A deformable model for fast visualization of Euclidean isosurfaces of the brain

F.P.G. Bergo *F.F. de Goes*
S.K. Goldenstein *A.X. Falcão*

Technical Report - IC-05-20 - Relatório Técnico

September - 2005 - Setembro

A Deformable Model for Fast Visualization of Euclidean Isosurfaces of the Brain

Felipe P.G. Bergo, Fernando F. de Goes, Siome K. Goldenstein, and Alexandre X. Falcão
Instituto de Computação – Universidade de Campinas
CEP 13084-970, Campinas, SP, Brazil
{felipe.bergo,fernando.goes,siome,afalcao}@ic.unicamp.br

Abstract

Curvilinear reformatting of MR images has been of great assistance to the diagnosis of dysplastic lesions in epilepsy patients. A recent improvement is its complete automation using *Euclidean isosurfaces* obtained from an "envelope" of the brain. Each isosurface consists of points with the same Euclidean distance from the envelope. The 3D visualization of these isosurfaces is usually computed using voxel projection and the voxel intensities as texture map. In this paper, we introduce a deformable model which combines a polygonal mesh and a graph, both obtained from the envelope. Our data representation can quickly create meshes for isosurfaces at arbitrary depths from the envelope. This representation fits well to modern GPUs, leading to high interactivity with the hardware currently available. We evaluate the visualization performance of our new deformable model side by side to an efficient implementation of voxel projection.

1 Introduction

Volumetric image data from magnetic resonance (MR) and computerized tomography (CT) have become common in clinical environments, but most computational tools do not exploit the 3D information of these datasets for diagnosis, relying on 2D slice views similar to those printed on film.

In Medicine, 3D visualization of CT/MR images can improve diagnosis and help treatment planning. The diagnosis of *dysplastic lesions* (which usually manifest themselves as blurred regions between gray and white matter) in epilepsy patients, for example, is much easier when we visualize MR images of the brain using a curvilinear reformatting [1]. The main idea is to display voxel intensities on 3D cuts, orthogonal to the sulci. This way, lesions are more evident than on the traditional MR slices [1]. However, the procedure requires careful manual delineation of lines, following the curvature of the brain in several slices, and interpolation of their shape [2]. The results are sensitive to this manual delineation, and the interpolation introduces curvature artifacts on the frontal and occipital lobes.

The technique presented in [3] represents a considerable improvement in the previous approach for curvilinear reformatting [1]. The method automatically extracts an *envelope* of the human brain — a surface that encloses the brain's surface as close as possible to the dura-mater — from MR images, and obtains *Euclidean isosurfaces* from that envelope. Each isosurface consists of points with the same Euclidean distance from the envelope. The isosurfaces are orthogonal to most of the sulci, enhancing the visualization of dysplastic lesions when the voxel intensities (texture) are mapped onto them. The method also eliminated the aforementioned curvature artifacts.

The 3D visualization of the isosurfaces is usually computed based on voxel projection [3]. In this paper, we introduce a deformable model which combines a polygonal mesh (all triangles) and a graph, both obtained from the envelope. Our data representation can quickly deform the original envelope mesh to describe any arbitrary depth isosurfaces. This scheme fits well to modern GPUs, and we compare it to an efficient software-based voxel projection implementation.

2 Background and Related Work

For a given set of MR images of the head, the method proposed in [3] automatically extracts the brain and obtains an envelope as close as possible to the surface of the dura-mater. This envelope can otherwise be obtained by interactive segmentation [4, 5]. For each voxel inside the envelope, the minimum Euclidean distance between that voxel and a voxel of the envelope must be computed. This operation is a 3D Euclidean distance transform (EDT). The distance map encodes several isosurfaces where voxels have the same Euclidean distance from the envelope. The entire process is based on the image foresting transform (IFT)— a graph-based approach to the design of image processing operators [6, 7].

In the IFT, each voxel is a node of an implicit graph whose arcs are defined by an adjacency relation between voxels. For a given set of seed voxels and suitable path-cost function, the IFT computes a minimum-cost path forest, whose trees are rooted at the seed set. The EDT of the envelope, for example, can be obtained by choosing its voxels as seeds and the Euclidean distance between voxels as path-cost function. After the Euclidean IFT, each seed becomes the root of a tree composed by its closest voxels. In this forest, the Euclidean IFT assigns to each voxel one optimal path from its closest seed and the cost of this path is the squared Euclidean distance between them. Both informations are exploited in this paper. The algorithm is essentially an extension of Dijkstra's algorithm [8] to multiple sources and more general path-cost functions. Note that, an Euclidean isosurface in the forest consists of the voxels with the same cost.

