Shear-Warp Volume Rendering Algorithms Using Linear Level Octree for PC-Based Medical Simulation

Zhenlan Wang[1], Chee-Kong Chui[1], Chuan-Heng Ang[2], Wieslaw L. Nowinski[1]

[1] Biomedical Imaging Lab, Singapore
zhenlan@lit.a-star.edu.sg
[2] School of Computing, National University of Singapore, Singapore

Abstract. We describe a new algorithm for 3D, and potentially higher, volume rendering. This algorithm takes advantage of the shear-warp factorization technique to reduce matrix computing and a hierarchical data structure for better utilization of spatial coherence. The algorithm represents an improvement over existing algorithms in terms of performance, with little compromise in image quality. The algorithm benefits 3D volume rendering as well as multi-modality volume rendering. Initial work on comparison and validation of algorithm are done. The results and analysis show that the algorithm is promising. Our preliminary work in using this method for medical simulation is also discussed. This method is particularly suitable for PC-based medical simulation where the computation load is high and the raw computing power is limited.

1 Introduction

Approximately eighty percents of all information perceived by human is through the eyes, while the visual system of humans is the most complex of all sensory modalities [1]. In medicine, visual information plays an essential role for accurate diagnosis and effective therapy planning. Medical images such as computed tomography (CT), magnetic resonance imaging (MRI) and nuclear medicine imaging are increasingly used in medical simulation for pre-treatment planning and outcome prediction. There is a demand for medical simulation to be executed on cost effective desktop workstations. The research work reported here represents our effort to develop efficient visualization solutions for PC-based medical simulation systems [2, 3].

In medical image applications, direct volume rendering is considered superior over surface rendering [4]. There are two broad categories of volume rendering algorithms, known as object and image space methods. The shear-warp algorithm [5] is widely regarded as the fastest volume rendering method to date. There are various methods to speedup rendering by using parallel/distributed approaches [6], hardware acceleration [7], and texture mapping techniques [8]. 4D rendering is relatively new, particularly in medicine. The recent advances including dynamic MRI and multi-modality imaging make new demands for highly sophisticated and efficient visualization techniques. Visualization of an actual procedure with patient-specific 4D volume datasets and visualization of multi-modal datasets of the same patient is critical for accurate simulation with high confidence.

Existing techniques lack the provision of high speed and low cost multi-modality volume renderer as well as 4D rendering that will be important in future medical simulation. For example, VolumePro [7] and texture mapping are restricted to a single dataset (single modality). Texture mapping does not support extensive shading.

A new volume rendering algorithm called LLO-based shear-warp algorithm is presented in this paper. Both suitable data structure and efficient algorithm are explored to achieve fast 3D volume visualization of medical images for simulation on a standard PC. The algorithm takes advantages of both the shear-warp factorization technique and the hierarchical data structure. In contrast to the conventional hierarchical data structure, the algorithm avoid the hierarchical traversal during the rendering. The volume data are encoded so that only the 3D regions containing imaging information need to be accessed and processed. The regions of both low presence and low variation can be visualized efficiently.

The paper is organized as follows. Data structure and algorithm are in sections 2 and 3, respectively. Initial results are in section 4. The application of our method for medical simulation is discussed in section 5, followed by the conclusion.

2 Linear Level Octree (LLO)

2.1 Data Structure

Octree can be labeled with different schemes. Linear Level Octree (LLO) [9], which is extended from linear octree scheme, is a label scheme with many advantages. LLO inherits all the advantages of linear octree. For instance, only leaf nodes are stored, which greatly reduces not only memory consumption but also, more essentially, the nodes that are required to be processed. It implicitly encodes the location, the size of the nodes, and the path from the root to the nodes. Therefore, it is easier to estimate the relations between arbitrary spatial points and LLO nodes with lower computing cost. More benefits from LLO will be mentioned in the description of the algorithm.

A volume with size $2^n \times 2^n \times 2^n$ is put into a coordinate system, where n is the resolution of the raster. The back-bottom-left corner of the volume is located at the origin, and its edges are aligned/overlapped with the coordinate axes respectively.

