Extending Marching Cubes with Adaptative Methods to obtain more accurate iso-surfaces

John Congote[1,2], Aitor Moreno[2], Iñigo Barandiaran[2], Javier Barandiaran[2], and Oscar Ruiz[1]

[1] CAD/CAM/CAE Laboratory, EAFIT University, Medellín, Colombia
[2] VICOMTech, San Sebastian, Spain

Abstract. This work proposes an extension of the Marching Cubes algorithm, where the goal is to represent implicit functions with higher accuracy 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 is reduced, and when the algorithm finishes, the implicit surface representation using the modified cubical grid is more accurate, as the results shall confirm. The proposed algorithm corrects some topological problems that may appear in the discretization 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 visualization of Magnetic Resonance Imaging (MRI) and Computer Tomography (CT) [1]. This representation is also widely used as an intermediate step for several graphical processes [2], 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 [3] [4], signed distance transform [5], discrete voxelisations or constructive solid geometry.

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) [6] 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 [7]. The output of the MC algorithm is an explicit surface represented as a set of connected triangles known as polygonal representation. The original results of the MC algorithm presented several topological problems as demonstrated by Chernyaev [8] and have already been solved by Lewiner [9].

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

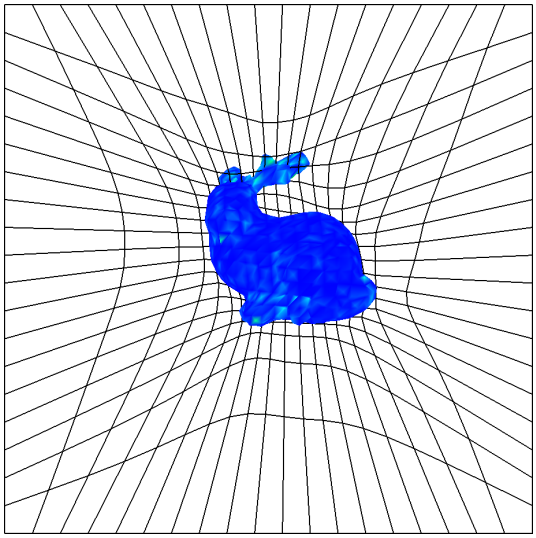


Fig. 1. Optimised Grid with 20^3 cubes representing the bunny.

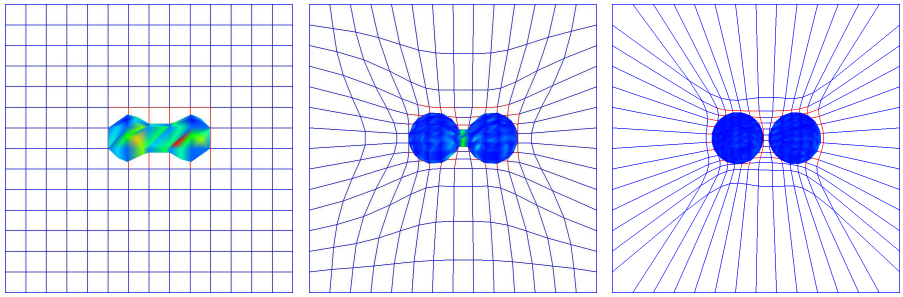
resolution of the generated polygonal surface depends directly on the input grid size. In order to increase the resolution of the polygonal surface it is necessary to increase the number of cubes in the grid, increasing the amount of memory required to store the values of the grid.

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 optimization 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) [6] 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 by the value of the implicit function evaluated at the vertex coordinates. Depending



(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

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

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 performance 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 the best suitable one for the current generation of graphic hardware because it has been optimized to this type of input.

MC variations were developed to enhance the resolution of the generated explicit surfaces, allowing the representation of geometrical lost details during MC discretization process. Weber [10] 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 joining all the reconstructed polygonal surfaces. It is necessary to generate a special polygonization 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 regions of interest(ROI) is the octree subdivision [11]. This method generates an octree in the region of existence of the function, creating a polygonization 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 [12] and implemented for algebraic functions by Pavia [13].

