ADAPTATIVE CUBICAL GRID FOR ISOSURFACE EXTRACTION

John Congote[1], Aitor Moreno[2], Iñigo Barandiaran[2], Javier Barandiaran[2] and Oscar E. Ruiz[1]
[1]*CAD/CAM/CAE Laboratory, EAFIT University, Medellín, Colombia*

[2]*VICOMTech, San Sebastian, Spain*

Keywords: Adaptive isosurface extraction, Adaptive tessellation, Isosurfaces, Volume warping, Marching cubes.

Abstract: This work proposes a variation on the Marching Cubes algorithm, where the goal is to represent implicit functions with higher resolution and better graphical quality using the same grid size. The proposed algorithm displaces the vertices of the cubes iteratively until the stop condition is achieved. After each iteration, the difference between the implicit and the explicit representations are reduced, and when the algorithm finishes, the implicit surface representation using the modified cubical grid is more detailed, as the results shall confirm. The proposed algorithm corrects some topological problems that may appear in the discretisation process using the original grid.

1 INTRODUCTION

Surface representation from scalar functions is an active research topic in different fields of computer graphics such as medical visualisation of Magnetic Resonance Imaging (MRI) and Computer Tomography (CT) (Krek, 2005). This representation is also widely used as an intermediate step for several graphical processes (Oscar E. Ruiz, 2005), such as mesh reconstruction from point clouds or track planning. The representation of a scalar function in 3D is known as implicit representation and is generated using continuous algebraic iso-surfaces, radial basis functions (Carr et al., 2001) (Morse et al., 2005), signed distance transform (Frisken et al., 2000) or discrete voxelisations.

The implicit functions are frequently represented as a discrete cubical grid where each vertex has the value of the function. The Marching Cubes algorithm (MC) (Lorensen and Cline, 1987) takes the cubical grid to create an explicit representation of the implicit surface. The MC algorithm has been widely studied as has been demonstrated by Newman (Newman and Yi, 2006). The output of the MC algorithm is an explicit surface represented as a set of connected triangles known as a polygonal representation. The original results of the MC algorithm presented several topological problems as demonstrated by Chernyaev (Chernyaev, 1995) and have already been solved by Lewiner (Lewiner et al., 2003).

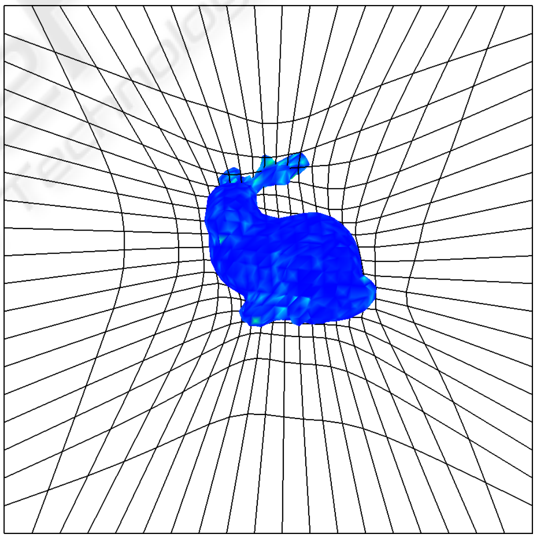


Figure 1: Optimised Grid with 20^3 cubes representing the bunny.

The MC algorithm divides the space in a regular cubical grid. For each cube, a triangular representation is calculated, which are then joined to obtain the explicit representation of the surface. This procedure is highly parallel because each cube can be processed separately without significant interdependencies. The resolution of the generated polygonal surface depends directly on the input grid size. In order to increase the resolution of the polygonal surface it is necessary to increase the number of cubes in the grid, increasing the amount of memory required to store the values of the grid.

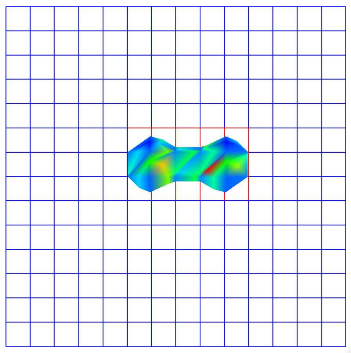
Alternative methods to the MC algorithm introduce the concept of generating multi-resolution grids, creating nested sub-grids inside the original grid. The spatial subdivision using octrees or recursive tetrahedral subdivision techniques are also used in the optimisation of iso-surface representations. The common characteristic of these types of methods is that they are based on adding more cells efficiently, to ensure a higher resolution in the final representation.