The Euclidean isosurfaces can be rendered using voxel projection [9] (i.e., voxel splatting [10, 11]). Fast surface rendering methods based on voxel projection could be used to store each isosurface in a separate data structure [12], but we provide a simpler and efficient solution in Section 3.3. Voxel projection does not require any special translation from the implicit raster model, but do not take advantage of the capabilities of modern GPUs to provide faster interaction. Medical doctors need to rotate, zoom and traverse the surfaces at various depths to locate and analyze the lesions, and speeding up the rendering time will provide a more pleasant user experience. To take advantage of GPU capabilities we are required to extract a polygon mesh model from the envelope [13]. The canonical method to accomplish this is the marching cubes [14] algorithm, but it leads to a prohibitively high number of triangles when applied to high-resolution volumetric data sets.

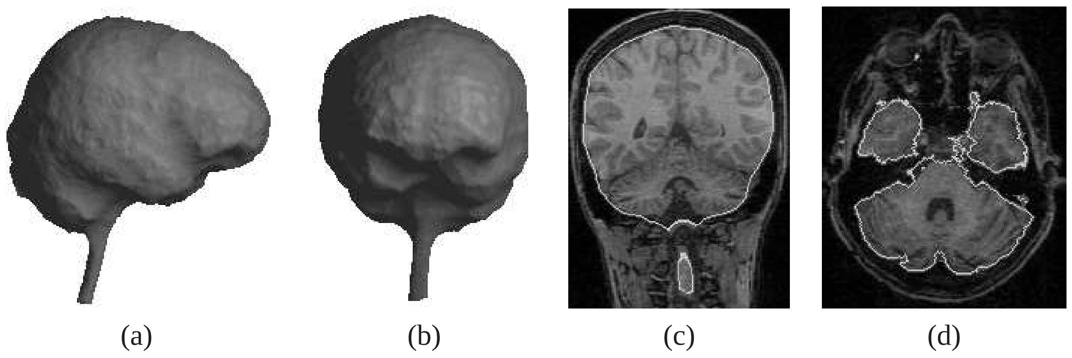


Figure 1: (a-b) 3D renditions of an envelope obtained by automatic segmentation. (c-d) Multiple contours on a coronal and axial slices.

In this paper, we present a simpler method to obtain a polygonal mesh that discretizes the envelope. Our method is similar to the approach reported in [15], but with resolutions selected by the user. We also use a second Euclidean IFT to create an optimal path forest. This way, we can efficiently deform the envelope's mesh to represent any inner isosurface.

3 Methods

In this section, we describe a our simple technique for obtaining the envelope's mesh (Section 3.1), its deformation process to obtain all inner isosurfaces' meshes (Section 3.2), and the efficient software-based algorithm for voxel projection (Section 3.3), which we later use as a baseline for performance comparison.

3.1 Computation of the Envelope's Mesh

In our approach, we find the contours of selected 2D cross sections of the envelope. We then tessellate the contours to compute the envelope's mesh.

Even though the envelope (Fig. 1a–b) is a single connected component, some of its 2D cross-sections may contain two or more contours. Coronal slices may have two contours at the spinal chord (Fig. 1c), and axial slices often have up to three contours around the basal regions (Fig. 1d). Segmentation mistakes can also lead to more contours on a given slice. Sagittal slices, on the other hand, are less likely to have multiple contours.