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 [14]. 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 [15] presented an algorithm based on the warping of the grid to a defined region of interest. The warping of the vertices is performed in a hierarchical procedure, the volume is considered as a single cell of the grid, and then, the central point of the grid is warped in the direction of the *ROI*. Then, this cell is divide then in eight cells and the process is repeated until the number of selected cells is achieved. The result is a new grid with the same number of cells, but with higher resolution near to the *ROI*. The algorithm was tested with discrete datasets, and the *ROI* is created by the user or defined by a crossing edges criteria.

The presented method generates a similar warping grid as Balmelli does, but we avoid the use of hierarchical procedures and our region of interest is automatically generated based in the input implicit function, obtaining dense distribution of vertices near the iso-surface. (see Figure 2)

3 Methodology

The presented algorithm in this work is an extension of the MC algorithm. The main goal is to generate a more accurate representations of the given implicit surfaces with the same grid resolution.

Applying a calculated displacement to the vertices of the grid, the algorithm reconfigure the position of the vertices of the grid to obtain more accurate representations of the iso-surface. In order to avoid self-intersections and to preserve the topological structure of the grid, the vertices are translated inside the cells of the neighbor of the vertex. The displacement to be applied to all the vertices are calculated iteratively until a stop condition is satisfied.

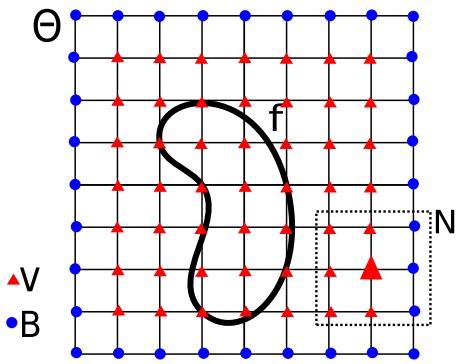
Let be Θ a rectangular prism tessellated as a cubical honeycomb, W the vertices of Θ [Eq. 1], B the boundary vertices of Θ [Eq. 2], and V the inner vertices of Θ [Eq. 3]. For each vertex $v_i \in V$, a N_i set is defined as the *26 adjacent* vertices to v_i, denoting each adjacent vertex as $n_{i,j}$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

$$W = \{w_i / w_i \in \Theta\} \tag{1}$$

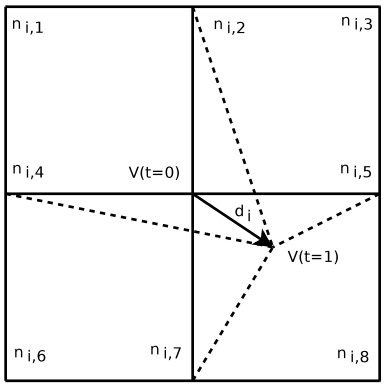
$$B = \{b_i / b_i \in \delta\Theta\} \tag{2}$$

$$V = W - B \tag{3}$$

$$N_i = \{n_{i,j} / n_{i,j} \text{ is } j\text{th neighborhood of } v_i\} \tag{4}$$



(a) Grid nomenclature, Θ cubical grid, $f(x, y, z) = 0$ implicit function, N vertex neighborhood, V vertices inside the grid, B vertices at the boundary of the grid



(b) two consecutive 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 displacement vector [Eq. 6], obtaining a new configuration of Θ, where *i)* the topological connections of the grid are preserved, *ii)* cells containing patches of f are a more accurate representation of the surface, 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 3).

The distance vector d_i is calculated as shown in [Eq. 6] and it can be seen as the resultant force of each neighboring vertex scaled by the value of f at the position of each vertex and the attraction value A. 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]

The attraction value A is empirical value which scale the value of the function in all the grid. This value control the attraction factor of the vertices of the grid to the iso-surface, values between 0 and 1 produces a grid avoids the iso-surface, values lesser than 0 generate incorrect behavior of the function. The recommended and useful values are equal or greater than 1, very high values of A could generate problems for the grid, produces big stepping factors for the displacement vectors d_i and then some characteristics of the iso-surface could be lost. The A value is highly related to the value of the distance of the bounding box of the grid, and the size of the objects inside the grid.