Each octant is labeled with a unique *code key*: (L_i, x_i, y_i, z_i), where L_i is the level of the node, and x_i, y_i, and z_i are the x, y, z level-coordinates of the node respectively. The code key of the root node is defined as (0, 0, 0, 0).

Assume an arbitrary node A at level L_i has code key (L_i, x_i, y_i, z_i). Then, the back-bottom-left subnode of A at level $L_i + ! L$ ($! L = 1, 2, \ldots$) has code key $(L_{i0}, x_{i0}, y_{i0}, z_{i0})$, where $L_{i0} = L_i + ! L$, $x_{i0} = x_i \bullet 2^{! L}$, $y_{i0} = y_i \bullet 2^{! L}$ and $z_{i0} = z_i \bullet 2^{! L}$. The code keys of A's other subnodes at level $L_i + ! L$ are given as follows.

$$\begin{array}{l} L_{i0}, x_{i0}, y_{i0}, z_{i0} \\ L_{i1}, x_{i1}, y_{i1}, z_{i1} \\ L_{i2}, x_{i2}, y_{i2}, z_{i2} \\ L_{i3}, x_{i3}, y_{i3}, z_{i3} \\ L_{i4}, x_{i4}, y_{i4}, z_{i4} \\ L_{i5}, x_{i5}, y_{i5}, z_{i5} \\ L_{i6}, x_{i6}, y_{i6}, z_{i6} \\ L_{i7}, x_{i7}, y_{i7}, z_{i7} \end{array} = (L_i, x_i, y_i, z_i) + \begin{array}{l} 0, 0, 0, 0 \\ 0, 1, 0, 0 \\ 0, 0, 1, 0 \\ 0, 1, 1, 0 \\ 0, 0, 0, 1 \\ 0, 1, 0, 1 \\ 0, 0, 1, 1 \\ 0, 1, 1, 1 \end{array}$$

We define the distance between two adjacent voxels as one unit. By exploiting the label scheme of LLO, we can infer the following properties of an arbitrary octant node A with code key (L, x, y, z):

- Location = $(x \bullet 2^{!L}, y \bullet 2^{!L}, z \bullet 2^{!L})$, where ! L = n – L.
  This is the absolute location coordinate (not level coordinate) of A's back-bottom-left corner voxel in the coordinate system. Normally, it is also regarded as the location of node A.
- Size = $2^{!L}$, where ! L = n – L.
  This is the side length of node A, or in other words, it is the number of voxels contained in one dimension of A.
- The code key of A's upper level (ancestor level), say level L_w ($L_w < L$), is:
  $(L_w, x/2^{!L}, y/2^{!L}, z/2^{!L})$, where ! L = L – L_w.

2.2 Conversion of Volume to LLO

To take advantage of the spatial coherency in the volume, LLO is employed in our algorithm. Before rendering, the classified volume is converted into the representation of LLO. We defined the following leaf criteria for LLO generation:

- The smallest octant is of size $2 \times 2 \times 2$, i.e., each dimension of the octant contains 2 voxels at least.
- Let *Max* and *Min* be the maximum and minimum value/intensity of all the voxels contained in an octant, respectively. The equation below must be satisfied:
  Max – Min " T, where T is a user predefined threshold (T # 0).
- If all the voxels contained in an octant are transparent, the octant is a "white" node and it will not be stored. Otherwise, it is a "black" node, and stored.

Besides the leaf node criteria, information saved in each leaf node (i.e. octant data structure) is also an important factor for efficient volume rending. We have the following information saved with the leaf nodes.

- Code key of each leaf node (L_i, x_i, y_i, z_i).
- Eight vertex voxels of the leaf node.

We have the eight voxels located at the corners of the octant, say vertex voxels, stored with the leaf node. It is based on the observation that since a group of neighboring voxels can be organized into a leaf octant, their value must be homogeneous (variation under a threshold value). Eight vertex voxels should be enough to represent the sub-volume. Thus, with the leaf node criteria and octant data structure, the whole volume is encoded into LLO and the original volume dataset is not needed any longer.