To extract the envelope's mesh, we take a number N_F of equally spaced sagittal slices. On each slice, we sample N_P equally spaced points of the largest contour. Finally, we tessellate these polygonal contours to form a mesh composed by only triangles. In Fig. 2, we illustrate this idea and present sample envelope meshes for different given values of N_F and N_P. The first and last contours are closed with flat caps built each with N_P triangles connected to the N_P contour points and the contour's geometric center. The number of triangles in the final mesh is $2N_P(N_F - 1) + 2N_P = 2N_P N_F$. The complete procedure for mesh extraction is:

1. For each of the N_F slices, sample N_P points on the largest contour.

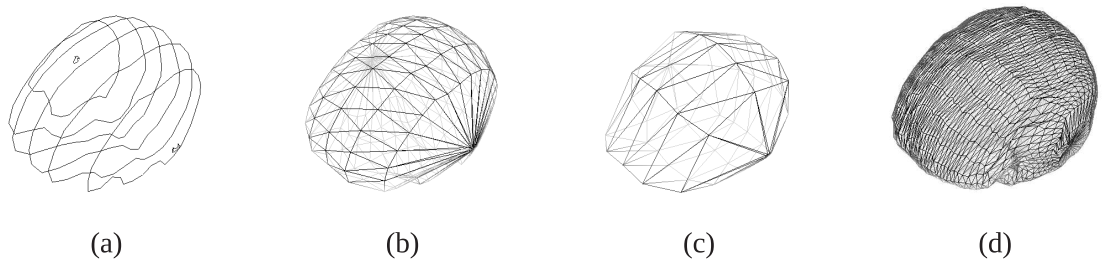


Figure 2: Mesh extraction: (a) Sampled sagittal contours with $N_F = 8$, $N_P = 30$. (b) Triangle tessellation between consecutive contours of (a). (c) The mesh obtained with $N_F = 8$, $N_P = 8$. (d) The mesh obtained with $N_F = 80$, $N_P = 40$.

2. For each $(N_F - 1)$ pair (α, β) of consecutive contours, add its $2N_P$ triangles $(\alpha_1, \alpha_2, \beta_1), (\beta_1, \beta_2, \alpha_1), \ldots, (\alpha_{N_P}, \alpha_1, \beta_{N_P}), (\beta_{N_P}, \beta_1, \alpha_{N_P})$ to the mesh.
3. For each of the first and last contours: compute the contour's geometric center c and add the sequence $(\alpha_1, \alpha_2, c), (\alpha_2, \alpha_3, c), \ldots, (\alpha_{N_P}, \alpha_1, c)$ of N_P triangles to the mesh.

3.2 Isosurface Mesh Deformation

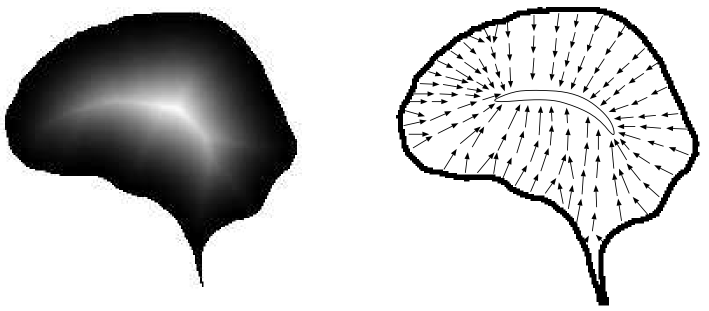
In order to locate dysplastic lesions, medical doctors need to visualize Euclidean isosurfaces with respect to the envelope at arbitrary depths, changed interactively with minimum delay.

While the mesh extraction algorithm of section 3.1 could be applied to each possible isosurface, the contour sampling step makes it prohibitively expensive (it takes a few seconds on a typical dataset, and the user expects response times of hundredths of second). We now present a method to "shrink" the envelope's mesh to obtain a mesh for any given target depth D with the same number of triangles. We want to find, for each vertex, the closest point in the discrete 3D Euclidean distance map with distance $\geq D$. We efficiently achieve this through a second Euclidean IFT [6, 7].

We already have the distance map of the envelope (Fig. 3a) and the maxima of this map are the farthest inner points from the envelope. These maxima are used as seeds of the second Euclidean IFT, leading to optimal paths that end at the envelope and converge to these maxima. For each vertex of the envelope's mesh, we can follow backwards its optimal path in order to find its closest point on any given isosurface. Fig. 3b depicts a representation of the forest resulting from the second Euclidean IFT, whose paths converge to the inner "core" of maximum depth. The computation of the second Euclidean IFT takes a few seconds, but then we can use it to obtain the mesh for any depth very quickly. Fig. 4 shows some meshes obtained by shrinking the envelope's mesh in this way. Some triangles will degenerate to lines as the mesh shrinks, but this does not lead to artifacts in the rendition.

