

Christopher M. Smemoe

C S 555

Paper 1

Volume Visualization and Recent Research in the Marching Cubes Approach

Introduction to Volume Visualization

Often, a person may have scalar values at each point in a three dimensional grid that they desire to display. An example is data from X-ray Computer Tomography (CT) scanners. CT scanners produce slices of a subject at a specified interval (at every 1-5 mm). Each slice contains a 512-by-512-by-12-bit two-dimensional array of X-ray absorption coefficients (Watt and Watt, 1992). A combination of these X-ray absorption coefficients can be used to generate scalar values at each point in each of the 2D array, creating a 3D grid of scalar values when the slices are combined. From these scalar values, a particular value can be selected as the *threshold* value. Using this threshold value, a 3D surface can be constructed that approximates the location of this value at different points in the 3D grid. This approach is called *volume rendering*. This 3D surface can be used to represent organ or bone tissue. The 3D grid of scalar data is called a *scalar field*.

The result of a finite difference simulation of contaminant concentration in a groundwater aquifer is another example of scalar values at each point in a three dimensional grid. In a finite difference simulation of groundwater contaminant transport, a 3D grid is constructed and a mathematical groundwater contaminant transport model is run using the initial conditions and boundary conditions assigned to the grid nodes. The output from the mathematical simulation is the concentration of the modeled contaminant at each node in the 3D grid. This output contaminant concentration at each node represents the scalar field for this groundwater contaminant problem. An isosurface (or a surface of equal contaminant concentration) can be generated using a threshold value as described above (See Figure 1).

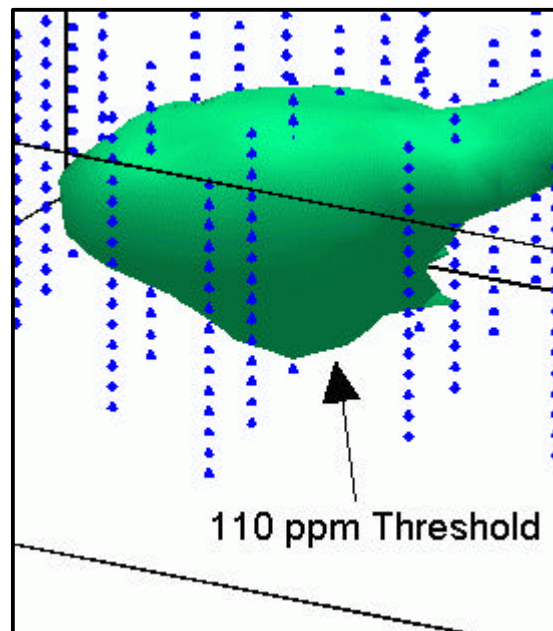


Figure 1: A volume rendering of the 110-PPM contaminant plume in an underground aquifer based on interpolated scattered data. From the Groundwater Modeling System (GMS) WWW page, 1995.

This paper will outline the history and methods of volume visualization, and will focus on the background and current research in an intermediate volume visualization technique called the marching cubes algorithm.

Volume Visualization Techniques

Watt and Watt (1992) divide volume visualization techniques into three methods: Early volume visualization techniques, intermediate geometric representation techniques, and volume rendering by ray casting.

Early Volume Visualization Techniques

One of the first approaches to volume rendering was by G. T. Herman and H. K. Liu (1979). In their approach, volume rendering located human organs from tomograms. The layers of images were divided into a 3D grid of voxels (volume elements) and each voxel was assigned to either be inside or outside the organ being rendered. Herman and Liu made the boundary voxels opaque and rendered them. One twist to this method is to determine the gradient of the scalar field at each boundary voxel. This gradient can then be used as the normal N in the shading calculation for a voxel. The gradient at each of the boundary voxels can be approximated using a central difference method (Equation 1).

$$\nabla V(X) \approx \begin{pmatrix} \frac{1}{2} \left(V(x_{i+1}, y_j, z_k) - V(x_{i-1}, y_j, z_k) \right), \\ \frac{1}{2} \left(V(x_i, y_{j+1}, z_k) - V(x_i, y_{j-1}, z_k) \right), \\ \frac{1}{2} \left(V(x_i, y_j, z_{k+1}) - V(x_i, y_j, z_{k-1}) \right) \end{pmatrix}$$

Equation 1: Determination of the gradient vector at a boundary volume element (used as the normal vector for shading at the voxel).