3 LLO-Based Shear-Warp Algorithm

3.1 Rendering Pipeline

Figure 1 shows the pipeline of our LLO-based shear-warp algorithm. An LLO can be either encoded from volume data or directly loaded from the storage media where an LLO is encoded and previously saved. The rendering procedure is enclosed in the dash-line square. After the LLO is built, octants are output one by one according to the current view direction. A shear-warp renderer is employed for octant rendering. The contribution of each octant is composited to an intermediate image. After all the octants are processed, the intermediate image is swapped for generation of the final image.

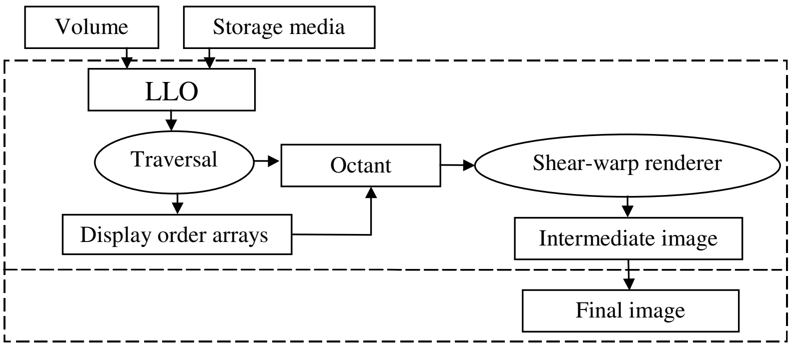


Fig. 1. Rendering pipeline of the LLO-based shear-warp algorithm

3.2 Display Order and Traversal of LLO

In this algorithm, we are taking advantage of an object-order approach to fulfill the image-order volume rendering. Octants instead of voxels become the primitive visualization elements, and the number of nonempty octants is far less than the number of voxels. Because the empty regions do not need to be processed or even checked, processing time can be reduced remarkably. It is realized based on the fact that, for a given set of parallel viewing directions, the octants can be processed in specific order without affecting the result image, and there are only finite (eight) sets of such viewing directions.

To distinguish the child nodes of a parent, a distinctive number is distributed to each of them as showed in Figure 2. A vector, say (V_x, V_y, V_z,), is used to represent view directions. Thus, display orders of nodes from different view directions are given in Table 1. The underlined node-numbers in same segments imply they have same display priority so that they can be processed in different orders without affecting the result image.

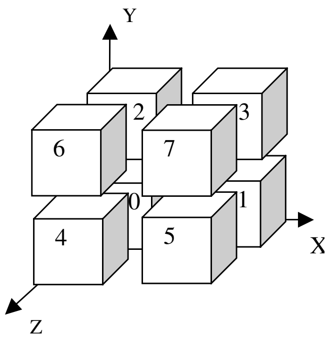


Fig. 2. Code of child nodes

(V_x, V_y, V_z,)	Display Orders
(<0, <0, <0)	7, 3, 5, 6, 1, 2, 4, 0
(>0, <0, <0)	6, 2, 4, 7, 0, 3, 5, 1
(<0, >0, <0)	5, 1, 4, 7, 0, 3, 6, 2
(>0, >0, <0)	4, 0, 5, 6, 1, 2, 7, 3
(<0, <0, >0)	3, 1, 2, 7, 0, 5, 6, 4
(>0, <0, >0)	2, 0, 3, 6, 1, 4, 7, 5
(<0, >0, >0)	1, 0, 3, 5, 2, 4, 7, 6
(>0, >0, >0)	0, 1, 2, 4, 3, 5, 6, 7

Table 1. Display orders of nodes

With the node-display-order table, the volume data encoded by LLO can be traversed easily with a recursive algorithm. By further exploiting Table 1, we find that there are only four distinctive display orders, while others are just in inverse order. Therefore before rendering, the LLO can be traversed once for each of the four distinctive view directions, and the pointers of the leaf nodes are stored in four different arrays in order. Thus during rendering, it is not necessary to traverse the LLO any longer. According to the current view direction, one of the four display order arrays is selected, and each octant is visualized from the beginning to the end or inverse in the array (Figure 1). Since the traversal of the hierarchical data structure is the most time consuming operation and it is also regarded as the major disadvantage of hierarchical data structure [10], a substantial reduction of computing cost is achieved by this improvement.