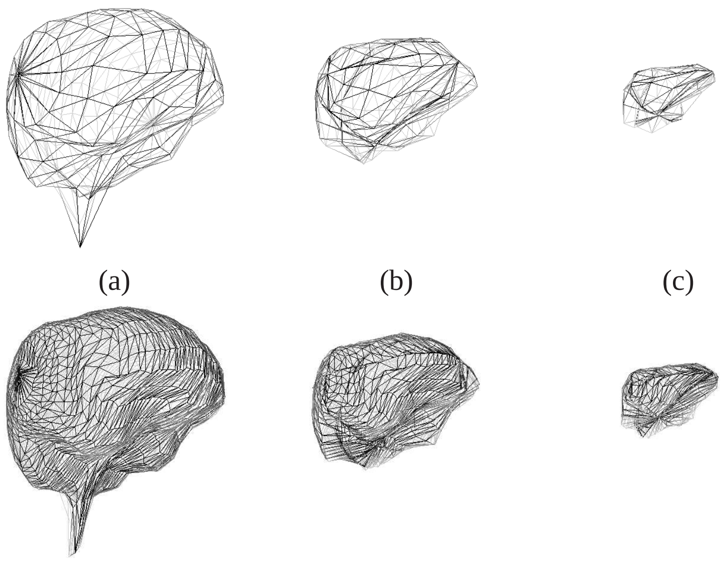
3.3 Efficient Voxel Projection

A straightforward way to render the Euclidean isosurface at depth D is doing a raster scan of the volumetric data projecting every voxel with depth $\geq D$ to a z-buffer [16]. However, only about



(a) (b)

Figure 3: Envelope's distance map and mesh shrinking: (a) Sample slice of the first EDT (brighter values are more distant from the envelope). (b) Topology of the forest resulting from the second EDT, computed from the maxima of (a): paths converge to the deepest isosurface.



(a) (b) (c)

(d) (e) (f)

Figure 4: Examples of mesh shrinking: (a) envelope's mesh with 450 triangles. (b–c): shrinkings for deeper isosurfaces, computed from (a). (d) envelope's mesh with 3840 triangles. (e–f): shrinkings for deeper isosurfaces, computed from (d).

15% of the voxels are inside the envelope, and even less voxels are inside the deeper isosurfaces. Since it is sufficient to project voxels within a thin shell from the desired isosurface, it is more efficient to discard voxels outside the envelope and keep the remaining voxels sorted according to their isosurface depth (Euclidean distance to the envelope). Voxel projection can then be carried out by finding the first voxel at the desired depth by binary search and then traversing the sorted voxel vector until a voxel with depth $\geq D + \delta$ is reached, where δ is the thickness of the shell. Some shell thickness is required to prevent holes that occur due to the discretization of the distance transform. We compute the distance transform with an adjacency radius of $\sqrt{3}$ voxels (26-neighborhood), so a 2-voxel shell is enough to prevent holes. Splatting [10, 17] is also useful to prevent holes without the need to trace rays using the inverse viewing transform.

4 Results

Two parameters must be taken into account when evaluating visualization methods: the initial setup time, and the frame rates achieved after initialization. The user only needs to wait the setup time once for each patient being analyzed. For the visualization of Euclidean isosurfaces, there is a third important parameter: the time required to switch between isosurfaces of different depths. We call *rotation frame rate* (RFR) the rate at which a given isosurface can be rotated and/or zoomed without changing the current polygon mesh. Since deeper isosurfaces lead to smaller rasterization areas, rendering is likely to become faster as the isosurface depth increases. To account for this, in all experiments we measured the frame rate for the envelope and for a very deep surface where the sulci are no longer present.

We call *mesh derivation time* (MDT) the time taken to generate an arbitrary mesh from the envelope's mesh. MDT does not account for the rendering of the generated meshes, but the frame rate between isosurfaces of different depths can be calculated by $\frac{1}{RFR+\frac{1}{MDT}}$. The next sections report the experiments used to evaluate and compare the hardware-accelerated visualization of the deformable model (the envelope's mesh and the optimal-path forest of the second Euclidean IFT) to the aforementioned software-based voxel projection approach. We used a variety of computers running Linux for the experiments, and they are listed and labeled in Table 1. Machines C and D are the same, with video cards switched between experiments.

