

# Generating Segmented Tetrahedral Meshes from Regular Volume Data for Simulation and Visualization Applications

Alex J. Cuadros-Vargas & Luis G. Nonato  
*University of São Paulo, Brazil*

Eduardo Tejada & Thomas Ertl  
*University of Stuttgart, Germany*

**ABSTRACT:** This paper describes a method to generate Delaunay tetrahedral meshes directly from regular volume data by vertex insertion driven by an error criterion. By adapting the mesh to the input data, the method produces meshes suitable for fast and precise volume ray-casting. A partitioning method that splits the mesh into homogeneous sub-meshes is also presented. Such mesh segmentation strategy turns out very useful in applications such as numerical simulation and focus-and-context rendering, where well defined structures are of paramount importance.

## 1 INTRODUCTION

Computational simulation in domains defined from regular volume data has stimulated the technological development in many branches of science, e.g. flow simulation (Steinman 2002) and virtual surgery (Bro-Nielsen and Cotin 1996). One of the main challenges in this context is to decompose the domain in a mesh suitable for simulation as well as visualization, as most algorithms proposed to perform this task are not aimed at generating meshes respecting the features of the whole volume, turning it difficult a realistic volume rendering of the mesh. Another drawback regarding mesh generation from regular volumes concerns the automation of the process. In general, the algorithms strongly rely on extensive pre-processing steps, which typically consist in segmenting regions of interest, using the boundary surfaces of such regions as input to a mesh generator (Cebal and Lohner 1999).

Techniques that act directly on the input volumetric data (Hale 2001), in general, line up the mesh with the volume features, but do not worry about segmenting structures contained in the volumes. The mesh segmentation is essential in applications such as physically-based deformation of volumetric bodies where different stiffness constants must be assigned to different materials within the mesh. Segmented meshes can also be exploited for rendering purposes, such as using focus-and-context techniques and combining surface and volume rendering to highlight the boundaries between different regions.

In this work we show a method that can be used

to generate segmented Delaunay tetrahedral meshes, directly from the input volume, aiming at obtaining useful characteristics for volume visualization. These properties have been exploited and incorporated in our GPU-based single-pass ray-caster for tetrahedral meshes (Tejada and Ertl 2005).

## 2 RELATED WORK

One of the earliest works on generating meshes from images was proposed by Terzopoulos and Vasilescu (1992), who approach the problem using adaptive meshes assembled as a set of nodal masses connected by springs. Their method moves the mesh nodes by gradient and curvature information so as to concentrate nodes in regions where a rapid variation of the data is present. A similar approach has been proposed by Hale (2001), which makes use of a potential energy function to line up points of a regular lattice with image features and generates the final mesh by Delaunay tessellation. A pre-processing step is employed to reduce noise and highlight sharp regions.

Aiming at minimizing the interpolation error between the mesh and the original image, Garland and Heckbert (1995) proposed an iterative method that inserts, in a Delaunay tessellation, the points of maximal interpolation error. Grosso et al. (1997) use an error measure derived from a least-square approximation to guide an iterative tetrahedral subdivision. The subdivision follows a set of rules applied over tetrahedra and octahedra marked for refinement. Roxborough and Nielson () also employ least-square approx-

imation to estimate the interpolation error, but, differently to Grosso et al., only a single subdivision rule that splits the longest edge of a tetrahedron is applied. Takahashi et al. (2004) proposed an adaptive tetrahedralization method for computing topological volume skeletons, using topological and geometrical criteria to iteratively sub-divide the tetrahedra.

The iterative method proposed by Garcia et al. (1999) controls the maximum root-mean-square error (RMS) by choosing the vertices of the mesh from a curvature image, that is, more vertices are placed in areas with high curvatures. The mesh model is built by generating the Delaunay tessellation from the chosen vertices. Regions with high RMS error are resampled and the Delaunay tessellation updated. An interesting non-iterative algorithm has been presented by Yang et al. (2003). Their method makes use of zero-crossing jointly with the Floyd-Steinberg error-diffusion algorithm to choose a set of points from which the Delaunay tessellation is calculated.

