Extraction of Level Sets from 3D Images

David Eberly
Geometric Tools, LLC
http://www.geometrictools.com/
Copyright © 1998-2008. All Rights Reserved.

Created: September 7, 2000
Last Modified: March 2, 2008

Contents

1 Introduction **2**

2 Extraction Using Linear Interpolation **2**

3 Consistent Triangle Ordering **4**

1 Introduction

For the purposes of this algorithm, a *3D image* is assumed to be a $B_x \times B_y \times B_z$ array of integer values. The voxel locations are (x, y, z) where $0 \le x < B_x$, $0 \le y < B_y$, and $0 \le z < B_z$. No restriction is assumed on the voxel values I_{xyz} other than they are integer-valued.

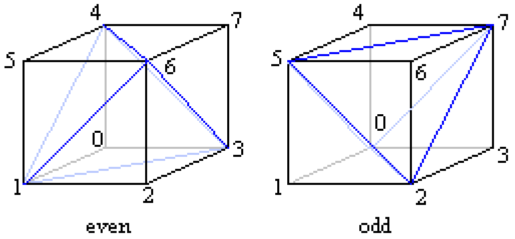
A continuous formulation of the image is required. For *domain cube* with corners (x_0, y_0, z_0), (x_0+1, y_0, z_0), (x_0, y_0+1, z_0), (x_0+1, y_0+1, z_0), (x_0, y_0, z_0+1), (x_0+1, y_0, z_0+1), (x_0, y_0+1, z_0+1), and (x_0+1, y_0+1, z_0+1), let $F(x, y, z)$ be a continuous function that represents the image on the cube. It is not necessary that F match the pixel values at the eight corners, but the algorithm discussed in this document does have that property.

If L is a value in the range of F, then the *level set of* F *for* L is the set of points (x, y, z) that satisfy $F(x, y, z) = L$. Generally one expects the level sets to consist of surfaces, but isolated points and isolated edges are possible. For example, if $F(x, y, z) = x^2 + y^2 + z^2$, the level set $F(x, y, z) = L > 0$ consists of a single sphere, but the level set $F(x, y, z) = 0$ is a single point. Another example is $F(x, y, z) = x^2 + y^2$. The level set $F(x, y, z) = L > 0$ is a single cylinder, but the level set $F(x, y, z) = 0$ is the line $(0, 0, t)$ for $t \in \mathbb{R}$. The algorithms in this document construct both surfaces, curves, and points. A level set for the entire image is constructed from the level sets for the functions on the domain cubes.

2 Extraction Using Linear Interpolation

The continuous formulation on domain cubes is based on decomposing each cube into five tetrahedra and using linear interpolation on each tetrahedron. The decomposition is symmetric in the same sense as that for domain squares of a 2D image (see `ExtractLevelCurves.pdf`). Figure 2.1 shows the decomposition for an even and an odd cube.

Figure 2.1 Even and odd decompositions of a cube.



For an even decomposition, the tetrahedra are $\langle 0,1,3,4\rangle$, $\langle 1,2,3,6\rangle$, $\langle 4,5,6,1\rangle$, $\langle 4,6,7,3\rangle$, and $\langle 1,3,6,4\rangle$. For an odd decomposition, the tetrahedra are $\langle 0,7,4,5\rangle$, $\langle 2,5,7,6\rangle$, $\langle 0,1,2,5\rangle$, $\langle 0,2,3,7\rangle$, and $\langle 0,2,5,7\rangle$. The linear interpolant $F(x, y, z)$ on a tetrahedron with vertices $\langle x_i, y_i, z_i; F_i\rangle$ for $0 \le i \le 3$ is defined by computing the barycentric coordinates of the (x_i, y_i, z_i) with respect to the tetrahedron, say the values are c_i for $0 \le i \le 3$, then computing $F = \sum_{i=0}^{3} c_i F_i$.

Given a level value L and a tetrahedron in the decomposition, each vertex value F satisfies $F < L$, $F = L$, or $F > L$, for a total of 81 possibilities for the tetrahedron. Each possibility corresponds to a tetrahedron that has no level set points, a single level set point, a single level set line segment, a single triangle, a single planar quadrilateral, or the entire tetrahedron (function is L at all vertices). Tables 2.1 and 2.2 show the cases and permutations.

Table 2.1 *Some cases and permutations for level sets of F on a tetrahedron.*

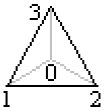
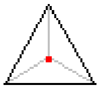
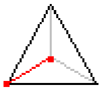
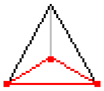
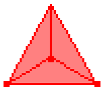
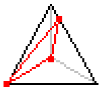
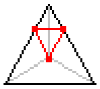
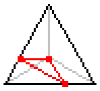
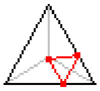
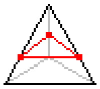
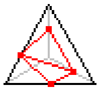
case	*0*	*1*	*2*	*3*	*4*	*5*
permute						
0	$++++$	$0+++$	$00++$	$000+$	0000	$00+-$
1	$----$	$+0++$	$+00+$	$00+0$		$-00+$
2		$++0+$	$++00$	$0+00$		$+-00$
3		$+++0$	$0++0$	$+000$		$0+-0$
4		$0---$	$0+0+$	$000-$		$0+0-$
5		$-0--$	$+0+0$	$00-0$		$-0+0$
6		$--0-$	$00--$	$0-00$		$00-+$
7		$---0$	$-00-$	-000		$+00-$
8			$--00$			$-+00$
9			$0--0$			$0-+0$
10			$0-0-$			$0-0+$
11			$-0-0$			$+0-0$

Table 2.2 *Some cases and permutations for level sets of F on a tetrahedron.*

case	*6*	*7*	*8*	*9*	*10*
permute					
0	$0++-$	$0+--$	$0+-+$	$+++-$	$++--$
1	$-0++$	$-0+-$	$+0+-$	$++-+$	$-++-$
2	$+-0+$	$--0+$	$-+0+$	$+-++$	$--++$
3	$++-0$	$+--0$	$+-+0$	$-+++$	$+--+$
4	$0--+$	$0-++$	$0-+-$	$---+$	$+-+-$
5	$+0--$	$+0-+$	$-0-+$	$--+-$	$-+-+$
6	$-+0-$	$++0-$	$+-0-$	$-+--$	$00-+$
7	$--+0$	$-++0$	$-+-0$	$+---$	$+00-$

The vertex ordering is shown in the figure for case 0 and is the same ordering for the figures of the remaining cases. Permutation 0 refers to that ordering. For example, case 5 and permutation 0 indicates that vertices 0 and 1 have zero value, vertex 2 has positive value, and vertex 3 has negative value.

3 Consistent Triangle Ordering

After extracting the level set as a manifold triangle mesh, it is not guaranteed (and usually not the case) that the vertex ordering for the triangles is consistent. The vertices for one triangle can be clockwise and the vertices for an adjacent triangle can be counterclockwise. This makes it difficult to compute smooth normals for the purposes of lighting the surface in a visualization. The image class provides a function call, `MakeConsistentOrdering` that reorders the triangle connectivity indices to form a consistent ordering. That call constructs a vertex-edge-triangle table that has triangle adjacency information. Each connected component of the mesh is traversed. For each component, the first triangle's ordering is used as is. Each adjacent triangle has its indices reordered to be consistent. If the two triangles share an edge $\langle I_0, I_1\rangle$, and if the first triangle contains that edge in the order listed, then the adjacent triangle must contain those indices in the reverse order. Once the adjacent triangles are processed, the algorithm is then applied to the adjacent triangles of each of these already processed triangles.