well defined spatial extent and a well defined representation for each field of interest over its spatial domain. The simplest example of a native data element is just a grid point that holds field values. Other possibilities are blocks of grid points treated as a unit, cells with a non-zero volume and a field representation defined over the entire cell, or blocks of such cells. For a given field, we assume that it is possible to bound the difference between the representation over one of our leaf cells and the representation over each

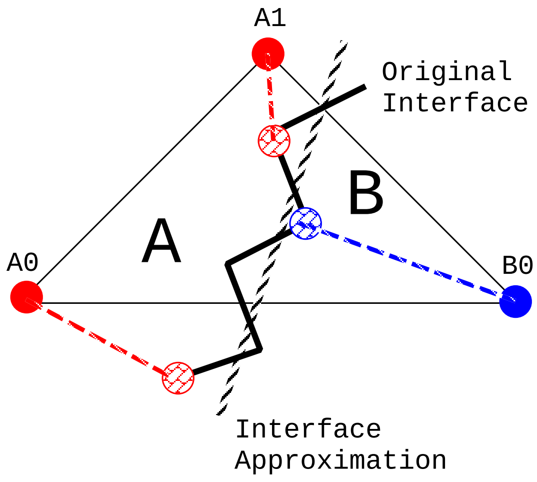

Fig. 6. Ghost value computation for a triangle containing two materials.

of the native data elements with which the given cell intersects. The error for the given field in the given cell is the maximum of the errors associated with each of the intersecting native data elements. Currently, we are dealing with native data elements that are grid points of zero volume.

The field error e_T for a non-leaf cell is computed from the errors associated with its two children according to:

$$e_T = \max\{e_{T_0}, e_{T_1}\} + |z(v_c) - z_T(v_c)|, \tag{1}$$

where e_{T_0} and e_{T_1} are the errors of the children; v_c is the vertex that splits the parent into its children; $z(v_c)$ is the field value assigned to v_c; and $z_T(v_c)$ is the field value that the parent assigns to the spatial location of v_c. The approximated value at v_c, $z_T(v_c)$, is calculated as:

$$z_T(v_c) = \frac{1}{2}(z(v_0) + z(v_1)), \tag{2}$$

where v_0 and v_1 are the vertices of the parent's split edge. This error is still a genuine bound on the difference between our representation and the representation of the original data set. However, it is looser than the bound computed directly from the data. The error computed from the children has the advantage that the error associated with a cell bounds not only the deviation from the original representation but also the deviation from the representation at any intermediate resolution level. Consequently, this error is *nested* or monotonic in the sense that the error of a child is guaranteed not to be greater than the error of the parent. Once the errors of the leaf cells are computed, the nested bound for all cells higher in the tree can be computed in time proportional to K, where K is the number of leaf cells in the tree. This can be accomplished by traversing the tree in a bottom-up fashion.

The material interface error associated with a leaf node is the maximum value of the errors associated with each of the material interfaces in the node. For each material interface, the error is the maximum value of the errors associated with the vertices constituting the triangle mesh defining the interface and being inside the cell. The error of a vertex is the absolute value of the distance between the vertex and the interface approximation. The material interface error **E** for a cell guarantees that no point in the original interface polygon mesh is further from its associated approximation than a distance of **E**. This error metric is an upper bound on the deviation of the original interfaces from our approximated interfaces. A cell that does not contain a material interface is considered to have an interface error of zero.

8 Results

We have tested our algorithm on a data set resulting from a simulation of a hypersonic impact between a projectile and a metal block. The simulation was based on a logically rectilinear mesh of dimensions 32x32x52. For each cell, the average density and pressure values are available, as well as the per-material densities and volume fractions. The physical dimensions in x, y, and z directions are [0,12] [0,12] and [-16,4.8].

There are three materials in the simulation: the projectile, the block, and *empty* space. The interface between the projectile and the block consists of 38 polygons, the interface between the projectile and empty space consists of 118 polygons and the interface between empty space and the block consists of 17574 polygons. Figure 7 shows the original interface meshes determined from the volume fraction information. The largest mesh is the interface between the metal block and empty space; the next largest mesh in the top, left, front corner is the interface between the projectile and empty space; the smallest mesh is the interface between the projectile and the block.

