Using a classification tree to speed up rendering of hybrid surface and volume models

Maria Ferre
URV Computer Science and Math. Dept.
Av. Països Catalans, 20
43007 Tarragona, Spain
mferre@etse.urv.es

Anna Puig
CREB Center of Biomedical Engineering Research,
ETSEIB, UPC
08028 Barcelona, Spain
anna@maia.ub.es

Dani Tost
CREB Center of Biomedical Engineering Research,
ETSEIB, UPC
08028 Barcelona, Spain
dani@lsi.upc.es

ABSTRACT

Hybrid rendering of volume and polygonal model is an interesting feature of visualization systems, since it helps users to better understand the relationships between internal structures of the volume and fitted surfaces as well as external surfaces. Most of the existing bibliography focuses at the problem of correctly integrating in depth both types of information. The rendering method proposed in this paper is built on these previous results. It is aimed at solving a different problem: how to efficiently access to selected information of a hybrid model. We propose to construct a decision tree (the Rendering Decision Tree), which together with an auxiliary run-length representation of the model avoids visiting unselected surfaces and internal regions during a traversal of the model.

Keywords

Volume Rendering, Hybrid Rendering, Decission Tree, Run-length encoding.

1. INTRODUCTION

There are two main approaches of the visualization of voxel models: rendering the volume as a whole and rendering isosurfaces. The former approach is achieved by Direct Volume Rendering (DVR), which computes the contribution of all the voxels to the image. Indirect Volume Rendering (IVR) can perform the latter approach. Isosurfaces are first extracted from the volume data with the popular Marching Cubes algorithm [Loc87], or by contouring and slicing [Mey92]. Then, they are rendered with the standard hardware-assisted polygon-rendering pipeline. Alternatively, Direct Volume Rendering (DVR) [Lev90] can also render surfaces, by computing the contribution to the image of the volume cells that contain the isosurfaces, and applying a surface shading without need of intermediate representations. The major advantage of IVR to visualize surfaces is that, once the polygonal model is extracted, its rendering is generally faster than DVR, even if its number of faces is large. In addition, any level of zoom can be applied during IVR rendering, whereas the lack of an actual polygonal model in DVR reduces its suitability when the surface is very near the observer. However, DVR does not require any preprocessing step and, thus, it provides more flexibility to visualize different isosurfaces.

Journal of WSCG, Vol.12, No.1-3, ISSN 1213-6972
WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.

Combining the two approaches, i.e. mixing surface and volume rendering is an interesting feature of volume visualization. It conveys more information than only surfaces but in a neater way than volume only. Therefore, it provides a better perception of the relationships between the different regions of the data. Mixing surfaces and volumes can also be used to show the interaction of external synthetic surfaces with a volume, as for example CAD models of a scalpel, bone prosthesis or a radiation beam with MR data or CT data.

DVR provides a natural way to mix surface and volume rendering by applying different shading models to the cells depending if they belong to the boundary of a feature or to its interior. In this paper, we call *hybrid shading* this rendering modality. Moreover, hybrid shading can be extend to external surfaces by voxelizing them in a pre-process

[Kau87]. A similar approach has been used for realistic rendering of complex scenes with a huge number of faces and repetitive patterns such as forests, in which trees and leaves can be represented with geometrical models if they are near or as a voxelization storing BRDF's if they are far [Nom95] [Ney98]. However, this type of rendering requires a low ratio image pixel per voxel, because it does not have a polygonal model of the surfaces.

An alternative to the voxelization is to keep separate representations of the volume and the surfaces (fitted as well as external) or a hybrid model representing both types of data, and to mix them during rendering. We will call *hybrid rendering* this approach to distinguish it from *hybrid shading*. Existing methods following this approach are based on ray casting [Lev90] [Fru91] [Miy92] [Sob94], Z-Buffer and Back-to-Front traversal [Goa89] [Kau90] [Eck99] [Tos93] and 3D texture-mapping [Kre99] [Boa03]. They are reviewed in the next section.

