EUROGRAPHICS 2011 / M. Chen and O. Deussen
(Guest Editors)

Volume 30 (2011), Number 2

Contouring Discrete Indicator Functions

J. Manson, J. Smith, and S. Schaefer

Texas A&M University, USA

Abstract

We present a method for calculating the boundary of objects from Discrete Indicator Functions that store 2-material volume fractions with a high degree of accuracy. Although Marching Cubes and its derivatives are effective methods for calculating contours of functions sampled over discrete grids, these methods perform poorly when contouring non-smooth functions such as Discrete Indicator Functions. In particular, Marching Cubes will generate surfaces that exhibit aliasing and oscillations around the exact surface. We derive a simple solution to remove these problems by using a new function to calculate the positions of vertices along cell edges that is efficient, easy to implement, and does not require any optimization or iteration. Finally, we provide empirical evidence that the error introduced by our contouring method is significantly less than is introduced by Marching Cubes.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

An indicator function of a closed object is a function that is one on the interior of the object and zero on the exterior. These functions may be complex to evaluate and store depending on the complexity of the shape they represent. In practice, we use a Discrete Indicator Function (DIF) that represents the indicator function using a uniform grid of function values. In a DIF, each cell in the grid stores the fraction of the cell contained by the object, which is equivalent to the integral of the continuous indicator function inside the cell volume divided by the volume of the cell. Cells that are entirely exterior or interior to the object will have values of either zero or one, but cells on the boundary of the shape will have values between zero and one. These DIFs form an implicit representation of an object, and we can reconstruct an approximation of the underlying object by taking a level set of the function at value $\frac{1}{2}$.

DIFs arise in several areas of Computer Graphics. Perhaps, the most commonly seen example of a DIF is the pixels used to display an anti-aliased font on a computer screen. Although the screen is composed of discrete pixels, fonts appear to have smooth outlines because, rather than using only white and black, values of gray are used to show how much of a glyph is in each pixel. Beyond fonts, there may be applications where arbitrary images need to be converted from a pixel to a vector format. For example, it may be desirable to magnify a low-resolution image that contains hard boundaries. While typical pixel-based magnification results in a blurred image, finding a contour first produces a far crisper and more aesthetic result, as demonstrated by Valve Software [Gre07] in their newest games. Once a discrete contour has been calculated, it is then possible to draw the contour with hardware acceleration [LH06].

In image processing, DIFs are also often found when compositing multiple images. For example, it is common to film objects in front of a blue or green screen and remove the background by calculating an anti-aliased mask at each pixel. A pixel-based composition of many overlaid objects may produce an undesirable, aliased result where multiple boundaries overlap unless an accurate vector-based composition is calculated using the contours of the objects.

Analogs of DIFs are also encountered in medical imaging. In a CAT (Computerized Axial Tomography) scan or MRI (Magnetic Resonance Image) there are structures such as bones or tumors that stand out from the background and have clear boundaries that need to be drawn. In the 3D case of an MRI, these boundaries define a 3D surface. Surface boundaries are also found in other areas of computer graphics. For example, Eulerian simulations of water physics define occupancies over a 3D grid. Any inaccuracies in calculating the air/water interface from the DIF results in high-frequency noise or ripples that do not exist in the simulation. Another

Figure 1: *A DIF representing a complex shape sampled over a 512^3 grid. From left to right: Marching Cubes, Marching Cubes after a Gaussian blur of size 3, Marching Cubes after a Gaussian blur of size 7, and our method without blurring. Blurring increases the smoothness of the Marching Cubes surface but sacrifices details in the shape.*

important occurrence of a DIF is during reconstruction of an object's surface from a point cloud. There are several surface reconstruction methods [Kaz05, KBH06, MPS08] that internally calculate a DIF of the reconstructed object from which they then extract a surface.

In the past, contouring methods have focused on smooth functions rather than on DIFs. We believe that DIFs are an important subclass of functions for which people typically calculate suboptimal contours. The inaccuracies in calculated surfaces are often small in scale, but can have large visual effects on the texture, lighting, refraction, and reflection of an object. In this paper, we describe a simple algorithm that is tailored to find accurate contours in the specific case of DIFs.

1.1. Contribution

We describe a simple modification to standard contouring algorithms, such as Marching Cubes (MC), that enhances the quality of contours calculated from DIFs. Our analysis shows that the linear interpolation in these methods can be replaced by our new interpolant to estimate the position of the contour more accurately. Our interpolant has four cases, and the two most common cases are extremely simple to calculate. We also provide qualitative and quantitative comparisons between the interpolants that show our method clearly improves the contours of DIFs.

2. Related work

The simplest and most often used strategy for calculating implicit contours of functions is to contour cells of a regular grid independently from one-another using algorithms like MC [WMW86, LC87]. In our examples, we assume that samples are taken at the centers of cells and therefore run MC over the dual grid. Because the possible values of the DIF are between zero and one, it is natural to define the contour to be where the DIF has a value of $\frac{1}{2}$. MC determines the topology of a cell from a lookup table indexed by an eight-bit integer where each bit corresponds to one of the eight corners of the cell and is set to 0 if the corner is outside the solid or 1 if inside. In MC, the function is assumed to be smooth so that over short distances, like the span of a cell, the function can be approximated as a linear function. Vertices of the surface passing through each cell are therefore placed along the edges of the cells where the line interpolating the connected grid values has a value of $\frac{1}{2}$.

