

Piecewise Tri-linear Contouring for Multi-Material Volumes

Powei Feng¹, Tao Ju², and Joe Warren¹

¹ Rice University

{pfeng,jwarren}@rice.edu

² Washington University in St. Louis

taoju@cse.wustl.edu

Abstract. The ability to model objects composed of multiple materials has become increasingly more demanded in scientific applications. The visualization of a discrete multi-material volume often suffers from voxelization of the boundary between materials. We propose a contouring method that can be efficiently implemented on the GPU to reduce the artifacts and jaggedness along the material boundaries. Our method extends naturally from the standard tri-linear contouring in a signed volume, and further provides sub-voxel accuracy for representing three or more materials.

1 Introduction

Many scientific modeling applications require the ability to model objects composed of multiple materials. Probably the most common examples are found in bio-medicine, where researchers are often interested in the decomposition of a biological structure, obtained by imaging techniques like MRI or EM, into individual function units. Figure 1 shows two such examples, a human skull segmented into anatomical subdivisions, and a molecular complex decomposed into protein subunits. Modeling of these smaller units helps biologists and medical researchers to understand the function of the entire entity.

One of the simplest ways to represent multiple materials in a grid volume is to attach an integer material label to each grid point. While this approach is fairly simple to implement, its drawbacks are obvious. The discrete labeling leads to blocky, voxelized material boundaries that are hard to shade in a natural manner [7] (see Figures 2(a) and 2(c)).

When only two materials (e.g., inside and outside) are present in the volume, a standard solution is implicit modeling [1]. In this approach, each grid point is associated with a positive or negative floating-point scalar, where the sign indicates whether the grid point lies inside or outside the object. These scalars can be considered as samples of a continuous function $f(x, y, z)$, and the boundary surface is defined as the set of all points where $f(x, y, z) = 0$. There are numerous contouring algorithms that can produce a polygonal approximation of this

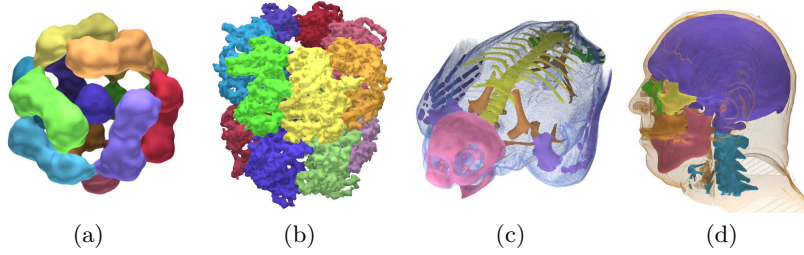


Fig. 1. Multi-material volumes representing structure of Hsp 26 (EMDB 1226) (a), subunits of the molecular chaperone GroEL (b), bones of a salamander (c), and the anatomical regions of a human head (d) visualized using tri-linear contours.

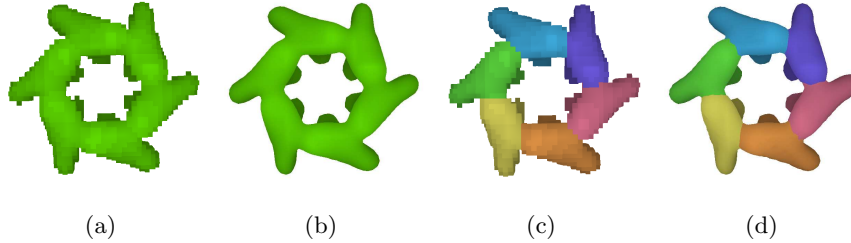


Fig. 2. Comparison of the typical voxelized material boundaries (a,c) with the same materials visualized using our tri-linear contours (b,d). This example is the replicative helicase G40P molecular structure, before (a,b) and after (c,d).

surface, such as Marching Cubes [11], Dual Contouring [9] and others [5, 10]. Alternatively, the continuous surface can be directly rendered on GPU [18, 3, 12]. The key idea behind these approaches are that the signed grid can be stored as a 3D texture and that a single texture fetch can be used to evaluate $f(x, y, z)$ via tri-linear interpolation at an arbitrary point. In practice, this tri-linear boundary surface provides better normals (for shading) and better silhouettes than either polygonal contours or voxelized boundaries (see Figure 2(b)).