A common characteristic of the strategies described above is that a mesh is generated directly from the original dataset, not demanding a pre-segmentation. Some algorithms, however, strongly rely on a pre-segmentation process to define regions of interest from which a mesh is built. An example is the algorithm proposed by Cebal and Lohner (1999) which makes use of a heavy user-assisted pre-processing to binarize the original image. The mesh is generated by an advancing-front technique that starts from the surface defined by the binary data set. Zhang et al. (2003) and Berti (2004) make use of thresholding and an oct-tree decomposition to generate an adaptive tetrahedral mesh inside the region stipulated by the threshold. Mesh improvement techniques are used by both methods in order to smooth and fit the mesh into the thresholded region.

Our method presents characteristics from both classes of algorithms described above. The mesh is generated directly from the regular volume data, thus avoiding the stressful pre-processing step. Since the generated tetrahedra tend to be fitted within homogeneous regions of the volume, the mesh can be segmented according to the volume data.

### 3 BASIC CONCEPTS

Let  $S$  be a set of points in  $\mathbb{R}^3$ . A *tetrahedral mesh* of  $S$  is a three-dimensional simplicial complex  $M$  whose vertices are the points in  $S$ , and any  $k$ -simplex of  $M$ ,  $k = 0, 1, 2$ , is contained in at least a 3-simplex (tetrahedron) of  $M$ . If the union of all simplices in  $M$  makes up the convex hull of  $S$  and the circumsphere of each tetrahedron in  $M$  does not contain any point of  $S$  in its interior then  $M$  is called a *Delaunay mesh*. Delaunay meshes are closely related to Voronoi diagrams, which can be obtained by associating to

each  $k$ -simplex of  $M$  a  $3 - k$  cell in the diagram. In fact, each vertex of the diagram is the center of the empty sphere circumscribing a Delaunay tetrahedron and each vertex  $v_i$  of the Delaunay mesh is contained in the three-dimensional cell of the diagram that contains the points in  $\mathbb{R}^3$  closest to  $v_i$  (Fortune 1992).

Let  $S$  be a set of points and  $M$  be a tetrahedral mesh of  $S$ . If  $M$  can be decomposed as  $M = M_1 \cup M_2 \cup \dots \cup M_k$ , where each  $M_i$  is also a mesh and  $M_i \cap M_j$ ,  $i \neq j$  is either empty or a  $k$ -dimensional simplicial complex,  $0 \leq k \leq 2$ , then  $\{M_1, M_2, \dots, M_k\}$  is said to be a *k-partitioning* of  $M$  in *sub-meshes*  $M_i$ ,  $i = 1, \dots, k$ .

A  $X \times Y \times Z$  *regular volume* is a function  $I : [0, \dots, X] \times [0, \dots, Y] \times [0, \dots, Z] \rightarrow \mathbb{R}^+$  that assigns to each point  $p \in [0, \dots, X] \times [0, \dots, Y] \times [0, \dots, Z] \subset \mathbb{Z}^3$  a non-negative scalar  $I(p)$ . The pair  $(p, I(p))$  is called *voxel*.

## 4 GENERATING DELAUNAY TETRAHEDRAL MESHES FROM STRUCTURED VOLUMETRIC DATA

As mentioned above, our method first generates a tetrahedralization that respects a desired feature of the volumetric data. In our case, we intend to generate a mesh whose tetrahedra do not cross different regions of the volume, that is, each tetrahedron should be contained in only one homogeneous region of the volume. Then, a partition of this mesh into homogeneous sub-meshes. We describe below these two processes in detail.

