Volume data visualization using fractal interpolation surfaces

Polychronis Manousopoulos[1], Vassileios Drakopoulos[1] and Theoharis Theoharis[1]

[1]Department of Informatics and Telecommunications, University of Athens, Greece

Abstract

Visualization of medical or experimental data is often achieved by extracting an intermediate geometric representation of the data. One such popular method for extracting an isosurface from volume data is the Marching Cubes (MC) algorithm, which creates a polygon mesh by sampling the data at the vertices of the cubes of a 3D grid. A method that uses the vertex extraction phase of the MC algorithm and represents the data by fractal interpolation surfaces (FISs) instead of a polygon mesh is presented. The proposed method is appropriate for isosurfaces that are not locally flat, such as natural structures. Another advantage is that a coarser grid resolution can typically be used, since FISs are particularly good at representing detailed, irregular or self-similar structures. Thus for many cases the resulting isosurface is more accurate or more compact. The multiresolution extension of the method is also straightforward. Experimental data verify the practical usefulness of the proposed method.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Curve, surface, solid, and object representations; I.3.7 [Computer Graphics]: Fractals

1. Introduction

The Marching Cubes (MC) algorithm ([LC87]) is one of the most popular algorithms for volume data visualization. It extracts from the data an intermediate geometric representation which is subsequently rendered. The intermediate representation is a triangular mesh, extracted by intersecting the cubes of a 3D grid with the isosurface implied by the volume data. The MC algorithm has been widely and effectively used in a diversity of applications, most notably biomedical ones. A variety of enhancements or extensions have been proposed; a comprehensive overview can be found in [BW01]. For example, it has been shown that in the original MC formulation, ambiguous cube configurations may result into "holes" in the approximated surface. Various solutions have been proposed to this issue, e.g. in [NH91] a consistent triangulation strategy is suggested, while in [Che95] and [LB03] an extended set of cube triangulation cases are defined in order to resolve the ambiguities. Alternatives to triangles as surface primitives have also been suggested. For example, bicubic patches are used in [GN89], while triangular cubic Bezier patches are employed in [The02]; in [Nie04] quad patches are used to model a surface dual to the isosurface. A multiresolution approach that reduces the number of resulting triangles is suggested in [SZK95].

In this paper, we introduce an extension of the MC algorithm that uses fractal interpolation surfaces, instead of triangles, in order to model the isosurface locally. Our motivation is to create an algorithm that produces better results for coarser grid resolutions, is suitable for irregular or self-similar volume data and, moreover, has the ability to generate a variable number of output primitives for a fixed grid resolution. The multiresolution extension of the proposed algorithm is also straightforward.

2. Background

2.1. Marching Cubes

The MC algorithm assumes that the volume data are available as function values $f_{i,j,k}$ at the vertices (x_i, y_j, z_k), $i = 1,\dots,I$, $j = 1,\dots,J$ and $k = 1,\dots,K$ of a rectangular 3D grid. The vertices of each "cube" of the grid are labelled as inside(including on) or outside the desired surface according to their function value, typically by setting a threshold that defines the respective isosurface. For each cube that intersects the surface, the approximate intersection points are determined by linear interpolation on the edges with opposite labelled vertices. Subsequently, a set of triangles connecting the intersection points is extracted, such that the topology of the cube and the surface intersection is properly described. Typically, $2^8 = 256$ cases exist, but symmetry reduces them

© The Eurographics Association 2008.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

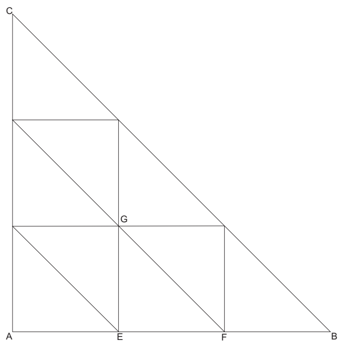

Figure 1: *Domains for fractal interpolation surfaces over triangular lattices.*

to 15 distinct ones, which can be conveniently implemented in a tabular form.