While the use of two signs to model two materials is simple and elegant, the idea of using three or more labels to represent a partition of space into multiple materials has received only limited attention. Most existing works in this direction focus on producing polygonal inter-material boundaries. For example, Dual Contouring [9] creates polygons from point and normal data stored on edges in the grid, and the works of [6, 15] use a generalized Marching Cubes look-up table to polygonalize a multi-labeled cell.

A work on smooth boundaries among multiple materials that is similar to our own is by Stalling et al. [16]. In their approach, a tri-linear function $f^k(x, y, z)$ is defined for each material k , and the boundary surfaces are located where the values of two or more functions are identical and higher than the remaining functions. To define $f^k(x, y, z)$, each grid point is associated with an array of

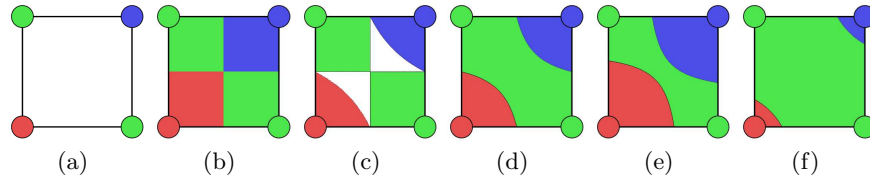


Fig. 3. Two-dimensional comparison of methods. (a) is a multi-material voxel with no intensity information. (b) is the naive approach of classifying by nearest neighbor. (c) Tiede et al. proposes a linear filter for classification [17], but it leaves points unclassified. (d) We propose a tri-linear representation that classifies all points within a voxel. (e) and (f) are examples of our approach that demonstrate the flexibility in representing contours.

scalars representing the “probabilities” that this grid point is classified as each material present in the volume. While this approach gives smooth material interfaces, the need to store multiple scalars per point not only increases the memory consumption, in comparison to the signed scalar representation of two materials, but also hampers fast GPU implementations as more texture fetches are needed. We build on their approach by providing a more compact representation that allows for fast GPU implementation.

Tiede et al. introduced a multi-material classification scheme for volume raycasting [17]. Hadwiger et al. integrated this classification scheme into their hardware implementation of high-quality volume rendering [8]. The classification scheme described by Tiede et al. focuses on segments produced by thresholding. In the case where the threshold ranges of multiple materials overlap, Tiede et al.’s approach is to linearly interpolate the binary mask associated with the material. For each material A , space where the interpolated value (with respect to A ’s tri-linearly interpolated binary mask) is greater than 0.5 is classified as A . Although linear filtering resolves the overlap of threshold ranges, it also produces unclassified regions within a single voxel (see Figure 3(c)). In the case where the input is a segmented volume without intensity information, Tiede et al.’s approach would produce a classification that has ripple-like effect (see Figure 4(b)). Our method guarantees classification for all points within a voxel (see Figure 3(d)) and provides greater flexibility in sub-voxel classification (see Figure 3(e)), hence capable of representing smooth inter-material boundary (see Figure 4(c)). Also note that both Tiede et al. and Hadwiger et al. tackled the problem from a visualization perspective, where they improved multi-material rendering for one particular visualization technique. Our approach is to present a geometric representation for multi-material volume that can be used for various visualization methods.

In this paper, we propose an alternative generalization of the idea of two-sign tri-linear contouring to that of multi-labeled tri-linear contouring with the goal of creating smooth multi-material boundary surfaces that can be efficiently rendered. The key difference between our generalization and that in [16] is that we only require a single scalar and a single integer label to be stored at each grid

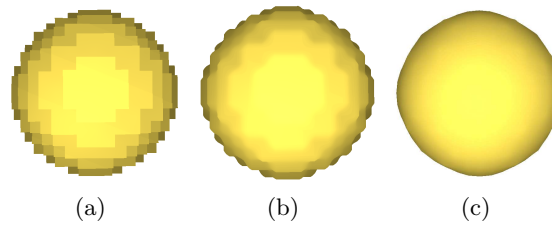


Fig. 4. Three-dimensional comparison of methods. (a) is the input of a sphere-like segment. (b) is rendered using Tiede et al.’s classification scheme [17]. Note that it produces a bumpy surface. (c) is our representation for the segment contour. Details for constructing (c) from the segment (a) is described in Section 4.

point. We show that the multi-material contours defined this way enjoys a number of properties, such as being piece-wise tri-linear and reproducing two-sign tri-linear contours where only two materials are present. The compact volume representation allows fast GPU-based rendering of the smooth contours. We demonstrate the use of tri-linear contouring in several examples of multi-labeled volumes.