### 4.1 Generating the mesh

Let  $T$  be the set of tetrahedra of a Delaunay mesh  $M$  whose vertices are points of a regular volume  $I$  and  $E : T \rightarrow \mathbb{R}^+$  be a function that associates an error measure to each tetrahedron in  $T$ . In fact, function  $E$  measures how good a tetrahedron is regarding a specific property, that is,  $E$  enables to decide whether or not a tetrahedron must belong to the tetrahedralization. Different strategies have been proposed to define the function  $E$ , which usually rely on evaluating  $E$  by traversing all voxels inside a tetrahedron  $t$  so as to decide, based on some characteristic of the image, whether or not  $t$  is a suitable tetrahedron. In general, when  $E$  indicates that  $t$  is a bad tetrahedron, the mesh is updated by inserting new points within  $t$  (Garland and Heckbert 1995). Although widely employed, this strategy presents two main drawbacks. Traversing all voxels within a tetrahedron may demand a high computational effort, as every time the mesh is updated all new tetrahedra must be scanned in order to re-evaluate  $E$ . Another problem arises during the insertion of new points, which, when not handled properly, can result in an accumulation of points around already existing vertices. To avoid this we adopt a strategy based on the medians of the tetrahedra to evaluate  $E$ .

Traversing only medians one can reduce the computational effort while being effective in detecting tetrahedra that cross different regions of the volume.

Let  $h_1, h_2, h_3, h_4$  be the four medians of a tetrahedron  $t \in M$ ,  $\mathcal{E}$  be a high-pass filter and  $c_{\mathcal{E}}$  be an user defined scalar. Consider the set of points

$$P_j^t = \{p \in h_j \mid \mathcal{E}(p) \geq c_{\mathcal{E}}\} \text{ for } j = 1, \dots, 4.$$

and let  $P^t = \bigcup_{j=1}^4 P_j^t$ . Then,  $P^t$  is the set of high frequency points (detected by  $\mathcal{E}$ ) of a tetrahedron  $t$ . In our experiments we used the Sobel 3D operator as the filter  $\mathcal{E}$ .

Let  $A \subset \mathbb{R}^3$  be a finite and non empty set. We define the distance between some point  $p \in \mathbb{R}^3$  and  $A$  as  $D(p, A) := \min\{d^2(p, a) \mid a \in A\}$ , where  $d(\cdot, \cdot)$  is the Euclidean distance. Note that, since  $A$  is a finite set,  $\exists a_* \in A$  such that  $d^2(p, a_*) = D(p, A)$ .

Let  $w^t$  be the circumcenter of  $t \in M$ . Consider  $P^t \neq \emptyset$ . As  $P^t$  is finite, we have that, there always exists some  $p_*^t \in P^t$  such that  $d^2(w^t, p_*^t) = D(w^t, P^t)$ . Let  $V$  be de set of vertices of  $M$ . From these definitions de error function  $E$  can be stated as:

$$E(t) = \begin{cases} 0 & \text{if } P^t = \emptyset \\ D(p_*^t, V) & \text{if } P^t \neq \emptyset \end{cases}$$

As  $p_*^t$  is a high frequency point, it is supposed to be on the boundary of different regions of the volume  $I$ . Therefore,  $D(p_*^t, V)$  measures how much a tetrahedron  $t$  invades a region of the volume. Thus, values of  $E(t)$  close to zero indicate that  $t$  is well fitted within a region contained in the volume. Therefore, a tetrahedron  $t$  is considered unsuitable if  $E(t) > c_E$ , where  $c_E$  is an user defined scalar. An unsuitable tetrahedron  $t$  is eliminated by inserting  $p_*^t$  in  $M$ .

It is a well known fact that circumcenters are distant points of the vertices of a Delaunay tetrahedralization. Since the point  $p_*^t$ , in a tetrahedron  $t$ , is chosen to be the closest point to a circumcenter  $w^t$ , then  $p_*^t$  is, in general, a distant point of the vertices of a Delaunay mesh  $M$ . Therefore, the problem of accumulating points around already existing vertices is also reduced.