Most of the existing hybrid rendering methods focus mainly at solving the problem of correctly depth sorting the volume samples and the polygons. They assume that all the volume and the surfaces must be rendered and thus, they essentially visit all the data. This is inefficient when not all the volume and not all the surfaces must be rendered, which is often the case in the exploration of a dataset. In fact, a desirable feature of hybrid rendering is the flexibility to render specific regions of the data, either their surface, their internal volume or both, while hiding others.

Trying to restrict the traversal of a volume data set to the relevant cells is a general problem in volume visualization. It has been addressed for the acceleration of isosurfacing (octrees [Wil92], span space [Liv96]) as well as for speeding up volume rendering (kd-trees [Sub90], octrees [Wil94], run-length encoding [Lac94]). However, as mentioned above, this problem has been little addressed in hybrid rendering [Lev90].

This paper addresses the problem of the fast exploration of hybrid models. Our primary goal is provide means of performing efficiently various visualizations of the models, changing the selection of the features to be rendered. We assume that the original model has been classified. Therefore, our method is not suitable for a first exploration of a dataset, but rather to efficiently manipulate it once its internal structure is known. The main application of our method is teaching by rendering atlas or case study models. It is suitable for *hybrid shading* with low ratio pixel/voxel as well as for *hybrid rendering* when zooming on the surfaces is required.

In the next section, we review the previous work on hybrid rendering. Next, we describe the proposed model and the traversal algorithm associated to it. Finally, we show the results of the simulations and the conclusions.

2. PREVIOUS WORK

Ray casting can handle simultaneously different models by tracing the ray against each of them and merging their contribution along the ray. Based on this strategy, the hybrid ray tracer proposed by Levoy [Lev90] reduces the aliasing problems near the surfaces by performing an adaptive sampling of the volume. Miyazawa and Koyamada [Miy92] improved the antialiasing by first classifying the surface inside the volume. For each voxel, they compute a list of intersecting polygons. Therefore, the voxels with a non-empty list of polygons can be over sampled when rays are cast. Fruhauf [Fru91] also proposes the use of ray casting for volume, combined with any rendering algorithm for the geometric primitives capable of outputting an image space sorted list of elements that can be merged along the rays. Sobierajski and Kaufman [Sob94] designed a general ray tracer capable of handling various surfaces and volume models. They propose a classification of the intersection types that, together with the use of bounding boxes for the objects, avoid useless intersection tests and volume sampling of occluded regions.

Z-Buffer has also been extended to mixed surfaces [Goa89], [Kau90]: two independent Z-buffer processes are realized and then, the image buffers are combined according to their associated depths. A similar idea is used in *Volumizer* [Eck99]. The disadvantage of these approaches is that they cannot handle correctly semi-transparent volumes and transparent surfaces simultaneously. In a different approach [Tos93], the synthetic surface is converted into a face-octree representation according to the orientation and resolution of the voxel model. The face-octree is traversed back-to-front simultaneously with the voxel model preserving the correct depth order and thus allowing transparency of both the volume and the surface.

More recently, a 3D texture-map-based volume rendering approach has been proposed, able to render opaque and translucent polygons together with semi-transparent volume at interactive rates [Kre99]. The volume is processed in a slice-by-slice basis. The volume slices and the translucent polygons clipped at the boundary of the slabs defined by two consecutive slices are rendered alternatively, preserving a correct depth composition. In order to avoid costly clipping operations of the polygons against the slices, the

authors propose to use a bucket sort of the translucent polygons according to the slabs that they traverse. This strategy has also been used [Boa03] in order to render a hybrid octree which encodes the volume as well as the surface. The major advantage of the hybrid octree is that the texture associated to the volume can be generated at different levels of resolution depending on the variation of the scalar field in the node or its relative importance to the visualization. This characteristic simplifies the sorting of the surface polygons between slices, and it can be used to obtain multiresolution hybrid visualizations.