2.2. Fractal Interpolation Surfaces on Triangular Domains

Fractal interpolation surfaces (see [GH93] for a detailed discussion) are based on the theory of *iterated function systems (IFSs)*. An IFS, denoted by $\{X; w_n, n = 1,2,\ldots,N\}$, consists of a complete metric space (X,ρ), e.g. $(\mathbb{R}^n, ||\cdot||)$, and a finite set of continuous mappings $w_n\colon X \to X$, $n = 1,2,\ldots,N$. If w_n are contractions, then the IFS is termed *hyperbolic* and the transformation $W\colon \mathscr{H}(X) \to \mathscr{H}(X)$ with $W(B) = \cup_{n=1}^{N} w_n(B)$, where $\mathscr{H}(X)$ denotes the space of nonempty compact subsets of X, has a unique fixed point $A_\infty = W(A_\infty) = \lim_{n\to\infty} W^n(B)$, for every $B \in \mathscr{H}(X)$, which is called the *attractor* of the IFS.

Let us represent the given set of *interpolation points* as $\{(x_i, y_i, z_i = z(x_i, y_i)) \in \mathbb{R}^3 : i = 0,1,\ldots,M\}$. These points define a triangular lattice on the *xy*-plane. For example, the triangular domain *ABC* in Fig. 1 is partitioned into subtriangles, such as EFG. Let N be the number of triangles of such a triangulation; e.g. in Fig. 1, N=9. Moreover, we assume that the interpolation points corresponding to the vertices of the subtriangles that lie on the domain triangle boundary are coplanar, in order for the resulting surface to be continuous. Let $\{\mathbb{R}^3; w_n, n = 1,2,\ldots,N\}$ be an IFS with affine transformations

$$w_n \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} k_n & l_n & 0 \\ p_n & q_n & 0 \\ a_n & b_n & s_n \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} m_n \\ r_n \\ c_n \end{bmatrix},$$

such that each transformation maps the triangular domain to a single subtriangle, e.g. in Fig. 1 *ABC* is mapped to EFG. This is achieved by having each transformation map the vertices of one triangle to the respective ones of the other, e.g. $A \to E, B \to F$ and $C \to G$. These conditions along with the coplanar boundary constraint, result in the equations

$$k_n = \frac{(x_0^n - x_1^n)(y_1 - y_2) - (x_1^n - x_2^n)(y_0 - y_1)}{(x_0 - x_1)(y_1 - y_2) - (x_1 - x_2)(y_0 - y_1)}$$

$$l_n = \frac{(x_1^n - x_2^n)(x_0 - x_1) - (x_0^n - x_1^n)(x_1 - x_2)}{(x_0 - x_1)(y_1 - y_2) - (x_1 - x_2)(y_0 - y_1)}$$

$$m_n = x_0^n - x_0 k_n - y_0 l_n$$

$$p_n = \frac{(y_0^n - y_1^n)(y_1 - y_2) - (y_1^n - y_2^n)(y_0 - y_1)}{(x_0 - x_1)(y_1 - y_2) - (x_1 - x_2)(y_0 - y_1)}$$

$$q_n = \frac{(y_1^n - y_2^n)(x_0 - x_1) - (y_0^n - y_1^n)(x_1 - x_2)}{(x_0 - x_1)(y_1 - y_2) - (x_1 - x_2)(y_0 - y_1)}$$

$$r_n = y_0^n - x_0 p_n - y_0 q_n$$

$$a_n = \frac{((z_0^n - z_1^n) - s_n(z_0 - z_1))(y_1 - y_2) - ((z_1^n - z_2^n) - s_n(z_1 - z_2))(y_0 - y_1)}{(x_0 - x_1)(y_1 - y_2) - (x_1 - x_2)(y_0 - y_1)}$$