3.3 Visualization of Octants with Shear-Warp Rendering

The share-warp algorithm is used for octant rendering. Octants are transformed into the sheared object space. Given the transformation of the parent node, the transformation of its sub-octant can be efficiently computed. In the sheared space, the octant is projected onto an intermediate image plane, which composite projections of all the octants. The intermediate image is run length encoded so that opaque pixels can be skipped quickly. However, run length encoding of the volume is not needed as only non-transparent sub-volumes are passed to the renderer.

The shaded samples are interpolated in the octant. Since all the samples are interpolated using the same eight vertices in one octant, computing cost can be greatly reduced by incremental computation. For efficient gradient interpolation, the gradients for all eight vertices of each octant are computed and saved with each octant at a preprocessing stage.

After all the octants have been processed, the intermediate image can be warped into the final image with an inexpensive bilinear filter. The rendering procedure is completed.

3.4 Multi-modality Visualization

Our LLO-based shear-warp algorithm is extended to render multiple modalities simultaneously. Octree is very efficient in set-theoretic operations. If multiple modalities have been encoded into LLO, they can be easily integrated by simple octree “OR” operations. Thus, while multiple modalities are integrated into one LLO, it can be easily visualized with the algorithm discussed in the previous section. The main advantage of this method is memory saving because there are no additional volume modalities existed.

Multiple modalities could be integrated either at the data pre-processing stage, the rendering stage, or post-processing composing stage. Therefore, an integration criterion, referred to as integration function, must be defined. An integration function in rendering stage can be defined as:

$$S_I = \alpha \cdot S_A + \beta \cdot S_B$$

where S_A and S_B are samples drawn from volume data set A and B, respectively, at the same virtual position. S_I is the final integrated sample value. ! and ″ are the integration factors. A lookup table as shown in Table 2 can be maintained for efficient multi-modality integration.

Table 2. Integration factor lookup table

Intensity value	!	″
1	0.6	0.7
2	0.2	0.5
…	…	…

Different samples according to their intensity values are weighted differently so that multiple structures in different modalities can be flexibly classified and visualized together without confusion. In this method, boundary check is necessary, because the integrated sample value may exceed the permitted maximum value.

4 Results and Discussion

We compared the performance of our LLO-based ray-casting renderer with that of the conventional ray-casting algorithm. We observed significant speedup in comparison with the latter and almost no change in image quality. The shear-warp algorithm is under implementation and we expect to have even better results.

Our LLO-based ray-casting renderer has achieved average 3.8 times of speed acceleration with comparable image quality compared to the conventional ray-casting method, on 4 CT, MRI, and rotational angiography datasets shown in Table 3. The results are obtained on a PC with 1.4GHz Pentium III CPU and 512MB physical memory. The rendered image has resolution of 256×256. We use a homogeneous threshold of 10 for LLO generation.

From Table 3, the smallest dataset brain vessel did not achieve the best speedup, whereas CT Head – Skull and MR Brain datasets achieved the best performance. There are two reasons. Firstly, the brain vessel data are most semi-transparent, while CT Head Skull and MR Brain datasets have opaque surfaces. So early ray termination due to the opacity of these surfaces has accelerated the speed of ray-casting algorithm on the latter two datasets. Secondly, the non-transparent voxels of the brain vessel dataset scatter in the volume space, while the non-transparent voxels of CT Head Skull and MR Brain datasets are scatter in small regions. There are more large empty octants in the LLOs of latter two datasets that help the volume rendering skip space more quickly and efficiently. On the other hand, because the brain vessel dataset consists of many small octants, it gains very good image quality. We also found that both the highest and the lowest speedup performance are acquired from the same dataset CT Head. By further manipulation of the dataset, we can found that there is a large noisy block located exactly behind of the head. In the Face image, the noisy block reduced the performance significantly. On the contrary, the noisy block is filtered in the Skull image, so the left large empty space encoded by LLO benefit the performance significantly.

Therefore, the acceleration performance of LLO for volume rendering is often dataset-dependent, and the LLO-based rendered images are similar to the images rendered by brute force ray-casting. The preliminary results achieved so far do not demonstrate the full capability of the method. Therefore, better performance can be expected in the future.

