ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 1 (2012) 1747–1755

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

An adaptive graph for volumetric mesh visualization

Diogo T. Robaina, Mauricio Kischinhevsky

Universidade Federal Fluminense, Instituto de Computação

Sanderson L. Gonzaga de Oliveira

Universidade Federal de Lavras, Departamento de Ciência da Computação

Diego N. Brandão, Esteban Clua, Anselmo Montenegro

Universidade Federal Fluminense, Instituto de Computação

Abstract

This work presents an adaptive strategy in order to visualize volumetric data generated from numerical simulations of partial differential equations. The mesh is represented by a graph data structure. Moreover, the Autonomous Leaves Graph is extended to the three-dimensional case. This scheme intends to achieve better transversal cost than a tree-like (e.g., bintree, quadtree and octree) space arrangement approach. Furthermore, this strategy intends to reduce the computational cost of constructing the discretization and the visualization of data. The total-ordering of the mesh volumes used in the discretization and the visualization processes is by the 3D Modified Hilbert space-filling Curve. To evaluate the performance, the strategy is applied on a Heat Conduction simulation problem using finite difference discretizations and the experimental results are discussed. Comparisons are made between numerical results obtained when using the Hilbert Curve and its modified version. In addition, experiments are shown when visualization is made from inside and outside the volume. The results expose the efficiency of using this strategy.

Keywords: Adaptive mesh refinement, Space-filling curve, Volumetric Visualization, Autonomous Leaves Graph, Heat Conduction simulation.

1. Introduction

Visualization of data from numerical simulations is not a easy task when its quantity is large. Volumetric modeling and rendering approaches for such task is usually a common choice. In this task, the software needs to scan an enormous quantity of voxels for each frame. Efficiency in processing, storage and access the data are essential in applications involving scientific visualization. The process shoule pursue low computational cost. Moreover, few nodes of the data structure should be traversed and updated in order to correctly represent a grid representation. It is

Email addresses: `{drobaina,kisch}@ic.uff.br` (Diogo T. Robaina, Mauricio Kischinhevsky), `sanderson@dcc.ufla.br` (Sanderson L. Gonzaga de Oliveira), `{dbrandao,esteban,anselmo}@ic.uff.br` (Diego N. Brandão, Esteban Clua, Anselmo Montenegro)

URL: `www.ic.uff.br` (Diogo T. Robaina, Mauricio Kischinhevsky), `www.dcc.ufla.br` (Sanderson L. Gonzaga de Oliveira), `www.ic.uff.br` (Diego N. Brandão, Esteban Clua, Anselmo Montenegro)

worth recalling that accessing the main memory of a RAM-model (which is attributed to von Neumanns report [1]) computer system is still currently a bottleneck in terms of performance.

One trivial strategy consists on exhibiting all the data, without any interpretation of the problems' features. Although it demands a simple codification, this strategy implies high computational cost. Another strategy is to build images without details in the regions where the phenomenon is easily interpreted and more discrete places in regions of interest such as those where singularities or discontinuities occur. The latter reduces the quantity of points needed and as a consequence, it reduces the computational cost in order to solve the task. In addition, this approach leads to a robust and efficient strategy for the solution of problems that involve partial differential equations by methods that use mesh discretizations, e.g. finite difference, volume and element discretizations.

In order to implement view-dependent level of detail for a non-uniform grid structure, one must be able to represent different parts of the grid at different resolutions. Usually, this is implemented by a hierarchical representation in which one can gradually refine further detail to different parts of the grid. In relation to data structures, there is a number of options available for achieving this multiresolution representation. The most common includes the quadtree and the binary tree. Usually, a quadtree structure represents a rectangular region divided uniformly into four quadrants. Each of these quadrants can then be successively divided into four smaller regions, and so on. Quadtrees have been used for a number of terrain level of detail systems. A binary tree structure, or simply bintree, works the same way as a quadtree, but instead of segmenting a rectangle into four, it segments a discrete place into equal halves [2].

Another common data structure used in this case is the octree (see, e.g. [3]). It is a simple hierarchical space-subdivision scheme in which the bounding box of an object or scene is recursively subdivided into eight equally sized octants, which may in turn be subdivided [2]. Moreover, in this representation, each volume of the scenario is a node of a tree and for each refinement operation a cell is subdivided into eight new ones. This process has a high traversal cost because it represents cells of different levels of refinement in the same structure [4].