Figure 8 shows a cross-section view of the mesh created by a cutting plane through the tetrahedral mesh. The darker lines are the original interface polygons, and the lighter lines are the approximation to the interface. The interface approximation error is 0.15. (An error of 0.15 means that all of the vertices in the original material interface meshes are no more that a physical distance of 0.15 from their associated interface approximation. This is equivalent to an error of (0.5 - 1.5)% when considered against the physical dimensions.) A total of 3174 tetrahedra were required to approximate the interface within an error of 0.15. The overall mesh contained a total of 5390 tetrahedra. A total of 11990 tetrahedra were required to approximate the interface to an error of 0.15 and the density field within an error of 3. The maximum field approximation error in the cells containing material interfaces is 2.84, and the average field error for these cells is 0.007. These error

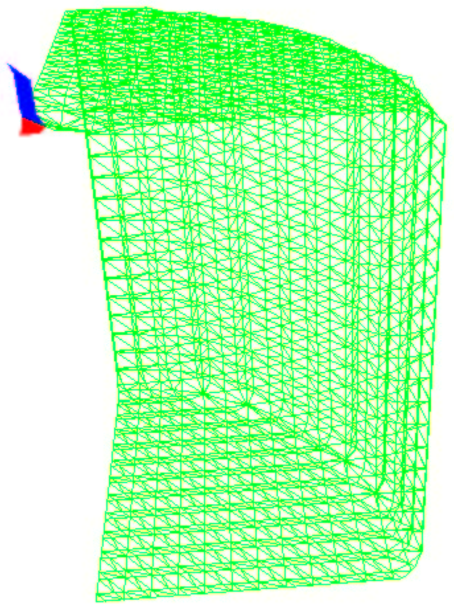

Fig. 7. Original triangular meshes representing material interfaces.

measurements indicate that separate field representations for the materials on either side of a discontinuity can accurately reconstruct the field.

Figures 8 and 1 compare the density fields generated using linear interpolation of the density values and explicit field representations on either side of the discontinuity. These images are generated by intersecting the cutting plane with the tetrahedra and evaluating the density field at the intersection points. A polygon is drawn through the intersection points to visualize the density field. In the cells where material interfaces are present, the cutting plane is also intersected with the interface representation to generate data points on the cutting plane that are also on the interface. These data points are used to draw a polygon for each material that the cutting plane intersects.

Figure 1 shows that using explicit field representations in the presence of discontinuities can improve the quality of the field approximation. This can be seen in the flat horizontal and vertical sections of the block where the cells approximate a region that contains the block and empty space. In these cells, the use of explicit representations of the discontinuities leads to an exact representation of the density field. The corresponding field representations using linear interpolation, shown in Figure 8, capture the discontinuities poorly. Furthermore, Figure 1 captures more of the dynamics in the area where the projectile is entering the block (upper-left corner). The linear interpolation of the density values in the region where the projectile is impacting the block smooths out the density field, and does not capture the distinct interface between the block and the projectile.

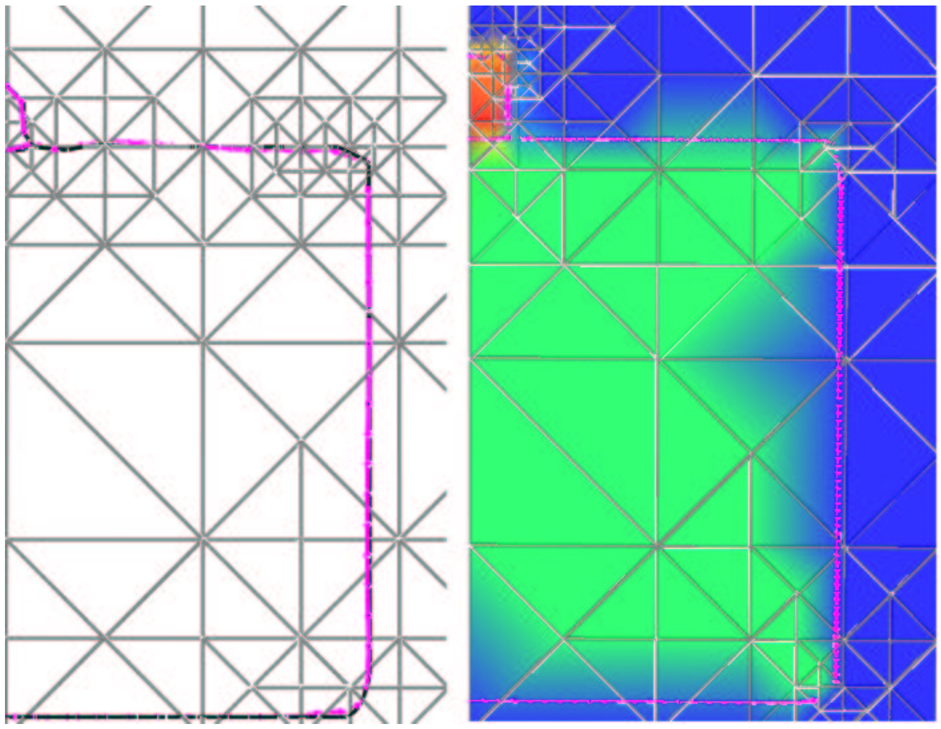

Fig. 8. Cross section of the tetrahedral mesh. The left picture shows the original interfaces and their approximations. The picture on the right shows the density field using linear interpolation.

