Improving Efficiency for CUDA-based Volume Rendering by Combining Segmentation and Modified Sampling Strategies

Dun ZHOU*, Hongwei Du*, Feng Zhao**, Hongxing Kan*, Geng Li*, and Bensheng Qiu*

* Center for Biomedical Engineering, University of Science and Technology of China, Hefei, Anhui 230027, China.
** School of Computer Science, University of Lincoln, Lincoln, LN6 7TS, United Kingdom.

***Abstract* — The objective of this paper is to present a speed-up method to improve the rendering speed of ray casting at the same time obtaining high-quality images. Ray casting is the most commonly used volume rendering algorithm, and suitable for parallel processing. In order to improve the efficiency of parallel processing, the latest platform-Compute Unified Device Architecture (CUDA) is used. The speed-up method uses improved workload allocation and sampling strategies according to CUDA features. To implement this method, the optimal number of segments of each ray is dynamically selected based on the change of the corresponding visual angle, and each segment is processed by a distinct thread processor. In addition, for each segment, we apply different sampling quantity and density according to the distance weight. Rendering speed results show that our method achieves an average 70% improvement in terms of speed, and even 145% increase in some special cases, compared to conventional ray casting on Graphics Processing Unit (GPU). Speed-up ratio shows that this method can effectively improve the factors that influence efficiency of rendering. Excellent rendering performance makes this method contribute to real-time 3-D reconstruction.**

Keywords - parallel processing; CUDA; ray segmentation; workload balance; distance weight

I. INTRODUCTION

Volume rendering [1] is an important branch of image visualization which is usually used in medical image visualization [2,3], meteorological analysis [4,5], molecular modeling [6], and so on. Among all the rendering algorithms, ray casting [7] stands out because of its simple principle and good image quality. The principle of ray casting makes it very suitable for parallel processing, for the reason that each ray can be independent and unrelated from others. K. L. Ma et al. [8] utilized computer clusters to realize the parallel processing in 1990s, they divide the original dataset into many sub-blocks, but it still takes tens of seconds. A similar distributed rendering method is accomplished using clusters of personal computers [9]. But when using computer clusters, communication capacity between computers is extremely limited, what's more, the connection between borders of sub-blocks needs extra data loading, leading to extra time consumption.

GPU (graphics processing unit) can solve the problems mentioned above, because it owns hundreds of processing cores and provides hardware interpolation [10] functions. More and more volume renderings on GPU appear after the emergence of GPU programming [11-14]. Klar O et al. [15] propose an improved parallel processing method in 2006 by dividing a volume dataset into parts, which are dynamically swapped in and out, rendering speed can reach over 10 fps (frames per second). Similar improvements are proposed by others [13,16]. But if the data is large enough that can't be loaded in the memory of a single GPU, multi-GPU approach is widely used [17].

Workload reduction is another effective method to improve rendering speed. One way is to reduce the sampling number directly, like empty-space skipping [18], early ray termination [19]and multiresolution algorithm [20,21], but the texture organization is complicated in multiresolution method, and the multidimensional transfer function design is hard [22], in addition, boundary fusion between different resolutions needs extra processing [23]. Another way is value estimation, for example, Zhang et al. [24] presents a strategy using cubic B-spline method to estimate sampling values from neighboring samples with rendering speed 17 to 34 fps, and T. L. Falch et al. [25] directly use scattered point data to estimate all the sampling values of volume dataset. Although workload reduction can realize speed-up, but in some cases, insufficient sampling rates may degrade image quality.

Until now, CUDA [10] has become the most widely used parallel processing platform on GPU. In addition to using powerful parallel processing performance provided by CUDA to accelerate the speed [26-28], researchers begin to study the effect of inherent features of CUDA on rendering performance. Sugimoto et al. [29] study the impact of TB (thread block) shape and texture organization on volume rendering speed, and they proposed a method which can dynamically determine the optimal TB shape to increase rendering speed. In this paper, we propose an improvement method to promote the rendering efficiency of ray casting implemented on CUDA. The method can not only improve the degree of parallel processing, but also reduce the workload, which will be helpful for high quality visualization and real-time 3-D reconstruction [26,30,31].