Levoy addresses the problem of avoiding irrelevant data during hybrid rendering in the paper mentioned above [Lev90]. Levoy proposes to use an octree representation of the volume to efficiently skip over empty regions. However, as the surface model is kept separately from the volume, this method does not provide a fast way of accessing directly to voxels traversed by an external surface or containing a given isosurface. The face octree proposed in [Tos93] grants a fast access to the codified surface voxels but it is restricted to the codification of only one external surface and it does not classify the volume into regions. The hybrid octree described by Boada at al. [Boa03] provides also a fast access to surface nodes. In addition, similarly to the BON structure [Wil92], it stores the maximum and minimum values of each node and thus, it provides means of skipping over non-relevant nodes of the volume. However, it is restricted to one fitted surface. Finally, sorting intersection elements as proposed by Sobierajski at al. [Sob94] can avoid traversing occluded voxels or computing unnecessary intersections ray surfaces but it does not eliminate unwanted traversals.

3. THE PROPOSED METHOD

The Rendering Decision Tree (RDT)

Our work is inspired on the *Decision Trees*, well known in the Information Theory field [Goo88] and used in decision analysis. Decision trees are boolean functions that classify variables of a multidimensional feature space into classes. They are such that each internal node of a tree tests a feature, each leaf node assigns a class or category and the arcs out of a node are labeled with the possible values of the features of this node. When rendering a scene, in general as well in volume and hybrid rendering, users must select and specify properties of the objects (or voxels) that should actually be visualized. This selection can be viewed as *feature vector* of a multidimensional feature space, in which the objects of a scene (voxels in a volume model) can be classified into semantic regions. Rendering queries are hardly arbitrary but rather follow the semantic structure of the scene. If this structure is known, it can be used to construct a decision tree, the *Rendering Decision Tree(RDT)* that will allow us to quickly determine the set of selected objects or voxels.

In the initial exploration of a voxel model, users select relevant ranges of values by conveniently specifying transfer functions [Kni01] that set to zero the opacity of the other value ranges. Thus, in the rendering, although all the volume is traversed, non-relevant regions are hidden. Let n be the number of voxels of a volume model. In order to render the model, only a subset of k voxels actually contribute to the image: those that fulfill the rendering specifications, i.e. those that belong to the selected class. Once the model has been classified, successive visualizations will drive to the selection of subsets within this classification. Traversing all the volume when these classes have already been characterized results in unnecessary visits to n-k remaining voxels. The aim of the RDT is provide a direct access to the selected subsets corresponding to the different classes.

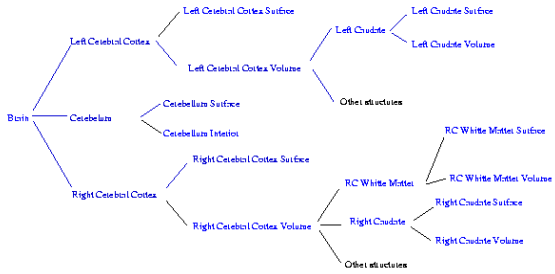


Figure 1. RDT Example for a model of the brain. Selected areas to be rendered are colored in blue.

Figure 1 shows an example of a RDT for a model of the brain. It classifies the brain into three regions: the right and left hemispheres cortex and the cerebellum. Each of these regions is subdivided into two categories: boundary and interior. The internal regions of the hemispheres are in turn classified into other structures, which again separate the boundary from the interior. If, as an example, the user wants to render the surface of the cerebellum of the right hemisphere together with the volume of the left hemisphere, the tree is traversed and voxels that belong to these classes are selected for rendering. In the next section we discuss how to associate to each node of such a tree the corresponding information of the model.

The run-length encoding

Our method is suitable for hybrid shading as well as hybrid rendering. In both cases, we take as input the

original voxel model. We use the classification step to construct an RDT that classifies the voxels into regions and into internal and boundary voxels. For hybrid shading, we first extract the relevant isosurfaces from the voxel model by applying a Marching Cubes algorithm. Each boundary voxel points to as many lists of polygons as extracted isosurfaces cross it. Furthermore, if a synthetic surface must be mixed with the volume, we classify and clip it with the volume cells as if it is a fitted surface.