## 4.2 Partitioning

Suppose that a mesh  $M$  has already been partitioned in  $n$  sub-meshes  $M_1, M_2, \dots, M_n$ ,  $n > k$ , where  $k$  is the target number of partitions. Thus,  $n - k$  sub-meshes must be merged in order to get a  $k$ -partitioning. The merge operation aims at gluing the most similar sub-meshes, where similarity is measured from a set of properties extracted from the sub-meshes.

Voxel information as well as topological and geometrical properties can be considered when comparing sub-meshes. Let  $S_I(M_i, M_j)$  and  $S_{TG}(M_i, M_j)$  be

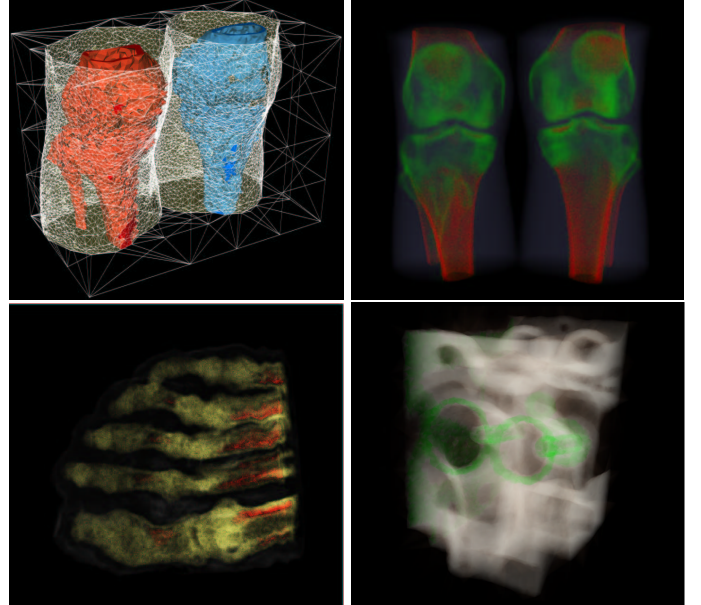


Figure 1: Partitioned mesh obtained for the Knee dataset and its respective volume rendering (top). Volume rendering of the meshes obtained for the Foot and Engine datasets (bottom).

functions that measure similarity regarding voxel information and topological/geometrical properties of  $M_i$  and  $M_j$  respectively. From  $S_I$  and  $S_{TG}$  we can define a general similarity function as  $S(M_i, M_j) = w_1 S_I(M_i, M_j) + w_2 S_{TG}(M_i, M_j)$ ,  $w_1 + w_2 = 1$ ;  $0 \leq w_1, w_2 \leq 1$ , where the weights  $w_1$  and  $w_2$  are used to adjust the relevancy of voxel information and topological/geometric properties in the similarity measure.

Function  $S_I$  compares an array of characteristics extracted from the voxels in  $M_i$  and  $M_j$ , assigning a value between 0 and 1. In our context, values close to zero mean highly similar sub-meshes whereas values close to one indicate not similar sub-meshes. A simple way to define  $S_I$ , which has been adopted in our implementation, is compare the average of voxels values from each sub-mesh, that is,  $S_I(M_i, M_j) = |H(M_i) - H(M_j)| / |H(M_i) + H(M_j)|$ , where  $H(M_k)$  is a function that associates to each sub-mesh  $M_k$ ,  $k = i, j$  the average of voxel values. In order to avoid to scan every voxel in a new sub-mesh generated from the merge operation, we compute  $H(t) = \frac{1}{n_t} \sum_{p \in t} I(p)$  for each tetrahedron  $t$ , where  $n_t$  is the number of voxels in  $t$ , before starting the mesh segmentation process. From the value of  $H$  in each tetrahedron, we can estimate  $H(M_k)$  as the weighted average of  $H$  in the tetrahedra, that is,  $H(M_k) = \sum_{t \in M_k} Vol(t) H(t) / \sum_{t \in M_k} Vol(t)$ , where  $Vol(t)$  is the volume of tetrahedron  $t$ . It is worth noting that at the beginning each tetrahedron is considered as a sub-mesh.