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

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

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

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).

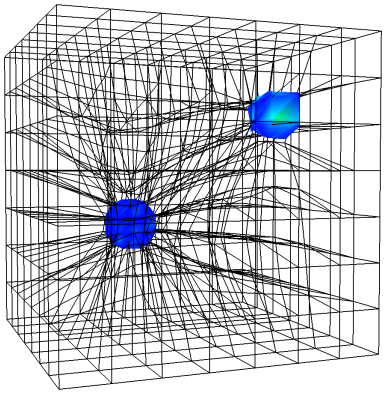
repeat
  s := 0;
  foreach *Vertex* v_i **do**
    $d_i := \frac{1}{26} \sum_{n_{i,j}} \frac{n_{i,j} - v_i}{1 + A|f(n_{i,j}) + f(v_i)|}$;
    mindist := MIN $\left(\frac{|n_{i,j} - v_i|}{2} \right)$;
    $d_i := \bar{d}_i$CLAMP $(|d_i|, 0.0, \text{mindist})$;
    $v_i := v_i + d_i$;
    s := s + $|d_i|$;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

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

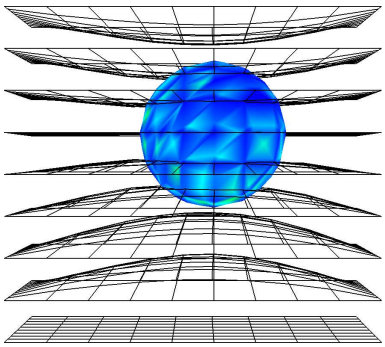
4 Results

The proposed algorithm was tested with a set of implicit functions as distance transforms (see Figure 3) of a set of spheres, the spheres are define as a point in the space with their radius. The result of the sphere data set (3(d),4(a),5(a)) and the two-sphere data set (3(c),4(b),5(b)) are presented, but the algorithm also has been tested with datasets composed of more than 1000 spheres (see Figure 1). The algorithm has been tested also with other non-distance transform implicit functions, but the generation of false *ROI* in the grid degenerates the structure of the grid resulting in bad representations of the iso-surface. For demonstration purposes, the number of cells has been chosen to enhance visual perception of the improvements produced by the algorithm. For the visualization process we use Marching Tetrahedra[16] because 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 shown in Figure 2. 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



(c) Two spheres in different positions with a scalar function as the distance transform, representing the behavior of the algorithm with different objects in the space.



(d) Sphere in the center of the space with a scalar function as the distance transform. The slides shows the different warping of the grid in the different positions

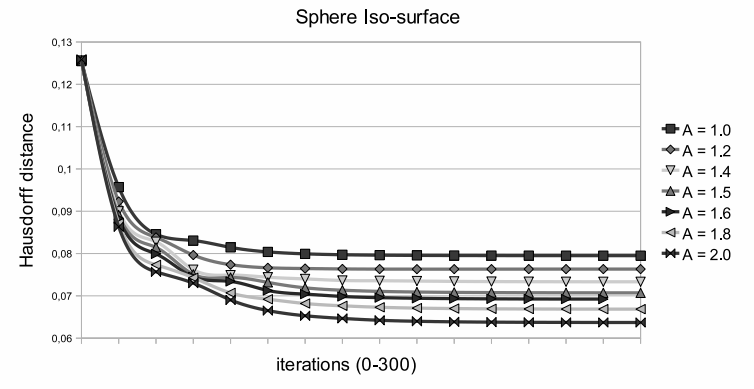
Fig. 3. Implicit function of spheres as distance transforms

algorithm, the resolution of the results has been increased and also topological errors of the original explicit representation were considerably reduce with the algorithm.