Table 3. Rendering timings and speedup performance

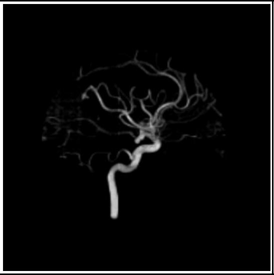
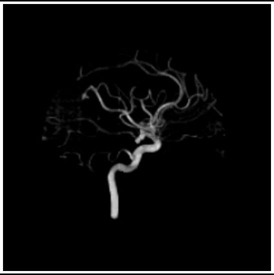
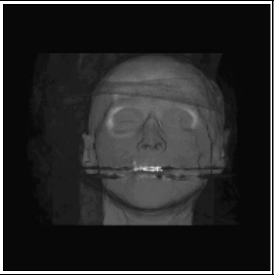
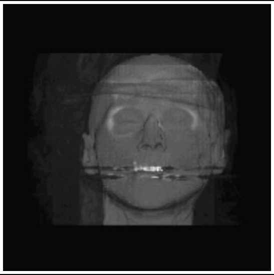
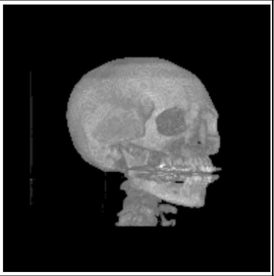
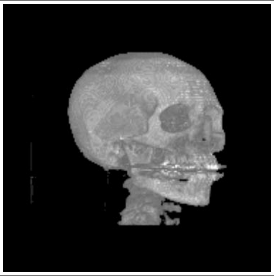
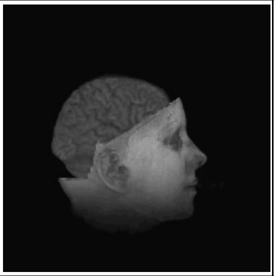
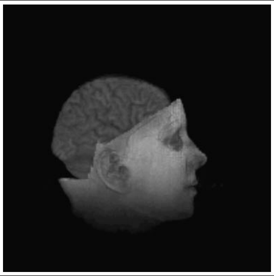
Dataset	Brute force ray-casting	LLO-based ray-casting	Speedup
Brain Vasculature (128 x 128 x 128)			3.2
CT Head – Face (256 x 256 x 113)			2.9
CT Head – Skull (256 x 256 x 113)			4.9
MR Brain (256 x 256 x 109)			4.2

Figure 3 demonstrates our LLO-based ray-casting algorithm for multi-modality rendering. The two datasets used are VHD Male and a patient's cerebral angiography. The former has resolution of 256 x 256 and involved a total of 85 slices with 5 mm inter-slice gap, and is acquired by multi-slice CT scanner. The latter is a reconstructed rotational x-ray angiography (XRA) data from GE. The CT images and reconstructed XRA are registered and visualized on a PC.

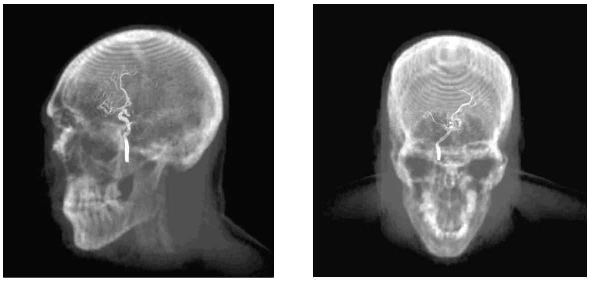
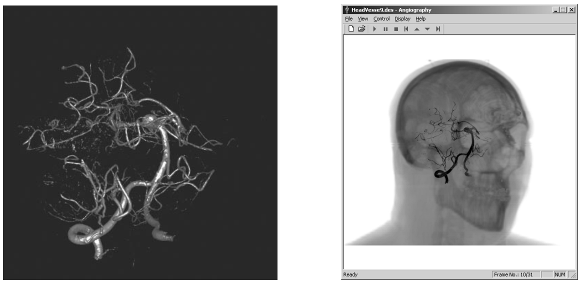


Fig. 3. Multi-modality rendering

5 Volume Rendering in Medical Simulation

The LLO-based volume renderer is part of our effort in developing PC-based interactive medical simulator [11, 12]. Figure 4 shows volume rendered images of cerebral vasculature and the GUI of a system that is developed to register the volume rendered images of the same patient vascular with the volume rendered CT images of the VHD Male dataset.



