Marching Cubes in Four and Higher Dimensions: Extended Abstract

Praveen Bhaniramka Roger Crawfis Ho-Seok Kang
DongLin Liang Rephael Wenger Zhi Yao

The Ohio State University
Columbus, Ohio

Abstract

The marching cubes algorithm is a popular visualization algorithm for constructing a 2D isosurface from a regular 3D grid. We generalize the marching cubes algorithm to four dimensions and higher. For each hypercube in the grid, we identify the edges which intersect the isosurface. We construct the isosurface within the hypercube by taking the convex hull of the midpoints of these edges and choosing a subset of the boundary of that convex hull.

1 Introduction

Given a scalar field, i.e., a scalar function on R^d, an isosurface is a set of points with identical scalar values. The marching cubes algorithm by Lorensen and Cline is a popular, simple, and efficient algorithm for constructing isosurfaces from scalar values in a three dimensional regular grid[1]. The regular grid divides the volume into cubes whose vertices are the grid vertices and the isosurface is constructed piecewise within each cube. Each grid vertex is labelled positive, "+", or negative, "-", depending upon whether its value is greater or less than the value of the isosurface. The structure of the isosurface in the cube depends only on the positive and negative labels on its eight vertices. Thus there are 2^8 ways in which the isosurface can intersect a cube. The marching cubes algorithm first builds a table of these 2^8 cases and then uses this table to determine the structure of the intersection of the surface and each cube. The actual location of the surface within the cube depends upon interpolation of the values of the cube vertices.

By exploiting symmetry, Lorensen and Cline reduced the 2^8 cases to fourteen. They analyzed these fourteen cases by hand, constructing a triangulated surface in each case. Nielson and Hamann added two more cases to resolve certain ambiguities and inconsistencies in Lorensen and Cline's original algorithm[2].

A hypercube in four dimensions has sixteen vertices and 2^{16} possible vertex labellings. Even after exploiting symmetry, we found that we were left with 222 cases. Analyzing all these cases by hand, would have been a tedious and error prone exercise. Higher dimensions are even worse. Instead, we looked for and found a systematic way of generating the surface and its triangulation for each case.

Weigle and Banks generalized a variation of the marching cubes algorithm by replacing the cubes with simplices[3, 4]. Using the barycentric subdivision, they broke each cube into simplices and then constructed the isosurface in each simplex. They triangulated the isosurface by recursively triangulating the various dimensional faces of the polyhedra composing the isosurface. Because of the simple structure and symmetry of a d-simplex, there are only $d+2$ cases, each case corresponding to a different number of vertices with positive orientation. However, a d-cube breaks into between $d!$ and $2^{d-1}d!$ simplices, depending upon the decomposition used[3]. The time and space used by the algorithm increase by a corresponding factor.

2 Marching Cubes in R^d

We present a direct generalization of the marching cubes algorithm to higher dimensions. More specifically, we give an algorithm to automatically generate a table of the isosurface and its triangulation for all the possible 2^{2^d} labellings of the hypercube.

The 1-skeleton of the hypercube is the graph of vertices and edges of the hypercube. We use a subgraph G of the 1-skeleton of the hypercube to determine the topology of the isosurface. The vertices of G are the vertices of the hypercube with positive labels. The edges of G are edges of the 1-skeleton whose endpoints are positive. Each connected component of G

determines a separate piece of the isosurface in the hypercube.

For each component G' of G, identify all the 1-skeleton edges which connect G' to $\overline{G'}$, the complement of G' in the 1-skeleton. Choose the midpoints of all these edges and perturb them slightly along their respective edges so that any $d+1$ which do not lie on a common facet of the hypercube are not on a common hyperplane. Take the convex hull of these perturbed midpoints. Because of the perturbation, the convex hull will be a full dimensional polytope in R^d.

The intersection of the boundary of the convex hull and the boundary of the hypercube partitions the boundary of the convex hull into connected regions. Each such connected region separates some portion of the hypercube boundary from the interior of the convex hull. We choose the connected region which separates the vertices in G' from the interior of the convex hull as part of our isosurface.

The isosurfaces for each connected component of G are generated separately. However, we claim that they do not intersect. We demonstrate this by giving an alternate construction of our isosurface. Instead of constructing the isosurface for each component of G separately, form the set of midpoints of all edges which connect G and $\overline{G}$, i.e., the edges which have one positive and one negative endpoint. Perturb the midpoints along their edges and construct the convex hull of all the midpoints.

The intersection of the boundary of the convex hull and the boundary of the hypercube partitions the boundary of the convex hull into connected regions. Again each such connected region separates some portion of the hypercube boundary from the interior of the convex hull. We choose the connected regions which separate the vertices with positive label from the interior of the convex hull as our isosurface in the hypercube. These isosurface "patches" exactly correspond to the patches of isosurface constructed individually in the previous method. Since these patches line on the boundary of a single convex polyhedron, they clearly do not intersect.

The key to the correctness of these algorithms is the claim that some region separates the vertices in G' and no other hypercube vertices from the interior of the convex hull. Alternatively, in the second construction each region separates either positive or negative vertices from the interior of the convex hull, but not both. A proof will appear in the full paper.

Instead of using the vertices with positive labels, we could have used the vertices with negative labels in either construction. Doing so gives a different although equally valid isosurface. However, using positive labels for some cases and negative labels for others can result in mismatches on the boundaries of the hypercubes. This was essentially the problem discovered by Nielson and Hamann in the original marching cubes algorithm[2].

3 Implementation

The marching cubes algorithm is usually implemented by constructing a complete table of all 256 labellings of the cube. A similar table for four dimensions would contain $2^{16} = 65,536$ entries. This is large but not prohibitively so. Alternatively, symmetry could be used to reduce the table size, at the expense of increasing the table lookup time. This would be necessary in five dimensions since such a table would contain 2^{32} entries. Finally, some or all entries in the table could be generated as needed.

Our algorithm constructs the isosurface for each connected component in the subgraph G separately. Thus two different labellings which have a common component in their subgraphs would have the same portion of isosurface corresponding to those components. This commonality could be exploited to aid in the table construction and lookup.

The 3D marching cubes algorithm uses interpolants of the positive and negative vertices to determine the isosurface, not simply midoints of the two. Using interpolants in four dimensions could potentially cause the surface to fold back on itself. We are investigating whether and when this is a problem. Generating the surface for each cube directly, instead of using a table, would avoid this problem, at the cost of increased time.

References

[1] Lorensen, W., and Cline, H. Marching cubes: a high resolution 3d surface construction algor ithm. *Comput. Graph. 21*, 4 (1987), 163–170.

[2] Nielson, G., and Hamann, B. The asymptotic decider - resolving the ambiguity in marching cubes. In *Proceedings of Visualization '91* (1991), IEEE Computer Society Press.

[3] Weigle, C., and Banks, D. Complex-valued contour meshing. In *Proceedings of Visualization '96* (1996), IEEE Computer Society Press.

[4] Weigle, C., and Banks, D. Extracting iso-valued features in 4-dimensional scalar fields. In *Proceedings of the 1998 Symposium on Volume Visualization* (1998), pp. 103–110.