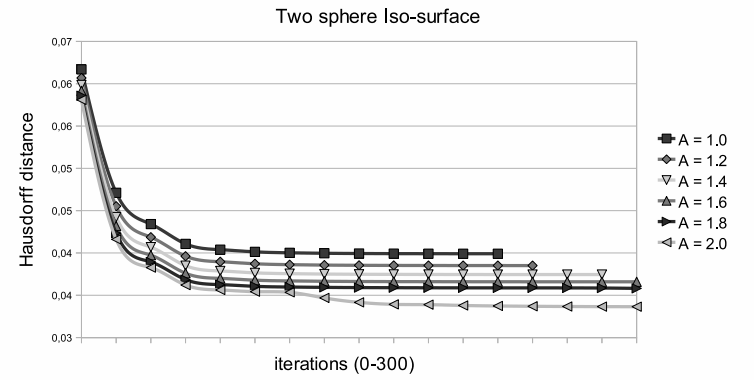
Accuracy of the explicit representations of the algorithm were measured using the methodology of De Bruin [17] which is based on the Hausdorff distance explain by Dubuisson [18]. The figures 4 and 5 shows the behavior of the algorithm where, in almost all the iterations, the Hausdorff distance between the implicit iso-surface and the explicit surface are decreasing. The iterations where there is an increment of the distance, could represent a point where the configuration of the grid is not suitable for optimization and then the algorithm needs to modify the grid looking for suitable configurations.

Figure 4 presents the results of the algorithm with the same implicit surfaces but with different attraction factor values (A). As the results show the accuracy of the iso-surface is highly related with the (A) value, because the allowed warping of the grid is bigger obtaining a dense grid near to the iso-surface, but this over-fit of the grid can be dangerous if the grid is going to be used for other implicit functions, like time varying functions, because the grid need more iterations to adapt to the new iso-surface.

Figure 5 shows the behavior of the algorithm with different grid sizes. The accuracy of the final explicit representation of the iso-surfaces shows an improve-



(a) Sphere implicit function



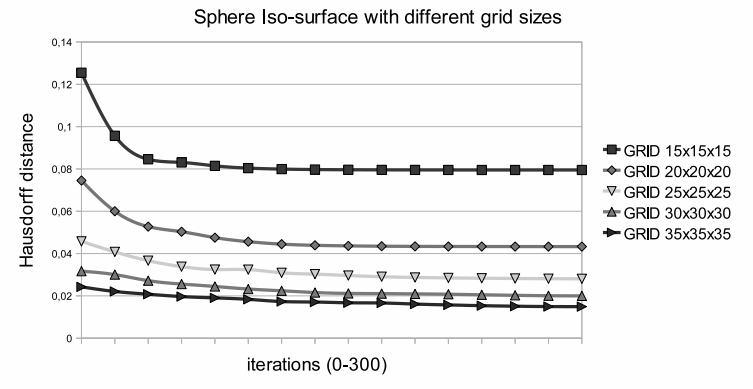
(b) Two sphere implicit function

Fig. 4. Hausdorff distance of the explicit representations of the figures in each iteration of the algorithm with different attraction values

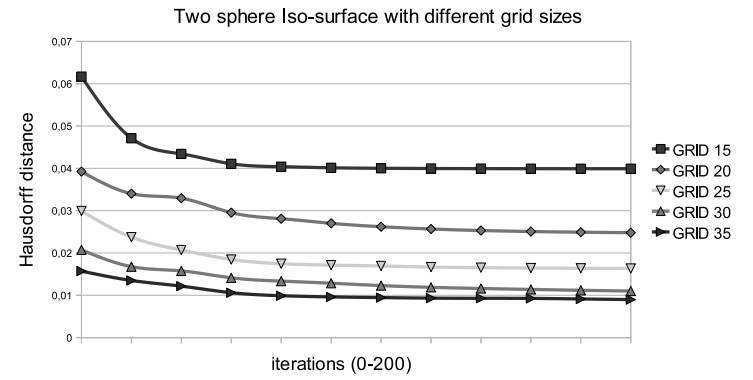
ment of the accuracy of the representation. Then it is possible with the algorithm to represent iso-surfaces with good accuracy and quality without increasing grid sizes. This characteristic allow us to use smaller grids for the representation without loss of accuracy or quality in the representation and saving computational resources.

