Extracting Boundary Surfaces/Solids from a Gridded Cube

Brandon J. Wilde and Clayton V. Deutsch

Interpolating a distance function is a useful method for modeling domain boundaries. It can be difficult to visualize the boundaries generated by this method. Extracting a surface/solid from the grid of interpolated distance function values provides a format more amenable for visualization, particularly in three dimensions. The marching cubes algorithm is able to extract the surface/solid. The surface/solid is written out in a format which can be used by a number of commercial software packages.

Introduction

The spatial extent of stationary a domain is represented by a boundary. The boundary indicates the transition from one domain to another. Interpolating a distance function has been shown to be a useful method for estimating the boundary location (McLennan, 2008; Munroe and Deutsch, 2008 a,b; Hosseini, 2009). Once the distance function has been interpolated the boundary location can be identified by various means. The most obvious is to apply a constant threshold of zero. Similarly, a constant positive threshold could be applied to arrive at a domain that is big everywhere; a constant negative threshold could be applied to arrive at a domain that is small everywhere. The boundary location is determined by identifying whether each location falls inside or outside the domain. Any locations with distance function greater than or equal to the threshold are outside the domain and any locations with distance function less than the threshold are inside the domain. Simulated distance function values could also be used to determine whether a location is inside or outside the domain. The boundary lies at the transition between locations outside and inside the domain.

In three dimensions, the boundary position is identified by a cube of interpolated distance function values. Viewing the boundary typically involves viewing two-dimensional slices through the cube. It can be useful to view the surface in three dimensions without the cube of interpolated distance function values obscuring the view of the surface. This requires extracting the surface from the cube. An algorithm for extracting the surface and a program with its GSLIB-style implementation are described.

Boundary Surface/Solid Extraction

It can be useful to have a boundary surface in a format that is amenable to 3D visualization. A grid of simulated and interpolated distance function values is not easily visualized in 3D. A tool called `ExtractSolid` is available to extract a surface from a regular 3D grid. This tool implements the marching cubes algorithm which forms a polygonal surface representation through a scalar field sampled on a rectangular 3D grid (Bourke, 1994). The marching cubes algorithm is one of the latest algorithms of surface construction used for viewing 3D data. It was first described by Lorensen and Cline (1987). It produces a triangle mesh which forms a surface representation of the boundary. This algorithm is well suited to extracting a surface from interpolated distance function. This algorithm has been applied in many fields including biochemistry, biomedicine, deformable modeling, digital sculpting, environmental science, mechanics and dynamics, natural phenomena rendering, visualization algorithm analysis, etc (Newman and Li, 2006).

The algorithm proceeds by considering groups of 8 adjacent cells in the configuration shown in Figure 1. The 8 cells are considered as the 8 vertices (black numbers) of a rectangular prism with the vertices connected by 12 edges (grey numbers). If all the vertices are outside or inside the domain then the boundary does not pass through the cell, but if some vertices are inside and others are outside the domain then the boundary does pass through the cell. If one vertex is outside the domain and an adjacent vertex is inside the domain then the boundary cuts the edge between these two vertices. The position that it cuts the edge is linearly interpolated from the distance function values at the vertices.

Consider that vertex 3 is the only vertex inside the domain and the other seven vertices are all outside the domain. Vertex 3 is connected to vertices 0, 2, and 7 by edges 3, 2, and 11 respectively. The boundary cuts through these three edges. Interpolating between the vertices to determine where the edges are cut yields the triangular facet shown in Figure 2. Consecutively considering all groups of 8 adjacent cells yields a surface composed of triangular facets, or a triangulated irregular network (TIN). Each of the 8 vertices has two possible values (outside or inside) for a total of 2^8=256 possible combinations. Accounting for reflections and rotations reduces the number of unique cases to 14, not including the case where all vertices are either outside or inside the domain. Reflective and rotational symmetry are illustrated in Figure 3. The 15 unique cases are shown in Figure 4. The determination of a vertex falling outside or inside the domain can be based on a constant threshold or the simulated distance function. The linear interpolation for both cases is shown in Figure 5. Where the linear interpolation of the interpolated distance function intersects the constant threshold or the linear interpolation of the simulated distance function is where the boundary is located.