A naive approach in order to associate to the RDT nodes the set of corresponding voxels is to keep a simple list of voxels per node. This approach presents serious drawbacks. First, being each list independent from the order, this model would not preserve the spatial ordering inherent to the voxel model between the different classes. Moreover, this structure would have huge memory occupancy, as it would require for each non-empty voxel one pointer per region to which it belongs.

We propose to construct an auxiliary voxel model that labels each voxel according to the leave of the RDT to which it belongs. We use a Run-Length (RL) codification of this model. Each leave of the RDT stores the label of its associated class in the RL model. Therefore, when a rendering selection is done, the RDT is traversed in order to compute the labels of the selected classes. If a terminal node matches the rendering criterion, its associated class is selected for rendering. If it is a non-terminal node that matches the rendering criterion, all its descendent classes are selected. The RL model is then traversed skipping over non-selected classes and accessing to the actual scalar or surface values of the voxels belonging to the selected classes only.

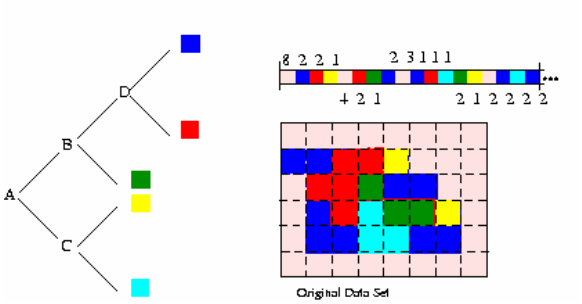


Figure 2. RL model based on the RDT labeling. At right, the RDT is depicted labeling the internal nodes with a letter and the leaves with a color. At bottom left, the classified voxel model is shown, and at top left top, its RL codification.

Figure 2 illustrates this process with a color codification. The voxel model has been classified according to the RDT tree shown at the right of the figure. The RL is depicted together with the classified voxel model. It should be observed that, for clarity, the voxel model depicted is the classified one, although in the voxel array, we actually keep the original gray values of the data. The traversal of the RDT selects the blue and red voxels. The run-length traversal skips over the other colors, and accesses to the actual scalar and surface values of only the blue and red voxels.

The traversal of the RL preserves the order of the voxel model, so it can be used for BTF and FTB traversals of the model, for splatting, shear-warp, or in order to compute 3D-texture maps of the model. However, it is not convenient for ray casting, as it does not provide a direct access to the voxels individually. As its primary goal is to speed up rendering, and being ray casting a slow method, this is not a major drawback. It should be observed that, if the camera rotates around the model, three run-length codifications must be computed corresponding to the three axes order permutations.

It should be observed that, if a unique run-length model is constructed, each voxel must belong to only one leave of the RDT. This means that the classification process must partition the data into disjoint regions or into regions enclosed one into each other following the hierarchy of the RDT. This cannot be guaranteed if the classification criterion separates the boundary voxels of each surface into different classes, since boundary voxels may be crossed by more than one isosurface. However, this problem can be avoided if the RDT classifies voxels into different groups according to the combination of surfaces that cross them. As a consequence, nodes of a tree can share descendents. Specifically, nodes representing adjacent semantic regions can share a descendent node representing the voxels crossed by the boundaries of these regions.

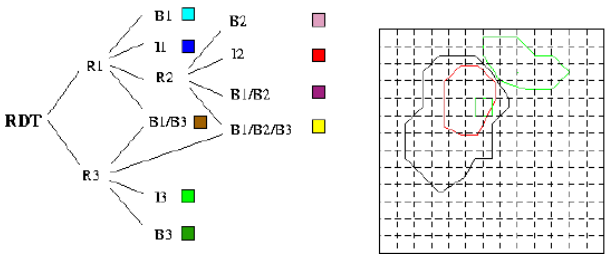


Figure 3. Example of an RDT labeling with multiple surfaces crossing voxels. The RDT tree is depicted at the left; the regions in the voxel model at the middle and the labeling of the voxels in the RL at the right.