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,



(a) Sphere implicit function



(b) Two sphere implicit function

Fig. 5. Hausdorff distance of the explicit representations of the figures in each iteration of the algorithm with different grid sizes

increasing the number of cells containing the surface. The algorithm was tested with algebraic functions, representing distance transforms of the models. The generated scalar field has been selected to avoid the creation of regions of false interest [19], 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. The accumulated distance converges quickly to the desired value. This behavior 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 re-adapted by running a few iterations of the presented algorithm. Whenever the surface changes can be considered no be small, the number of iterations until a new final condition is reached will be low. Then the obtained results will be a better real-time surface representation using a coarser cube grid.

The value Δ is sensitive to the grid size, so a better stop condition should be evaluated which represent the state of the quality of the representation and reduce the number of iteration which are unnecessary. The model used in this algorithm is close to a physics spring model, a close comparison of the proposed algorithm with the spring model could be done.

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

[1] Krek, P.: Flow reduction marching cubes algorithm. In: Proceedings of ICCVG 2004, Springer Verlag (2005) 100–106

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

[3] Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: 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, New York, NY, USA, ACM (2001) 67–76

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

[5] Frisken, S.F., Perry, R.N., Rockwood, A.P., Jones, T.R.: Adaptively sampled distance fields: a general representation of shape for computer graphics. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co. (2000) 249–254

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

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

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

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

[10] Weber, G.H., Kreylos, O., Ligocki, T.J., Shalf, J.M., Hamann, B., Joy, K.I.: Extraction of crack-free isosurfaces from adaptive mesh refinement data. In: Data Visualization 2001 (Proceedings of VisSym '01), Springer Verlag (2001) 25–34
[11] Shekhar, R., Fayyad, E., Yagel, R., Cornhill, J.F.: Octree-based decimation of marching cubes surfaces. In: VIS '96: Proceedings of the 7th conference on Visualization '96, Los Alamitos, CA, USA, IEEE Computer Society Press (1996) 335–ff.
[12] Schaefer, S., Warren, J.: Dual marching cubes: Primal contouring of dual grids. In: PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference, Washington, DC, USA, IEEE Computer Society (2004) 70–76
[13] Paiva, A., Lopes, H., Lewiner, T., de Figueiredo, L.H.: Robust adaptive meshes for implicit surfaces. SIBGRAPI **0** (2006) 205–212
[14] Kimura, A., Takama, Y., Yamazoe, Y., Tanaka, S., Tanaka, H.T.: Parallel volume segmentation with tetrahedral adaptive grid. ICPR **02** (2004) 281–286
[15] Balmelli, L., Morris, C.J., Taubin, G., Bernardini, F.: Volume warping for adaptive isosurface extraction. In: Proceedings of the conference on Visualization 02, IEEE Computer Society (2002) 467–474
[16] Carneiroz, B.P., Y, C.T.S., Kaufman, A.E.: Tetra-cubes: An algorithm to generate 3d isosurfaces based upon tetrahedra. In: IX Brazilian symposium on computer, graphics, image processing and vision (SIBGRAPI 96). (1996) 205–10
[17] Vos, D.B., Bruin, P.W.D., Vos, F.M., Post, F.H., Frisken-gibson, S.F., Vossepoel, A.M.: Improving triangle mesh quality with surfacenets. In: In MICCAI. (2000) 804–813
[18] Dubuisson, M.P., Jain, A.K.: A modified hausdorff distance for object matching. In: Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on. Volume 1. (1994) 566–568 vol.1
[19] Congote, J., Moreno, A., Barandiaran, I., Barandiaran, J., Ruiz, O.: Adaptative cubical grid for isosurface extraction. In: 4th International Conference on Computer Graphics Theory and Applications GRAPP-2009, Lisbon, Portugal (Feb 5-8 2009) 21–26