$$b_n = \frac{((z_1^n - z_2^n) - s_n(z_1 - z_2))(x_0 - x_1) - ((z_0^n - z_1^n) - s_n(z_0 - z_1))(x_1 - x_2)}{(x_0 - x_1)(y_1 - y_2) - (x_1 - x_2)(y_0 - y_1)}$$

$$c_n = z_0^n - a_n x_0 - b_n y_0 - s_n z_0,$$

for every $n = 1,2,\ldots,N$, where (x_0, y_0, z_0), (x_1, y_1, z_1), (x_2, y_2, z_2) are the vertices of domain triangle (e.g. *ABC*) and (x_0^n, y_0^n, z_0^n), (x_1^n, y_1^n, z_1^n), (x_2^n, y_2^n, z_2^n) the vertices of the n-th subtriangle (e.g. EFG). The s_n are *free* parameters of the transformations satisfying $|s_n| < 1$. It is known (see [GH93]) that the attractor $G = \bigcup_{n=1}^{N} w_n(G)$ of the aforementioned IFS is the graph of a continuous function that passes through the interpolation points (x_i, y_i, z_i), $i = 0,1,\ldots,M$. This function is called *fractal interpolation surface (FIS)* corresponding to these points.

3. Marching Cubes using fractal interpolation surfaces

We introduce an extension of the MC algorithm using FISs, called *Fractal Marching Cubes (FMC)* hereafter. The FMC algorithm uses the vertex extraction phase of MC, but subsequently uses FISs, instead of triangles, in order to model the isosurface locally. The FISs are constructed so as to capture the detail of the, not necessarily locally flat, isosurface. The FMC algorithm is outlined as follows:

For each cube in the 3D grid that intersects the isosurface :

1. Determine the MC triangles.
2. For each triangle :
    a. Calculate the triangle centroid and displace it along the triangle normal in order to achieve the isosurface value.
    b. Transform the triangle such that it lies on the *xy*-plane. Let this transformation be T. Apply T to the displaced centroid as well.
    c. Create the IFS of the FIS for the four triangle points (three vertices and displaced centroid) as shown in Fig. 2.
    d. Generate the attractor of the IFS as a set of triangles using the Deterministic Iteration Algorithm.
    e. Apply T^{-1} to the generated triangles.

The labelling of the cube vertices as inside or outside and the determination of the (potential) intersection points are

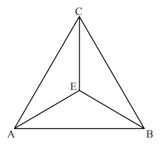

Figure 2: *The triangular lattice for constructing a fractal interpolation surface construction inside a cube.*

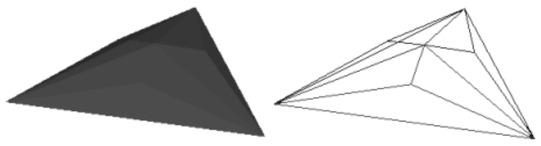

Figure 3: *An approximated fractal interpolation surface using the DIA with n=2.*