$S_{TG}$  is concerned with topological and geometrical properties to compare two sub-meshes. Aiming at favoring the merging of “small” neighbor sub-meshes

into the largest ones, we define  $S_{TG}$  as follows:

$$S_{TG}(M_i, M_j) = \begin{cases} \frac{\min\{Vol(M_i), Vol(M_j)\}}{\max\{Vol(M_i), Vol(M_j)\}} & \text{if } M_i \cap M_j \neq \emptyset \\ 1 & \text{if } M_i \cap M_j = \emptyset \end{cases}$$

Therefore,  $S_{TG}(M_i, M_j)$  assumes smaller values when  $M_i$  and  $M_j$  are neighbor sub-meshes ( $M_i \cap M_j \neq \emptyset$ ) and the volumes are very different.

Using a priority queue sorted by  $S$  we can recover the most similar sub-meshes in a very efficient way. Thus, the sub-meshes with highest priority (the smallest value of  $S(M_i, M_j)$ ) are merged up to a  $k$ -partitioning is reached.

Figure 1 (top-left) shows an example of a segmented mesh generated with this process. Notice that multiple structures are meshed simultaneously without pre-processing the original data.

## 5 RENDERING OF SEGMENTED TETRAHEDRAL MESHES

We adapted our single-pass ray-caster for tetrahedral grids (Tejada and Ertl 2005) to exploit the characteristics of the meshes generated with the approach described in Section 4, namely the segmentation and the fact that the scalar value is constant within a tetrahedron. Also, since the mesh is convex, no convexification pre-processing or re-entry handling must be performed.

Since the scalar is constant within a tetrahedron the accumulated associated color  $\tilde{C}_k$  and opacity  $\alpha_k$  for the ray segment between the entry point  $x_k^i$  and exit point  $x_k^o$  of the  $k$ -th tetrahedron  $t^{(k)}$  intersected by the ray can be calculated as  $\tilde{C}_k = \tilde{c}(H(t^{(k)}))\alpha_k$  and  $\alpha_k = 1 - \exp(-\tau(H(t^{(k)}))d(x_k^i, x_k^o))$  respectively, where  $H(t^{(k)})$  is the scalar value at tetrahedron  $t^{(k)}$  and  $\tilde{c}$  and  $\tau$  are transfer functions for color and extinction densities respectively. Therefore, approximation improvement approaches such as pre-integrated volume rendering (Engel, Kraus, and Ertl 2001) are not necessary. The ray-integral is then accumulated as usual as  $\tilde{C}_{acc} = \tilde{C}'_{k-1} + (1 - \alpha'_{k-1})\tilde{C}_k$  and  $\alpha_{acc} = \alpha'_{k-1} + (1 - \alpha_{k-1})\alpha_k$ , where  $\tilde{C}'_{k-1} = \tilde{C}_{acc}$  and  $\alpha'_{k-1} = \alpha_{acc}$  are the accumulated associated color and opacity after the  $(k-1)$ -th intersected tetrahedron respectively. Renderings obtained using this modified ray-caster are shown in Figure 1.

Another consequence of having constant scalars within the tetrahedra is that the gradient within it becomes zero. However a rough way to obtain a non-zero gradient within each tetrahedron is setting the scalar value  $s(v_i)$  at vertex  $v_i$  as  $s(v_i) = \sum_{t_i \in St(v_i)} H(t_i) Vol(t_i) / \sum_{t_i \in St(v_i)} Vol(t_i)$ , where  $St(v_i)$  is the star of  $v_i$ . Thus, a constant gradient per tetrahedron is computed as done in previous work (Tejada and Ertl 2005). Since the gradient is also constant within

each tetrahedron, two-dimensional transfer functions indexed by scalar and gradient magnitude can be used (Fig. 2:bottom-left).