Figure 3 illustrates such a structure. The RDT classifies the voxels into 9 regions: interior voxels (I1, I2, I3), voxels crossed by only one surface (B1, B2, B3) and voxels crossed by more than one surface (B1/B2, B1/B2/B3). The drawback of this solution is that it increases the number of classes and, therefore it may result in a higher fragmentation of the RL and thus, higher memory requirements. In order to solve this problem, instead of one RL model, several RL could be created, associated to the intermediate nodes of the RDT and traversed simultaneously. We are currently working on these types of structure, but the results shown in the next section correspond only to the former one.

Rendering

As mentioned above, our model is suitable for sorted traversals of the data. In our simulations, we have used it to render the model using the splatting strategy. The traversal of the run-length models accesses to the selected voxels, which can fall into three categories, depending on the visualization query: interior voxels, surface voxels and hybrid voxels. The former ones are purely volumetric. They are splatted according to their emission and absorption. When a surface voxel is reached, the polygons inside it are projected. It should be noted that if more than one surface crosses the voxel and the surfaces are translucide, the order of the projection of the polygons is relevant. We use the approach proposed by Kreeger and Kaufman [Kre99] to solve this problem, using the GL z-tests. Finally, for the third type of voxels, we first project the surface and next splatt the interior. As in other previous approaches, this actually composes erroneously the surface and the volume. Removing this error would require knowledge of the decomposition of the voxel into subvolumes according to the surfaces that cross it. We do not address this problem in this paper.

4. RESULTS

All the simulations have been carried out on a Sun Ultra 60 360MHz using our multimodal rendering software platform Hipo [Pui02]. For all the simulations, we have first executed a non-optimized version of the traversal algorithm using the original voxel model without RDT and RL. This rendering serves us as the unit of CPU cost. All the CPU costs shown in the tables are relative to this unit cost in order to effectively measure the improvements provided by the proposed method.

Two datasets have been used: a 32x32x32, one byte intensity phantom model and a real dataset composed of 190x220x178 MR (Magnetic Resonance) data of the brain. The phantom model is composed of two disjoint voxelized spheres. The surfaces of these spheres have been extracted using a Marching Cubes algorithm. The RDT subdivides the volume into these two regions at the first level of the tree, and into interior and boundary regions at the second level.

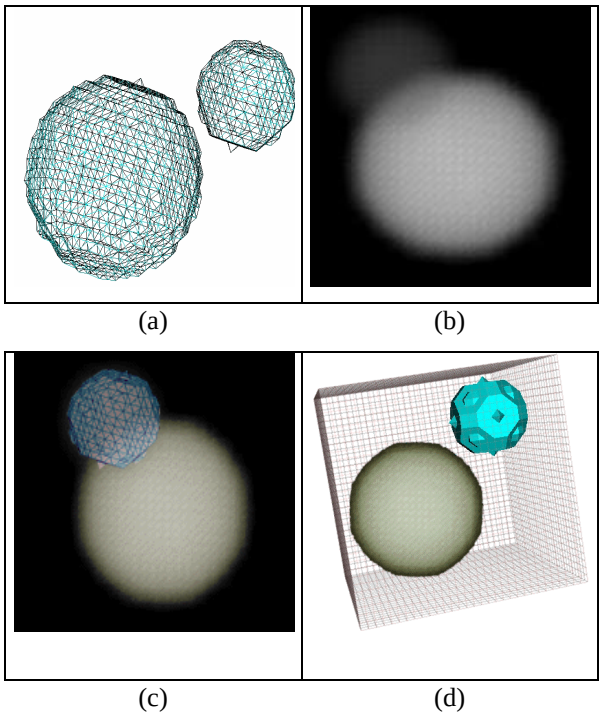


Figure 4. Different images of the phantom model: (a) Triangle meshes of the extracted fitted surfaces, (b) Volume rendering, (c) Hybrid rendering of the volume of two regions and the wireframe surface of one region, (d) Hybrid rendering of one region of the volume and the shaded surface of the other region framed into the voxel model.

Figure 4 shows several rendered images of this model, selecting either the two surfaces, the two internal regions or mixing surfaces and volume. Table 1 gives information on the size of the model, its internal regions and the number of triangles of the surfaces.