For a 3D grid with *nx* x *ny* x *nz* cells, there are (*nx*-1) x (*ny*-1) x (*nz*-1) groups of 8 cells to be checked for boundary crossing. The polygons that define the boundary are determined by considering all possible groups of 8 cells. Fortunately the algorithm is fast. The result is a polygon in 2D or a TIN in 3D representing the boundary between the domains. There are a number of options for outputting the TIN in a format that can be imported by various commercial software packages. For Petrel, the TIN is exported as 'Irap classic lines (ASCII)' format; for Gocad, the TIN is exported as 'Irap RMS triangle surface' format; for other software packages such as Vulcan, the TIN is exported as 'Drawing interchange format' (DXF).

Ambiguities

There are cases where the arrangement of inside and outside vertices can have multiple solutions for the cutting of the edges. Consider the 2D case where the two vertices opposite from each other are inside (black filled) and the other two vertices are outside (white filled) the domain as shown in Figure 6. There are two options for how the edges could be cut. This is referred to as face ambiguity (Lewiner et al., 2003). In 3D, in addition to face ambiguity, there can also be internal ambiguity. An example is case 4 (Figure 4) where two diagonally opposite vertices of the cube differ in their classification from the rest of the vertices. As shown in Figure 7, there are two plausible solutions. Figure 7-left shows the standard facetization which contains two disjoint facets; Figure 7-right shows a different facetization where the facets form a tube or tunnel. A number of authors have discussed various means for resolving these types of ambiguities ensuring that the final surface generated by the marching cubes algorithm is topologically correct, in particular Chernyaev (1995) and Lewiner et al. (2003). The `ExtractSolid` program described herein does not correct any such ambiguities. It is assumed that for geological purposes the basic implementation of the marching cubes algorithm is sufficient. This is likely true as the coordinates of the vertices of the TIN are of primary interest while the nature of their connections is of secondary interest.

Example

Figure 8-left shows a number of slices from a 3D cube of interpolated distance function values. Applying the typical threshold of zero distance to the interpolated values yields the distinct domains shown in Figure 8-right. From these slices the practitioner starts to get a feel for what the boundary looks like. A number of slices can be viewed in 3D simultaneously to give perhaps a better idea of what the boundary looks like as shown in Figure 9. This format is still not ideal as slices on top cover portions of slices below. The marching cubes algorithm is applied to this cube of distance function values creating a TIN representation of the surface as shown in Figure 10. This surface can be loaded into software capable of rendering a 3D view of the surface. The surface can be rotated and magnified to give a clear view of the surface location and features.

Conclusions

Visualizing geologic boundaries defined by a grid of interpolated distance function values is difficult. Such boundaries are more easily visualized as a surface/solid. The marching cubes algorithm is an efficient way of extracting a surface from a regular grid. This algorithm is implemented in the program `ExtractSolid`. This program reads in a gridded model and, using the marching cubes algorithm, extracts the surface corresponding to the boundary between domains. The surface is made up of many triangles and is a triangulated irregular network (TIN). The surface can be exported using different formats which can be utilized by various commercial software packages. Only the basic marching cubes algorithm is implemented. None of the various enhancements to the algorithm that detect and correct ambiguities in the surface are included. This is a potential area for future work.

References

Bourke P, 1994. Polygonising a scalar field. Accessed at http://paulbourke.net/geometry/polygonise/ on June 24, 2011.

Chernyaev C, 1995. Marching cubes 33: construction of topologically correct isosurfaces. Technical Report CERN CN 95-17, CERN.

Hosseini AH, 2009. Probabilistic Modeling of Natural Attenuation of Petroleum Hydrocarbons. Ph.D. Thesis. University of Alberta. 359p.

Lewiner T, Lopes H, Vieira AW, and Tavares G, 2003. Efficient implementation of marching cubes' cases with topological guarantees. Journal of Graphics Tools, 8:1-15.

