Approximating Material Interfaces during Data Simplification

Benjamin F. Gregorski*
Bernd Hamann * Kenneth I. Joy *

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

Abstract

We present a new method for simplifying large 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 a cell. Representing material interfaces explicitly in a data simplification process, allows us to use separate field representations for each material over a single cell.

1 Introduction

Computational physics simulations are generating larger and larger amounts of data. 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. The basis for our simplification algorithm is the subdivision of a tetrahedral mesh as presented by Zhou et al. [?].

2 Material Interfaces

2.1 Motivation

A material interface defines the boundary between two distinct materials. Field representations, i.e. density, pressure, etc., 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.

2.2 Extraction and Approximation

Material interfaces are represented as triangle meshes. Within one of our cells, i.e. a tetrahedron, an approximate material interface is represented as the zero set of a signed distance function. The signed distance from a vertex $\mathbf{V}$ to an interface $\mathbf{I}$ is determined by finding the vertex $\mathbf{V}_i$ in the triangle mesh describing $\mathbf{I}$ closest to $\mathbf{V}$. 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 $\mathbf{V}$, then $\mathbf{V}$ is considered to be on the "positive side" of the interface.

Figure 1 shows an example of a triangle with several material interfaces and their approximations. The jagged lines

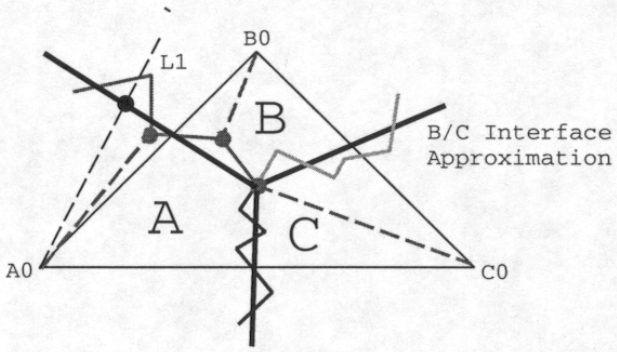

Figure 1: Triangle with three materials (A, B, and C) and three interfaces.

are the original boundaries and the black lines are the approximations derived from using the signed distance values. The dashed lines and dots show the points on the interface between materials A and B used to compute the signed distance values for the vertices A_0, B_0, and C_0. The signed distance function is assumed to vary linearly within a cell. A vertex has at most one signed distance value for each interface. This ensures that the interface representation is continuous across cell boundaries.

3 Discontinuous Field Representations

3.1 Motivation

Cells that contain material interfaces typically 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 each material. 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. These extrapolated values are called ghost values.

The ghost value computation is illustrated in Figure 2. The known field values are indicated by the solid circles. Vertices A_0 and A_1 are in material A, and thus need ghost values for material B. Vertex B_2 lies in material B, and needs a ghost value for material A.

3.2 Computation of Ghost Values

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

*{gregorsk,hamann,joy}@cs.ucdavis.edu

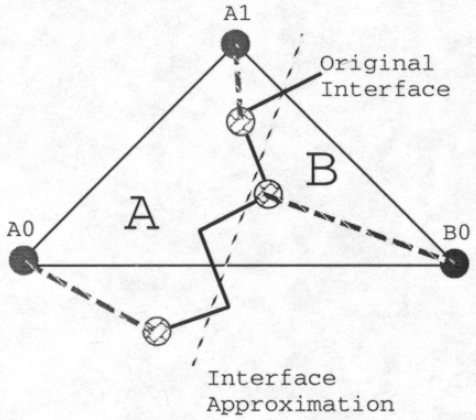

Figure 2: Triangle containing two materials.

1. For each material interface present in the cells that share the vertex, find the vertex on the interface V_{min} closest to **V**. This is shown in Figure 2.

2. Evaluate the data set on the far side of the interface at V_{min} and use this as the ghost value at **V** for the material on the opposite side of the interface.

Only one ghost value exists per material and vertex. This ensures that the field representations are continuous across cell boundaries.

4 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. Figure 4 shows a cross section view of the mesh created by a cutting plane through the tetrahedral mesh (i.e our approximation mesh). Figures 3 and 4 compare the density fields generated using linear interpolation of the density values and explicit field representations on either side of the discontinuity. Figure 4 shows that using explicit field representations 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.

5 Conclusions

We have presented a simplification method for scientific data sets that explicitly represents material interfaces in mesh cells. Our algorithm constructs an approximation that can be used in place of the original data set for visualization purposes. Explicitly representing the material and implicit field discontinuities allows us to use multiple field representations to better approximate the field within each cell.

References

[1] Y. Zhou, B. Chen, and A. E. Kaufman, "Multiresolution tetrahedral framework for visualizing regular volume data," in *IEEE Visualization '97* (R. Yagel and H. Hagen, eds.), pp. 135–142, IEEE Computer Society Press, Nov. 1997.

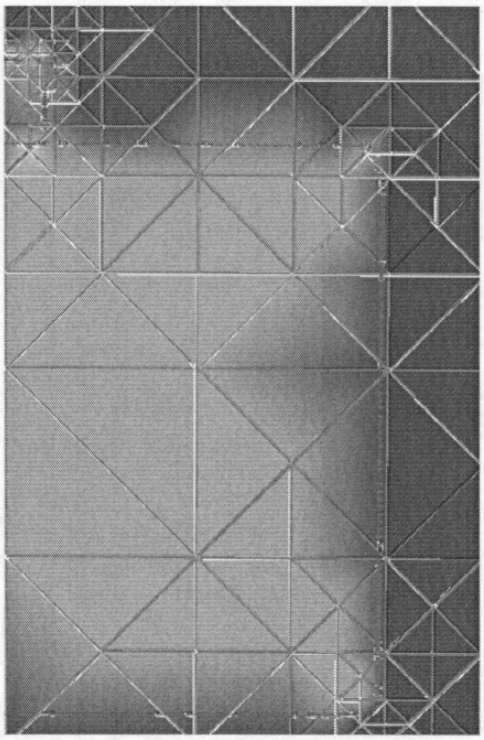

Figure 3: Density field using linearly interpolated density values.

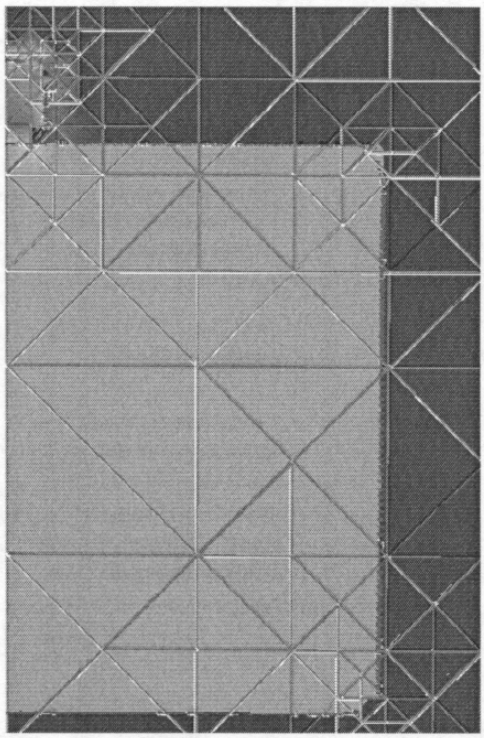

Figure 4: Density field using explicit interface representations and separate field representations.