II. METHOD IMPROVEMENT

Figure 1a illustrates the geometry of ray casting. Let V be the volume to be rendered which is a cubic of $X \times Y \times Z$ voxels (X, Y, Z represent length, width, and height, respectively), and (x, y, z) be the voxel coordinates. Pixel value accumulation is done as follow,

$$S(u,v)=\sum_{i=1}^{n}(\partial(e_i)*C(e_i)*\prod_{j=0}^{i-1}(1-\partial(e_j))) \quad (1)$$

where e_i represents the i^{th} penetrated voxel along ray R, n denotes the total number of penetrated voxels, $C(e_i)$ and $\partial(e_i)$ are the color and opacity values, respectively. Moreover, this accumulation procedure could be divided into two parts, one from samples 1 to k (0 < k < n), the other from samples k+1 to n, shown as

$$\begin{aligned} S(u,v)=&\sum_{i=1}^{k}(\partial(e_i)*C(e_i)*\prod_{j=0}^{i-1}(1-\partial(e_j))) \\ &+\sum_{i=k+1}^{n}(\partial(e_i)*C(e_i)*\prod_{j=k}^{i-1}(1-\partial(e_j)))*(\prod_{j=0}^{k-1}(1-\partial(e_j))) \end{aligned} \quad (2)$$

Figure 1b illustrates the situation when divided into three parts, in which three segments are represented by three different colors. After accumulation operation, each segment becomes a voxel, in this way, segmentation method changes into nested ray casting.

Theoretically, parallelism is proportional to the number of segments. However, more segments also cost more time inevitably during intermediate results merging period, therefore, we propose a method which automatically determines the number of segments according to rotation angle illustrated in Figure 2, where the cube represents volume dataset, the top row is for rotations about the z-axis, and the bottom row stands for pattern about arbitrary angles. The number in Fig. 2 represents the number of segments of respective rays.

Length calculation is needed to determine the optimal number more accurately, as shown in Fig. 3a. The longest one is r3 (length of diagonal), which is calculated in advance according to dataset information on CPU, usually proportional to the size of dataset. The number of segments on each ray is calculated on GPU by ray length. We define the thresholds as 0.5×r3 and 0.8×r3, when ray length larger than 0.8×r3, the ray is split into three segments, like r3 in Fig .3b; when ray length larger than 0.5×r3 but smaller than 0.8×r3, the ray is split into two segments, like r2 in Fig .3b; when ray length smaller than 0.5×r3, no segmentation is necessary, like r1 in Fig .3b. All rays are split equally according to the number, see more in Fig .3c. After segmentation, each segment of is processed by an individual thread in the same TB, as shown in Fig .3b. In this way, no matter how visual angle changes, corresponding optimal segmentation pattern can be determined automatically.

To take advantage of the segmentation method, we adopt distance-weight sampling strategy, i.e., for every segment, its depth determines its sampling density, as shown in Figs. 3b r1, r2 and r3. Besides, both starting and ending locations of each segment should be recorded to solve the boundary sampling problem, sampling begins at the starting location (e.g., points a and b in Fig. 3c) and ends before the ending location to avoid overlapping. This sampling strategy has the advantage of reducing the amount of sampling at the same time avoiding image distortion. Empty space skipping and early ray termination algorithms can also be added into this strategy to further improve performance.

III. MAIN RESULTS

Experiments are conducted on a desktop PC with a GeForce GTX 980 graphics card, which has 2048 CUDA cores and 1.28 GHz clock rate. The machine runs on Windows 7 and CUDA 6.5 platform. For comparison, we implement conventional ray casting as the control group, each thread is responsible for handling one ray and the sampling interval is fixed. The experiments are carried out on six datasets, namely bucky (32×32×32, resolution is 32×32, 32 slices), spinal vascular (125×154×145), aneurysm (256×256×256), foot (256×256×256), head (256×256×256) and skull (256×256×256). bucky datasets is supplied by "The Volume Libraty" (http://www9.informatik.uni-erlangen.de/External/vollib/), all the others are downloaded for free from the "Volren" (http://www.volren.org). The display window is composed of 512×512 pixels, i.e., W = H = 512.

Figure 4 shows the image quality of our proposed method, it can be seen that even though the number of sampling has decreased caused by distance-weight sampling strategy, the image quality is still visually acceptable.

A table about PSNR (Peak Signal to Noise Ratio) values is shown in Table 1, in which PSNR values of six datasets from three different viewing angles are recorded. In most cases, PSNR values are above 40, which means the image quality of our method is almost equal to that of original ray casting method.

Previous research has demonstrated that the TB shape influences the rendering performance [29], we thus adopt variable TB shapes and rotation angles in both the experimental and control groups. The TB shapes vary from 2×128 to 128×2 (W×H). For rotation angle, we use x-rotation and z-rotation, ranging from 0 to 180 degrees. In this way, we can evaluate the rendering speed in different workload distributions.