Lorensen W and Cline HE, 1987. Marching Cubes: A high resolution 3D surface construction algorithm. Computer Graphics (SIGGRAPH 87 Proceedings) 21(4), p. 163-170

McLennan J, 2008. The Decision of Stationarity. Ph.D. Thesis. University of Alberta. 191p.

Munroe MJ and Deutsch CV, 2008a. A methodology for modeling vein-type deposit tonnage uncertainty. Center for Computational Geostatistics Annual Report 10. University of Alberta. 10p.

Munroe MJ and Deutsch CV, 2008b. Full calibration of C and β in the framework of vein-type deposit tonnage uncertainty. Center for Computational Geostatistics Annual Report 10. University of Alberta. 16p.

Newman, T.S. and Yi, H., 2006, A survey of the marching cubes algorithm, Computers and Graphics, 30:854-879.

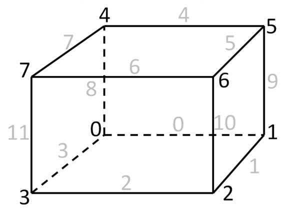


Figure 1: Vertex and edge numbering convention for marching cubes (modified from Bourke, 1994).

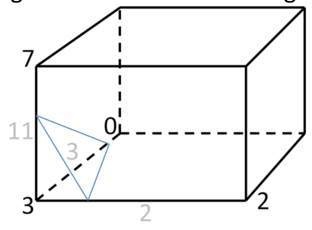


Figure 2: Triangular facet representing boundary through the cell (modified from Bourke, 1994).

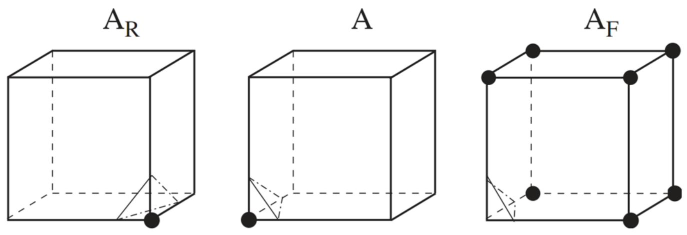


Figure 3: Illustration of reflective (A with A_F) and rotational (A with A_R) symmetries.

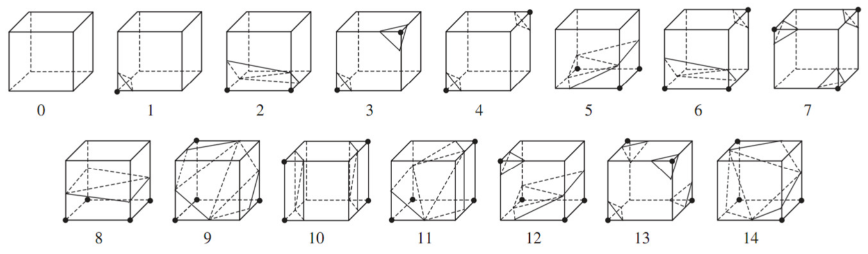


Figure 4: The 15 surface intersection cases (from Newman and Yi, 2006).

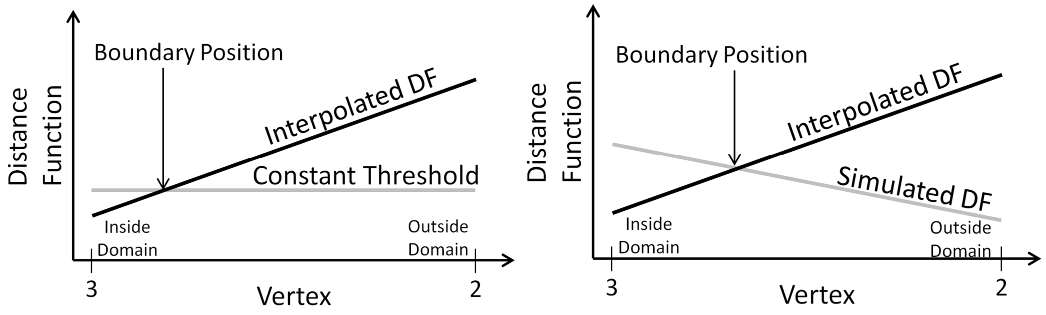


Figure 5: Linear interpolation based on interpolated DF and threshold to determine boundary position.