When a function is smooth, it can be approximated by linear functions at small scales, but indicator functions are inherently discontinuous, which means that a linear interpolant is not a suitable approximation of a DIF. Therefore, using a linear interpolant on a DIF will produce a contour that deviates from the ideal contour. Although these deviations may be small in absolute scale, they are easily visible on a lit surface because angular deviation of surface normals is independent of sampling resolution. Inaccuracies in the surface follow a regular pattern determined from the grid and create noticeable aliasing. This results in an easily visible, high-frequency pattern rather than uniform noise (see Figure 1).

Because MC only works well with smooth functions, a common solution to the aliasing problem is to generate a smooth function by applying a low-pass filter to the input function. Although applying a Gaussian filter [MGS96] is the simplest solution, more sophisticated methods have been used [WK93, WK94]. Unfortunately, the contour extracted from the smoothed function may lose high-resolution details present in the input function.

While many researchers have worked on modifying and extending MC, they have focused on areas other than the interpolation function. For example, there has been a lot of interest in calculating closed contours over adaptive grids. Many of these methods, such as Dual Marching Cubes [SW04] and Unconstrained Isosurface Extraction on

Arbitrary Octrees [KKDH07], reduce to MC in regions that have uniform sampling.

In contrast, some methods [CWA*07, GF98, RBB05, WJ03] assume the function provides no information besides classifying the vertices as belonging to a particular solid. In the simplest case of binary segmented data (inside/outside), these methods have a similar goal to our method in that they attempt to calculate a smooth boundary to an indicator function. The difference is that the grids they operate on do not store fractional occupancies like DIFs. This means that any initial guess of these methods is necessarily non-smooth, and a filtering pass is required to fix the poor initial guess.

The Surface Nets algorithm [GF98] segments a DIF with a contour that is both smooth and guaranteed to be within one pixel of the true contour, but it does so at a high computational cost. Surface Nets first creates an initial guess of the contour by creating a contour that is dual to the input grid. The method then performs an iterative relaxation of the surface while enforcing constraints that prevent surface vertices from exiting their dual cells. Similarly, Pressing [CWA*07] calculates an initial guess but, before applying iterative relaxation along cell edges, constrains patches that are detected as planar. In these methods, a constrained, iterative smoothing operation is required to remove the obvious aliasing that is present in the initial guess. Our method is much more efficient because we produce a contour that is free of aliasing without any iteration or optimization.

Several methods have also been proposed for contouring multi-material volume fractions stored in a grid. Contouring a DIF can be viewed as the two-material case of this more general problem. The multi-material contouring method discussed by Bonnell et al. [BDS*03] extends Marching Tetrahedra (MT) to the multi-material setting. However, this method degenerates to linear interpolation with two materials and generates surfaces with the same oscillations as those introduced by MC on DIFs.

Another approach to solving the multi-material problem is to use particles that repel each other and are attracted to the interface between materials [MWK*08]. A costly Delaunay triangulation of the particles can then be used to produce triangles with good aspect ratios. However, the restorative forces that place particles on the boundary require a smooth function, which the authors achieve by blurring the material composition functions. This results in the same loss of quality and volume preservation that affects the surface reconstructed from a blurred function in MC.

Anderson et al. propose two methods of contouring multi-material volume fractions. The first method [AGDJ08] finely subdivides each cell and randomly assigns subcells to be different materials so that the ratio of materials in the subcells approximate the volume fractions in the original cells. Subcells are then randomly swapped in a simulated annealing minimization of the surface area between material types. Because each subcell can be only one material type, the resulting surface contoured over the subcells will still be blocky. In their second paper [AGDJ10], the authors address this problem by solving for new vertex positions that minimize surface curvature. Unfortunately, the accuracy of volume preservation is determined by the level of subdivision and, even for a highly subdivided grid, the proposed energy functional does not prevent oscillations in the contoured surface.

There have also been several recent techniques designed to improve the quality of the triangles in surfaces produced by MC [HW90, SSS06, RW08, DSC*08, DSS*09]. These methods typically modify the topology of the surface produced by MC to avoid long, skinny triangles. Unfortunately, the visual artifacts from using MC on DIFs arise from the incorrect assumption that the underlying function is smooth. Our technique improves surface geometry by correcting that assumption. Methods that improve triangle quality of MC surfaces solve an orthogonal problem and can (perhaps even should) be used in conjunction with our method.

3. Calculating contours from DIFs

Our input is a DIF in which we are given the fractional occupancies for cells in a regular grid. Since every sampled value corresponds to a cell volume, we consider samples to be located at the center of cells, and we connect the samples with a dual grid. Our contouring method is a simple modification of MC in which we replace the function that calculates vertex positions along cube edges. For simplicity we will first analyze the 2D case where areas are stored for each cell and contours are lines rather than surfaces. We then apply the technique to 3D data in Section 4. Instead of assuming that the contoured function is smooth along a dual edge and can be approximated by linear interpolation, we solve for a contour line that separates the inside/outside of the object so that the inside area matches the occupancy values of the cells. We then place the contour vertex at the intersection of the contour line and the dual edge.

