J. Biomed. Eng. Res.
Vol.26, No.1, 23-30, 2005

A Fast Volume Rendering Algorithm for Virtual Endoscopy

Jong Beom Ra, Sang Hun Kim[1], Sung Min Kwon

Department of Electrical Engineering and Computer Science, KAIST
[1]Broadcast Technical Research Team, KBS
(Received October 24, 2004. Accepted November 29, 2004)

Abstract: 3D virtual endoscopy has been used as an alternative non-invasive procedure for visualization of hollow organs. However, due to computational complexity, this is a time-consuming procedure. In this paper, we propose a fast volume rendering algorithm based on perspective ray casting for virtual endoscopy. As a pre-processing step, the algorithm divides a volume into hierarchical blocks and classifies them into opaque or transparent blocks. Then, in the first step, we perform ray casting only for sub-sampled pixels on the image plane, and determine their pixel values and depth information. In the next step, by reducing the sub-sampling factor by half, we repeat ray casting for newly added pixels, and their pixel values and depth information are determined. Here, the previously obtained depth information is utilized to reduce the processing time. This step is recursively performed until a full-size rendering image is acquired. Experiments conducted on a PC show that the proposed algorithm can reduce the rendering time by 70 - 80% for bronchus and colon endoscopy, compared with the brute-force ray casting scheme. Using the proposed algorithm, interactive volume rendering becomes more realizable in a PC environment without any specific hardware.

Key words: Fast volume rendering, Perspective ray casting, Virtual endoscopy

INTRODUCTION

Endoscopy is an effective medical treatment and diagnostic technique that enables a doctor to examine the interior surface of hollow organs such as the bronchus and colon in a human body. During the endoscopic procedure, an optical probe is inserted into an organ's interior. As such, endoscopy has several intrinsic defects, such as causing considerable uncomfortableness to patients, risk of perforation through an organ, and a limited observation area. In this context, virtual endoscopy can be an alternative to the optical endoscopy, since it enables doctors to examine the interior surface of target organs in a virtual environment reconstructed from a series of CT images without inserting an optical probe [1]. A virtual endoscopy has various merits compared to the conventional endoscopy. First, sedation or anesthesia to reduce patient pain is not necessary. Second, probability of perforation through organs is zero. Third, various types of images can be acquired by simply controlling camera parameters and changing the camera path. Lastly, many organs such as the bronchus, colon, and blood vessels can be easily examined, whereas some organs (e.g. blood vessels) cannot be examined via a conventional endoscopy [2-3].

For virtual endoscopy, two rendering methods are usually used; surface rendering and volume rendering. In surface-rendering-based virtual endoscopy, the boundary surface is automatically (or manually) extracted from 3D volume data at the pre-processing step. Then, the extracted surface is modeled into triangular meshes and is rendered using a graphics pipeline [4]. Even though surface rendering can be performed rapidly due to graphic hardware acceleration, it still needs to alleviate the preprocessing burden as well as high hardware requirement [5]. It may also lose some important information due to the binary decision in extracting the surface, and the number of triangles to model the surface. In addition, its application areas may be limited because it cannot visualize tissues beneath the interior surface. In contrast, volume-rendering-based virtual endoscopy provides realistic rendering images, and make various operations and applications possible because they can visualize the data beneath the interior surface. Although volume rendering can be a good alternative to surface rendering, it requires a high computational cost, especially for perspective rendering as is the case for virtual endoscopy. Therefore, most volume-

Corresponding Author: Jong Beom Ra, Dept. of EECS, KAIST, 373-1, Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea
Tel. +82-42-869-5434, Fax. +82-42-869-8360
E-mail. jbra@ee.kaist.ac.kr

rendering-based virtual endoscopy systems use a high price graphics workstation with a parallel architecture, and require a large-size memory to provide desirable image quality and rendering speed [3], [6]. However, high hardware cost may limit the general use of virtual endoscopy. Also, if lossy rendering is allowed in order to maintain desired rendering speed in a less expensive hardware, some important information may be lost [2].

To alleviate this problem, we consider pure software-based volume rendering. There are several software-based algorithms available, based on volumetric ray casting [7], voxel projection [8-9], and shear-warp factorization approach [10], and so on. In this paper, we adopt the ray casting algorithm, which is known as a more accurate method for perspective rendering, and widely used in virtual endoscopy. The ray casting algorithm consists of three steps; re-sampling along rays, calculation of color and opacity values at the sampling points, and combining colors and opacity to acquire final pixel values. Since these steps are repeated on every pixel in an image, the algorithm has a high computational cost; thus real-time or interactive rendering is not feasible. Therefore, various space leaping schemes to accelerate ray casting have been suggested by using object data coherence and/or temporal coherence [6], [11-17].