Figure 5 illustrates the significant improvement of rendering speed (fps) using our method, no matter what the TB shapes or visual angles are. In these experiments, we use 'head' dataset (Fig .4d). Figures 5a, b, c, d are control group, where Figs. 5a, b depict the results of convent-

Table 1. Results of PSNR/dB

Image Name	Front View	Top View	Side View
	PSNR/dB	PSNR/dB	PSNR/dB
Aneurysm	50.798	44.340	47.846
Spinal Vascular	43.010	47.225	39.645
Foot	47.586	49.607	41.364
Head	42.321	35.839	40.894
Skull	44.534	42.361	40.412
Bucky	41.648	41.436	42.814

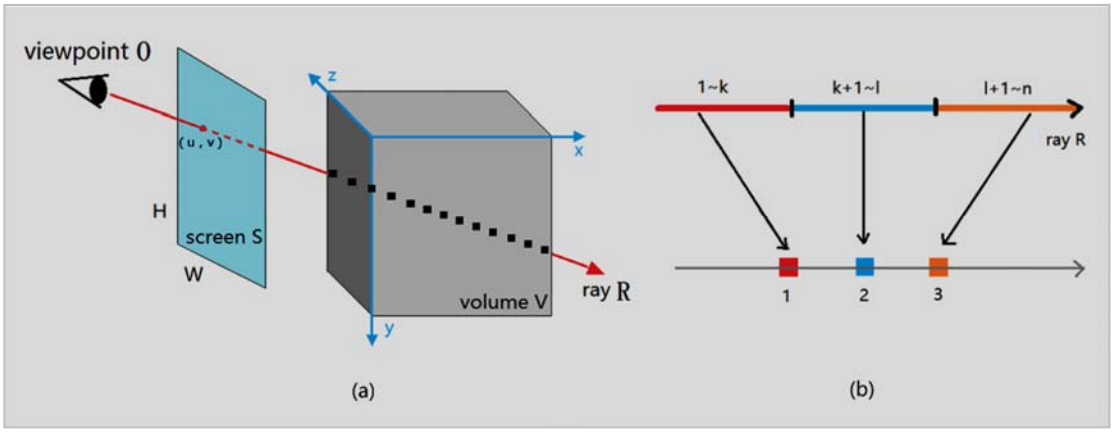

Figure 1. Illustration of (a) ray casting and (b) segmentation method

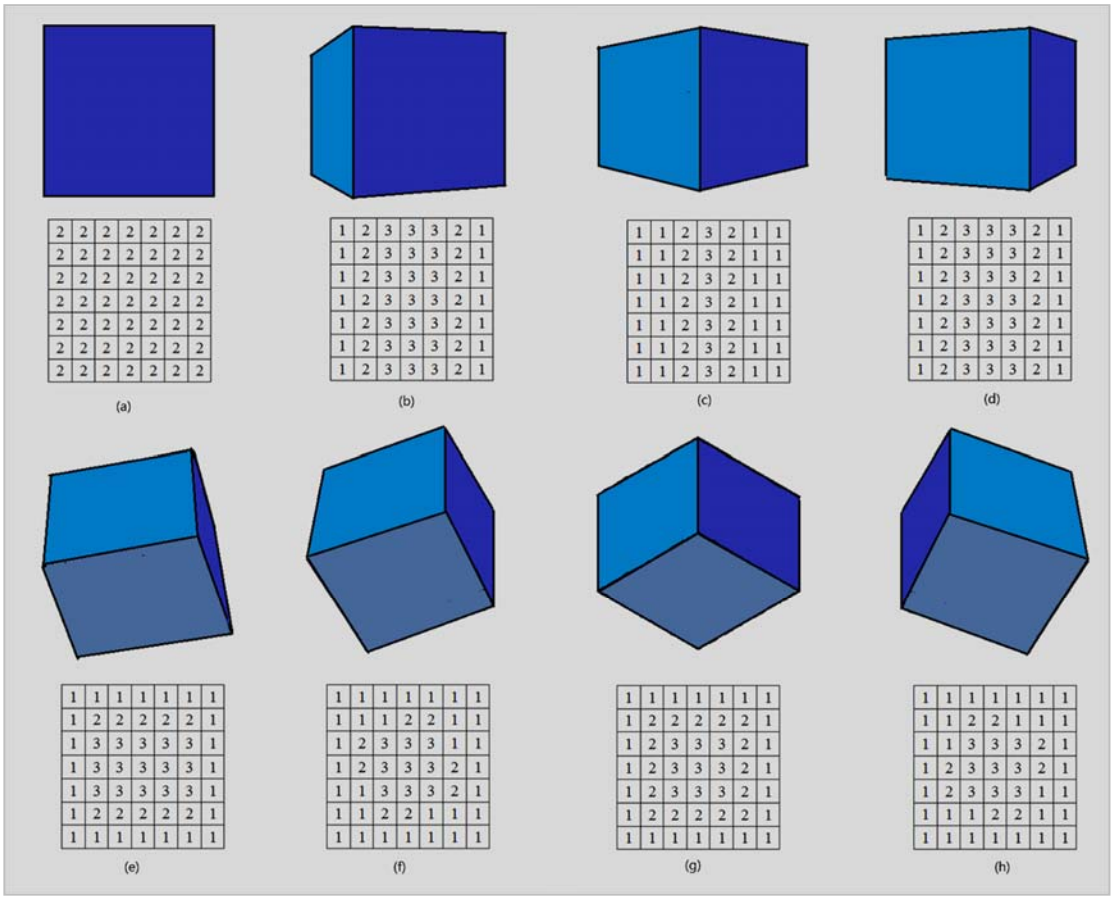

Figure 2. Automatic segmentation patterns based on rotation angles

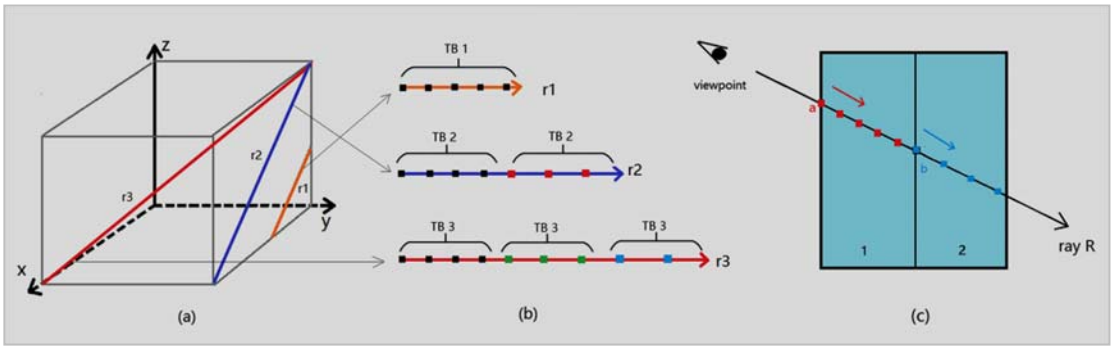

Figure 3. Illustration of automatic segmentation method

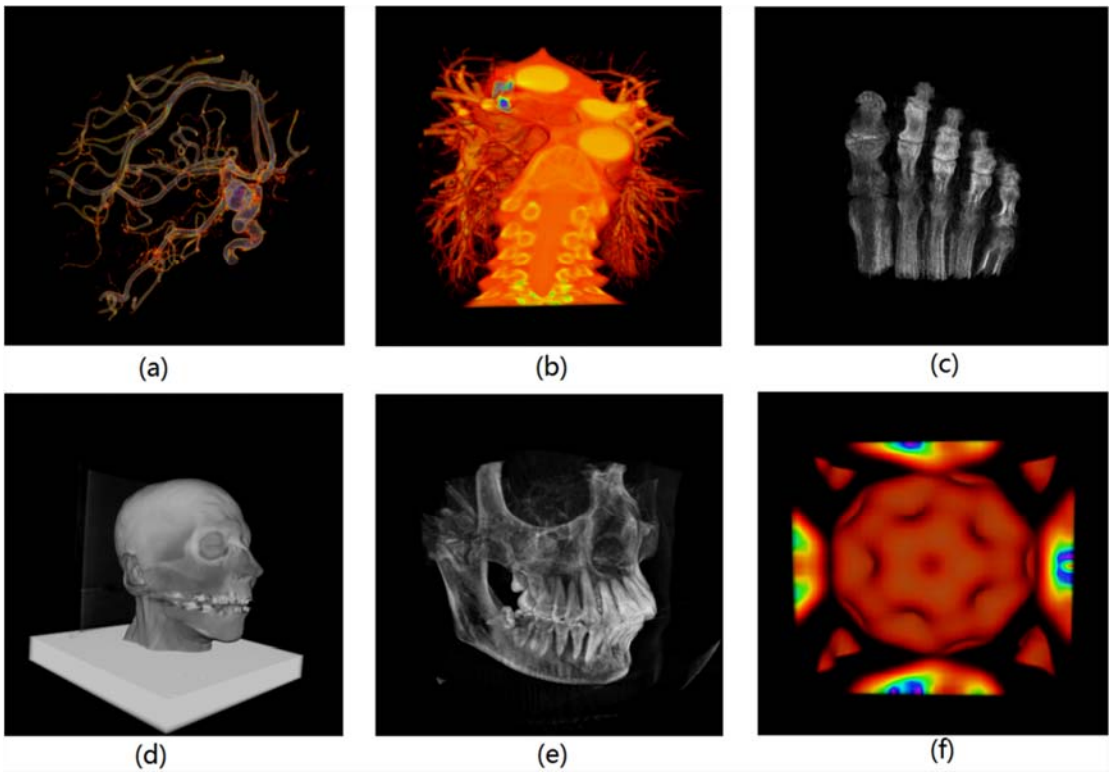

Figure 4. Image quality results of (a) aneurysm; (b) spinal vascular; (c) foot; (d) head; (e) skull and (f) bucky