9 Conclusions and Future Work

We have presented a method for constructing multiresolution representations of scientific datasets that explicitly represents material interfaces. Our algorithm constructs an approximation that can be used in place of the original data for visualization purposes. Explicitly representing material and implicit field discontinuities allows us to use multiple field representations to better approximate the field within each cell. The use of the tetrahedral subdivision allows us to generalize our algorithm to a wide variety of data sets and to support interactive level-of-detail exploration and view-dependent simplification.

Future work will extend our error calculations to support complex native data element types such as tetrahedra and curvilinear hexahedra. Our current ghost value computation assumes that the field is constant on the other side of the interface. Higher-order extrapolation methods will be investigated for ghost value computation to determine if an improved field approximation can be obtained. Similarly, material interfaces are defined by approximations based on linear functions. The tradeoff between cell count and higher-order approximation methods should be investigated to determine if a better approximation can be obtained without a great increase in computational complexity. We also plan to apply our algorithm to more complex unstructured data sets.

10 Acknowledgments

This work was supported by, Los Alamos National Laboratory and by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 awarded to the University of California, Davis. We thank the members of the Visualization Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis and the members of the Advanced Computing Laboratory at Los Alamos National Laboratory.

References

1. Kathleen S. Bonnell, Daniel R. Schikore, Kenneth I. Joy, Mark Duchaineau, and Bernd Hamann. Constructing material interfaces from data sets with volume-fraction information. In *Proceedings Visualization 2000*, pages 367–372. IEEE Computer Society Technical Committee on Computer Graphics, 2000.
2. Paolo Cignoni, Enrico Puppo, and Roberto Scopigno. Multiresolution Representation and Visualization of Volume Data. *IEEE Transactions on Visualization and Computer Graphics*, 3:352–369, October 1997.
3. M. A. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In R. Yagel and H. Hagen, editors, *IEEE Visualization '97*, pages 81–88. IEEE Computer Society Press, 1997.
4. Mark A. Duchaineau, Martin Bertram, Serban Porumbescu, Bernd Hamann, and Kenneth I. Joy. Interactive display of surfaces using subdivision surfaces and wavelets. In T.L. Kunii, editor, *Proceedings of 16th Spring Conference on Computer Graphics, Comenius University, Bratislava, Slovak Republic*, 2001.
5. Mark A. Duchaineau, Serban Porumbescu, Martin Bertram, Bernd Hamann, and Kenneth I. Joy. Dataflow and re-mapping for wavelet compression and

view-dependent optimization of billion-triangle isosurfaces. In G. Farin, H. Hagen, and B. Hamann, editors, *Hierarchical Approximation and Geometrical Methods for Scientific Visualization*. Springer-Verlag, Berlin, Germany, 1999.

6. M. Garland and P. S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization '98 (VIS '98)*, pages 263–270, Washington - Brussels - Tokyo, October 1998. IEEE.
7. Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209–216. ACM SIGGRAPH, Addison Wesley, August 1997.
8. Thomas Gerstner and Renato Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings Visualization 2000*, pages 259–266. IEEE Computer Society Technical Committee on Computer Graphics, 2000.
9. B. Hamann. A data reduction scheme for triangulated surface. In *Computer Aided Geometric Design*, volume 11, pages 197–214, 1994.
10. Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report.
11. Hugues Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.
12. Hugues H. Hoppe. New quadric metric for simplifying meshes with appearance attributes. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 59–66, San Francisco, 1999. IEEE.
13. Reinhard Klein, Gunther Liebich, and Wolfgang Straßer. Mesh reduction with error control. In Roni Yagel and Gregory M. Nielson, editors, *Proceedings of the Conference on Visualization*, pages 311–318, Los Alamitos, October 27–November 1 1996. IEEE.
14. Leif P. Kobbelt, Jens Vorsatz, Ulf Labsik, and Hans-Peter Seidel. A shrink wrapping approach to remeshing polygonal surface. In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 119–130. The Eurographics Association and Blackwell Publishers, 1999.
15. Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 95–104. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
16. Peter Lindstrom. Out-of-Core simplification of large polygonal models. In Sheila Hoffmeyer, editor, *Proceedings of the Computer Graphics Conference 2000 (SIGGRAPH-00)*, pages 259–262, New York, July 23–28 2000. ACM-Press.
17. Anwei Liu and Barry Joe. Quality local refinement of tetrahedral meshes based on bisection. *SIAM Journal on Scientific Computing*, 16(6):1269–1291, November 1995.
18. Oliver G. Staadt and Markus H. Gross. Progressive tetrahedralizations. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *Proceedings of Visualization 98*, pages 397–402. IEEE Computer Society Press, October 1998.

19. Isaac J. Trotts, Bernd Hamann, Kenneth I. Joy, and David F. Wiley. Simplification of tetrahedral meshes. In Holly Rushmeier David Ebert and Hans Hagen, editors, *Proceedings IEEE Visualization '98*, pages 287–296. IEEE Computer Society Press, October 18–23 1998.
20. Yong Zhou, Baoquan Chen, and Arie E. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization '97*, pages 135–142. IEEE Computer Society Press, November 1997.