Contributions In the context of implicit modeling and multi-material modeling, our work makes several novel contributions:

- We introduce a generalization of the standard two-sign tri-linear contouring to multi-labeled volumes, and demonstrate the properties of the resulting contours.
- We present an efficient GPU implementation for rendering the tri-linear multi-material contours.
- We generalize the common set operations, union and intersection, from two-signed volume to a multi-material volume.
- We demonstrate the use of our multi-material representation and routines in several examples.

2 Multi-material contouring

Conventionally, the tri-linear contour in a two-material grid is defined by signed scalars associated with the grid points. Our approach for multi-material contouring is to replace the signed scalar at each grid point with a *scalar* and a *material label*. In the two material case, our method would reproduce the standard tri-linear contours defined by signed scalars. In the case of three or more materials, the contouring method would generate piecewise tri-linear contours that form a continuous surface. Our method adds small overhead over the traditional signed volumes, and allows efficient hardware-accelerated rendering.

In the following, we first introduce the definition of contours in our volume representation. We next present a number of properties of such contours. We end this section by a discussion of means to evaluate the contours.

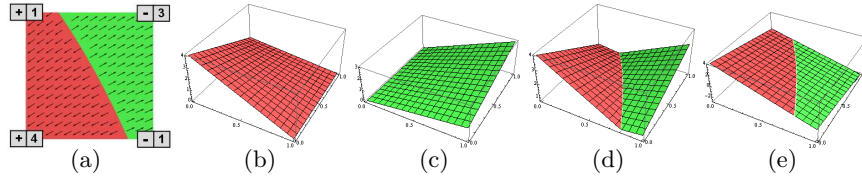


Fig. 5. Two material example: material classification in a 2D cell with $+/-$ labels (the arrows denote the gradient) (a), the bi-linear function for each material label (b,c) and their maximum (d). (e) shows the bi-linear function defined by treating the two-labeled scalars at cell corners as signed scalars, and the function's zero contour.

2.1 Defining contours

To define the contour surfaces that partition the space into regions with different materials, we first consider the dual problem of *classifying* the material of an arbitrary point in space. Given a grid cell whose corners have associated non-negative scalars s_i and material labels m_i , the following method can be used to determine the material label of a point x inside the cell. Here, index i ranges from 0 to 7, representing the eight corners of a cell.

- For each distinct material label k present in the cell, construct a set of scalars t^k associated with the corners of the cell via the following rule:

$$\begin{aligned} t_i^k &= s_i & \text{if } k = m_i \\ t_i^k &= 0 & \text{otherwise} \end{aligned}$$

- Compute the values of the tri-linear interpolant $t^k(x)$ for each distinct label k . The trilinear coefficients are t_i^k for $i = 0, 1, \dots, 7$.
- Return the material label k for which $t^k(x)$ is maximum.

With this classification, the contour between two regions with material labels k and j is simply where tri-linear functions $t^k(x), t^j(x)$ both reach maximum. This contour is an iso-surface of the form:

$$t^k(x) = t^j(x) \geq t^m(x), \quad \forall m \neq k, j \quad (1)$$

Figure 5(a) illustrates the classification method and the resulting contours in a 2D cell. In this example, only two materials are present at the cell corners, which we label as $+$ (red) and $-$ (green). Figure 5(b) and 5(c) shows the two bi-linear functions $t^+(x)$ and $t^-(x)$, respectively. Figure 5(d) shows a plot of the maximum of these two functions, where the white curve indicates the contour.

Figure 6(a) shows another 2D example in which the four corner of the cell have three distinct materials, red, green and blue. Figure 6(b) show plots of the three bi-linear functions associated with the materials. Finally, Figure 6(c) shows a plot of the maximum of these functions and the associated partition of the cell into three distinct materials via three contours that meet at a common point.

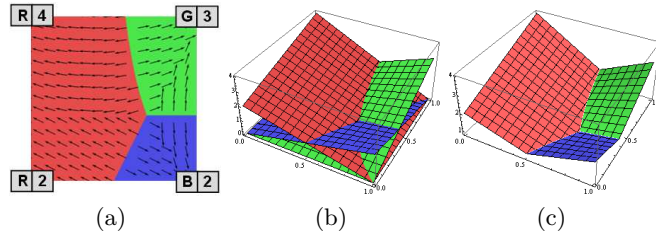


Fig. 6. Three material example: material classification in a 2D cell with $R/G/B$ labels (the arrows denote the gradient) (a), the three bi-linear functions, one for each material label (b), and their maximum (c).