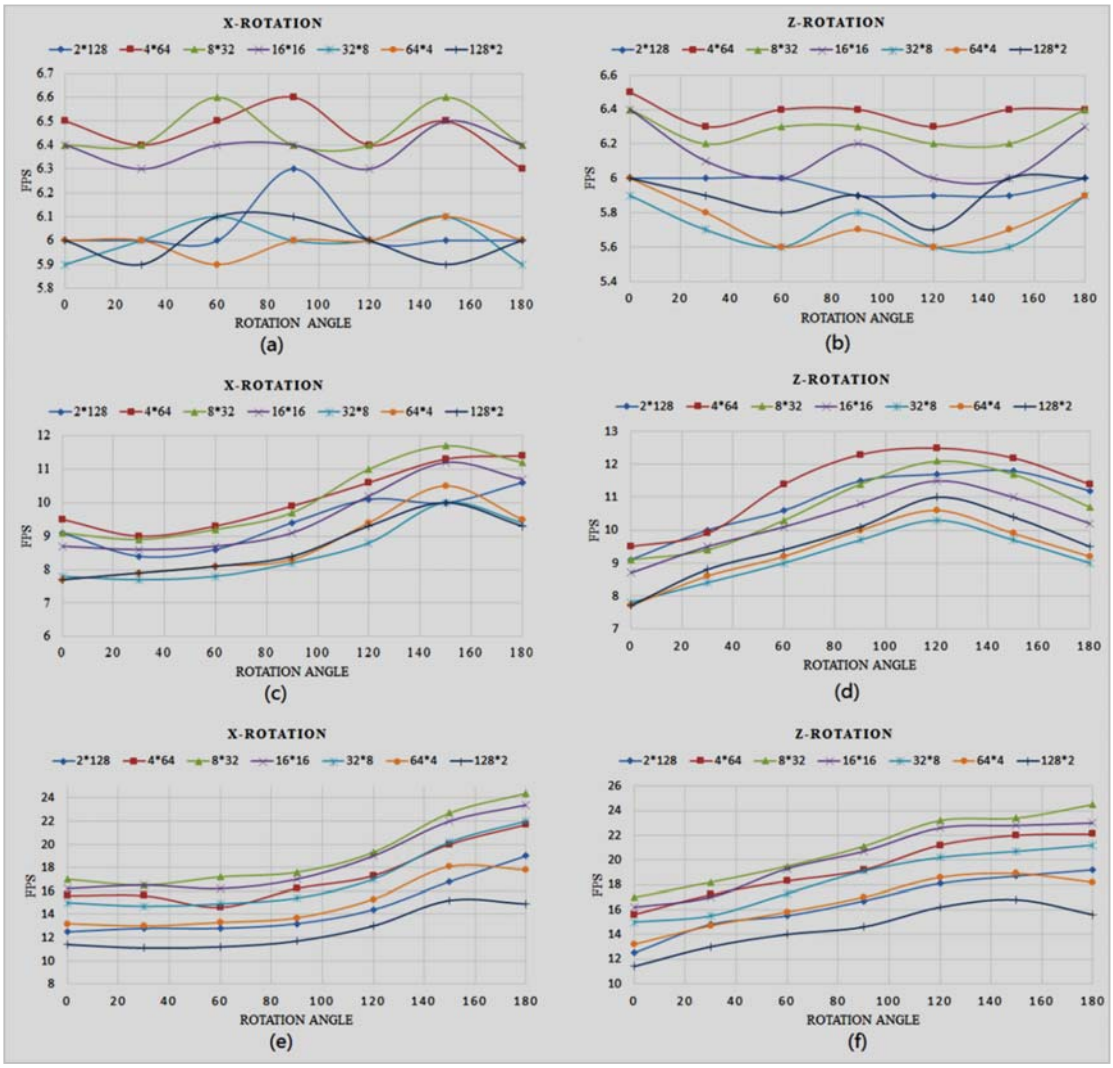

Figure 5. Relationship between rendering speed (fps) and rotation angle with different TB shapes using head dataset

Figure 6. Instructions per warp (IPW) performance of both control and experimental groups