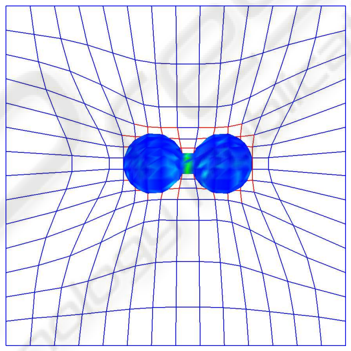
This work is structured as follows: In Section 2, a review of some of the best known MC algorithm variations is given. Section 3 describes the methodological aspects behind the proposed algorithm. In Section 4 details the results of testing the algorithm with a set of implicit functions. Finally, conclusions and future work are discussed in Section 5.

2 RELATED WORK

Marching Cubes (MC) (Lorensen and Cline, 1987) has been the *de facto* standard algorithm for the process generating of explicit representations of iso-surfaces from scalar functions or its implicit definition The MC algorithm takes as an input a regular scalar volumetric data set, having a scalar value residing at each lattice point of a rectilinear lattice in 3D space. The enclosed volume in the region of interest is subdivided into a regular grid of cubes. Each vertex of all cubes in the grid is set the value of the implicit function evaluated at the vertex coordinates. Depending on the sign of each vertex, a cube has 256 (2^8) possible combinations, but using geometrical properties, such as rotations and reflections, the final number of combinations is reduced to 15 possibilities. These 15 surface triangulations are stored in Look-Up Tables (LUT) for speed reasons. The final vertices of the triangular mesh are calculated using linear interpolation between the values assigned to the vertices of the cube. This polygonal mesh representation is ideally suited to the current generation of graphic hardware because it has been optimised to this type of input.

MC variations were developed to enhance the resolution of the generated explicit surfaces, allowing the representation of geometrical details lost during MC discretisation process. Weber (Weber et al., 2001) proposes a multi-grid method. Inside an initial grid, a nested grid is created to add more resolution in that region. This methodology is suitable to be used recursively, adding more detail to conflictive regions. In the final stage, the explicit surface is created by


(a) Original Grid. The two spheres are displayed as a singular object due to the poor resolution in the region.


(b) Intermediate Grid. Both spheres are displayed well, but are still joined.

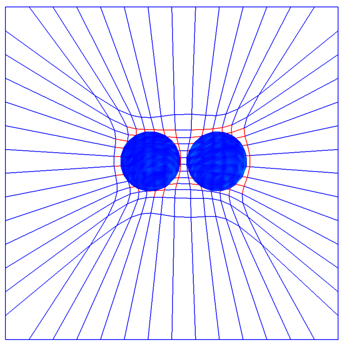

(c) Final Grid. The new resolution displays two well shaped and separated spheres with the same number of cubes in the grid.

Figure 2: 2D slides representing three different states in the evolution of the algorithm of two nearby spheres.

joining all the reconstructed polygonal surfaces.

It is necessary to generate a special polygonisation in the joints between the grid and the sub-grids to avoid the apparition of cracks or artifacts. This method has a higher memory demand to store the new values of the nested-grid.

An alternative method to refine selected region of interest is the octree subdivision (Shekhar et al., 1996). This method generates an octree in the region of existence of the function, creating a polygonisation of each octree cell. One of the flaws of this method is the generation of cracks in the regions with different resolutions. This problem is solve with the Dual Marching Cubes method (Schaefer and Warren, 2004) and implemented for algebraic functions by Pavia (Paiva et al., 2006)

The octree subdivision method produces edges with more than two vertices, which can be overcome by changing the methodology of the subdivision. Instead of using cubes, tetrahedrons were used to subdivide the grid, without creating nodes in the middle of the edges (Kimura et al., 2004). This method recursively subdivides the space into tetrahedrons.

The previous methodologies increment the number of cells of the grid in order to achieve more resolution in the regions of interest. Balmelli (Balmelli et al., 2002) presented an algorithm based on the movement of the grid to a defined region of interest using a warping function. The result is a new grid with the same number of cells, but with higher resolution in the desired region.