performed as in the MC algorithm. In Step 1. the set of triangles in the cube is extracted using the MC configurations. Note that an approach such as [NH91] should be adopted in order to avoid "holes" in the resulting surface. In Step 2.a., the triangle centroid is displaced along the triangle normal in order to achieve the isosurface value, calculated by trilinear interpolation. In this way, we use information from the cube interior and not only from its edges as in the MC case. Note that the centroid selection is the simplest possible; we can select any number of points in the triangle interior and properly displace them in order to construct the FIS. In Step 2.b., we transform the triangle so that it lies on the *xy*-plane; the transformation should be invertible, consisting only of rotations and translations. This is done in order for the FIS to be based on the plane defined by the triangle and not on the default *xy*-plane. The latter would result into aliasing effects, since all FISs would have the same vertical orientation. In Step 2.c., the affine transformations of the IFS for the FIS interpolating the triangle vertices and centroid are calculated. The free parameters s_n are set to a fixed value in $(0,1)$ and can be chosen according to the desired roughness of the isosurface, as well as to constrain it inside the cube. In Step 2.d., the FIS is generated as a set of triangles so that it can be efficiently rendered for visualization. The Deterministic Iteration Algorithm (DIA) for constructing an IFS attractor is employed. We start with the initial triangle, which is transformed by all affine transformations, resulting into $N(= 3$ in our case$)$ subtriangles. These are again transformed by all affine transformations, resulting into N subtriangles each, and so on. At the n-th step of the algorithm N^n triangles are produced; an example is given in Fig. 3. By setting $n = 0$, the FMC reduces to the original MC. The value of n can be adjusted by the user in order to control the detail of the resulting surface; it can also be variable across the cubes in order to avoid an excessive number of triangles. The steps 2.c. and 2.d. are similar to a subdivision procedure. However, the formulation using FISs provides an automated application of any subdivision pattern and can be extended by using starting shapes other than the triangle for the DIA. In Step 2.e., the inverse transformation to that of Step 2.b. is applied to the generated triangles, in order to restore them inside the cube. The multiresolution extension of the FMC algorithm is straightforward. Indeed, one may chose different numbers of points in the interior of each triangle for calculating the FISs and, moreover, different numbers of iterations in the DIA for generating them. These may be chosen according to local surface properties, in order to allow variable levels of detail.

4. Results

We have used volume data from the CT of a cadaver head. The data set is part of the "University of North Carolina Volume Rendering Test Data Set", courtesy of North Carolina Memorial Hospital, and is available at http://graphics.stanford.edu/data/voldata/. The data consist of 113 slices, each of resolution 256×256, defining a rectangular voxel grid with X:Y:Z aspect ratio equal to 1:1:2 for each voxel. In Figures 4–6, the results of CT visualization using the MC and FMC algorithms are presented. The data have been visualized at grid resolutions of $64 \times 64 \times 26$, $128 \times 128 \times 52$ and $256 \times 256 \times 113$. For the FMC algorithm, the number of DIA iterations was set to one. As can been seen in the figures, both algorithms produce better results for higher grid resolutions. However, this is more evident for the MC algorithm, where the differences between the various resolutions are greater, while for the FMC algorithm the decrease in the grid resolution is less crucial. For a fixed grid resolution, the FMC algorithm produces visually better results than the MC algorithm, especially at coarser grid resolutions. But even at the highest grid resolution, the FMC algorithm captures more accurately the local surface detail, as can be seen in the figures. Therefore, we conclude that the use of FISs as adopted in the FMC algorithm, can enhance the results by modelling more accurately the topology of the cube and the surface intersection.

5. Conclusions and Further Work

We have presented a novel extension to the popular MC algorithm. The FMC algorithm employs the vertex extraction phase of MC, but subsequently uses FISs in order to model the topology of the cube and the surface intersection. Results indicate that the FMC algorithm can achieve better results than the MC algorithm at the same grid resolution. Therefore, it can be especially useful when lower resolution volume data are available or prefered for efficiency, but is also capable of describing more accurately the local surface detail for higher resolutions. Moreover, the FMC algorithm is appropriate for surfaces that exhibit self-similarity or irregularity, properties which are inherently modeled by the FISs.

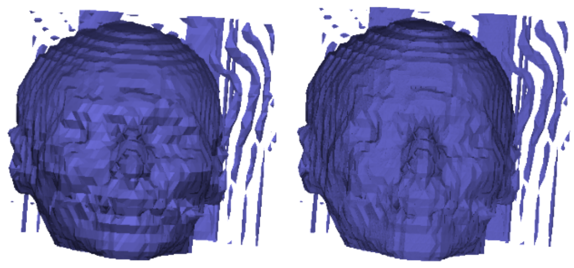

Figure 4: *(left) CT visualization of tissue using the MC algorithm and grid resolution* $64 \times 64 \times 26$. *(right) CT visualization of tissue using the FMC algorithm and grid resolution* $64 \times 64 \times 26$.