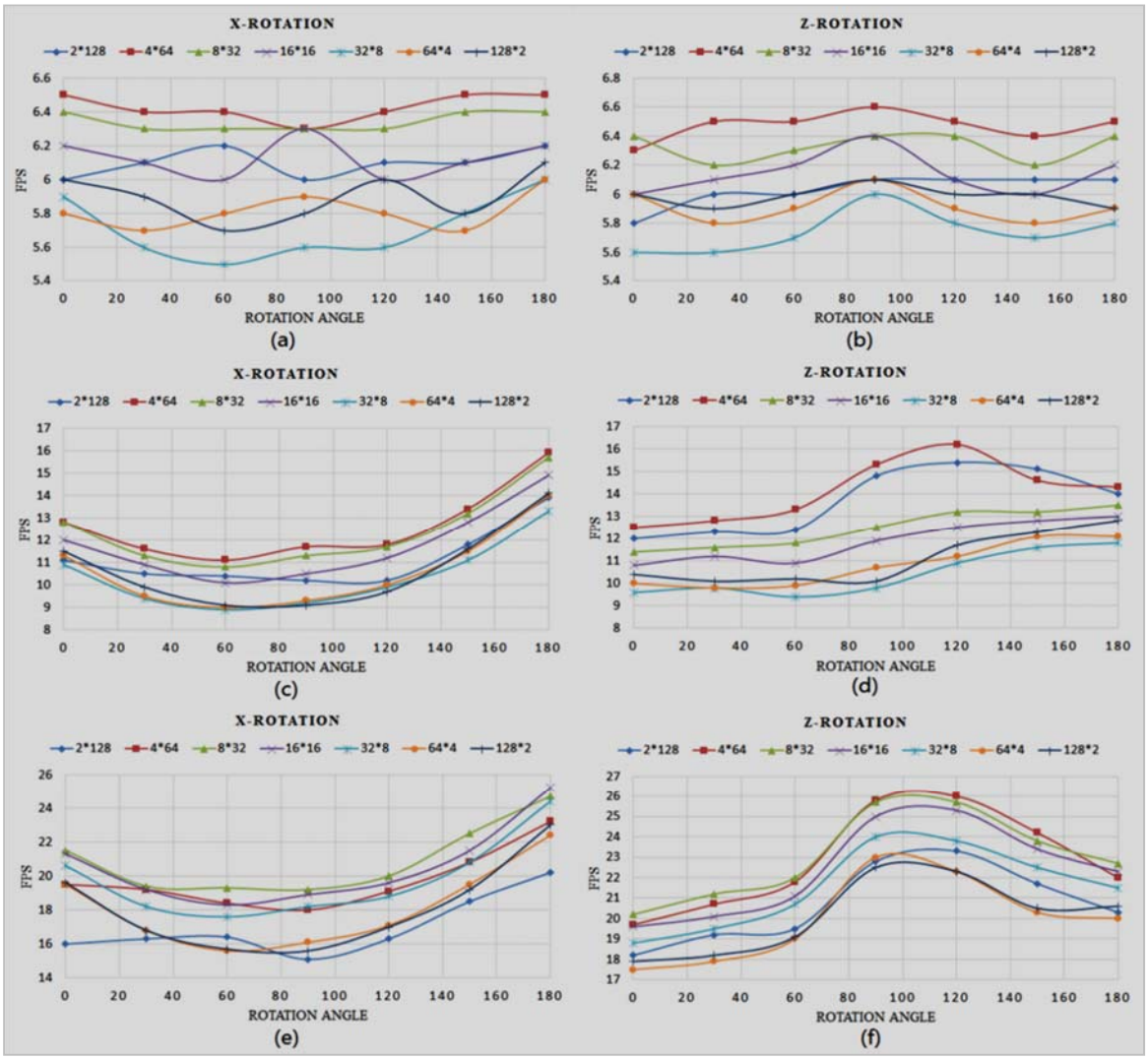

Figure 7. Relationship between rendering speed (fps) and rotation angle with different TB shapes using spinal vascular dataset

ional ray casting with only empty space skipping without early termination method, and Figs. 5c, d depict the results with both empty space skipping and early termination methods. Figs. 5e, f are experimental group using our methods.

FPS in Figs .5c, d is larger than the one in Figs .5a, b, because early termination method can reduce sampling number tremendously. Detailed explanation is shown in Fig .6, in which Fig .6c represents IPW (instructions per warp) when using both empty space skipping and early termination methods, but Fig. 6a represents IPW when using empty space skipping only. We can see that instructions in a warp dramatically decreases after applying early termination method. Warp mentioned here is the most basic scheduling unit on CUDA that consists 32 threads which run the same instruction at one time. FPS is promoted further after applying our method, see more in Figs .5e and f, the main reason is the reduction of sampling numbers and workload imbalance, details are also shown in Fig .6e, in which instructions number decreases further and workload distribution is much more balanced compared to Figs .6a and c.

In addition, Fig. 5 gives the relationship between frame rate and TB shape. Here, various TB shapes are used, like W×H=4×64 (red line) and 64×4 (orange line) in Fig. 5a. The results show that the vertical TB shapes (W<H), like W×H=4×64 and 8×32, mostly outperform those horizontal sizes (W>H), like W×H=64×4 and 32×8, when using the original ray casting method

(see Figs. 5a, b). This is due to the impact of TB shape on cache hit rate, i.e., warp size varies according to TB shape [29].

Another experiment is done using 'spinal vascular' dataset. Results are shown in Fig .7, in which similar results emerge that frame rate is greatly promoted when using our method no matter how rotation angle or TB shape changes, and vertical TB shapes (W<H), like W×H=4×64 (red line) and 8×32 (green line), mostly outperform those horizontal sizes (W>H), like W×H=64×4 (orange line) and 32×8 (blue line). The reason is also explained in Fig .6, where Figs .6b, d and f represent the performance of IPW when using spinal vascular dataset, tremendous decrease of instruction number is obvious and improvement of workload balance is also distinct.