(a) **(b)**

Fig. 4. PC-based medical simulator:(a) Cerebral vessels; (b) Cerebral vessels registered with VHD head rendered as a fluoroscopic image

6 Conclusion

In this paper, we present our initial work on a new shear-warp algorithm based on spatial data structure. The proposed algorithm has several advantages. Firstly, because of the employment of a shear-warp renderer, each octant is considered only once during rendering, and only the relevant octant is considered each time. Secondly, the algorithm takes advantage of 3D coherency of the volume data rather than 1D coherency in pure shear-warp algorithm so that in 3D regions of both low presence and low variation, the algorithm can work efficiently. Additionally, because the samples in the same octant are computed by incremental interpolation among the same set of vertices, supersampling in three directions (x, y and z) can be performed with little cost compared with pure shear-warp algorithm. Finally, this method only keeps "black" octants of one LLO instead of 3 copies of RLE-encoded data as in shear-warp algorithm. We plan to have this new LLO-based shear-warp algorithm to form the basis for multi-modal and 4D volume visualization in our PC-based simulator.

Acknowledgements

Support of this research development by Agency of Science, Technology and Research is gratefully acknowledged.

References

1. Demiris, A. Mayer, H.P. Meinzer, 3-D Visualization in Medicine: An Overview, Contemporary Perspectives in Three-Dimensional Biomedical Imaging, C. Roux and J.-L. Coatrieux (Eds.) IOS Press, 1997, pp. 79-105.
2. J.H. Anderson, C.K. Chui, Y. Cai, Y. Wang, Z. Li, X. Ma, W.L. Nowinski, M. Solaiyappan, K. Murphy, A.C. Venbrux and P. Gailloud, Virtual reality training in interventional radiology, to appear in *Seminars in Interventional Radiology*, Thieme Medical, 2002.
3. Z. Li, C.K. Chui, J.H. Anderson, X. Chen, X. Ma, W. Huai, Q. Peng, Y. Cai, Y. Wang and W.L. Nowinski, Computer environment for interventional neuroradiology procedures, *Simulation and Gaming*, Vol. 32, No. 3, September 2001, pp. 405-420.
4. T.T. Elvins, A Survey of Algorithms for Volume Visualization, *Computer Graphics*, 26(3), 1992, pp. 194-201.
5. P. Lacroute, and M. Levoy, Fast volume rendering using a shear-warp factorization of the viewing transformation, *SINGGRAPH'94*, 1994, pp. 451-458.
6. W.L. Nowinski, A hybrid parallel ray caster for medical imaging, *Mathematical Research*, 1994, pp. 305-316.
7. H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, The VolumePro real-time ray-casting system, *Proc. SIGGRAPH'99*, 1999, pp. 251-260.
8. M. Meissner, U. Hoffmann, and W. Strasser, Enabling Classification and Shading for 3D Texture Mapping based Volume Rendering using OpenGL and Extensions, *IEEE Visualization*, 1999, pp. 207-214.
9. C.K. Chui, Z.M.Yin, and R.B. Shu, K.F. Loe, An efficient algorithm for volume display by linear level octree, *Proceedings of Seminar on Computer Graphics DISCS/NUS*, National University of Singapore, Singapore, November, 1991, pp. 46-62.
10. R. Yagel, Efficient Techniques for Volume Rendering of Scalar Fields, *Data Visualization Techniques*, Edited by C. Bajaj, Chichester: Wiley, 1999, pp.15-30.
11. C.K. Chui, Z. Li, J.H. Anderson, K. Murrphy, A. Venbrux, X. Ma, Z. Wang, P. Gailloud, Y. Cai, Y. Wang and W.L. Nowinski, Training and Planning of Interventional Neuroradiology Procedures - Initial Clinical Validation *Proceedings of 10th Annual Medicine Meets Virtual Reality Conference (MMVR 2002)*, Newport Beach, USA, February 2002, pp. 96-102.
12. W.L. Nowinski and C.K. Chui, Simulation of interventional neuroradiology procedures, Proc. Medical Imaging and Augmented Reality (MIAR 2001), Hong Kong, June 2001, pp. 87-94.