4.1 Deformable Envelope Mesh Experiments

We implemented a visualization tool in C++ and OpenGL [13] to render the deformable mesh with hardware acceleration, and tested it on a variety of PCs running Linux.

The MR texture was defined as a 3D texture in OpenGL. The tests were not run on machine A due to the lack of 3D acceleration in its video adapter, and machines B and G did not support the $256 \times 256 \times 256$ 3D texture. Experiments were run with 10 different values for N_F and N_P, leading to meshes sized between 200 and 8000 triangles. Table 2 shows the setup times and frame rates for the case $(N_F = 30, N_P = 30)$, which leads to 1800 triangles and provides reasonable quality.

On each test, the model was rotated 1440 degrees (4 complete turns) or 10 seconds (whichever expired first) with 1 degree steps. Fig. 5 shows plots for the frame rates on all mesh sizes. Except for machine E, all computers were able to render the scenes fast enough to provide a responsive

Name	CPU	Video Adapter
A	Pentium III 1.0 GHz	Intel 815
B	Athlon 1.1 GHz	nVidia MX4000
C	Pentium 4 2.4 GHz	nVidia FX5700LE
D	Pentium 4 2.4 GHz	nVidia 6800GT
E	Xeon 2.4 GHz	nVidia MX440
F	Pentium 4 2.8 GHz	nVidia FX5200
G	Pentium 4 3.0 GHz	nVidia MX4000
H	Xeon 3.06 GHz	ATI Radeon 9800Pro
I	Athlon FX64 3500+	nVidia FX5700LE

Table 1: Computers used in the experiments.

Machine	Setup time	RFR Envelope	RFR Inner	MDT
C	13.6	412	518	0.173
D	13.6	1172	1172	0.173
E	12.4	15	70	0.270
F	11.9	263	355	0.287
H	11.5	697	709	0.264
I	5.8	424	536	0.110

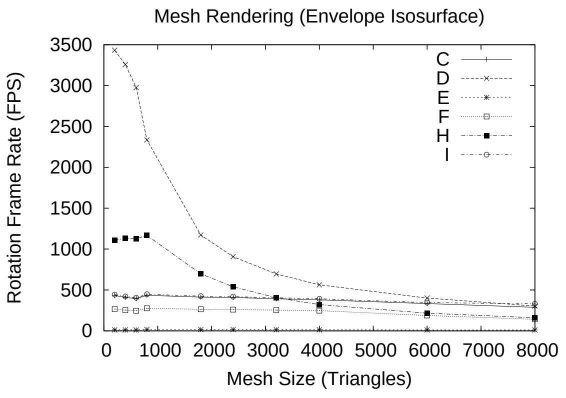
Table 2: Hardware-accelerated rendering performance for a mesh with 1800 triangles ($N_F = 30$, $N_P = 30$). Setup time and MDT (mesh derivation time) are given in seconds, RFR (rotation frame rate) is given in frames per second.

feeling to the user.

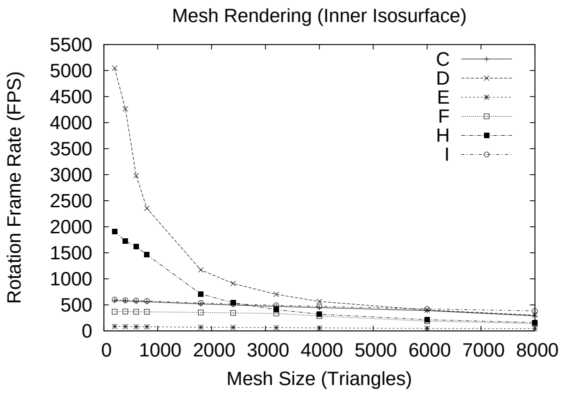
We were surprised to find out that machines B and G were unable to render the scenes, since their graphic adapter is newer than E's. The time required to derive a new mesh with our approach was kept below 0.5 seconds, still providing good response times. MDT was measured deriving 50 possible meshes (from the 1-voxel-deep one to the 50-voxel-deep one) and taking the average time.