2.2 Characterization of the contours

The multi-material contours produced by this method have several important properties.

Piecewise tri-linear and continuous surfaces By Definition 1, the multi-material contours defined within each cell are piecewise contours of various tri-linear functions. The contours are also continuous across neighboring cells. This fact follows from the observation that two cells sharing a common grid point, edge or face have the same scalars and material labels on that common grid element. Since the restriction of the tri-linear functions used in defining our multi-material contour on each grid point, edge or face depend only the scalar and material labels on that grid element, the multi-material contours must agree across adjacent grid elements.

Reproducing tri-linear contours on signed grids A key property of our contour definition is that it can exactly reproduce the contours defined by the standard tri-linear interpolation on a signed grid. Consider a cell whose corners are associated with signed scalars \hat{s}_i . The tri-linear interpolant $\hat{s}(x)$ defined by these signed scalars is either positive or negative, and the standard tri-linear contour is defined as the surface $\hat{s}(x) = 0$.

To reproduce this surface using our method, we construct the scalars and labels at the cell corners as follows. If \hat{s}_i is positive at corner i , we let $s_i = \hat{s}_i$ and label m_i as $+$. Otherwise, we let $s_i = -\hat{s}_i$ and label m_i as $-$. Note that this coefficient set satisfies the relation

$$\hat{s}_i = t_i^+ - t_i^- \quad (2)$$

Hence the associated tri-linear functions $t^+(x)$ and $t^-(x)$ also satisfy

$$\hat{s}(x) = t^+(x) - t^-(x). \quad (3)$$

Therefore, the zero contour of the function $\hat{s}(x)$ coincides with the locations of x where $t^+(x)$ equals $t^-(x)$, which is the contour defined by our classification method.

Figure 5(e) plots the bi-linear function $\hat{s}(x)$ for the 2D cell configuration in Figure 5(a), treating each labeled scalar as a signed scalar. Observe that the zero contour of this bi-linear function is identical to the contour defined using our method as where the two bi-linear functions are identical (Figure 5(d)).

Gradient One useful property from standard implicit modeling is that the gradient of the implicit function is normal to the contours of that function. In the multi-material case, a similar property holds. Given a contour formed by the iso-surface $t^k(x) = t^j(x)$, the gradient of the function $t^k(x) - t^j(x)$ is simply the normal to this surface. The key observation here is that the pair of material labels j and k change as the point x varies over the cell. For the three material case, $t^k(x)$ and $t^j(x)$ denote the largest and second largest tri-linear interpolant at x . This will produce the exact gradient field since we can view the local neighborhood of a point on the two-material boundary as being defined by the two dominant tri-linear functions. Note that points where more than two materials meet are degenerate with unknown gradients. Figure 5(a) show the gradient field for a two material cell while Figure 6(a) shows the gradient field for a three material cell.

2.3 Evaluating contours

We will discuss two ways to evaluate the multi-material contours, one utilizing the graphics hardware for direct surface rendering, and the other resorting to polygonization of the contours. Note that all examples in this paper are presented using the first approach.

Direct rendering A key motivation of our multi-material representation is to utilize graphics hardware for efficient rendering of the contours. We use texture-slicing volume rendering as our algorithm for visualizing multi-material volumes, as done in standard implicit modeling on a signed grid [18, 2]. Texture-slicing volume rendering approximates the ray-integral in traditional ray-casting volume rendering by rendering view perpendicular slices and compositing the slices using hardware blending.

The scalars and material labels, along with auxiliary data such as a density map, are stored as 3D textures. Coloring a single screen fragment involves a number of texture fetches to determine the density, color, and shade of the fragment in texture space, utilizing the underlying tri-linear interpolation capability of GPU. For our classification algorithm, we need to fetch an additional 8 scalar values and 8 integer labels as part of the fragment shader program. Texture fetches are typically expensive operations in shader programming. We note that, however, these 16 fetches can be reduced to 4 fetches by packing the values into the RGBA channels for a single texel. As we shall see in the Results section, our algorithm achieves interactive rendering rates, even for volumes with complex material composition like those in Figure 1.