Our method is also based on the displacement of the vertices of the grid, obtaining dense distribution of vertices near to the iso-surface. (see Figure 2)

3 METHODOLOGY

The proposed algorithm is presented as a modification of the MC algorithm. The principal goal is the generation of more detailed approximations of the given implicit surfaces with the same grid resolution.

Applying a selective displacement to the vertices of the grid, the algorithm increases the number of cells containing the iso-surface. In order to avoid self-intersections and to preserve the topological structure of the grid, the vertices are translated in the direction of the surface. The displacement to be applied to each vertex is calculated iteratively until a stop condition is satisfied.

Let be Θ a rectangular prism tessellated as a cubical honeycomb, W the vertices of Θ [Eq. 1], B the boundary vertices of Θ [Eq. 2], and V the inner vertices of Θ [Eq. 3]. For each vertex $v_i \in V$, a N_i set is defined as the *26 adjacent* vertices to v_i, denoting each adjacent vertex as $n_{i,j}$ [Eq. 4]. (see Figure 3)

$$W = \{w_i / w_i \in \Theta\} \quad (1)$$
$$B = \{b_i / b_i \in \delta\Theta\} \quad (2)$$

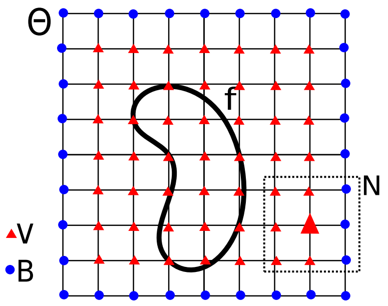


Figure 3: Grid nomenclature, Θ cubical grid, $f(x,y,z) = 0$ implicit function, N vertex neightboor, V vertices inside the grid, B vertices at the boundary of the grid.

$$V = W - B \quad (3)$$
$$N_i = \{n_{i,j} / n_{i,j} \text{ is } j\text{th neighbourgh of } v_i\} \quad (4)$$

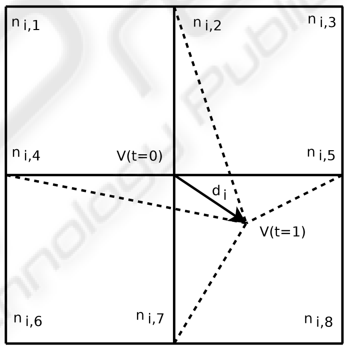


Figure 4: two consecutives iterations are show where the vertex v is moved between the iterations $t = 0$ and $t = 1$. The new configuration of the grid is shown as dotted lines.

The proposed algorithm is an iterative process. In each iteration, each vertex v_i of the grid Θ is translated by a d_i distance vector, obtaining a new configuration of Θ, where *i)* the topological connections of the grid are preserved, *ii)* the number of cells containing patches of f are greater than, or equal to, the previous value, and *iii)* the total displacement [Eq. 7] of the grid is lower and is used as the stop condition of the algorithm when it reach a value Δ(see Figure 4).

The distance vector d_i is calculated as shown in [Eq. 6] and it can be seen as the resultant force of each neighbouring vertex scaled by the value of f at the position of each vertex. In order to limit the maximum displacement of the vertices and to guarantee the topological order of Θ, the distance vector d_i is clamped in the interval expressed in [Eq. 5]

$$0 \leq |d_i| \leq \text{MIN}\left(\frac{|n_{i,j} - v_i|}{2}\right) \quad (5)$$

$$d_i = \frac{1}{26}\sum_{n_{i,j}} \frac{n_{i,j} - v_i}{1 + |f(n_{i,j}) + f(v_i)|} \tag{6}$$

$$\sum_{v_i} |d_i| \geq \Delta \tag{7}$$

The algorithm stops when the sum of the distances added to all the vertices in the previous iteration is less that a given threshold Δ [Eq. 7] (see Algorithm 1).