The derivation process can be sped up if we allow it to use more memory, keeping a "half-way mesh" (say, the 25-voxel-deep one if 50 is the maximum depth) such that the meshes inside it are derived from it instead of the envelope's mesh. This reduces the search along the optimal paths of the second Euclidean IFT. The time required to find the 1-voxel-deep mesh was never less than half the average MDT, so this optimization would lead to a speed up factor below 2.

Mesh derivation takes time linearly proportional to the number of triangles, as shown in Fig. 6 with timings obtained on machine F. Fig. 7 shows the renditions obtained with the 10 mesh sizes as well as with the voxel projection method.



(a)



(b)

Figure 5: Plots of rotation frame rate vs. number of triangles: (a) envelope's mesh. (b) inner isosurface's mesh.

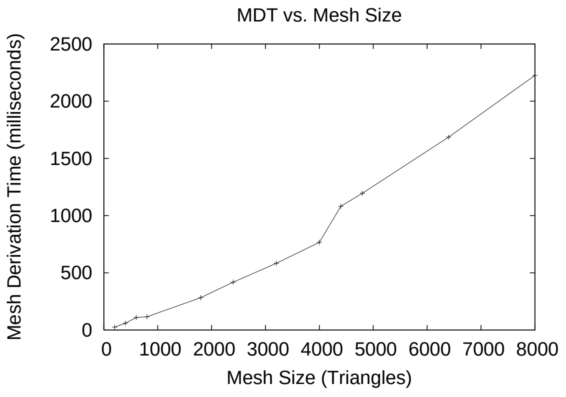


Figure 6: Mesh derivation time vs. mesh size, on machine F.

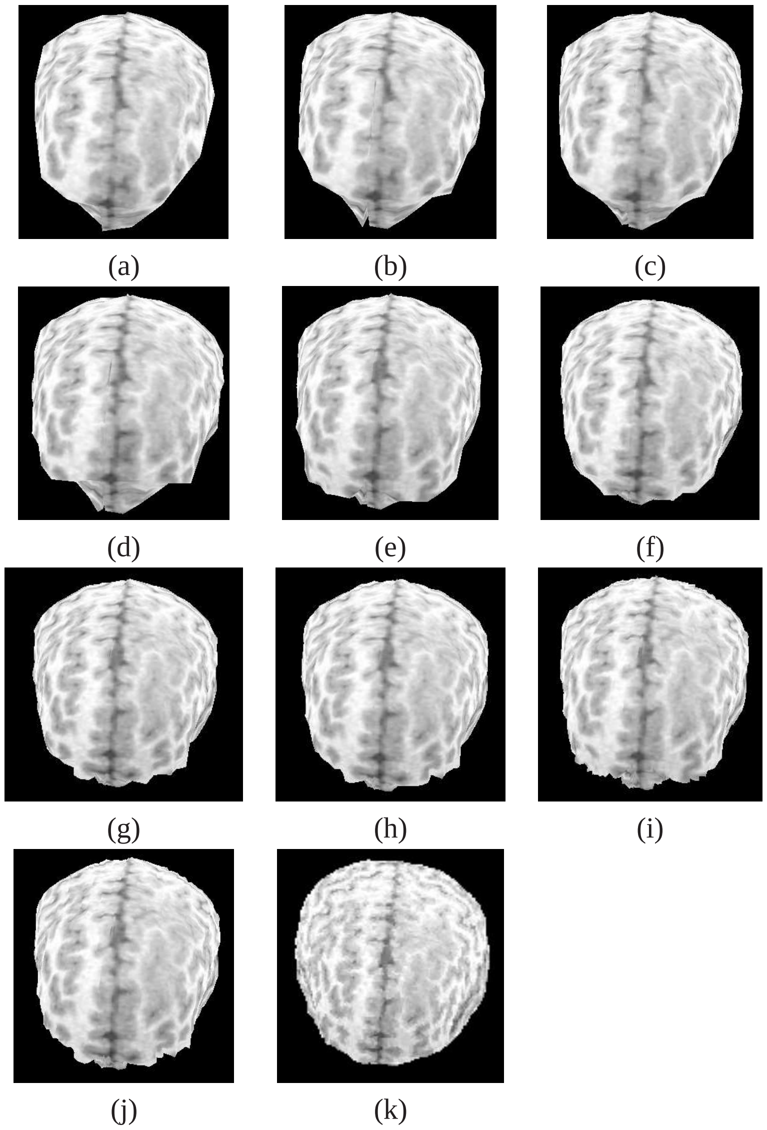


Figure 7: A dysplastic lesion (blurred gray matter on the right side of the figures) shown in renditions with various mesh sizes (a–j) and by voxel projection (k). Mesh sizes: (a) 200 triangles; (b) 400 triangles; (c) 600 triangles; (d) 800 triangles; (e) 1800 triangles; (f) 2400 triangles; (g) 3200 triangles; (h) 4000 triangles; (i) 6000 triangles and (j) 8000 triangles.