The segmentation of the grid can be exploited by focus-and-context techniques. Also, the boundaries between the sub-meshes can be highlighted by rendering them as translucent surfaces during ray-casting (Fig. 2:bottom-right). This is accomplished by simply storing a further scalar for each face of the tetrahedra and fetching it during ray-traversal for the current tetrahedron. This scalar is the identifier of the boundary to which the face belongs. If the face is not a boundary face, a negative value is set. In order to interactively support color and opacity picking for the boundaries, the boundary identifier is used to fetch the corresponding color and opacity from a one-dimensional color table. The fetched color  $c_k^s$  and opacity  $\alpha_k^s$  are then accumulated as  $\tilde{C}'_k = \tilde{C}'_{acc} + (1 - \alpha'_{acc})\alpha_k^s c_k^s$  and  $\alpha'_k = \alpha'_{acc} + (1 - \alpha'_{acc})\alpha_k^s$  (assuming that  $c_k^s$  is a non-associated color).

## 6 RESULTS

The tests performed were carried out on a standard PC with a 3.20 GHz 64-bits processor and 2 GB of RAM and an NVidia 7800 GTX graphics board.

In Table 1 performance results are shown. In the table we also present ratios between the size in bytes of the tetrahedral mesh stored in a format we defined (detailed below) and the original volume stored as a stack of PGM files. The format we used for storing the tetrahedral mesh is as follows:

---

```

Keyword_Sub_meshes #Sub-meshes
List of Sub-meshes IDs

Keyword_Points #Points
X-Coord. Y-Coord. Z-Coord. Scalar Type*
...

Keyword_Tetrahedra #Tetrahedra
ID-Vertex0 ID-Vertex1 ID-Vertex2 ID-Vertex3
Scalar ID-Sub-mesh
...

```

---

\* Type is used to classify the vertex as boundary vertex or internal vertex.

---

Results concerning the performance of the hardware-implementation of the ray-caster are also presented in Table 1. It is important to mention that the ray-traversal included the boundaries surface check performed for each tetrahedron intersected by the ray and that two-dimensional transfer functions were used in all cases.

As discussed before, the segmentation can be exploited, not only for rendering purposes, but for applications where different materials present in the mesh must be identified. An example of this is volume de-

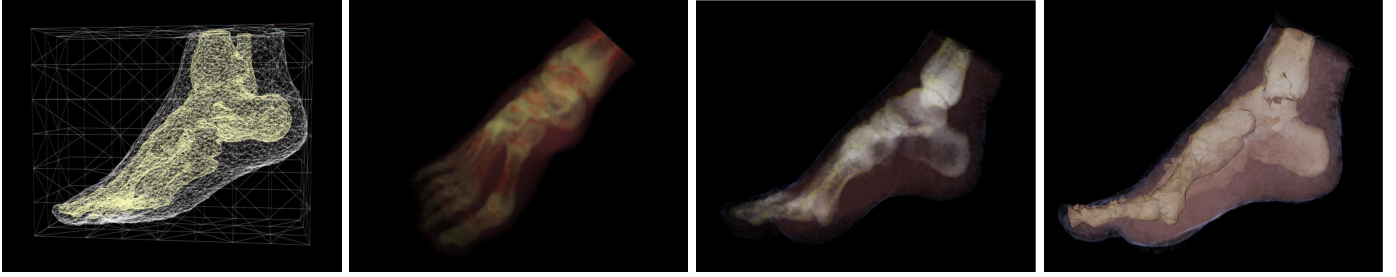


Figure 2: Mesh generated for the Visible Male Foot dataset (left). Volume rendering of the mesh (middle-left). Visualization using two-dimensional transfer functions (middle-right) and combined volume and surface rendering using the segmented mesh (right).