Algorithm 1: Vertex Displacement Pseudo-algorithm. $|x|$ represents the magnitude of x, $\overline{v}$ represents the normalised vector of v.

```
repeat
    s := 0;
    foreach Vertex $v_i$ do
        $d_i := \frac{1}{26}\sum_{n_{i,j}} \frac{n_{i,j}-v_i}{1+|f(n_{i,j})+f(v_i)|}$;
        mindist := MIN $\left(\frac{|n_{i,j}-v_i|}{2}\right)$;
        $d_i := \overline{d_i}$CLAMP $(|d_i|, 0.0, \text{mindist})$;
        $v_i := v_i + d_i$;
        s := s + $|d_i|$;
    end
until s $\geq \Delta$ ;
```

4 RESULTS

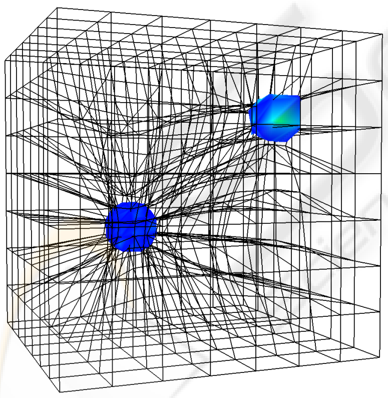


Figure 5: Two balls in different positions with a scalar function as the distance transform, representing the behaviour of the algorithm with different objects in the space.

The proposed algorithm was tested with a set of implicit functions as distance transforms (see Figure 5) and algebraic functions (see Figure 6(a)). For demonstration purposes, the number of cells has been chosen to be very low to aid in the visual detection of the improvements produced by the algorithm. For the visualisation process we use Marching Tetrahedra because

Table 1: Quality is measured as the average distance between the mesh vertices and the real surfaces. The columns 1 and 2 represent the quality of the bunny model (MC. QLTY.) using the given cubical grid size (GS_1) with the standard MC algorithm. The columns 3 and 4 shows the quality (AMC. QLTY.) using the given cubical grid size (GS_2) with the proposed algorithm after 30 iterations. The quality values (columns 2 and 3) have been aligned to be as equal as possible, showing that to achieve a target quality, the grid size using the proposed algorithm is less than the required using the standard MC algorithm.

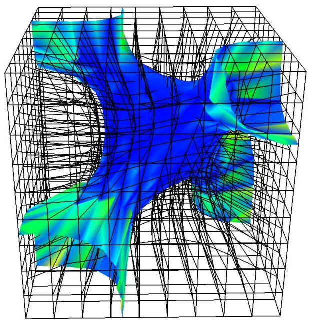
GS_1^3	MC. QLTY.	AMC. QLTY.	GS_2^3
10	0.958555	-	-
20	0.369976	0.32257	10
30	0.188298	0.186994	20
40	0.129414	0.127588	30
50	0.094878	0.092761	40

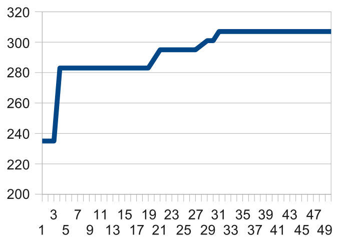
it produces correct topological representation of the iso-surface, and allows the identification of the topological correctness of the algorithm.

The obtained results of the algorithm are visually noticeable, as is shown in Figure 6. Without using the algorithm, the two spheres model is perceived as a single object (see Figure 2). In an intermediate state the spheres are still joined, but their shapes are more rounded. In the final state, when the algorithm converges, both spheres are separated correctly, each one being rendered as a near perfect sphere. Thus, using the same grid resolution and the proposed algorithm, the resolution of the results has been increased.

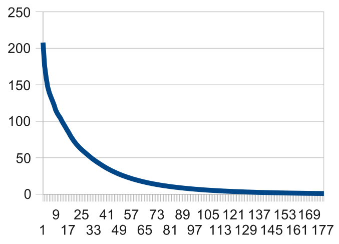
The proposed algorithm iteratively increases the number of cells containing the surface, adding more detail to the new representation. Figure 6(b) shows the incremental evolution of such a number of cubes containing the surface, tending toward a doubling of the number. The average distance triangulation vertices to the original surface was calculated as presented in table 1. The proposed algorithm can represent the surface with a good quality with a fraction of the amount of cells required with the original MC algorithm.

The total displacement of the vertices in each iteration is decreasing rapidly toward zero after a number of iterations as show in the Figure 6(c). Despite the seemingly high number of iterations, the algorithm is executed only once for static functions, and can be processed in the background. Even when the implicit function is a time variant, the cubical grid can be reused as the input cubical grid for the next algorithm execution. When the implicit function changes smoothly, the algorithm quickly re-converges after just a few iterations significantly reducing the computational effort.