	Subtree Sphere 1	**Subtree Sphere 2**	**Global Model**
Interior voxels	2.705	437	3.142
Boundary voxels	1.440	464	1.904
Empty voxels			27.722
Surface triangles	3.704	1.352	5.056

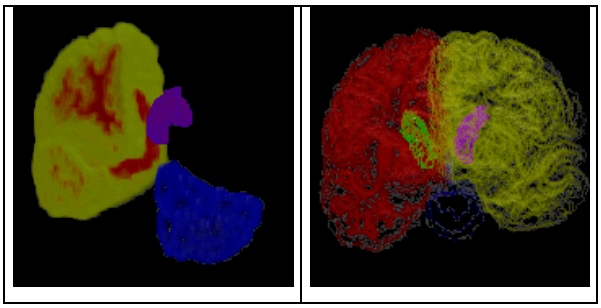
Table 1. Description of the Phantom hybrid model.

Table 2 shows the cost of different traversals of the proposed model in comparison to full traversals of the structure. The simulations correspond to the images shown in Figure 4.

Sub1	Sub2	kvol	ksur	khyb	Oc. rat	ren rat
Surface	Surface	0	1904	0	0.058	0.283
Volum	Volum	5046	0	0	0.095	0.359
Volum	Both	437	1440	464	0.071	0.325
Surface	Volum	4145	464	0	0.140	0.362

Table 2. Simulation results on the Phantom model. The first two columns indicate the branch of each subtree that has been visualized (volume, surface or both). The next three columns indicate the number of selected voxels for each category: volume voxels (kvol), surface voxels (ksur) and hybrid voxels (khyb). Column 6 indicates the relative occupancy of the selected features, i.e. kvol+ksur+khyb divided by the total occupancy of the model 32768. Column 7 shows the relative cost of the rendering in relation to the cost of the same rendering without using the proposed structure.

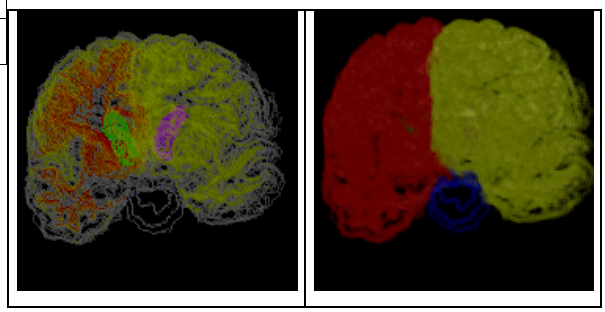
Similar simulations have been performed on the MR model of a human head as shown in Figure 5, 6 and 7. In some figures, we have rendered the regions with a constant color, and in some others we have used the gray value of the model. In the former case, it is only necessary to traverse the RL model, since it is not required to access to the actual voxel array.



(a) (b)

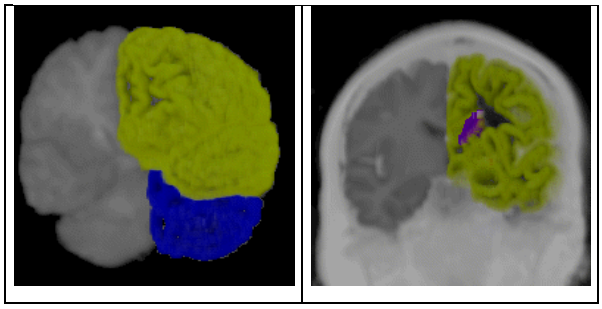
Figure 5. Rendered images of the brain model. (a) volume of the left caudate, volume of the right cerebral cortex and the right cerebral white matter and surface of the left cerebellum cortex, (b) surfaces of the left and right cerebral cortex and of the left and right caudate.

However, in order to have comparable results in the simulations, we have performed this access for all selected voxels. The RDT is composed of two main branches: the brain, which is subdivided into regions as depicted in the RDT of Figure 1 and the rest of the head. The fitted surfaces correspond to the regions labeled as: right cerebral cortex, right cerebral white matter, left and right caudate, left cerebral cortex and left and right cerebellum cortex. Table 3 shows the occupancy of these regions in terms of number of voxels and surface triangles. The simulation results are listed in Table 4.