Speed-up ratio, defined as (new fps - old fps)/old fps where new fps represents fps using our method, old fps represents fps using conventional ray casting with both empty space skipping and early termination methods, can reach average 70% promotion when using 'spinal vascular' dataset (shown in Fig. 5), in particular, when TB shape is W×H=32×8 and rotation angle is 90 degree, the speed-up ratio can reach 145%. On the whole, the speed-up ratio is higher where the original frame rate is lower, and vice versa, especially when TB shape is W×H=32×8, speed-up reaches highest among all TB shapes. In addition, speed-up ratio is much higher when rotation angles are around 90 degree compared to other situations.

IV. DISCUSSION

Image quality results have proved that this improved method can guarantee visual effect, the main reason is that although sampling strategy is applied based on distance weight, but among segments with closer distance to screen, intensive sampling interval has guaranteed image quality, sampling workload is reduced in further segments. But there is one situation where sampling regions clustered in the further areas from screen, artifact may occur because of low sampling frequency. In our experiments, this kind of situation appears only once, where the lowest PSNR value occurs in Table 1.

The workload of ray casting algorithm can be reflected by instruction numbers. The results in Figs .5, 7 have shown great promotion in rendering speed, reasons are shown in Fig .6. The IPW is largest when just using empty space skipping without early termination method, which means the sampling and instruction numbers are huge, uneven distribution of instructions is also obvious; after applying early termination method, the IPW is reduced dramatically, and workload distribution gains improvement; IPW is further reduced after applying our method. Instructions in one warp are now distributed to 2 or 3 warps after segmentation, the drop of instruction divergence and sampling number contributes to IPW decrease, as shown in Figs .6e, f, and GPU processing efficiency. The difference of instruction number between 'head' and 'spinal vascular' datasets in Figs .6c, d, e, f is because of the degree distribution of data opacity, early termination occurs in different timing.

Influencing factors like TB shape and data distribution are also improved by using our method, as shown in speed-up ratio results. Calculation results show that the speed-up ratio is higher where the original frame rate is lower, and vice versa, like TB shape W×H=32×8 and rotation angle around 90 degree when using 'spinal vascular' dataset. The reason for explaining this phenomenon is that our method can weaken the effect of these factors by optimizing workload distribution and balance.

V. CONCLUSION

In this paper, an improved method using automatic segmentation and modified sampling strategy is proposed using CUDA, in which the integrity of ray casting algorithm is preserved. Automatic segmentation utilizes features of CUDA like thread communication through shared memory and task switching mechanism, improving the degree of parallelism. Sampling strategy based on distance weight is mainly used to reduce sampling workload. Easy to implement, low cost and good performance make this method have broad prospects, which will contribute to real-time 3-D reconstruction in all disciplines. Future work will include improving sampling strategy to achieve better display results, refinement of ray intersection and practical implementation combined to MRI machine, where original MRI data flow, other than intermediate RAW image files, will be used.

ACKNOWLEDGMENTS

This work was supported by National Science Foundation of China (Grant number: 81371537, 91432301), Major State Basic Research Development Program of China (973 Program) (Grant number: 2013CB733803), and Fundamental Research Funds for the Central Universities of China (Grant number: WK2070000033). The authors would also like to thank Haiyan Zheng, Yuchuan Jia, Chang Zhai, Chunxiao Wang for their help on paper revision.

REFERENCES

[1] Drebin R A, Carpenter L, Hanrahan P, Volume rendering, ACM Siggraph Computer Graphics, vol.22, no.4, pp.65-74, 1988.

[2] Yong H W, Bade A, Muniandy R K, 3D RECONSTRUCTION OF BREAST CANCER FROM MAMMOGRAMS USING VOLUME RENDERING TECHNIQUES, Jurnal Teknologi, vol.75, no.2, 2015.

[3] Preim B, Baer A, Cunningham D, et al., A Survey of Perceptually Motivated 3D Visualization of Medical Image Data, Computer Graphics Forum, vol.35, no.3, 2016.

[4] Liu P, Gong J, Yu M, Visualizing and analyzing dynamic meteorological data with virtual globes: A case study of tropical cyclones, Environmental Modelling & Software, vol.64, pp.80-93, 2015.

[5] Liu P, Gong J, Yu M, Graphics processing unit-based dynamic volume rendering for typhoons on a virtual globe, International Journal of Digital Earth, vol.8, no.6, pp.431-450, 2015.

[6] M. Krone et al., Fast visualization of gaussian density surfaces for molecular dynamics and particle system trajectories, EuroVis-Short Papers, vol.1, pp.67-71, 2012.

[7] H. Ray et al., Ray casting architectures for volume visualization, Visualization and Computer Graphics, IEEE Transactions, vol.5, no.3, pp.210-223, 1999.

[8] K. L. Ma et al., A data distributed, parallel algorithm for ray-traced volume rendering, Proceedings of the 1993 symposium on Parallel rendering, pp.15-22, 1993.