Intermediate Volume Visualization Techniques

Intermediate volume visualization techniques use an intermediate representation composed of polygons to represent the isosurface. Several methods of intermediate volume representation techniques exist, and Watt and Watt (1992) present two techniques.

Connecting Two-Dimensional Contours

In one technique, the 2D contours of the scalar value are determined for each slice of the grid. Lines, forming polygons between the contours, connect the vertices along these contours. One way to connect these vertices is to use a shape blending method, where lines are created between the two contour lines such that the distance of the lines is minimized (Sederberg, 1994). After polygons are formed between the 2D contours, the polygons can be rendered using a shading algorithm.

The Marching Cubes Algorithm

The other intermediate technique of volume visualization presented by Watt and Watt is called the *marching cubes* algorithm. This algorithm involves two phases. In the first phase, a pass is made through the grid to determine which voxels the surface intersects. In the second phase, a surface is generated from the boundary cubes. For each boundary cube, the vertices of the cube inside and outside the isosurface are determined. A template can then be used to determine the shape and location of the resulting polygon in each cube. Normally, $2^8=256$ templates would exist for each cube, but due to symmetry, there are only 14. Of course, if the isosurface is being generated from objects other than regular cubes, other templates will need to be used. In this algorithm, a triangle mesh is generated from the isosurface. Vertex normals can be found either from the triangles themselves or from Equation 1 above.

Volume Rendering using Ray Casting

The final technique discussed by Watt and Watt is the technique of volume rendering by ray casting. In this technique, a ray or set of rays is cast through the volume being rendered for each pixel. For each voxel intersected by each ray, the voxel is assigned an R, G, B, and O value (O is for opacity, which is a measure of the “opaqueness” of the material). R, G, and B colors are accumulated and opacity is accumulated until the accumulated opacity is greater than the threshold opacity. At this point, the pixel is rendered using the accumulated color value. The ray casting method of volume rendering provides some advantages over the intermediate surface algorithms just described, including the ability to incorporate transparency and the ability to adjust the depth of volume penetration by adjusting the opacity value.

Description of the Marching Cubes Algorithm for Volume Visualization

With the background discussion in volume visualization techniques, the marching cubes algorithm can now be discussed in more detail.

Step 1: Finding all the cubes (voxels) intersected by the isosurface

In his original paper, B. Wyvill used a recursive method to determine the cube intersected by the isosurface. In his algorithm, Wyvill started with a “seed” cube, checked all the cubes surrounding this seed cube, and added each surrounding cube that intersected the isosurface to the queue. Each of the cubes on the queue was examined until all the cubes intersecting the isosurface were found (Watt and Watt, 1992).

Recent research has focused on speeding up the process of finding the cubes intersected by the isosurface. Cignoni et al (1997) uses an interval tree approach to locate the cells intersected by the isosurface. An interval tree is a method of finding intervals along the real line that contain a given query value. Itoh and Koyamada (1995) uses an extrema graph approach to determine the cells intersecting closed isosurfaces. An extrema graph is a graph whose nodes represent extreme points in the scalar field. The graph’s arcs have a list of cell ID’s intersected by the extreme point. The cells intersected by the closed isosurface are found by visiting the cells along the extrema graph, finding the cells intersecting the isosurface, and propagating to the adjacent cells.

Step 2: Examining the cubes intersected by the isosurface and generating a set of connected polygons

After the cubes intersected by the isosurface are determined, a set of connected polygons can be generated. This set of connected polygons will form the isosurface and can be rendered.

The connected polygons are determined by considering which vertices are inside and which are outside the isosurface for each cube. After the vertices inside and outside the cube are determined, a template is used to determine the topology of the polygon within the cube. From the template, a set of triangles is generated within the cube. Linear interpolation of the scalar function represented by the isosurface is used to find the where the triangle vertices intersect the cube edges.