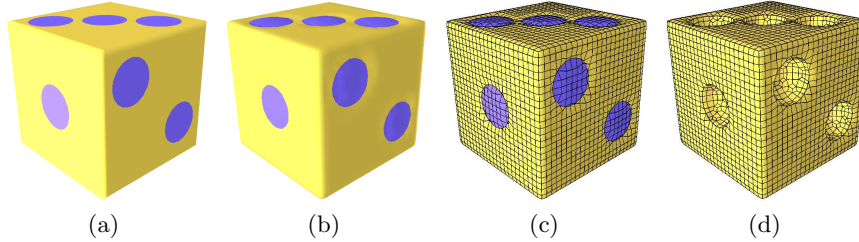


Fig. 7. The perspective view of a multi-material volume of size 33^3 rendered using GPU tri-linear contouring (a) and as polygonal contours generated by Dual Contouring (b), showing the grid structure (c). (d) depicts the mesh generated from Dual Contouring without the dots.

Both the scalars and material labels are 8-bit textures, which allows up to 256 number of materials. With only 8-bits of precision for the scalars, we need to ensure that the precision is not spent on non-essential portions of the representation. We note that the scalars are used for arbitration only on the border between different materials. This implies that the scalars only need to be accurate for cells that intersect the inter-material boundary.

Lastly, we use a single 1D transfer function that maps the voxel density to an opacity value. The color of each pixel is determined by the classification, where each material label is associated with a single color. Note that our method is a general classification scheme, and it can be extended to include multiple transfer functions (one for each material label) as described by Hadwiger et al. [8].

Polygonization Besides GPU-based rendering, an alternative way to evaluate the contours is using polygonal methods such as Dual Contouring. Under Dual Contouring, we create a vertex within each cell that exhibits a label change, and then form a polygon for each grid edge that exhibits a material label change by connecting the vertices created within the cells sharing that edge.

The vertex in a cell should be located closest to the intersection of all the pairwise tri-linear surfaces within the cell. More formally, let M be the set of materials in the cell and let x be a point inside the cell. Consider the function

$$E(x) = \sum_{j \neq k \in M} (t^k(x) - t^j(x))^2 \quad (4)$$

The minimum of this function describes a point that is close to the intersection of all the surfaces that satisfy $t^k(x) = t^j(x)$. This is a non-linear optimization problem that can be costly to compute. We approximate the solution using an QEF-based approach that was described by Schaefer et al [14]. Using this method, we first locate the intersections, p_i , of the surfaces along the cell edges. These intersections and the gradient directions (as described in Section 2.2), n_i , at these points describe a set of planes per cell. We then find a point that

minimizes

$$E'(x) = \sum_i (n_i \cdot (x - p_i))^2 \quad (5)$$

This minimization gives an approximation of Eq. 4 that is reasonable for our purpose. Note that this is an outline for computing the contour point. Please refer to the work of Schaefer et al. for more implementation details [14]. An example of the Dual Contouring polygonization is shown in Figure 7.

3 Set Operations on Multi-material Contours

One of the primary attractions of implicit modeling is the ease with which it can model Boolean operations from constructive solid geometry [13]. To enable interactive construction of multi-material models in our representation, we next develop analogs of the set operations Union and Intersection for multi-material contours. These operations can be applied in several context with regards to interactive segmentation. Due to space limitation, we direct the reader to [4] for more detail.

In the signed (two-material) case, the typical convention is to represent a solid as the set of solutions to the inequality $f(x, y, z) < 0$. Now, given two solids $f(x, y, z) < 0$ and $g(x, y, z) < 0$, the union of these two solids is simply the set $\min(f(x, y, z), g(x, y, z)) < 0$ while the intersection of two solids is the set $\max(f(x, y, z), g(x, y, z)) < 0$.

If the functions f and g are represented by signed grids, a standard technique for approximating their union or intersection is to take the min or max of their associated sign grids. Our goal is to develop equivalent rules for the two-material case that generalize to the multi-material case in a natural manner.

Our approach is as follows; consider two materials A and $\neg A$ (not A). A can be interpreted as being the inside of a solid (i.e; negative in the implicit model) and $\neg A$ can be interpreted as being the outside of a solid (i.e; positive in the implicit model.). Given a multi-material map consisting of only these two materials, we can attempt to construct rules for computing new non-negative scalars and material labels on the grid that reproduce the operations Union and Intersection.

In particular, given a grid point with two associated pairs (s_1, k_1) and (s_2, k_2) (where both the s_i are non-negative), our goal is to compute a scalar/label pair (s, k) for the union of the material S . This new pair can be compute using the case look-up given in Table 3.