There are only a few ways in which a linear surface can pass through a pair of cells in a 2D grid. Without loss of generality, we consider a pair of unit cells whose bottoms are on the $y = 0$ axis and that join along $x = 0$ as shown at the top of Figure 2. In this figure, the black lines are the primal cells, the blue line is a dual edge between function samples, and the red line is the estimated line separating exterior regions (white) and interior regions (pink). We also assume that the occupancy of the left cell a_1 is less than that of the right cell a_2 so that a contour intersects the dual edge when $a_1 \in \left[0, \frac{1}{2}\right), a_2 \in \left[\frac{1}{2}, 1\right]$. To remove symmetric cases, we also assume that the contour line we estimate is oriented upward (i.e. the y-component of the normal is positive). In each cell, the contour can pass through either a horizontal or vertical border, excluding the shared border. Since there are two cells, there are a total of four possible configurations, which we enumerate in Figure 2; Case 1 is side-side, Case 2 is bottom-top, Case 3 is bottom-side, and Case 4 is side-top.

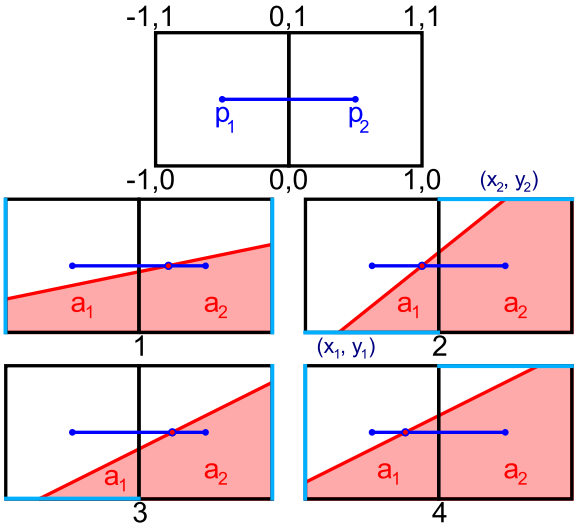

Figure 2: *We show the coordinate systems of adjacent cells and their dual edge on top. Below, we show the four configurations in which a line can intersect the cell edges. Intersected edges are highlighted in light blue. The end points of the contour line are labeled in case 2.*

Because there are multiple ways a line can intersect the pair of cells, there are two steps to finding an intersection with the contour. First, we must determine which case to use based on the values of a_1, a_2. Second, we must find the intersection between the dual edge and the estimated contour.

Our strategy for all of the cases is to first calculate the equation for the contour line. Our lines will always be of the form $y = mx + b$ with $m = \frac{y_2 - y_1}{x_2 - x_1}$ and $b = \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1}$ where (x_1, y_1) and (x_2, y_2) are the points at which the line intersects the boundary of the two cells. Notice that one component of each of the end-points is constrained for each of the four cases. Once we determine the equation of the line, we intersect the line with the dual edge $p(t)$ to find the parametric value t where $p(t) = p_1(1-t) + p_2 t$ and p_1, p_2 are the centers of the corresponding cells. Notice that because t is independent of translation, scale, and orientation, we can apply this procedure to all edges in the grid.

3.1. Case 1

In this case, x_1 and x_2 are constrained to the left and right sides of the cells, respectively. Therefore, $x_1 = -1$ and $x_2 = 1$. By integrating the line equations restricted to the cells, we can also write the area bound by this line for each cell as

$$a_1 = \int_{x_1}^{0} (mx+b)dx = \int_{-1}^{0} (mx+b)dx = b - \frac{m}{2},$$

$$a_2 = \int_{0}^{x_2} (mx+b)dx = \int_{0}^{1} (mx+b)dx = b + \frac{m}{2}.$$

Notice that y_1 and y_2 are the only free variables since a_1 and a_2 are provided by the DIF, x_1 and x_2 are constrained, and m, b are in terms of (x_1, y_1), (x_2, y_2). Furthermore, $0 \le y_1 \le 1$ and $0 \le y_2 \le 1$. If y_1 and y_2 are outside of this range, then the values of a_1 and a_2 are incompatible with the way the line intersects the edges of the cell and must correspond to one of the other three remaining cases. Hence, we test for this case by checking if the solutions for y_1 and y_2 are within their valid ranges, which yields

$$\begin{aligned} 0 \le \tfrac{3a_1 - a_2}{2} \le 1, \\ 0 \le \tfrac{3a_2 - a_1}{2} \le 1. \end{aligned} \tag{1}$$

If Equation 1 is true, then the intersection of our estimated line with the dual edge is at

$$t = \frac{a_1 - \frac{1}{2}}{a_1 - a_2}. \tag{2}$$

Interestingly, this case is identical to linear interpolation (i.e. $(1-t)a_1 + t a_2 = \frac{1}{2}$).

3.2. Case 2

In this case, intersection points are constrained to the bottom of the left cell, $y_1 = 0$, and the top of the right cell, $y_2 = 1$. The area bounded by the line in this configuration is given by the integrals

$$a_1 = \int_{x_1}^{0} (mx+b)dx = \frac{b^2}{2m},$$

$$a_2 = 1 - \int_{0}^{x_2} (1 - mx - b)dx = 1 - \frac{(1-b)^2}{2m}.$$

Again, x_1 and x_2 are the only free variables in these equations. Furthermore, $-1 \le x_1 \le 0$ and $0 \le x_2 \le 1$ yields the conditions

$$\begin{aligned} 0 \le 2a_1 + 2\sqrt{a_1 - a_1 a_2} \le 1, \\ 0 \le 2 - 2a_2 + 2\sqrt{a_1 - a_1 a_2} \le 1. \end{aligned} \tag{3}$$

Intersecting this line with the dual edge provides an extremely simple solution for the parameter t.

$$t = \frac{3}{2} - a_1 - a_2 \tag{4}$$

Notice that a_1 and a_2 are quadratic in b, so m and b have two solutions. In Case 2 the solution for t is identical regardless of which solution we use. Cases 3 and 4 produce different solutions depending on the choice of m and b, but only one of these solutions is valid; that is, the line formed by m, b intersects the boundary between adjacent cells in only one of the two solutions. We show only the valid solutions to Cases 3 and 4, and these solutions are valid over the entire domain.

