Isosurface Mesh Extraction from Scalar Fields
Marching Cubes vs Dual Contouring

Kieran Sockalingam
University of Oxford

October 2016

Introduction

In this essay I shall compare two of the most popular algorithms for extracting a polygonised mesh representing an ISOSURFACE of a given SCALAR FIELD. For a particular scalar field function $f(x, y, z)$, an isosurface with ISOVALUE k is defined as the surface containing all points at which $f(x, y, z) = k$. An isosurface extraction algorithm is a method for constructing a mesh of polygons to approximate such an isosurface. Generally speaking, the isosurface at which $f(x, y, z) = 0$ is generated, so a simple transformation can be used to generate an isosurface with a different isovalue.

In theory, a scalar field would usually be continuous and defined at every point in a given interval. For practical purposes, the data must be sampled at discrete points, and depending on the kind of data that is being used, it may only be available at discrete points, e.g. medical scan data. Different algorithms sample the data in various ways, but both the algorithms I shall discuss use the data as sampled on a structured, uniform grid. Thus, it is useful to imagine the data as an array of cubes, with the values of the scalar field tagged on the cubes at their vertices. It is common to interpolate as a trilinear function to define values for the scalar field within the cube.

Marching Cubes

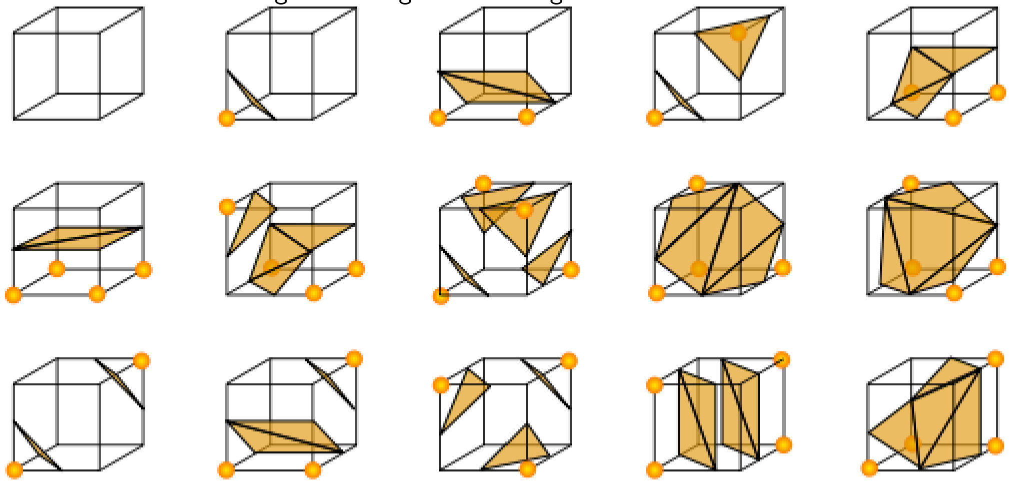
In many ways the original, and perhaps the most well-known of all the algorithms that have been developed to solve this problem is MARCHING CUBES. Introduced in 1987 by Lorensen and Cline in SIGGRAPH [LORENSEN AND CLINE 1987], it was the first of many similar algorithms, which can all be considered to build upon it. Marching Cubes takes a simplistic divide and conquer approach, breaking the problem down into a separate subproblem for each cube – neighbouring cubes do not affect each other's resulting geometry. Thus the algorithm is easily implemented in a fully data-parallel manner, which is a very useful when implementing Marching Cubes for a parallelized system such as a GPU, where parallel computation units perform the same instructions on different data in parallel very efficiently.

For each edge that exhibits a sign change on each cube, linear interpolation is used to place a vertex at the approximate location where the field takes a zero value. These vertices are joined together to form the triangles to represent the surface within the cube. Unlike the positions of the new vertices, the way in which they are connected together only depends on the signs of the scalar field at each corner of the cube, and not the exact value. This means there are only $2^8 = 256$ cases, making a case-table feasible and efficient. For each cube, the value of the scalar field at each of the eight vertices is checked to determine whether the vertex is above (outside) or below (inside) the isosurface. Using these eight binary values as an index, the polygon topology information is retrieved from a 256-value lookup table. The table describes which of the new vertices should be connected together into triangles.

In the original paper, Lorensen and Cline make use of two different symmetries to reduce the number of cases for which the triangulation must be calculated by hand. They note that the topology is the same when the vertices which are above the surface are exchanged with those that are below the surface, thus reducing the possible cases to 128. The number of cases is further reduced to just 15 by considering rotational symmetries by hand.