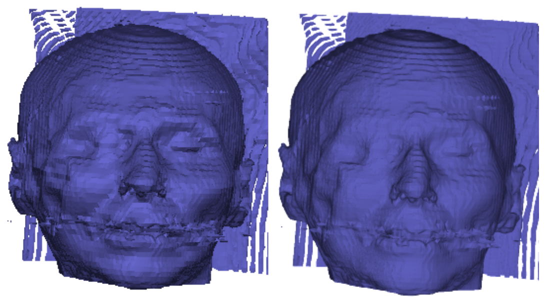

Figure 5: *(left) CT visualization of tissue using the MC algorithm and grid resolution* $128 \times 128 \times 52$. *(right) CT visualization of tissue using the FMC algorithm and grid resolution* $128 \times 128 \times 52$.

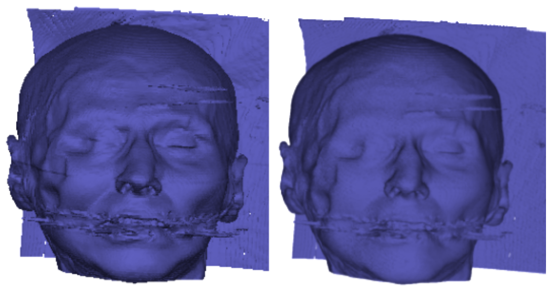

Figure 6: *(left) CT visualization of tissue using the MC algorithm and grid resolution* $256 \times 256 \times 113$. *(right) CT visualization of tissue using the FMC algorithm and grid resolution* $256 \times 256 \times 113$.

The number of output primitives generated by the FMC algorithm is variable for a fixed grid resolution, thus providing the ability to adjust the balance between performance and accuracy. Its multiresolution extension is also straightforward, by adjusting the interpolation points of each constructed FIS and the number of steps in the DIA. Further work will focus on the multiresolution extension of FMC, such that the optimal (i.e. following the surface detail) internal triangle points (Step 2.a. of FMC) and iterations of DIA (Step 2.d. of FMC) are automatically determined, and the use of recurrent FISs which are expected to produce even better results.

Acknowledgements

This research project is co-financed by E.U.-European Social Fund (75%) and the Greek Ministry of Development-GSRT (25%), under the PENED programme 70/3/8405.

References

[BW01] Brodlie K., Wood J.: Recent advances in volume visualization. *Computer Graphics Forum 20*, 2 (June 2001), 125–148.

[Che95] Chernyaev E.: Marching cubes 33: Construction of topologically correct isosurfaces. Technical Report CERN CN 95-17, http://wwwinfo.cern.ch/asdoc/psdir/mc.ps.gz, 1995.

[GH93] Geronimo J., Hardin D.: Fractal interpolation surfaces and a related 2-d multiresolution analysis. *Journal of Mathematical Analysis and Applications 176*, 2 (July 1993), 561–586.

[GN89] Gallagher R., Nagtegaal J.: An efficient 3-d visualization technique for finite element models and other coarse volumes. *Computer Graphics 23*, 3 (July 1989), 185–194.

[LB03] Lopes A., Brodlie K.: Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (Jan. 2003), 16–29.

[LC87] Lorensen W., Cline H.: Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics 21*, 4 (July 1987), 163–169.

[NH91] Nielson G., Hamann B.: The asymptotic decider: Resolving the ambiguity in marching cubes. In *Visualization '91* (1991), pp. 83–91.

[Nie04] Nielson G.: Dual marching cubes. In *Visualization '04* (2004), pp. 489–496.

[SZK95] Shu R., Zhou C., Kankanhalli M.: Adaptive marching cubes. *The Visual Computer 11*, 4 (Apr. 1995), 202–217.

[The02] Theisel H.: Exact isosurfaces for marching cubes. *Computer Graphics Forum 21*, 1 (Mar. 2002), 19–31.