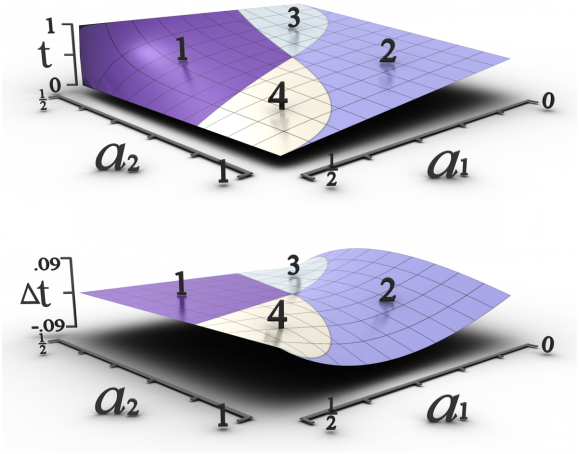

Figure 3: *For each of the four cases, we color the values of a_1 and a_2 that correspond to that case and plot the function value of t provided by that case (top). The bottom shows the difference between our function and the values provided by linear interpolation.*

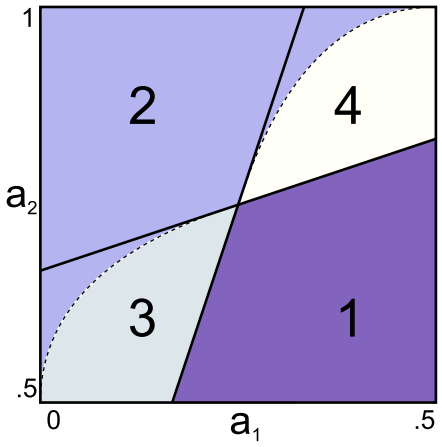

Figure 4: *The half-spaces (solid lines) used in the check for case 1 segment the domain into 4 regions. The two smaller regions are partitioned by quadratics (dashed lines).*

3.3. Case 3

In this case, our estimated line intersects the bottom of the left cell, $y_1 = 0$, and the right of the right cell, $x_2 = 1$. Expressing a_1 and a_2 in terms of integrals gives

$$a_1 = \int_{x_1}^{0} (mx+b)dx = \frac{b^2}{2m},$$

$$a_2 = \int_{0}^{1} (mx+b)dx = b + \frac{m}{2}.$$

In this situation, x_1 and y_2 are the free variables and must satisfy $-1 \le x_1 \le 0$ and $0 \le y_2 \le 1$. Solving for these variables gives the conditions

$$\begin{aligned} 0 \le \tfrac{a_1+\sqrt{a_1(a_1+a_2)}}{a_2} \le 1, \\ 0 \le 2a_1+2a_2-2\sqrt{a_1(a_1+a_2)} \le 1. \end{aligned} \tag{5}$$

Finally, intersecting this line with the dual edge gives the intersection parameter

$$t = 1 - \frac{2a_2-1}{8a_1+4a_2-8\sqrt{a_1(a_1+a_2)}}. \tag{6}$$

3.4. Case 4

The final case we consider is the situation where the estimated line intersects the left side of the left cell, $x_1 = -1$, and the top of the right cell, $y_2 = 1$. Again, we express a_1 and a_2 in terms of integrals, which yields

$$a_1 = \int_{-1}^{0} (mx+b)dx = b - \frac{m}{2},$$

$$a_2 = 1 - \int_{0}^{x_2} (1-mx-b)dx = 1 - \frac{(1-b)^2}{2m}.$$

For this case to be valid, $0 \le y_1 \le 1$ and $0 \le x_2 \le 1$. To simplify the equations and show the symmetry with Case 3, we substitute $\bar{a}_1 = 1-a_2$ and $\bar{a}_2 = 1-a_1$. Solving for y_1, x_2 gives the conditions

$$\begin{aligned} 0 \le \tfrac{\bar{a}_1+\sqrt{\bar{a}_1(\bar{a}_1+\bar{a}_2)}}{\bar{a}_2} \le 1, \\ 0 \le 2\bar{a}_1+2\bar{a}_2-2\sqrt{\bar{a}_1(\bar{a}_1+\bar{a}_2)} \le 1. \end{aligned} \tag{7}$$

When these conditions are true, we intersect our estimated line with the dual edge and find that

$$t = \frac{2\bar{a}_2-1}{8\bar{a}_1+4\bar{a}_2-8\sqrt{\bar{a}_1(\bar{a}_1+\bar{a}_2)}}. \tag{8}$$

3.5. Efficient case selection

From the previous description, it appears that 16 checks are necessary to choose the correct case (4 inequalities are shown for 4 cases). However, we reduce the number of checks to 2 simple checks against lines with one additional check that we perform $\frac{1}{3}$ of the time.