Note that the rule for computing k is straightforward. For Union, the new material label is A if and only if at least one of the material labels is A . For Intersection, the new material label is A if and only if both of the material labels are A . The rule for computing the new scalar s is only slightly more involved. The key is to convert back to the signed case and then return the result of taking the minimum of the converted scalars. For example, if both material labels are A , we take the negative of both scalars s_1 and s_2 , computed their min and then

Union				Intersection			
k_1	k_2	k	s	k_1	k_2	k	s
A	A	A	$\max(s_1, s_2)$	A	A	A	$\min(s_1, s_2)$
A	$\neg A$	A	s_1	A	$\neg A$	$\neg A$	s_2
$\neg A$	A	A	s_2	$\neg A$	A	$\neg A$	s_1
$\neg A$	$\neg A$	$\neg A$	$\min(s_1, s_2)$	$\neg A$	$\neg A$	$\neg A$	$\max(s_1, s_2)$

Table 1. Rules for performing Intersection and Union operations. We consider the pairs (s_1, k_1) and (s_2, k_2) as the input. The output of Union and Intersection is denoted as pair (s, k) .

negate the result. These three operations are simply the equivalent of taking the max of the original scalars. In particular, if both s_1 and s_2 are non-negative,

$$\max(s_1, s_2) = -\min(-s_1, -s_2) \quad (6)$$

Similar argument can be used to derive the given formulas for the remaining cases.

Another interpretation of these operations on the scalars s_1 and s_2 is to view these numbers as estimate of the distance from the grid point to the boundary of the region A . In the case of Union, the rule is that if both grid points lie in A , a good estimate of the distance from the grid point to the boundary of the union is the maximum of these two distances. Similar arguments again apply in the other cases.

3.1 Operations for Three or More Materials

Given the method for union and intersection defined above, the generalization of these operations to the three or more material case is relatively easy. We suggest two operations analogous to Union and Intersection for the multi-material case. The first operation *Overwrite* takes a multi-material map and a two-material map (with material A and $\neg A$) and performs the multi-material analog of Union. In particular, it treats the material in the first multi-material map as being either A or $\neg A$ and applies the two material rules for Union described above. The result of an Overwrite operation is that the material A in the second map overwritten onto any existing materials in the first map. The resulting map contains the union of the materials A in both maps.

The second operation *Restrict* again takes as input a multi-material map and a two-material map (with materials A and $\neg A$). In this case, the Restrict operation modifies the second map to return the intersection of the first map (viewed as materials A and $\neg A$) and the second map. Essentially, the second map is restricted to only those regions where the material A exists in the first map.

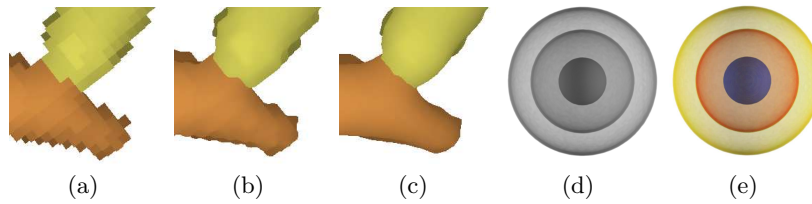


Fig. 8. Implementation details. Viewing a multi-material volume with only integer labels: direct rendering of the voxelized boundaries (a), tri-linear contouring using uniform assignment of scalars (b) and using Gaussian-filtered scalars (c). Viewing nested iso-surfaces in a density volume using transfer function (d) as a multi-material volume (e).

4 Implementation

We first consider visualizing a given multi-material volume where grid points are attached with only integer labels. Direct visualization of the voxelized material boundaries gives a blocky look (see Figure 8(a)). To create piece-wise smooth tri-linear contours, our method needs additional scalars besides the material labels. A simple approach is to assign $s_i = 1$ uniformly for every grid point. Although locally smooth, such contours “wobble” a lot and do not represent a globally smooth surface (see Figure 8(b)).

To alleviate local surface undulations, we use an improved scalar assignment based on blurring. Recall in our classification method that t_i^k is a scalar at grid point i for material label k , determined by the scalar s_i and material label m_i at that point. First, we assign $s_i = 1$ for every grid point, and hence t_i^k would be binary (0 or 1). Next, for each label k , we compute a blurred value, \bar{t}_i^k , by applying a truncated $3 \times 3 \times 3$ gaussian filter to t_g^k at neighboring grid points g as mentioned in [7]. Finally, we assign s_i to be

$$s_i^{new} = \bar{t}_i^k - \bar{t}_i^j \quad (7)$$

where \bar{t}_i^k and \bar{t}_i^j are the largest and second largest values among the blurred scalars for all material labels. Accordingly, the material label is set to be $m_i = k$.