Related to level of details in order to numerically solve differential equations (PDEs), [5] proposed a strategy to solve bidimensional PDEs based on the construction of a locally refined mesh with low computational cost. Moreover, the Autonomous Leaves Graph (ALG) is a graph that represents an adaptive mesh refinement coupled with a numerical solver of PDEs. ALG was initially associated with the Finite Volume Method. Specifically, the authors proposed the technique with quadrangular volumes in second-order bidimensional problems. In order to number the mesh nodes, a space-filling curve, called Modified Hilbert Curve (MHC), was proposed. Recently, [6] applied this scheme in a aerodynamic problem. Besides, [7] coupled ALG with square-shaped finite elements. Also, [8] coupled a similar graph data structure based on finite-volume triangular meshes applied to second-order evolutionary elliptical and parabolical PDEs.

In this ongoing work, ALG is tailored to the 3D cubic case. Moreover, this paper applies ALG for visualization of data from numerical simulations using a procedure that adaptively refines and unrefines volumes to be drawn, using the 3D- MHC to order the set of volumes. The outline of this paper is as follows. Section 2 presents relevant aspects about volumetric visualization for this work. Section 3 presents the ALG and its refinement and unrefinement processes. Space-filling curves and the 3D-MHC are discussed in Section 4. Numerical results are presented in Section 5. In Section 6 some concluding remarks which address effectiveness and performance of the proposed method are discussed.

2. Volumetric visualization

In many problems, the acquisition of data associated with volume regions is necessary. These data can be obtained directly from measurements of quantities in nature or via simulation, in which data are generated from the solution of the mathematical model of the problem.

There are two main strategies for the visualization of volumetric data: direct volume rendering [9], where Fig. 1a is an example, and isosurface extraction [10], where Fig. 1b is an example. In the first form, data is displayed directly and in the other one, polygonal surfaces for characteristics of interest are built and displayed using conventional rendering techniques.

The Direct Volume Rendering allows the visualization of real volumetric data providing more information than techniques based on surface rendering. However, color mapping and the use of transparency can introduce difficulties to the visualization and interpretation of the results of a numerical simulation.

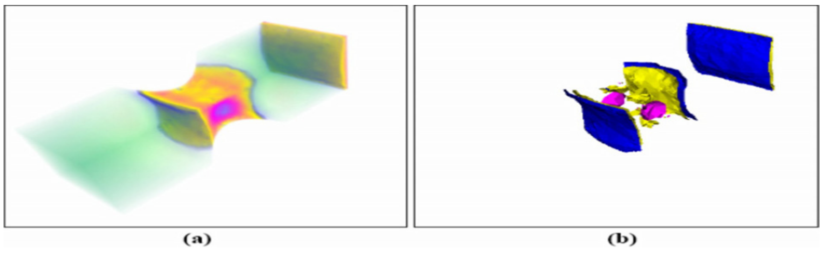

Figure 1: Techniques for volumetric visalization [11]: (a) direct volumetric rendering and (b) isosurface extraction.

This drawback can be observed in the scalar field of Fig. 2. On the other hand, surface extraction allows the identification of structures in the simulation or measurements. However, in this process, information about the content of the data set is lost. Besides, some data cannot be easily represented by surfaces, where clouds of points are examples. The adaptive strategy presented in this work allows the visualization of regions of interest as well as the interior of the volumes with large sharpness. To summarize, this paper provides a method for the analysis of results obtained from discretized mathematical models.

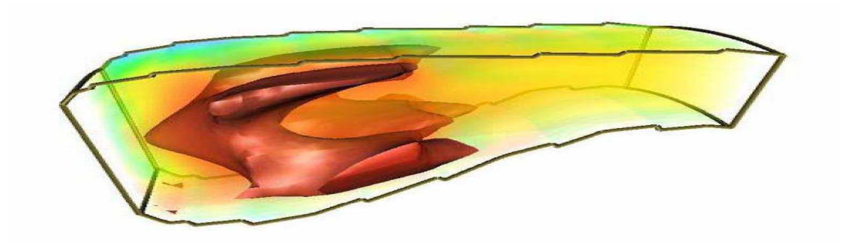

Figure 2: Direct Volume Rendering of a scalar field [12].