First note that the two boundary half-spaces ($a_2 \le 3a_1$ and $3a_2 \le a_1 + 2$) of Case 1 segment the domain into four regions (shown in Figure 4). The region where the half-spaces intersect is exactly Case 1. Notice also that the region that intersects neither half-space contains only Case 2. Each of these large regions occupies $\frac{1}{3}$ of the domain. The remaining regions contain Case 2 and either Case 3 or Case 4. We differentiate these small regions by a single additional test against a quadratic. Specifically, we select Case 3 if $(2a_1+2a_2-1)^2 < 4a_1(a_1+a_2)$ and we select Case 4 if $(2\bar{a}_1+2\bar{a}_2-1)^2 < 4\bar{a}_1(\bar{a}_1+\bar{a}_2)$.

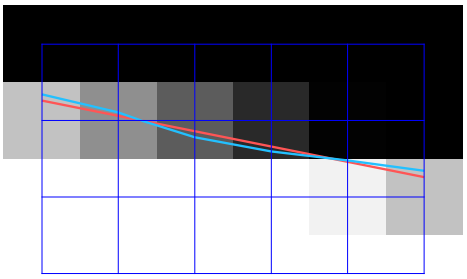

Figure 5: *Values of the DIF calculated from a linear function are shown in gray-scale. From the DIF, we calculate contours using our method (red) and MC (cyan) where vertices are placed along the blue lines of the dual grid.*

3.6. 2D Summary

To summarize our algorithm in 2D, we check if there is an intersection along each dual edge by determining whether the values of the cells span $\frac{1}{2}$. If so, we use the method described in Section 3.5 to determine which case the cell values correspond to. We then compute the parameter value t along the edge using the corresponding formula for t in either Equation 2, 4, 6, or 8 to find the intersection point and generate the topology of the surface in each dual cell using the MC algorithm.

Figure 4 shows the domains of the four cases according to Equations 1, 3, 5, and 7. From the figure, it is easy to see that the domains are disjoint except along the shared boundary of the cases and that the domains cover the entire valid parameter range for a_1, a_2. Hence, for a given set of values a_1, a_2 in a DIF, exactly one of the four cases will apply. The top of Figure 3 also shows a plot of the value of t given by the corresponding cases and shows that the function is smooth when transitioning between cases. Notice also that Case 1 corresponds exactly to linear interpolation, which produces the discontinuity as a_1, a_2 both approach $\frac{1}{2}$.

Figure 3 (bottom) shows the difference between parameter values found using our intersection function and standard linear interpolation. Since Case 1 is identical to linear interpolation, the difference is 0. However, the other three cases are different from linear interpolation, and our function smoothly diverges from linear interpolation up to nearly a tenth of a grid cell. Our method reproduces straight lines exactly because we directly calculate the intersection of lines with cells. Therefore, linear interpolation underestimates the distance to linear contours in cases where a_1 is small. Likewise, linear interpolation overestimates the distance to linear contours when a_2 is large.

Figure 5 shows a 2D example where we reconstruct a straight line from a DIF. The red line represents our method and the light blue line represents the contour created by MC using linear interpolation. Our method exactly reproduces the original linear function represented by the DIF. The oscillation effect that is apparent in the MC reconstruction results in normals that vary significantly from those of the original surface.

Although we reproduce lines exactly, we do not preserve the area of objects along curved boundaries. The linear functions between pairs of cells that we show in Figure 2 preserve the areas a_1 and a_2, and we place our vertex at the point where these lines intersect the dual edges. However, when we connect these vertices using the MC table, we do not reproduce the areas in the cells if the resulting contour is not a straight line. In general, our method generates shapes that underestimate areas in regions with positive curvature and overestimate areas in regions with negative curvature.

4. Extension to 3D

In 2D we were able to derive the intersection function defined in Figure 3 because the equation for a line has two degrees of freedom. The two areas in adjacent cells, a_1 and a_2, therefore determine a unique line. However, a 3D solution is more complicated because a plane has three degrees of freedom and is not fully constrained by two samples. Although only one more grid sample is required to fully determine a plane, there is no symmetric choice of a single additional sample. Preserving symmetry therefore requires all neighbor cells and over-constrains the problem. While we could find a separating plane using a non-linear least squares minimization, we propose a simpler solution.

Given an edge of the 3D dual grid, let a_1 and a_2 represent the occupancies of the cubes that the edge connects. If $a_1 < \frac{1}{2} \leq a_2$ or $a_2 < \frac{1}{2} \leq a_1$, the surface intersects that edge, and we place a vertex on the edge using the 2D method in Section 3. Specifically, we choose between the four cases using a_1, a_2 as described in Section 3.5 and find t using Equations 2, 4, 6, and 8.

The intuition for why this works is that one can imagine extruding Figure 2 out of the page into 3D. When the separating plane aligns with a Cartesian axis, the 3D case reduces to 2D. This 2D reduction is imperfect for unaligned planes, but still provides a reasonable approximation. While we cannot reproduce all 3D linear functions exactly from just a_1 and a_2, we find that this simple, efficient technique works remarkably well in practice.

5. Analysis

Our algorithm exactly reproduces two dimensional lines. This property is important because it is precisely in flat regions where aliasing patterns are most noticeable. Moreover, accurate reproduction of normals is especially important in 3D because we perceive surfaces mainly through the way light interacts with the surface, which depends on the normals of the object.