This heuristic is guided by the same intuition given in the *gradient* discussion of Section 2.2. In the two-material case, the contour resulted from the blurred assignment reproduces the standard tri-linear contour after blurring a signed scalar grid. In the case of three or more materials, the contour is formed by the top two dominant tri-linear interpolants ($\bar{t}^k(x)$ and $\bar{t}^j(x)$), and the assignment in Eq. 7 results in an approximation of that contour in the cell. Figure 8(c) shows the result of our heuristic. Note that the resulting contours improve over those of uniform assignment. More in-depth implementation details, including segmentation generation, can be found in [4].

5 Results

Our method has been implemented and tested on an Intel Xeon 5150 machine with 2 dual-core CPUs running at 2.66GHz. We use an nVidia GTX280 graphics card with 1GB of video RAM. The shaders were written in GLSL. We use OpenMP to enable multi-core processing for easily parallelizable portions of the code.

model	size	binary (fps)	tri-linear (fps)
Hsp26	$128 \times 128 \times 128$	69	23
GroEL	$240 \times 240 \times 240$	44	18
head	$128 \times 256 \times 256$	46	13
engine	$256 \times 256 \times 256$	40	17
sea turtle	$256 \times 256 \times 397$	29	11

Table 2. The rendering speed for a selection of datasets. All results were measured in frames per second (fps).

We gather rendering times for a selection of our test cases. The models are displayed in Figure 1. The running time is largely dependent on the maximum dimension of the volume as we use that to determine the number of quads to use as proxies for rendering. The rendering screen is 512×512 pixels. In addition, shading only occurs for visible fragments, and intensive computation only occurs for inhomogeneous cells. The variability of computing load in the fragment shader accounts for the differences in rendering times for volumes such as the “head” and “engine” datasets.

It is also worthy to note that the rendering speed is also dependent on the pixel estate required to display each volume. In other words, the smaller the volume appears on the screen, regardless of input size, the faster the rendering will be, which is an expected result. Our results are taken from the slowest rendering time for each of the test sets. Our method maintains a reasonable frame-rate when rendering the tri-linear contours. Lastly, Figures 9 illustrates the rendering improvement of the tri-linear contours versus binary classification.

6 Conclusions and Future Work

We present a contouring technique for multi-material volumes, aimed at providing both a smoother inter-material boundary than in typical voxelized approaches and efficient GPU-based rendering. The technique is a generalization of the standard tri-linear contouring in a signed volume, and only requires a small overhead to offer sub-voxel representations of multiple materials. Our technique can be used to improve the visualization of an existing multi-labeled volume and in interactive painting of a density volume [4].

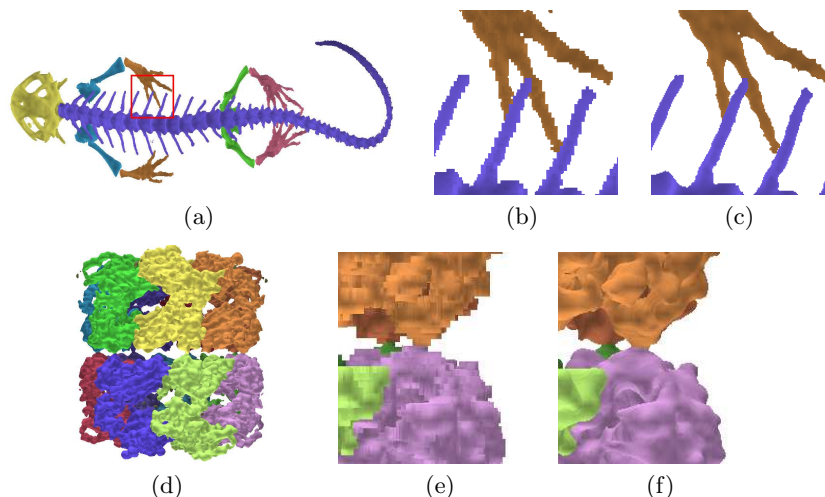


Fig. 9. A close up example of rendering using binary classification (b,e) and our piecewise tri-linear representation (c,f). The top example is the CT scan of Tarich Torosa (salamander) provided by the Digital Morphology Library. The bottom example is the GroEL structure, which was obtained from Electron Microscopy Data Bank (EMDB) under entry number 5002.