3. Data structure

Consider a unit cube $\Omega = [0, 1]^3$ as the domain problem. This cube is subdivided into eight ones with the same volume in Fig. 3a. The barycenter of each 3D volume of Fig. 3a is represented by black dots in Fig. 3b. Graph nodes, that represent 3D volumes, are also represented by black dots in Fig. 3c. Moreover, the barycenter of each 3D volume of Fig. 3b is represented by a graph node depicted in Fig. 3c. In these nodes, the related volume information, such as geometric coordinates and volumetric information, is stored. In order to represent the neighbors of a given node *n*, six other nodes are associated to the edges that reach node *n*, from east, north, west, south, front and back directions in Fig. 3c.

In order to represent volumes with different level of refinements, another type of node is used, called transition node and represented as white circles in Fig. 3c. Transition nodes are created in order to connect neighboring volumes that are at different refinement levels. Specifically, white circles in the graph of Fig. 3c link to the boundary of the domain.

3.1. *Refinement process*

In Fig. 3d, a 3D cube of Fig. 3a is refined into eight new ones. In a local refinement, the refined graph node, that represented the refined 3D volume, is deleted and replaced by a sub-graph represented in Fig. 3e. This sub-graph

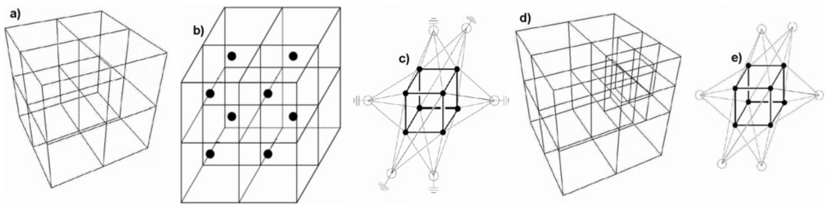

Figure 3: An initial domain and its representation [12]: a) Initial mesh; b) unit cube subdivided into eight ones and their barycenters represented by black dots; c) ALG data structure in 3D case; d) Refined mesh; e) Sub-graph that replaces a graph node in a local refinement.

is a pack of new nodes with level of refinement increased of one. This new pack is formed by eight nodes, that represent the eight new 3D cubic volumes, and six transition nodes, that link the six faces of the original volume. Moreover, those nodes of the new pack maintain a identifier indicating that they belong to a same pack for a possible unrefinement. In addition, those nodes of the new pack point to each other if they have a face in common. Furthermore, two neighboring graph nodes can be connected directly only if they are at the same level. The directed graphs of Fig. 3 are represented as undirected ones. Moreover, each edge indicates that its incident nodes point to each other. If they have different levels of refinement their connection are performed through transition nodes, one at a level.

Since the sub-graph data structure depicted in Fig. 3e replaces a graph node in a local refinement, it can be considered as a *plug-in* structure. Notice that the initial mesh, depicted in Fig. 3a, has four cubes.

In each local refinement, a pack of eight cubes (Fig. 3d) replaces an original cube. Similarly, this fractal characteristic is noticed in the graph. Notice that the initial graph, depicted in Fig. 3c, has eight graph nodes. In each local refinement, a pack of eight graph nodes, six transition nodes and their pointes (Fig. 3e) replaces an original graph node. This process may be repeated, increasing the level of refinements of each graph node. Figure 4 represents the graph with local refinements.

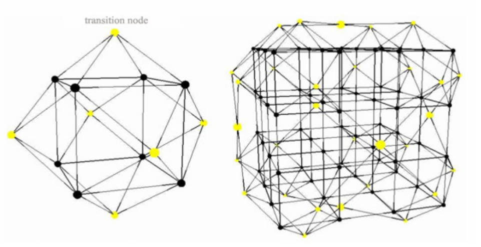

Figure 4: ALG representing local refinements [12].

3.2. Unrefinement process

The unrefinement process is allowed only to a pack of nodes originated from a same refinement. This means that eight graph volumes that are in the $n - th$ level are replaced by only one volume in the $(n - 1) - th$ level. These volumes must be originated from the same volume graph node to ensure the previous configuration of the mesh. The unrefinement process is implemented through the following steps:

- create a graph node;
- set spatial coordinates and other informations to the newly created graph node;
- link the graph node to its neighbors;

- realease memory space allocated to the transition and graph nodes;
- renumber the mesh by the 3D-MHC.

Figure 5 exemplifies an unrefinement process. Both refinement and unrefinement processes can lead to simplification of transition nodes since volumes share a same face.

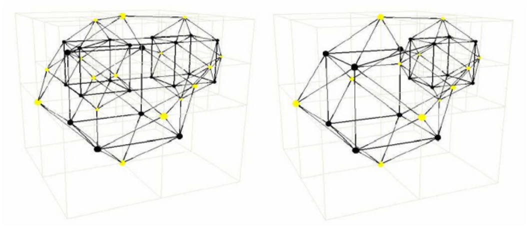

Figure 5: Unrefinement process [12].