(a) Algebraic function rendered with an optimised grid



(b) Cube increment (Y axis) vs. Iteration evolution (X axis)



(c) Total displacement of the grid (Y axis) vs. Iteration evolution (X axis)

Figure 6: Grid evolution for an algebraic function, comparing the number of cubes which contains the iso-surface and the total displacement of the vertices plotted against the stage of execution of the algorithm.

5 CONCLUSIONS AND FUTURE WORK

Our proposed iterative algorithm has shown significant advantages in the representation of distance transform functions. With the same grid size, it allows a better resolution by displacing the vertices of the cube grids towards the surface, increasing the number of cells containing the surface.

The algorithm was tested with algebraic functions, representing distance transform of the models. The generated scalar field has been selected to avoid the creation of regions of false interest, which are for static images in which these regions are not used.

The number of iterations is directly related to the chosen value Δ as it is the stop condition. The algorithm will continuously displace the cube vertices until the accumulated displacement in a single iteration is less than Δ. In the results, it can be seen that this accumulated distance converges quickly to the desired value. This behaviour is very convenient to represent time varying scalar functions like 3D videos, where the function itself is continuously changing. In this context, the algorithm will iterate until a good representation of the surface is obtained. If the surface varies smoothly, the cube grid will be continuously and quickly readapted by running a few iterations of the presented algorithm. As the surface changes may be assumed to be small, the number of iterations until a new final condition is achieved will be low. The obtained results will be a better real-time surface representation using a coarser cube grid.

6 ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Administration agency CDTI, under project CENIT-VISION 2007-1007. CAD/CAM/CAE Laboratory - EAFIT University and the Colombian Council for Science and Technology -Colciencias-. The bunny model is courtesy of the Stanford Computer Graphics Laboratory.

REFERENCES

Balmelli, L., Morris, C. J., Taubin, G., and Bernardini, F. (2002). Volume warping for adaptive isosurface extraction. In *Proceedings of the conference on Visualization 02*, pages 467–474. IEEE Computer Society.

Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., and Evans, T. R. (2001). Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA. ACM.

Chernyaev, E. (1995). Marching cubes 33: Construction of topologically correct isosurfaces. Technical report, Technical Report CERN CN 95-17.

Frisken, S. F., Frisken, S. F., Perry, R. N., Perry, R. N., Rockwood, A. P., Rockwood, A. P., Jones, T. R., and Jones, T. R. (2000). Adaptively sampled distance fields: A general representation of shape for computer graphics. pages 249–254.

Kimura, A., Takama, Y., Yamazoe, Y., Tanaka, S., and Tanaka, H. T. (2004). Parallel volume segmentation with tetrahedral adaptive grid. *ICPR*, 02:281–286.

Krek, P. (2005). Flow reduction marching cubes algorithm. In *Proceedings of ICCVG 2004*, pages 100–106. Springer Verlag.

Lewiner, T., Lopes, H., Vieira, A., and Tavares, G. (2003). Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15.

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):169–169.

Morse, B. S., Yoo, T. S., Rheingans, P., Chen, D. T., and Subramanian, K. R. (2005). Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 78, New York, NY, USA. ACM.

Newman, T. S. and Yi, H. (2006). A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879.

Oscar E. Ruiz, Miguel Granados, C. C. (2005). Fea-driven geometric modelling for meshless methods. In *Virtual Concept 2005*, pages 1–8.

Paiva, A., Lopes, H., Lewiner, T., and de Figueiredo, L. H. (2006). Robust adaptive meshes for implicit surfaces. *SIBGRAPI*, 0:205–212.

Schaefer, S. and Warren, J. (2004). Dual marching cubes: Primal contouring of dual grids. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 70–76, Washington, DC, USA. IEEE Computer Society.

Shekhar, R., Fayyad, E., Yagel, R., and Cornhill, J. F. (1996). Octree-based decimation of marching cubes surfaces. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 335–ff., Los Alamitos, CA, USA. IEEE Computer Society Press.

Weber, G. H., Kreylos, O., Ligocki, T. J., Shalf, J. M., Hamann, B., and Joy, K. I. (2001). Extraction of crack-free isosurfaces from adaptive mesh refinement data. In *Data Visualization 2001 (Proceedings of VisSym '01)*, pages 25–34. Springer Verlag.