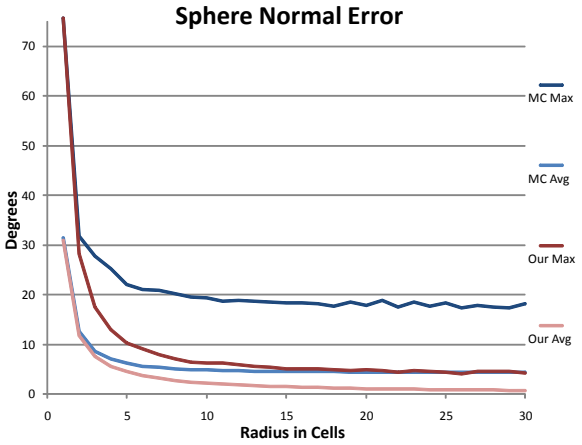

Figure 6: *The maximum and average errors in normal direction are plotted for spheres with radius varying from one to thirty cells. Once the radius is approximately fifteen, the maximum normal error from our method is equal to the average error of Marching Cubes.*

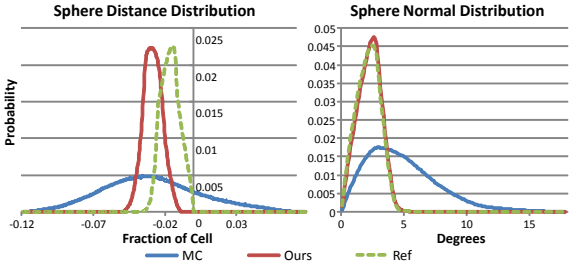

Figure 7: *For a sphere with radius of ten cells, we calculated contours that positioned vertices using linear interpolation (MC), using our intersection function (Ours). We also measure the error from placing vertices directly on the sphere's surface (Ref). The distribution of distances from an exact sphere is shown on the left, and the distribution of normal errors is shown on the right.*

A visual comparison of the MC contours and our contours shown in Figures 1, 8, and 9 indicates that applying the 2D equations to 3D works well. For example, Figure 9 shows an example of reconstructing an armadillo man from a DIF using MC (top) and our method (bottom). The MC surface is extremely noisy and obscures small details in the model, whereas our reconstruction produces a much higher quality surface that does not obscure small details in the reconstructed surface. It is even possible to see the edges from the polygons of the original surface used to create the DIF in our reconstruction.

Beyond providing visual comparisons between methods, we quantitatively compare surfaces with different curvatures. We compare MC and our method on surfaces of constant curvature by looking at spheres that vary in radius from one up to thirty cell widths. We calculate the percent occupancy of each boundary cell over a million samples and then compare the difference between the reconstructed surfaces and the true sphere by firing ten million rays from the sphere's center in different directions. For each ray, we measure the distance between intersections of the reconstructed surfaces and the ideal sphere as well as differences in their normals. We eliminate bias introduced from the position of the sphere with respect to the grid by adding random offsets to the spheres averaged over one hundred trials.

Figure 6 shows the difference in the error of surface normals between the surfaces created using our intersection function and MC as curvature decreases. This figure shows both the maximum difference between normals of the extracted surfaces and the analytic spheres as well as the average error. The error for the MC surface is shown in blue and the error for our surface is shown in red. As the radius of the spheres increases, the surface becomes locally planar and the average error in the normals of our surface almost vanishes, whereas the error for the MC surface ceases to improve. In fact, for a sphere with a radius of fifteen cells or more, even the maximum error from our method is as good as or better than the average error for surfaces produced by MC.

Figure 7 shows the distribution of error in positions and normals for spheres with a radius of ten cells. We compared the MC surface (blue), our surface (red), and a reference surface with vertices directly on the surface of the sphere (dashed green). The reference surfaces bound the quality our vertex placement algorithm. Notice that even the reference surface underestimates the radius of the sphere by an average of about 0.02 cells, because vertices are connected with planar polygons. Our method and MC both underestimate the sphere's radius by approximately 0.03 cells, but our method has a much smaller standard deviation akin to that of the reference surface. Our contour is very accurate for nearly planar surfaces (spheres with large radii), but linear interpolation is up to ten percent different from our interpolant, which gives MC a larger error distribution. The surface produced by our method reproduces normals of the sphere nearly as well as the reference surface does. In contrast, the normals of MC surfaces deviate significantly from those of the sphere.

In 2D, we exactly preserve areas of lines, but we do not preserve volume of planes in 3D. To determine how accurately we preserve the volumes of planes in 3D, we analytically calculate the occupancies of cells from planes with random orientations and positions. We contour these functions with both MC and our method and compute the absolute difference in volume between the reconstructed surfaces and the exact plane on a per cell basis. The standard deviation of the difference in reconstructed volumes sampled over 5000 planes was 2.76% for MC, whereas the standard deviation