4. Numbering the volumes

Numerical methods that discretize the domain problem, e.g. Finite Volume, Element and Difference Methods, in order to solve a partial differential equation result in a linear system to be solved. Those discrete places of the mesh need to be numbered in order to relate each unknown with its specific discrete place.

In [5], a space-filling curve was proposed in order to solve this task, the 2D Modified Hilbert Curve (2D-MHC). In a 3D domain, a curve (with endpoints) is a continuous function whose domain is the unit interval $[0, 1]^3$. Moreover, the limit of the sequence given by curves of order $1, 2, \ldots$, is the curve that passes through each point in the unit cube $[0, 1]^3$, or in a closed continuous volume. Furthermore, space-filling curves are curves whose ranges contain the entire 3D unit cube.

4.1. 2D and 3D Modified Hilbert Curves

A 2D Hilbert Curve is the limit of a sequence of curves $H_1, H_2, H_3, \ldots.$ represented in Figs. 6a-c and Fig. 7. In [5], the 2D Hilbert Curve was tailored to non-uniform quadrangular meshes. Moreover, ALG employs the 2D-MHC in order to number the mesh.

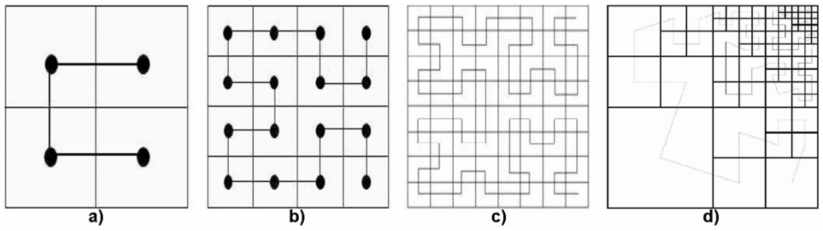

Figure 6: Total ordering of a quadrangular meshed geometry [5]: a-c) Sequence of 2D Hilbert Curves; d) example of a 2D-MHC.

In order to implement the 3D Modified Hilbert Curve (3D-MHC), in each graph node, this work defined an extra pair of pointers. These pointers are used to implement the total ordering of the graph nodes. Moreover, a double

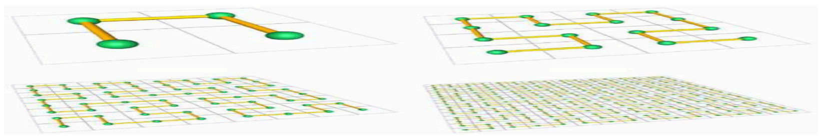

Figure 7: Sequence of 2D Hilbert Curve approximations [12].

linked list connects all graph nodes. As a result, a structure with local properties in the process of refinement and unrefinement of the mesh is obtained.

In the adaptive process, the mesh can become non-uniform and 2D Hilbert Curves cannot be used to represent these meshes. In [5], the 2D-MHC was proposed, depicted in Fig. 6d, in order to number non-uniform quadrangular meshes. Specifically, [5] used the 2D-MHC in order to number the barycenters of finite volumes.

The algorithm to implement the 2D-MHC can be found in [5] and [12]. The 2D-MHC allows volumes with different levels of refinement in the same stage. This feature may allow, in the case of scientific visualization, a significant reduction in the processing time of the images to be displayed. In this work, this approach was performed to visualize numerical solutions of partial differential equations.

In this work, in order to implement the 3D-MHC, the procedures described by [12] were applied. Figure 8 shows the 3D-MHC ordering a refined scenario.

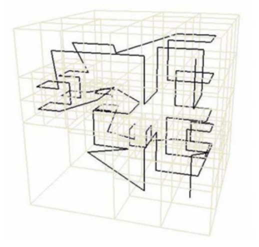

Figure 8: 3D Modified Hilbert Curve [12].

5. Numerical results

In order to investigate this adaptive 3D technique for visualization, finite difference approximations were applied to the 3D Heat Conduction Equation Problem $u_t = a^2(u_{xx} + u_{yy} + u_{zz})$, where a represents the coeficient of termical conductivity. This problem was applied in a unit cube built with a homogeneous isotropic material. Moreover, visualization of a volumetric mesh representing variation of temperature in the considered volume of study was evaluated. Figure 9a shows a mesh visualization. In this case, a temperature variation ranging from 0^o to 100^o with thermal coefficient equal to 1 was considered.