For our future work, we will experiment with using higher level interpolants such as B-splines for classification. This will give us greater flexibility and accuracy in defining the boundary between materials.

Acknowledgement

The work of Powei Feng is supported in part by the NIH grant R01GM079429. The work of Tao Ju is supported in part by NSF grants IIS-0705538, IIS-0846072, CCF-0702662 and DBI-0743691. We thank Matt Baker, Wah Chiu, Lu Liu, and Travis McPhail for helpful discussions. We thank Timothy Rowe and Jennifer Maisano of Digital Morphology Library for providing the salamander and sea turtle datasets. We thank The Volume Library, volvis.org, Electron Microscopy Data Bank, and the Protein Data Bank for providing the rest of the datasets.

References

1. Bloomenthal, J.: Implicit surfaces. *Computer Aided Geometric Design* 5, 341–355 (1997)
2. Cullip, T.J., Neumann, U.: Accelerating volume reconstruction with 3d texture hardware. Tech. rep., Chapel Hill, NC, USA (1994)
3. Engel, K., Kraus, M., Ertl, T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In: *HWWS '01: Proceedings of the ACM*

- SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware. pp. 9–16. ACM, New York, NY, USA (2001)
4. Feng, P.: Segmentation and Visualization of Volume Maps. Master’s thesis, Rice University, Texas, United States (2010)
 5. Frisken, S.F., Perry, R.N., Rockwood, A.P., Jones, T.R.: Adaptively sampled distance fields: a general representation of shape for computer graphics. In: SIGGRAPH ’00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. pp. 249–254. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000)
 6. Fujimori, T., Suzuki, H.: Surface extraction from multi-material ct data. In: Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on. pp. 6 pp.– (Dec 2005)
 7. Gibson, S.F.F.: Using distance maps for accurate surface representation in sampled volumes. Volume Visualization and Graphics, IEEE Symposium on 0, 23–30 (1998)
 8. Hadwiger, M., Berger, C., Hauser, H.: High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In: VIS ’03: Proceedings of the 14th IEEE Visualization 2003 (VIS’03). p. 40. IEEE Computer Society, Washington, DC, USA (2003)
 9. Ju, T., Losasso, F., Schaefer, S., Warren, J.: Dual contouring of hermite data. In: SIGGRAPH ’02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques. pp. 339–346. ACM, New York, NY, USA (2002)
 10. Kobbelt, L.P., Botsch, M., Schwanecke, U., Seidel, H.P.: Feature sensitive surface extraction from volume data. In: SIGGRAPH ’01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 57–66. ACM, New York, NY, USA (2001)
 11. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: SIGGRAPH ’87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques. pp. 163–169. ACM, New York, NY, USA (1987)
 12. Rezk-Salama, C., Engel, K., Bauer, M., Greiner, G., Ertl, T.: Interactive volume on standard pc graphics hardware using multi-textures and multi-stage rasterization. In: HWWS ’00: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware. pp. 109–118. ACM, New York, NY, USA (2000)
 13. Ricci, A.: A Constructive Geometry for Computer Graphics. *The Computer Journal* 16(2), 157–160 (may 1973)
 14. Schaefer, S., Warren, J.: Dual contouring: ”the secret sauce” (2003), rice University, Department of Computer Science Technical Report
 15. Shamma, M.H., Suzuki, H., Ohtake, Y.: Extraction of isosurfaces from multi-material ct volumetric data of mechanical parts. In: SPM ’08: Proceedings of the 2008 ACM symposium on Solid and physical modeling. pp. 213–220. ACM, New York, NY, USA (2008)
 16. Stalling, D., Zckler, M., Hege, H.C.: Interactive segmentation of 3d medical images with subvoxel accuracy. In: Proc. CAR98 Computer Assisted Radiology and Surgery. pp. 137–142 (1998)
 17. Tiede, U., Schiemann, T., Höhne, K.H.: High quality rendering of attributed volume data. In: VIS ’98: Proceedings of the conference on Visualization ’98. pp. 255–262. IEEE Computer Society Press, Los Alamitos, CA, USA (1998)
 18. Wilson, O., VanGelder, A., Wilhelms, J.: Direct volume rendering via 3d textures. Tech. rep., Santa Cruz, CA, USA (1994)