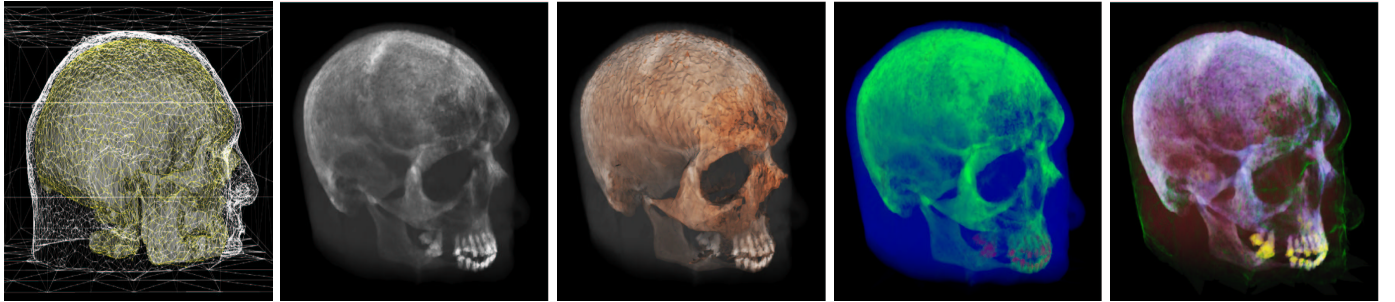


Figure 3: Renderings of a Delaunay tetrahedral mesh extracted from the Visible Human Head regular volume with our approach. From left to right: wireframe of the extracted mesh, volume rendering, combined volume rendering and surface rendering (for the boundary between two sub-meshes), rendering with one-dimensional transfer function and rendering with two-dimensional transfer function.

formation, where tetrahedral grids have proven to be more suitable than regular grids and different material properties, e.g. stiffness parameter, are desirable for different parts of the mesh (See Fig. 4 extracted from the work by Tejada and Ertl (2005)).

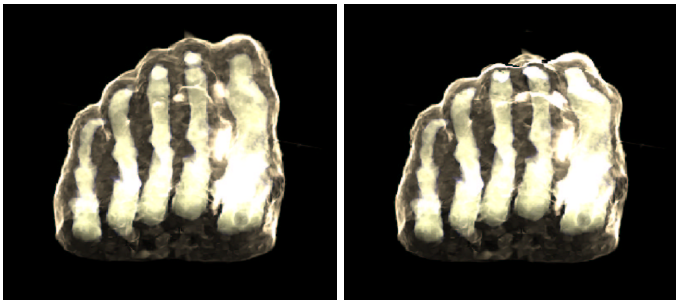


Figure 4: Mesh generated with our algorithm from the Foot data set suffering a physically-based deformation.

## 7 CONCLUSION

We have shown that the meshes generated by our strategy are suitable for volume visualization. In fact, the storage of a scalar representing the average of the voxel values in each tetrahedron allows the usage of multi-dimensional transfer functions in the hardware implementation of the ray-caster whilst avoiding pre-integration. The quality of the volume rendering carried out on the tetrahedral mesh generated by our method is comparable with the traditional regu-

lar grid volume visualization. However, the generated tetrahedral meshes can also be used in applications other than visualization, such as physically-based deformation.

However, it is necessary to point out that our approach does not generate tetrahedral meshes suitable for all kinds of numerical simulation, since the tetrahedra do not present a high quality shape. A technique devoted to improve the quality of the tetrahedra whilst preserving the alignment of the mesh with the volume is currently being implemented. This technique should turn the meshes suitable for a larger set of numerical applications.

Another point that deserves some comments is the mesh segmentation strategy. Although very simple, the mesh segmentation scheme presented good results in most of the tests we carried out, but its performance is not satisfactory in some cases. Aiming at improving the results obtained, we are adapting a powerful statistical segmentation strategy to work with tetrahedral meshes. We see such statistical approach as a promising tool.