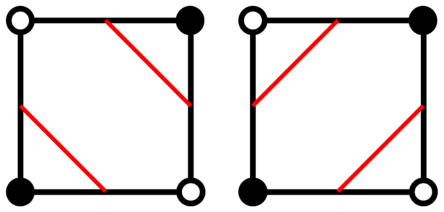


Figure 6: 2D ambiguous case.

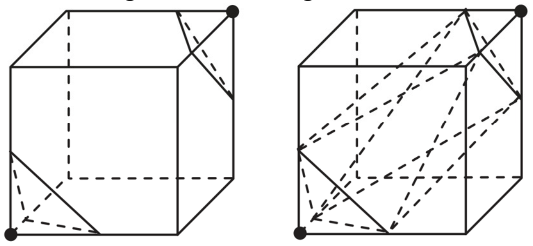


Figure 7: 3D ambiguous case (from Newman and Yi, 2006).

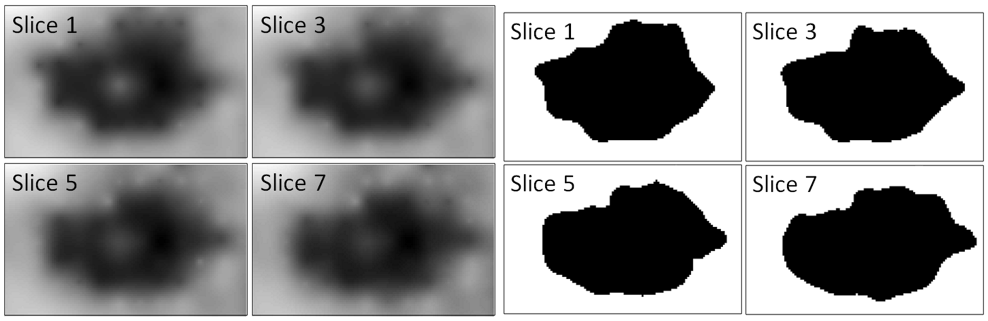
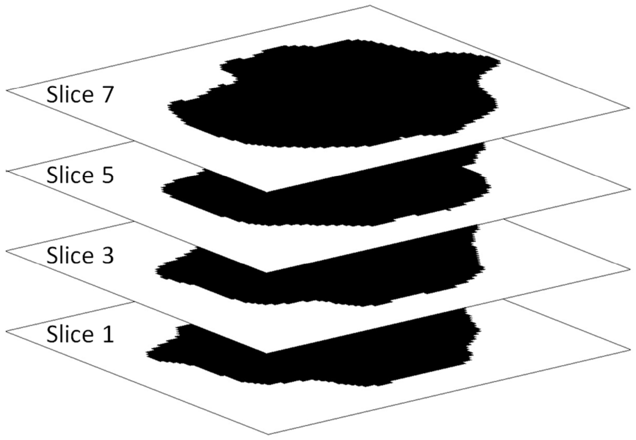


Figure 8: Interpolated distance function on the left identifies separate domains by applying a threshold of zero.



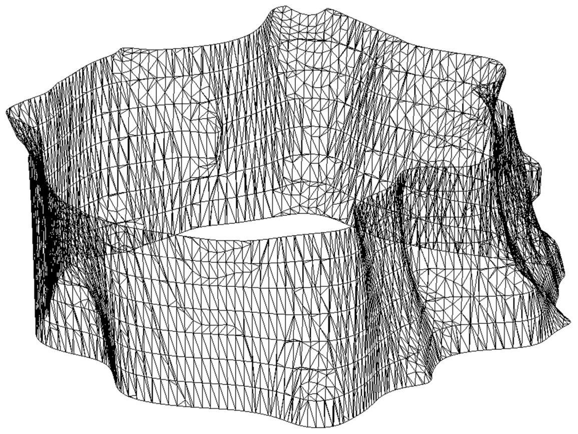


Figure 9: Viewing the boundary in 3D by stacking 2D slices.

Figure 10: 3D TIN surface view of boundary.