Machine	Setup time	RFR Envelope	RFR Inner
A	5.02	9	98
B	4.86	10	104
C	2.35	22	280
D	2.35	22	280
E	2.26	24	339
F	1.92	26	308
G	1.96	18	329
H	1.74	28	397
I	0.99	40	422

Table 3: Software-based voxel projection rendering performance. Setup time is given in seconds, and RFRs (rotation frame rates) are given in frames per second.

4.2 Voxel Projection Experiments

We tested software-based voxel projection on all 9 machines of Table 1. The volume was rotated 360 degrees with 1 degree increments (360 frames) to find the RFR for both the envelope and the same deep surface used to obtain the RFR on the 4^{th} column of Table 2.

The results are shown in Table 3. MDT is not measured since no extra computation is required when changing isosurface depths: the new isosurface is rendered just like a different viewing transform would be rendered, leading to the same frame rate listed in the RFR columns.

4.3 Discussion

Hardware-based rendering depends on a large number of factors – graphic adapter hardware, CPU speed, driver quality and support for the scene description language, and performance varies too much between machines. Hardware-accelerated rendering also leads to artifacts due to the approximation of surfaces by meshes of triangles.

In the case of the brain cortex, these artifacts are negligible (Fig. 7), but for other parts of the brain and other organs this can be a serious limitation (Fig. 8). Increasing the number of triangles indefinitely compromises the rendering time, since it increases the time required to derive the mesh for isosurfaces at different depths (Fig. 6) and graphic adapters deliver poorer performance as the mesh size increases (Fig. 5).

Software-based voxel projection provides faster setup times, does not require a translation from the volumetric raster representation in which the datasets are provided, and depends on CPU speed alone.

Dysplastic lesions are often in the “shallow” isosurfaces, closer to the envelope, where frame rates are lower with both methods. In this case, the frame rates achieved with voxel projection (9–40 FPS, from Table 3) make user interaction less pleasant than with the hardware-accelerated approach, provided that the user has a good graphics adapter with proper drivers. On machine E, software-based voxel projection beats the hardware-accelerated approach on all measurements – setup time, envelope frame rate and inner isosurface frame rate, and machines B and G can be used

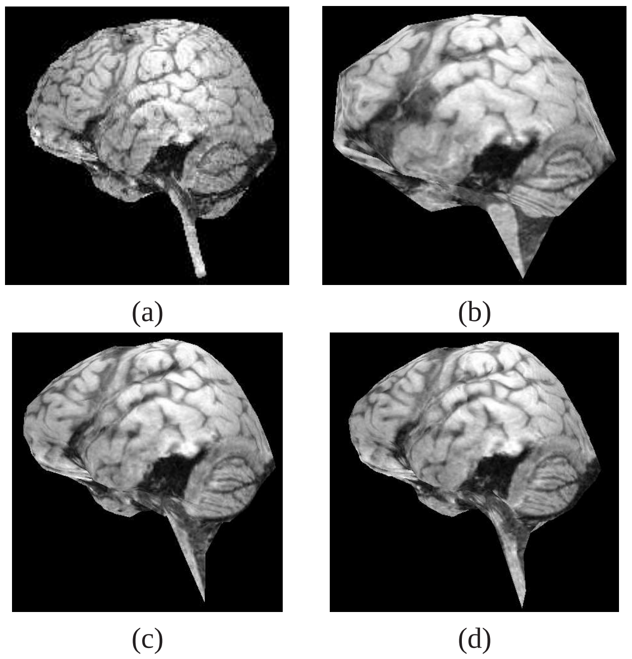


Figure 8: Spinal chord artifact due to the mesh discretization of the envelope: (a) voxel projection (no artifact). (b) rendition with 400-triangle mesh. (c) rendition with 2400-triangle mesh. (d) rendition with 8000-triangle mesh.

for visualization with voxel projection, while hardware-based rendering simply does not work on them.