Finally, concerning the rendering algorithm, the most critical problem that must be addressed arises due to the constant gradient within each tetrahedron, which affects the rendering for large tetrahedra when two-dimensional transfer functions dependent of the gradient are used or when non-polygonal iso-surfaces are rendered. Interpolation techniques are being inves-

Dataset	Processing time [sec]				Volume size [voxels]	Mesh size [tetrahedra]	Size ratio [bytes]	Rendering time [fps]	
	Mesh generation	Scalar values	Segmentation	Total				640 × 480	1600 × 1200
Knee	802.80	22.45	4.48	829.72	26471255	258600	0.218	3.76	0.80
Visible male head	961.14	28.79	3.75	993.68	77070336	389060	0.122	2.92	0.63
Visible male foot	218.32	6.08	1.24	225.65	4944354	101932	0.297	4.46	0.93
Foot	84.47	16.06	3.16	103.69	16777216	258600	0.256	3.41	0.74
Engine	264.37	7.40	1.38	273.15	8388608	122910	0.275	4.24	0.89

Table 1: Results obtained for the mesh generation algorithm and modified single-pass ray-caster. Processing times in seconds, input volume size, generated mesh size and compression ratio in bytes are shown for the mesh generation method. Performance results for the ray-caster modified for the tetrahedral meshes generated for two different viewport sizes are shown in frames per second.

tigated in order to sort out this problem.

## ACKNOWLEDGMENTS

We acknowledge the financial support of FAPESP - the State of Sao Paulo Research Funding Agency - (Grants #03/02815-0 and #02/05243-4), CNPq - the Brazilian National Research Council - (Grant #307268/2003-9), and DAAD - the German Academic Exchange Service - (Grant #A/04/08711).

## REFERENCES

- Berti, G. (2004). Image-based unstructured 3D mesh generation for medical applications. In *ECCOMAS*.
- Bro-Nielsen, M. and S. Cotin (1996). Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum* 15(3), 57–66.
- Cebral, J. and R. Lohner (1999). From medical images to CFD meshes. In *International Meshing Roundtable*, pp. 321–331.
- Engel, K., M. Kraus, and T. Ertl (2001). High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In *Workshop on Graphics Hardware*, pp. 9–16.
- Fortune, S. (1992). Voronoi diagrams and Delaunay triangulation. In *Computing in Euclidean Geometry*, Volume 1 of *Lecture Notes Series on Computing*, pp. 193–233.
- García, M., A. Sappa, and B. Vintimilla (1999). Efficient approximation of gray-scale images through bounded error triangular meshes. In *IEEE Image Processing*, pp. 168–170.
- Garland, M. and P. Heckbert (1995). Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, Carnegie Mellon University.
- Grosso, R., C. Lürig, and T. Ertl (1997). The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data. In *IEEE Visualization*.
- Hale, D. (2001). Atomic Images - a method for meshing digital images. In *10th International Meshing Roundtable*, pp. 185–196.
- Roxborough, T. and G. M. Nielson. Tetrahedron based, least squares, progressive volume models with application to freehand ultrasound data. In *IEEE Visualization*.
- Steinman, D. (2002). Image-based computational fluid dynamics modeling in realistic arterial geometries. *Ann. Biomed. Eng.* 30, 483–497.
- Takahashi, S., Y. Takeshima, G. Nielson, and I. Fujishiro (2004). Topological volume skeletonization using adaptive tetrahedralization. In *Geometric Modeling and Processing*, pp. 227–236.
- Tejada, E. and T. Ertl (2005). Large Steps in GPU-based Deformable Bodies Simulation. *Simulation Practice and Theory. Special Issue on Programmable Graphics Hardware* 13(9), 703–715.
- Terzopoulos, D. and M. Vasilescu (1992). Sampling and reconstruction with adaptive meshes. In *IEEE Int. Conf. Comp. Vision, Pattern Recog.*, pp. 829–831.
- Yang, Y., M. Wernick, and J. Brankov (2003). A fast approach for accurate content-adaptive mesh generation. *IEEE Trans. on Image Processing* 12(8), 866–881.
- Zhang, Y., C. Bajaj, and B.-S. Sohn (2003). Adaptive and quality 3D meshing from imaging data. In *ACM Symposium on Solid modeling and applications*, pp. 286–291.