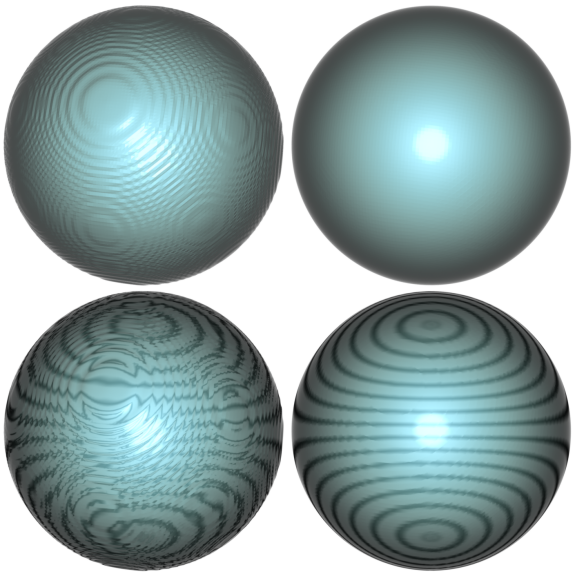

Figure 8: *When lighting and reflection are applied to 3D surfaces, the ripple patterns produced by Marching Cubes (left) are especially obvious compared to our method (right). Top: surfaces using diffuse lighting and specular highlights. Bottom: reflection lines.*

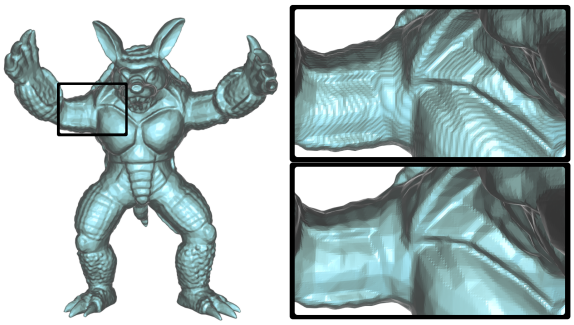

Figure 9: *Our interpolant can find accurate surface intersections for complicated discrete indicator functions like an armadillo man sampled over a 512^3 grid. For the highlighted region, we show a zoomed picture of the surface found using linear interpolation (top) and the surface found using our interpolant (bottom).*

was only 0.07% for our method. The average absolute difference in volume was 8.36% for MC and only 0.46% for our method. These results demonstrate that our method provides a good approximation of arbitrary 3D planes.

Although statistics show a clear difference between contouring methods, these numbers cannot adequately convey the displeasing banding patterns present in MC surfaces. Figure 8 shows a sphere generated by MC (left) and our method (right) under typical lighting conditions (top) and with a reflection map (bottom). The error in vertex positions depends on the size of the grid used, but normal error is independent of grid resolution and is clearly visible.

We have focused our analysis on spheres of various curvature, but our method works equally well for complex surfaces that have varying curvature. For example, the result of contouring a DIF of the armadillo man with our method and MC, shown in Figure 9, is striking. Figure 1 shows another complex surface. In both of these examples, the DIF is calculated to very high precision so that errors in contouring dominate errors in sampling. MC produces a surface with noticeable ridges and, even after applying a blur with a Gaussian kernel of size 3 and 7 to the DIF, the ridges in the MC surface are still faintly visible on the chest. Although blurring makes the surface smoother, it also destroys details of the original shape. Moreover, blurring the DIF [MGS96] or performing an iterative, constrained surface fairing [GF98] takes a significant amount of time in addition to running MC. In contrast, our method produces a high quality surface without any iteration. In terms of speed, we were unable to determine any significant difference in time between running MC with linear interpolation and our method.

Although the formulation of our method assumes that the input is a perfect DIF, perfect DIFs are hard to come by in practice. We tested our contouring method in the presence of noise by calculating an approximate DIF from laser range scans of the head of Michelangelo's David. We computed the DIF with a wavelet reconstruction method [MPS08] using Haar wavelets and no blurring. The DIF is imperfect both because of noise in the scanned data and from structured noise resulting from the scanned surface being open at David's neck. Notice in Figure 10 that ridges in the surface are greatly reduced in our surface compared to MC, but the improvement in quality is less than with perfect DIFs. Block artifacts from using a Haar basis are visible in both contours, but are partially obscured by the ridges from MC.

6. Conclusion and Future Work

Discrete indicator functions are an important class of implicit functions that require special consideration to produce accurate contours. Replacing the interpolation function used in MC is both simple and effective. Indeed, the apparent visual improvement in the surfaces generated by our method over MC is confirmed by our statistical analysis.

We should note that our method is specific to DIFs and should not be applied to arbitrary functions. Our technique is not appropriate in these situations, and linear interpolation typically suffices. However, DIFs do arise in practice and, for these functions, our method greatly improves the quality of the extracted surface with essentially zero cost.

One drawback of our method is that we do not preserve

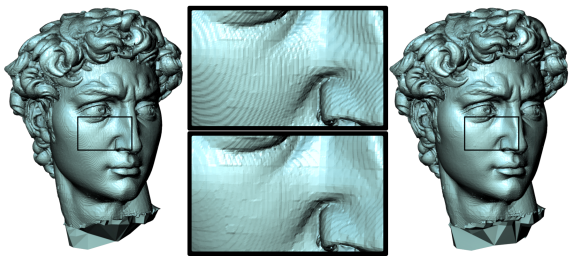

Figure 10: *Reconstruction of the head of Michelangelo's David using MC (left and top) and our interpolant (right and bottom).*

the volume of cells that intersect curved surfaces. Most methods preserve volume by performing a global optimization, but it may be possible to locally estimate the curvature of a surface from the DIF or locally fit a curved object like an ellipsoid to calculate a volume preserving surface. Furthermore, our method is restrictive in that we only handle two-materials as opposed to multiple materials considered in volume fraction methods. We would like to investigate the possibility of extending our approach to multiple materials and preserving volume in the future.