[9] Rodrigues J, Balan A, Zaina L, et al., A Survey on Distributed Visualization Techniques over Clusters of Personal Computers, arXiv preprint arXiv:1506.06968, 2015.

[10] Nvidia C, Nvidia cuda c programming guide, Nvidia Corporation, 2011.

[11] Liang R, Wu Y, Dong F, et al., Accumulation of local maximum intensity for feature enhanced volume rendering, The Visual Computer, vol.28, no.6-8, pp.625-633, 2012.

[12] Gobbetti E, Marton F, Guitián J A I, A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets, The Visual Computer, vol.24, no.7-9, pp.797-806, 2008.

[13] Rodríguez M B, Gobbetti E, Guitián J I, et al., A survey of compressed GPU-based direct volume rendering, Proc. Eurographics, vol.13, 2013.

[14] Bozorgi M, Lindseth F, GPU-based multi-volume ray casting within VTK for medical applications, International journal of computer assisted radiology and surgery, vol.10, no.3, pp.293-300, 2015.

[15] Klar O, Interactive GPU based segmentation of large medical volume data with level sets, CESCG 2007, 2006.

[16] A. Maximo et al., Memory Efficient GPU-Based Ray Casting for Unstructured Volume Rendering, Volume Graphics, pp.155-162, 2008.

[17] Fogal T, Childs H, Shankar S, et al., Large data visualization on distributed memory multi-GPU clusters, Proceedings of the Conference on High Performance Graphics, Eurographics Association, pp.57-66, 2010.

[18] Lee E S, Lee J H, Shin B S, A bimodal empty space skipping of ray casting for terrain data, The Journal of Supercomputing, pp.1-15, 2015.

[19] Fogal T, Schiewe A, Krüger J, An analysis of scalable GPU-based ray-guided volume rendering, IEEE Symposium on Large-Scale Data Analysis and Visualization, NIH Public Access, pp.43, 2013.

[20] Boada I, Navazo I, Scopigno R, Multiresolution volume visualization with a texture-based octree, The visual computer, vol.17, no.3, pp.185-197, 2001.

[21] Suter S K, Makhynia M, Pajarola R, Tamresh–tensor approximation multiresolution hierarchy for interactive volume visualization, Computer Graphics Forum, Blackwell Publishing Ltd, vol.32, no.3pt2, pp.151-160, 2013.

[22] Cai L, Nguyen B P, Chui C K, et al., A two-level clustering approach for multidimensional transfer function specification in volume visualization, The Visual Computer, pp.1-15, 2015.

[23] P. Ljung, C. Lundström, and A. Ynnerman, Multiresolution interblock interpolation in direct volume rendering, Lisbon, Portugal, pp.259-266, 2006.

[24] Zhang C, Xi P, Zhang C, CUDA-based volume ray-casting using cubic B-spline, Virtual Reality and Visualization (ICVRV), 2011 International Conference on, IEEE, pp.84-88, 2011.

[25] T. L. Falch et al., GPU-Accelerated Visualization of Scattered Point Data, IEEE Access, vol.1, pp.564-576, 2013.

[26] Y. Zhao, X. Cui, and Y. Cheng, High-performance and real-time volume rendering in CUDA, Biomedical Engineering and Informatics, BMEI'09. 2nd International Conference, pp.1-4, 2009.

[27] Kumar P, Agrawal A, CUDA based interactive volume rendering of 3D medical data, Intelligent Interactive Technologies and Multimedia, Springer Berlin Heidelberg, pp.123-132, 2013.

[28] Shi Z, Jinyi C, Improved algorithm of volume rendering combined texture mapping with ray casting based on CUDA, Application Research of Computers, vol.6, pp.66, 2015.

[29] Sugimoto, Yuki, Fumihiko Ino, and Kenichi Hagihara., Improving cache locality for GPU-based volume rendering, Parallel Computing, vol.40, no.5, pp.59-69, 2014.

[30] M. Nießner et al., Real-time 3d reconstruction at scale using voxel hashing, ACM Transactions on Graphics (TOG), vol.32, no.6, pp.169, 2013.

[31] J. Vlietinck, Method and system for real-time volume rendering on thin clients via render server, Google Patents, 2013.

[32] Lv, Z., Yin, T., Song, H., & Chen, G. (2016). Virtual Reality Smart City Based on WebVRGIS. IEEE Internet of Things Journal.

[33] Li, X., Lv, Z., Hu, J., Zhang, B., Shi, L., & Feng, S. (2015, June). Xearth: A 3d gis platform for managing massive city information. In 2015 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA) (pp. 1-6). IEEE.

[34] Jiang, D., Xu, Z., & Lv, Z. (2015). A multicast delivery approach with minimum energy consumption for wireless multi-hop networks. Telecommunication systems, 1-12.

.