Tests were applied in a Intel Pentium IV 2.28GHz, 256MB of main memory and NVIDIA GeForce4 MX 4000. Tests were performed with same initial and boundary conditions. The following results show times after 40 tests for each test in the process of maximum order refinement applied in the mesh. Figure 9a represents a mesh with 16^3 volumes using 3D Hilbert Curve. Table I compares the scheme performance in seconds using the 3D Hilbert Curve and the 3D-MHC in order to show the curve and mesh. It shows the mininum and maximum obtained values. The % column, rounded up to integers, shows the percentual of time decreased using the 3D-MHC in relation to using the 3D Hilbert Curve. One can analyze that, in general, the more volumes in the scenario, the more gain in using the 3D-MHC.

Table I: 3D Hilbert Curve (HC) × 3D Modified Hilbert Curve [12].

	curve						mesh					
	min			max			min			max		
vols.	HC	MHC	%	HC	MHC	%	HC	MHC	%	HC	MHC	%
64^3	0.44	0.06	14	0.54	0.06	11	0.44	0.17	39	0.56	0.18	32
128^3	0.64	0.09	14	0.81	0.12	15	2.08	0.25	12	2.25	0.32	14
256^3	1.82	0.21	12	1.89	0.24	13	7.64	0.68	9	7.85	0.75	10

Table II shows times in seconds of image generations using the 3D-MHC considering a viewer from inside and outside the cube. In relation to a viewer inside the volume, the times are smaller than the ones from a viewer outside the volume, even when the generated curve goes through the whole dots of the domain, including the ones outside the scene.

Table II: Image generations using the 3D MHC [12].

	inside		outside	
volumes	min	max	min	max
64×64×64	0.03	0.05	0.09	0.12
128×128×128	0.10	0.13	0.21	0.24
256×256×256	1.04	1.43	1.13	2.55

A code based on strategies proposed by [11] was implemented through the CG language in order to program the graphical device ([13]). Figure 9b exemplifies this result using direct volumetric rendering. The better rendering time tested with this approach and 16^3 volumes was 7.45 seconds. Table I shows that using ALG and 3D-HC (the regular version) with 256^3 volumes, the time was 7.85 seconds for rendering the mesh, a bit slower than the approach presented by the coded based on [11] (with 16^3 volumes). On the other hand, Table I shows that using ALG (with adaptive mesh refinement) and 3D-MHC with 256^3 volumes, the time was 0.75 seconds for rendering the mesh.

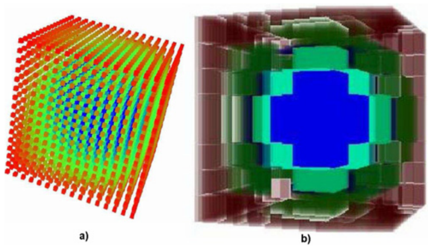

Figure 9: Mesh visualizations [12]: a) Mesh with 16^3 volumes; b) Direct Volume Rendering.

Figure 10a shows a 3D-MHC ordering a mesh with 64^3 volumes. It is worth pointing out that the implementation allows to visualize only low or high temperatures. Exemplyfing, the areas with low temperature are shown in Fig. 10b and the areas with high temperature are shown in Fig. 10c.

The implementation allows to visualize only a selected area. Figure 11a shows an adaptive mesh refinement applied to an initial 32^3 volumes using the 3D-MHC. Figure 11b shows a selected area of this mesh.

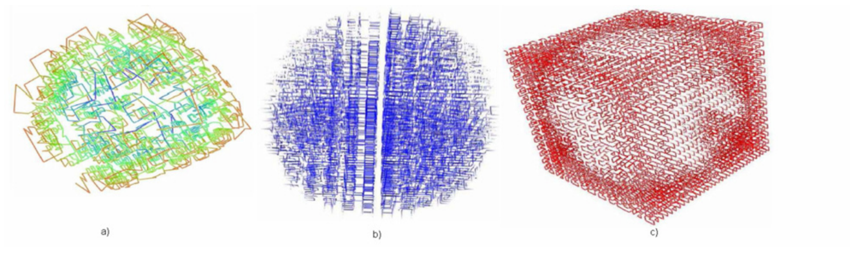

Figure 10: 3D-MHC ordering a mesh [12]: a) 3D-MHC ordering a mesh with 64^3 volumes; b) representation of regions with low temperature; c) representation of regions with high temperature.