In this paper, we have only considered situations in which cells that intersect the contour are uniform in size, but it may be possible to apply the principles of our method to adaptive grids such as octrees. This may be more complicated because a plane passing through a large cell may pass through any of the smaller cells bordering it so that more cells must be considered. Additionally, is not clear that the problem is always well defined, even in the 2D case.

Acknowledgments

The armadillo man is from the Stanford 3D Scanning Repository and Poseidon is by ZBrush/Jim Kalogiratos.

References

[AGDJ08] ANDERSON J. C., GARTH C., DUCHAINEAU M. A., JOY K.: Discrete multi-material interface reconstruction for volume fraction data. *Computer Graphics Forum (Proceedings of Eurographics) 27*, 3 (2008), 1015–1022. 3

[AGDJ10] ANDERSON J. C., GARTH C., DUCHAINEAU M. A., JOY K.: Smooth, volume-accurate material interface reconstruction. *IEEE Trans. Vis. Comp. Grap. 16*, 5 (2010), 802–814. 3

[BDS*03] BONNELL K. S., DUCHAINEAU M. A., SCHIKORE D. R., HAMANN B., JOY K. I.: Material interface reconstruction. *IEEE Trans. Vis. Comp. Grap. 9*, 4 (2003), 500–511. 3

[CWA*07] CHICA A., WILLIAMS J., ANDUJAR C., BRUNET P., NAVAZO I., ROSSIGNAC J. R., VINACUA A.: Pressing: Smooth isosurfaces with flats from binary grids. *Computer Graphics Forum 27*, 1 (2007), 36–46. 3

[DSC*08] DIETRICH C., SCHEIDEGGER C., COMBA J., NEDEL L., SILVA C.: Edge groups: An approach to understanding the mesh quality of marching methods. *IEEE Trans. Vis. Comp. Grap. 14* (2008), 1651–1666. 3

[DSS*09] DIETRICH C. A., SCHEIDEGGER C. E., SCHREINER J., COMBA J. L., NEDEL L. P., SILVA C. T.: Edge transformations for improving mesh quality of marching cubes. *IEEE Trans. Vis. Comp. Grap. 15* (2009), 150–159. 3

[GF98] GIBSON S., FRISKEN F.: Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention* (1998), pp. 888–898. 3, 8

[Gre07] GREEN C.: Improved alpha-tested magnification for vector textures and special effects. In *SIGGRAPH Courses* (2007), pp. 9–18. 1

[HW90] HALL M., WARREN J.: Adaptive polygonalization of implicitly defined surfaces. *IEEE Comput. Graph. Appl. 10* (1990), 33–42. 3

[Kaz05] KAZHDAN M.: Reconstruction of solid models from oriented point sets. In *Proceedings of SGP* (2005), pp. 73–84. 2

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of SGP* (2006), pp. 61–70. 2

[KKDH07] KAZHDAN M., KLEIN A., DALAL K., HOPPE H.: Unconstrained isosurface extraction on arbitrary octrees. In *Proceedings of SGP* (2007), pp. 125–133. 3

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 163–169. 2

[LH06] LEFEBVRE S., HOPPE H.: Perfect spatial hashing. In *Proceeding of SIGGRAPH* (2006), pp. 579–588. 1

[MGS96] MOR A., GIBSON S., SAMOSKY J.: Interacting with 3-dimensional medical data: Haptic feedback for surgical simulation. In *Phantom User Group Workshop* (1996). 2, 8

[MPS08] MANSON J., PETROVA G., SCHAEFER S.: Streaming surface reconstruction using wavelets. *Computer Graphics Forum (Proceedings of SGP) 27*, 5 (2008), 1411–1420. 2, 8

[MWK*08] MEYER M., WHITAKER R., KIRBY R. M., LEDERGERBER C., PFISTER H.: Particle-based sampling and meshing of surfaces in multimaterial volumes. *IEEE Trans. Vis. Comp. Grap. 14*, 6 (2008), 1539–1546. 3

[RBB05] REITINGER B., BORNIK A., BEICHEL R.: Constructing smooth non-manifold meshes of multi-labeled volumetric datasets. In *Proceedings of WSCG* (2005), pp. 227–234. 3

[RW08] RAMAN S., WENGER R.: Quality isosurface mesh generation using an extended marching cubes lookup table. *Computer Graphics Forum 27*, 3 (2008), 791–798. 3

[SSS06] SCHREINER J., SCHEIDEGGER C., SILVA C.: High-quality extraction of isosurfaces from regular and irregular grids. *IEEE Trans. Vis. Comp. Grap. 12*, 5 (2006), 1205–1212. 3

[SW04] SCHAEFER S., WARREN J.: Dual marching cubes: Primal contouring of dual grids. In *Proceedings of Pacific Graphics* (2004), pp. 70–76. 2

[WJ03] WU Z., JR J. M. S.: Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering 58*, 2 (2003), 189–207. 3

[WK93] WANG S. W., KAUFMAN A. E.: Volume sampled voxelization of geometric primitives. In *Proceedings of IEEE Visualization* (1993), pp. 78–84. 2

[WK94] WANG S. W., KAUFMAN A. E.: Volume-sampled 3d modeling. *IEEE Comp. Graph. and Appl. 14*, 5 (1994), 26–32. 2

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for soft objects. *The Visual Computer 2*, 4 (1986), 227–234. 2