One area of research (see Xia, El-Sana, and Varshney, 1997) focuses on decreasing the number of triangles representing the isosurface based on the viewing direction, lighting, and visibility. In many cases, huge decreases in the number of triangles representing the object are obtained, which translates to huge decreases in time required to render the object. In one instance, a sphere with 8192 triangles was converted to a sphere with 537 triangles with no significant differences in the final rendered image.

Other areas of research include the handling of small isosurfaces the size of one voxel. Zhou, Shu, and Kankanhalli (1994) present a simple, efficient approach to represent these small types of isosurfaces.

Step 3: Rendering the isosurface or multiple isosurfaces

After the geometry of the isosurface has been generated, the isosurface can be rendered. Any of several rendering methods may be used to render the isosurface. Also, if the isosurface is rendered as a transparent object, several isosurfaces can be rendered...one within the other.

Two issues exist with respect to rendering the isosurface. One issue is the method of calculating vertex normals. The other issue is the rendering method used.

One of two methods can be used to calculate vertex normals. In the first method, the normals are calculated from the isosurface itself by taking the normalized cross product of the two edges adjacent to each vertex. In the second method, the normalized gradient of the cube at the location of each vertex can be used to render the isosurface. This gradient can be calculated by an approximation method as in Equation 1.

Any rendering algorithm can be used to render the isosurface. A constant shading value can be assigned to each triangle or a ray-tracing method can be used to render the triangles.

Zundel (1998), who used ray tracing to render multiple isosurfaces, proposed one interesting idea. Zundel rendered several isosurfaces, each surface being a different color along a color ramp. Each isosurface was ray traced as a transparent, non-reflective object. The outer isosurfaces were the most transparent, while the innermost isosurface was opaque. This was done using an opacity function that increased the opacity (or decreased the transparency) of an isosurface as the distance from the eye increased.

Conclusions and Further Research

As the reader can see from this paper, several advances have come about in volume visualization since Herman and Liu's paper in 1979. Advances in the marching cubes algorithm itself include:

- Faster methods for determining the cubes intersecting the isosurface.
- Faster methods for rendering the isosurface.
- The ability to render extremely small isosurfaces the size of 1 voxel.
- Unique methods of rendering multiple isosurfaces.

With Xia et al's (1997) research in decreasing the number of triangles representing the isosurface and the fast isosurface generation techniques presented, isosurfaces can now be generated and rendered fairly swiftly.

Future research in the marching cubes algorithm could involve isosurface rendering. An isosurface-rendering algorithm where multiple isosurfaces are displayed as transparent surfaces could be implemented. The opacity function suggested by Zundel (1998) could be adjusted to produce a more useful visualization of multiple isosurfaces. Also, colors can be assigned to the isosurfaces based on specified, user-defined colors at different scalar data values. If an isosurface scalar value does not have a specific RGB color defined, an RGB color could be interpolated from the colors defined at the different scalar values.

References

- Alan Watt and Mark Watt, “Advanced Animation and Rendering Techniques”, ACM Press, New York, New York, pp. 297-321: *Volume Rendering Techniques*, 1992.
- G. T. Herman and H.K. Liu, “Three-dimensional Display of Organs from Computed Tomograms”, *Computer Graphics and Image Processing*, vol. 9, no. 1, pp. 1-21, 1979.
- Thomas W. Sederberg, “CE 571: Automatic Shape Blending”, A handout provided to Sederberg’s students in CE/ME 571, Winter 1994.
- Paolo Cignoni, Paola Marino, Claudio Montani, Enrico Puppo, and Roberto Scopingo, “Speeding Up Isosurface Extraction Using Interval Trees”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 158-169, Apr-June 1997.
- Koji Koyamada and Takayuki Itoh, “Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 4, pp. 319-327, Dec 1995.
- Julie C. Xia, Jihad El-Sana, and Amitabh Varshney, “Adaptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 171-183, Apr-June 1997.
- Chen Zhou, Renben Shu, and Mohan S. Kankanhalli, “Handling Small Features in Isosurface Generation Using Marching Cubes”, *Computers & Graphics*, vol. 18, No. 6, pp. 845-848, 1994.
- Alan K. Zundel, A personal interview by the author, 1998.