5 Conclusions

We have presented a simple and efficient approach to compute a deformable mesh from the brain's envelope [3], exploiting the graph representation resulting from an Euclidean IFT [6].

The 3D visualization of the Euclidean isosurfaces of the human brain is of great assistance to the diagnosis of dysplastic lesions, which are the most frequent cause of refractory epilepsy [1].

Representing Euclidean isosurfaces as polygon meshes is a required step to use the capabilities of modern GPU-based graphic adapters [13]. We evaluated the visualization performance of our deformable model and compared it to an efficient voxel projection approach on a variety of PC computers. Our method outperformed the software-based approach in all computers except on the one with the oldest graphic adapter. However, the mesh representation may introduce artifacts, and causes the application to be dependent of a large number of parameters.

We can conclude that, while hardware-accelerated rendering based on polygon meshes delivers better performance on most commonly available hardware, voxel projection should be offered as a fall back to deal with driver issues and objects with complex shapes. Also, voxel projection delivers reasonable performance to achieve interactivity on most currently available hardware.

Acknowledgments

We thank Dr. Li M. Li and Pablo A. Rio for the computational facilities at ASPE/FCM-UNICAMP, Dr. Fernando Cendes for some of the data used in these experiments, and CAPES, CNPq (Proc. 302427/04-0), FAPESP (Proc. 03/13424-1), and FAEPEX-UNICAMP (Proc. 1679/04) for the financial support.

References

[1] A. C. Bastos, R. M. Comeau, F. Andermann, D. Melanson, F. Cendes, F. Dubeau, S. Fontaine, D. Tampieri, and A. Olivier. Diagnosis of subtle focal dysplastic lesions: Curvilinear reformatting from three-dimensional magnetic resonance imaging. *Annals of Neurology*, 46(1):88–94, 1999.

[2] Rogue Research. BrainSight. http://www.rogue-research.com/B/epilepsy.htm.

[3] F. P. G. Bergo, A. X. Falcão, M. A. Montenegro, and F. Cendes. Automatic extraction of euclidean isosurfaces for visualization of dysplastic lesions from brain MRI. Technical Report IC-05-08, Institute of Computing, University of Campinas, May 2005.

[4] A. X. Falcão and F. P. G. Bergo. Interactive volume segmentation with differential image foresting transforms. *IEEE Trans. on Medical Imaging*, 23(9):1100–1108, 2004.

[5] Bluevoxel Software. Bluevoxel Alberio. http://www.bluevoxel.com/alberio_en.php.

[6] A. X. Falcão, J. Stolfi, and R. A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.

[7] A. X. Falcão, L. F. Costa, and B. S. da Cunha. Multiscale skeletons by image foresting transform and its applications to neuromorphometry. *Pattern Recognition*, 35(7):1571–1582, 2002.

[8] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[9] G. Frieder, D. Gordon, and R. A. Reynolds. Back-to-front display of voxels based objects. *IEEE Computer Graphics and Applications*, pages 52–60, Jan 1985.

[10] J. Huang, K. Mueller, N. Shareef, and R. Crawfis. Fastsplats: Optimized splatting on rectilinear grids. In *IEEE Visualization*, pages 219–227, Salt-Lake City, Oct 2000.

[11] M. Zwicker, H. Pfister, J. van Barr, and M. Gross. Ewa splatting. *IEEE Visualization and Computer Graphics*, 8(3):223–238, 2002.

[12] J.K. Udupa and D. Odhner. Shell rendering. *IEEE Computer Graphics and Applications*, 13(6):58–67, 1993.

[13] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL Programming Guide*. Addison-Wesley, New York, fourth ed. edition, 2003.

[14] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer & Graphics*, pages 163–169, July 1987.

[15] H. Fuchs, Z.M. Kedam, and S.P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, pages 693–702, Oct 1977.

[16] J.D. Foley, S.K. Feiner A. van Dam, and J.F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, New York, second ed. edition, 1990.

[17] F. P. G. Bergo. Segmentação interativa de volumes baseada em regiões. Master's thesis, Universidade Estadual de Campinas, Instituto de Computação, Feb 2004.