(a) (b)

Figure 6. Rendered images of the brain model. (a) same surfaces as Figure 5b plus surface and volume of the right cerebral white matter, (b) surface and volume of the right and the left cerebral cortex and surfaces of the right and the left cerebellum cortex.



(a) (b)

Figure 7. Rendered images of the brain model. (a) volume of the right cerebral cortex and the right cerebellum cortex and surface and volume of the left cerebral cortex and the left cerebellum cortex, (b) volume of the non-brain voxels of the head and of the right cerebral cortex, volume of the left caudate and surface of the left cerebral cortex.

Region	Interior voxels	Boundary voxels	Surface triangles
Right cerebral cortex	511036	77792	382609
Right cerebral white matter	165956	103513	256691
Right caudate	1462	2693	7056
Right cerebellum cortex	60600	16410	52430
Left cerebral cortex	509588	83890	388824
Left caudate	1588	2576	6609
Left cerebellum cortex	60484	16664	53871
Non-brain head region	4423855	464540	958439

Table 3. Description of the MR labeled model of the brain.

Fig	kvol	ksur	khyb	ocup ratio	render cost ratio
5.a	879125	0	0	0.118	0.294
5.b	0	166951	0	0.224	0.194
6.a	332907	0	103513	0.058	0.291
6.b	1020624	106301	161682	0.172	0.469
7.a	1235910	0	100554	0.179	0.447
7.b	4936479	83890	0	0.674	0.867

Table 4. Simulation results for the MR head dataset. The first column indicates the corresponding figure; the three next columns show the number of volume voxels (kvol), surface voxels (ksur) and hybrid voxels (khyb); Column 5 shows the ratio of occupancy of the selected features in terms of number of selected voxels divided by the size of the model 190x220x178; Column 6 shows the ratio of cost of our implementation in relation to full traversal of the model for the same selection.

The relative cost of our method, in comparison to full traversal, ranges between 20% and 30% for the phantom model and 20% and 70% for the head model. This is an important speedup of the rendering. Taking into account that the cost of creation of the structure in relation to the basic rendering cost is 0.49, the proposed method speeds up the rendering, even for only one traversal. It should be noted that this reduction in the cost is attributable to the efficiency of the proposed traversal, since the rendering cost itself (shading, projecting and compositing in the selected voxels) is the same in our method as a in full traversal. However, the rendering cost influences the overall improvement of the method, as it is part of the total cost. This explains the variation of the cost ratio, depending on the number, type of selected voxels and number of triangles per voxels. The efficiency of our method is due to the fact that the occupancy ratios are low, at most 30% in the five first simulations on the head model, which is not a biased data, since the simulations correspond to real physician's queries. The worst efficiency is obtained in the last simulation in which almost 70% of the voxels are selected. We expect the occupancy of relevant features to be low in other applications. This observation was the primary motivation of our work.

5. CONCLUSIONS

The method proposed in this paper is aimed at speeding the traversal of hybrid classified models. The Rendering Decision Tree (RDT) together with the auxiliary Run-Length encoding (RL) of the model provides means of accessing directly to the regions and the surfaces selected for rendering avoiding unnecessary traversals of the entire model. The simulations performed show that the method can improve the efficiency of the traversal in 60 to 70 % percent. Several development stem from this work. First, we would like perform more measurements of the relative efficiency of our structure in comparison to substituting the auxiliary RL model by multiple overlapping less fragmented RL at different levels of the tree. Comparison of its efficiency with octree structures are also desirable. In addition, more work should be done to enhance the depth composition for zooming on boundary voxels crossed by one or more surfaces. Finally, we are currently investigating means of reducing the IO operations for successive rendering with a similar rendering selection.

6. ACKNOWLEDGMENTS

This work has been funded by the project MAT2002-04297-C03-02 from the Ministerio de Educación y Ciencia.

7. REFERENCES

[Boa03] Boada, I, and Navazo, I., 3D texture-based hybrid visualizations. Computers and Graphics, (27):41-49, 2003.