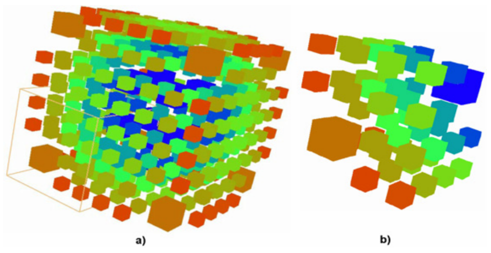

Figure 11: Visualization of data with 32^3 volumes [12]: a) mesh represented by ALG using a 3D-MHC; b) a selected area of the mesh.

6. CONCLUSIONS

In this ongoing work, a technique for 3D mesh rendering is applied. It stores and handles data arising from numerical simulation of partial differential equations. The strategy is to build algorithms and 3D data structures upon the same concepts that guided the development of the Autonomous Leaves Graph. The procedure intends to be efficient in providing traversal times for saving the cost of visiting volumes that are not part of a scene, if the point of view is from within the domain to be visualized. Even when the point of view is from the exterior of the volumetric region to be visualized, the technique intends to present competitive results.

In addition, this paper implements a 3D space-filling curve. The total-numbering of 3D meshes is performed by the 3D-MHC coupled to the refinement and unrefinement processes. The parts of the curve that link cubic volumes belonging to a region outside the scene to be exhibited can be traversed through the unrefined nodes. Those cubic volumes in the scene can also be coupled to the transparency model and may employ the same traversal time saving strategy. Continuing this work, the technique shall be applied to other general simulations in order to compare its efficiency and accuracy to competing visualization algorithms.

Acknowledgements

The authors gratefully acknowledge CAPES, FAPERJ and CNPq for the financial support.

References

[1] J. von Neumann, First draft of a report on the EDVAC, 1945. contract no. w-670-ord-492, Moore School of Electrical Engineering, University of Pennsylvania. Reprinted (in part), in: Origins of Digital Computers: Selected Papers, Springer-Verlag, 1982, pp. 383–392.

[2] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, R. Huebner, Level of Detail for 3D Graphics, Morgan Kaufmann, San Francisco, 2003.

[3] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, Massachusetts, 1989.
[4] I. Boada, I. Navazo, R. Scopigno, Multiresolution volume visualization with a texture-based octree, The Visual Computer 17 (3) (2001) 185–197.
[5] D. D. Burgarelli, M. Kischinhevsky, R. J. Biezuner, A new adaptive mesh refinement strategy for numerically solving evolutionary PDE's, Journal of Computational and Applied Mathematics 196 (2006) 115–131.
[6] S. L. O. Gonzaga, M. Kischinhevsky, Autonomous Leaves Graph applied to the Boundary Layer Problem, in: International Conference on Computational Science - ICCS 2009, Vol. 5544 of Lecture Notes in Computer Science, Springer, 2009, pp. 560–569.
[7] D. Brandão, S. L. O. Gonzaga, M. Kischinhevsky, Finite-element non-conforming h-adaptive strategy based on Autonomous Leaves Graph, in: International Conference on Computational Science - ICCS 2009, Vol. 5544 of Lecture Notes in Computer Science, Springer, 2009, pp. 570–579.
[8] S. L. O. Gonzaga, M. Kischinhevsky, A graph-based adaptive triangular-mesh refinement applied to classical elliptic and parabolical problems, in: XXXII Congresso Nacional de Matemática Aplicada e Computacional, 2009, pp. 607–612.
[9] C. Dyken, G. Ziegler, High-speed marching cubes using histogram pyramids, in: Proceedings of Eurographics 2007, Vol. 26, 2007, pp. 163–170.
[10] W. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, in: Computer Graphics (SIGGRAPH87 Proceedings), Vol. 21, 1987, pp. 163–170.
[11] R. S. Espinha, Visualização volumétrica interativa de malhas não estruturadas utilizando placas gráficas programáveis, Master's thesis, PUC-Rio (2005).
[12] D. T. Robaina, Um visualizador para navegação através de cenários tridimensionais baseado em malhas adaptativas, Master's thesis, Universidade Federal Fluminense (2007).
[13] C. Nvidia, Nvidia GPU Programming Guide Version 2.2.1, 20 april, 2005, http://www.nvidia.com..object/gpu_programming_guide/html, 2005.