The visualisation in Figure 1 depicts the 15 cases. The most trivial case is the first, where none of the vertices are underneath the surface. There is a single case when only one vertex is included; this can easily be rotated to the relevant position. There are three cases where two vertices are included; when

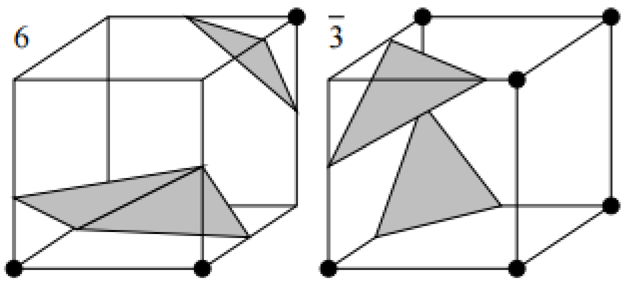
Figure 1: Original Marching Cubes case table



they share a line, when they are diagonally opposite on a face, and when they are opposite on the cube, and so on for cases with three and four vertices included. For cases where more than four vertices are included, symmetry is used.

The standard 15 case table has been criticized and improved upon by several authors, because its simplicity can cause at least two main problems. The first, and perhaps most serious, is that there is a possibility for holes to appear in the mesh due to vertices on a shared face being connected differently in two adjacent cells. An example of this is shown in Figure 2, where the shared face has the new vertices connected differently in each cube.

Figure 2: Marching Cubes hole issue, from the Marching Cubes 33 paper



The second problem is that the behaviour of the field within the cube is not considered when selecting from the case table, so there are ambiguities in some of the cases. For example, in the case where two diagonally opposite vertices are below the surface (first case on the last row in the above diagram), the two regions may or may not meet within the cube. In MARCHING CUBES 33 [CHERNYAEV 1995], the author solves this internal ambiguity by treating the behaviour of the field inside the cube as the trilinear function between the values at the vertices, and constructing a 33 case table that, with extra computations, creates a mesh which is topologically equivalent to the trilinear function.

The case table can also be extended in other ways, to solve other issues: Raman and Wenger index a table using three possible labels for each vertex, differentiating the case when the vertex has a value equal to the isovalue, resulting in $3^8 = 6561$ entries. This allows their version of the algorithm to avoid generating small or zero area triangles, short edges and small angles, resulting in a cleaner, more efficient mesh. [RAMAN AND WENGER, 2008]

Dual Contouring (of Hermite Data)

There have also been many improvements to the core mechanics of the Marching Cubes algorithm since the 1987 paper. There is a new class of algorithms which are so-called 'DUAL' methods, as opposed to the original 'PRIMAL' methods [SCHAEFER AND WARREN 2003]. Traditional primal algorithms generate one or more polygons inside each cube that intersects the surface, with the vertices lying on the edges of the cube. A dual algorithm is one which places a vertex within each cube, not necessarily on an edge, and makes a polygon connecting the four new vertices in cubes which share a given edge when that edge exhibits a sign change, thus generating a mesh that is dual to the mesh generated by a primal method in the sense that vertices and polygons are interchanged.

Dual methods have certain advantages. Firstly, the mesh polygons are more evenly sized, because the vertices roughly correspond to a regular grid, whereas Marching Cubes can produce both large and small polygons. Secondly, the resulting mesh better represents the intended isosurface because there is more freedom in the placement of vertices, which can be exploited by carefully picking the best location inside the cube. Primal methods also tend to create visible grid-like structures in the resulting mesh due to the constraint that vertices must be on the edges of the grid.

One of the earliest of the dual methods is SurfaceNets [Gibson 1998], which places the vertex within each cube at the centroid of the intersection points on the edges that would have been generated by Marching Cubes.

Dual Contouring, the algorithm introduced in Dual Contouring of Hermite Data [Ju et al 2002], makes changes to the way the scalar field data is represented. Instead of a flat grid, the method uses an octree structure (a tree structure where each node represents a cubic space and has eight children to represent the eight smaller cubes that fit into it with half the side length). The algorithm requires that the scalar field it operates upon is "hermite data", which means that the normal at a given point is also available. This can be provided analytically, or by simply calculating a numerical approximation to the normal by evaluating the scalar field at multiple points with a small offset. The normal data is used to preserve sharp features such as edges and corners in the resulting mesh. This is done using a technique whereby the vertex is placed in the location that minimizes a Quadratic Error Function (QEF). The QEF is constructed based upon not only the position of the intersection points, but also the normal of the scalar field at those points. This extra information means that features within the cube are preserved much better than by Marching Cubes, resulting in the distinctive sharp edges which are smoothed-out by Marching Cubes.