[Eck99] Eckel, G., OpenGL volumizer programer's guide. Document nº 0073720001, 1999.

[Fru91] Fruhauf, M., Combining volume rendering with line and surface rendering. Proceedings of Eurographics'91, pages 21-32. Elsevier Science Publisher (North Holland), 1991.

[Goa89] Goadsell, D.S., Mian, S. and Olson, A.J., Rendering volumetric data in molecular systems. Journal of Molecular Graphics, 7, March 1989.

[Goo88] Goodman, R.M. and Smyth, P. Decision tree design from a communication theory standpoint. IEEE Transactions on Information Theory, 34(5):979-994, 1988.

[Kau87] Kaufman, A. Efficient algorithms for 3D scan-conversion algorithms of parametric curves, surface and volumes. ACM Computer Graphics, 21:171-179, July 1987.

[Kau90] Kaufman, A., Yagel, R. and Cohen, D. Intermixing surface and volume rendering. 3D Imaging in Medicine: Algorithms, Systems and Applications, pp. 217-228, 1990.

[Kre99] Kreeger, K. and Kaufman, A. Mixing translucent polygons with volumes. Proc. IEEE Visualization, pp. 191-198, 1999.

[Kni01] Kniss, J., Kindlmann, G. and Hansen, C. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. Proceedings of the conference on Visualization 2001, pp. 255-262. IEEE Press, 2001

[Lac94] Lacroute, P and Levoy, M. Fast volume rendering using a Shear-Warp factorization of the viewing transformation. ACM Computer Graphics, 28(4):451-458, July 1994.

[Lev90] Levoy, M. A hybrid ray tracer for rendering polygon and volume data. IEEE Computer Graphics & Applications, 10(8):33-40, March 1990.

[Liv96] Livnat, Y., Shen, H.W. and Johnson, C.R. A near optimal isosurface extraction algorithm using the span space. IEEE Transactions on Visualization and Computer Graphics, 2(1), March 1996.

[Loc87] Lorensen, W.E. and Cline, H.E. Marching Cubes: A high resolution 3D surface construction algorithm. ACM Computer Graphics, 21(4):163-169, July 1987.

[Mey92] Meyers, D., Skinner, S. and Sloan, K. Surfaces from contours. ACM Transactions on Graphics, 11(3):228-258, 1992.

[Miy92] Miyazawa, T. and Koyadama, K. A high-speed integrated renderer for interpreting multiple 3D volume data. The Journal of Visualization and Computer Animation, 3:65-83, 1992.

[Ney98] Neyret, F. Modeling, animating and rendering complex scenes using volumetric textures. IEEE Transactions on Visualization and Computer Graphics, 4(1):55-70, 1998.

[Nom95] Noma, T. Bridging between surface rendering and volume rendering for multiresolution display. Proceedings of the 6th Eurographics Workshop on Rendering, pp 57-67. Eurographics, 1995.

[Pui02] Puig, A., Tost, D. and Ferre, M. Design of a multimodal rendering system. Proc. 7th International Fall Workshop Vision, Modeling and Visualization 2002, Greiner G., Niemann H., Ertl T., Girod B. and Seidel HP. Editors, pp 488-496, 2002.

[Sob94] Sobierajski, L.M. and Kaufman, A. Volumetric ray tracing. Proc. 1994 Symposium on Volume Visualization, october 1994.

[Sub90] Subramanian, K.R. and Fussell, D.F. Applying space subdivision techniques to volume rendering. Proc. Visualization'90, pp 150-159, 1990.

[Tos93] Tost, D., Puig, A. and Navazo, I. Visualization of mixed scenes based on volume and surface. Proc. European Workshop on Rendering, pp 281-294, 1993.

[Wil92] Wilhems, J. and Van Gelder, A. Octrees for faster isosurface generation. ACM Transactions on Graphics, 11(3):201-227, July 1992.

[Wil94] Wilhems, J. and Van Gelder, A. Multidimensional trees for controlled volume rendering and compression. Proc. ACM Symposium on Volume Visualization, 11:27-34, October 1994.