Each intersection point and its respective normal define a tangent plane to the isosurface. The QEF is the sum of the distances to each of the planes squared, represented equivalently as a function of the intersection points and the normal as:

$$E[x] = \sum_i (n_i \cdot (x - p_i))^2$$

where $E[x]$ is the QEF at a point x, and (p_i, n_i) are the (intersection point, normal) pairs.

Figure 3: Dual Contouring example, from the original dual contouring paper

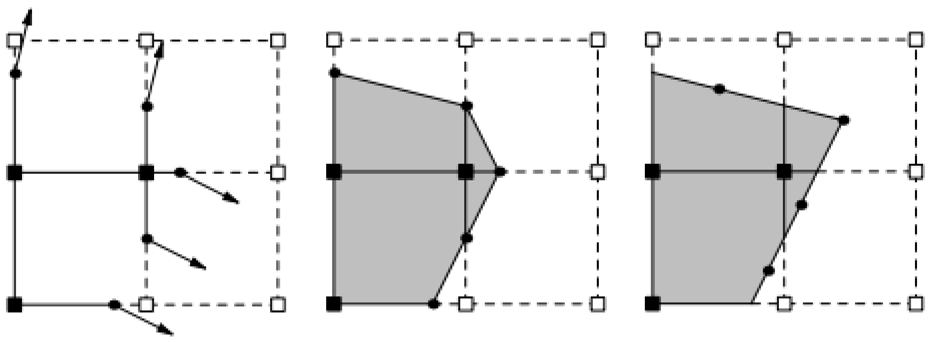


Figure 3 is 2D example, showing the scalar field data (left) with the Marching Cubes contour (middle) and Dual Contouring contour (right). In the upper-right cell, Dual Contouring captures a sharp corner feature which Marching Cubes fails to represent accurately, due to the placement of the vertex so as to be as close as possible to the position where the extension of the incoming tangent lines would meet.

Since its publication, Dual Contouring has also been extended, modified, and improved, with versions of the algorithm offering various advantages, such as Manifold Dual Contouring [Schaefer et al 2007], which better preserves the manifold nature of the surface, or Dual Marching Cubes [Schaefer and Warren 2004], which produces meshes of comparable quality with far fewer polygons by representing the scalar field data on a more flexible grid that is dual to the structured grids used by Marching Cubes and Dual Contouring.

Conclusion

Dual Contouring is a far more complex algorithm than the original Marching Cubes, and it reproduces the desired isosurface much more accurately. This makes it ideal for use in applications such as visualising medical scan data, where the accuracy of the visualisation is critical, and the processing time is less of a concern. However, the older Marching Cubes algorithm still finds use today, due to its simplicity, ease of implementation on parallel graphics hardware, and its fast execution time using lookup tables. For real-time applications, even recreating the mesh from scratch every frame is much more feasible with Marching Cubes. In applications such as computer games, where the isosurface mesh can be used to model a variety of different objects procedurally, or metasurfaces like liquids, the accuracy of the mesh is not critical, whereas speed of execution is paramount.

Bibliography

Chernyaev, E.V., 1995. Marching cubes 33: Construction of topologically correct isosurfaces. Institute for High Energy Physics, Moscow, Russia, Report CN/95-17, 42.

Gibson, S.F., 1998, October. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 888-898). Springer Berlin Heidelberg.

Ju, T., Losasso, F., Schaefer, S. and Warren, J., 2002, July. Dual contouring of hermite data. In ACM Transactions on Graphics (TOG) (Vol. 21, No. 3, pp. 339-346). ACM.

Lorensen, W.E. and Cline, H.E., 1987, August. Marching cubes: A high resolution 3D surface construction algorithm. In ACM siggraph computer graphics (Vol. 21, No. 4, pp. 163-169). ACM.

Schaefer, S. and Warren, J., 2003. Dual contouring:" the secret sauce", rice University. Department of Computer Science Technical Report.

Schaefer, S. and Warren, J., 2004, October. Dual marching cubes: Primal contouring of dual grids. In Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on (pp. 70-76). IEEE.

Schaefer, S., Ju, T. and Warren, J., 2007. Manifold dual contouring. IEEE Transactions on Visualization and Computer Graphics, 13(3), pp.610-619.

Raman, S. and Wenger, R., 2008, May. Quality isosurface mesh generation using an extended marching cubes lookup table. In Computer Graphics Forum (Vol. 27, No. 3, pp. 791-798). Blackwell Publishing Ltd.