Among these schemes, let us focus on the ones using object data coherence. The depth-based skipping algorithm acquires depth information by surface rendering, and uses this information in improving the rendering speed [6], [15-16]. However, these methods need a pre-processing step to extract the boundary surface and requires additional memory for the surface. Also, it is not guaranteed that the depth information acquired by surface rendering is the same as the real depth information. Meanwhile, the potential-field-assisted skipping method measures the Euclidean distance between every voxel located inside a hollow organ and the closest interior surface [11]. By using the measured distances, empty space can be leaped over with a locally adaptive skipping method during the rendering in order to reduce the rendering time. However, this method requires much time in the pre-processing step and needs large-size of memory for calculating and storing the distances of all voxels. In the block-based skipping method, the volume data is divided into blocks and they are classified into opaque or transparent blocks [12]. Then, by eliminating transparent blocks from the rendering process, the rendering speed can be improved. However, since some target organs have small diameters and frequent winding, a speedup by using a block structure has limitations. To improve the rendering speed, the exponential-region perspective ray casting methods make rays uniformly sample the data throughout the viewing volume [18-19]. However, their speed improvement is not noticeable compared to the other methods, because empty space skipping is not considered.

In this paper, we propose a fast volume rendering algorithm for virtual endoscopy on a general PC. This approach does not require any additional hardware or a large-size of memory, and provides lossless image quality with a considerably improved rendering speed.

PROPOSED ALGORITHM

To obtain a rendered image, the proposed algorithm performs hierarchical ray casting by gradually decreasing the sub-sampling interval of image pixels. Thereby, the rendered image becomes finer step-by-step. Meanwhile, to reduce the rendering time, the depth information at each step is stored and used to estimate the starting depths of rays to be cast in the next step. Here, the depth information of a pixel denotes the distance between a virtual camera and a suitable location before the interior surface along the corresponding ray. An overall block diagram of the proposed algorithm and the corresponding image refining procedure are shown in Fig. 1. The algorithm consists of a pre-processing step and initial rendering and refining steps. In the following subsections, the method is explained in detail.

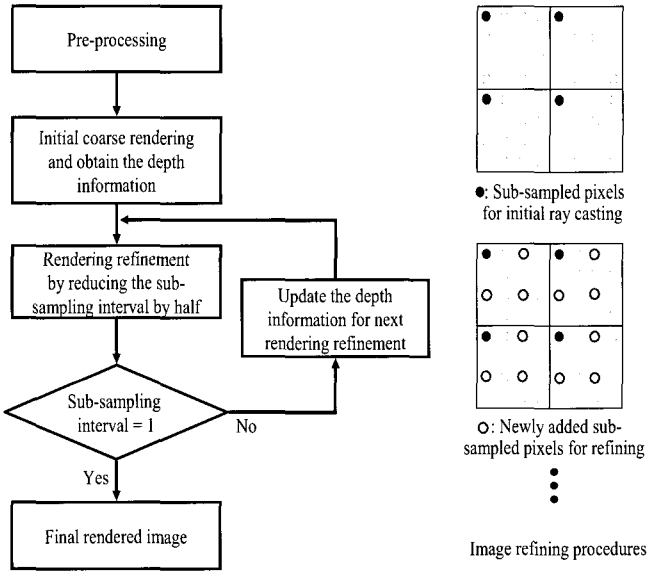


Fig. 1. Overall block diagram of the proposed algorithm and the corresponding rendered image structure.

Pre-Processing

In this step, the volume is divided into hierarchical blocks and they are classified into opaque or transparent blocks. If the total summation of opacity in a block equals zero, the block is designated a

transparent block. Otherwise, the block is designated as opaque. This pre-processing step can be performed without any noticeable computational burden.

Initial Rendering and Depth Information

In the initial rendering step, ray casting is performed only for coarsely sub-sampled pixels on the image plane to determine their pixel values and depth information. To reduce the rendering time in this step, we use the block transparency information, which is obtained in the preprocessing step [12]. If sampling points on a ray are located in a transparent block, we can skip the block and move to the first sampling point in the next block.

Depth Information

In utilizing the depth information in the rendering refinement, the proposed algorithm is based on two important points. First, the starting depth of a ray can be easily estimated from the depth information of its neighbor rays, which is determined in the previous step. Second, since the ray density is usually much larger than the voxel density near the virtual camera, empty space skipping based on the estimated starting depth is effective. It should be noted that loss of information is not allowed in reducing the processing time. Therefore, any approximation in the depth estimation, which may cause image quality degradation, is not permitted. As the depth information, we use either the voxel-wise depth or block-wise depth, depending on the length of the voxel-wise depth. Here, the voxel-wise depth, d_v, denotes the distance along a ray, from a camera to the transparent point just before the first sampling point having nonzero opacity (see Fig. 3); the block-wise depth, d_b, denotes the distance from a camera to the sampling point just before the first opaque leaf block on a ray (see Fig. 4).

Now, let us consider the decision strategy between the two kinds of depths. Fig. 2 depicts a 2D side view showing the relationship among the camera, the rendered image of $N \times N$, and corresponding sub-sampled rays of $N/N_S \times N/N_S$, where N_S is a sub-sampling interval. In this figure, the angle α_0 between the two sub-sampled rays can be represented as

$$\alpha_0 = 2\tan^{-1}\left(\frac{N_S}{N}\cdot\tan\frac{\theta}{2}\right). \quad (1)$$

Since α_0 is the maximum among the angles produced by a pair of subsequent sub-sampled rays, it would be considered to drive the most reliable criterion. Then, the voxel-wise maximum depth, r_v, is defined as

$$r_v = (1/\sin\frac{\alpha_0}{2}), \quad (2)$$

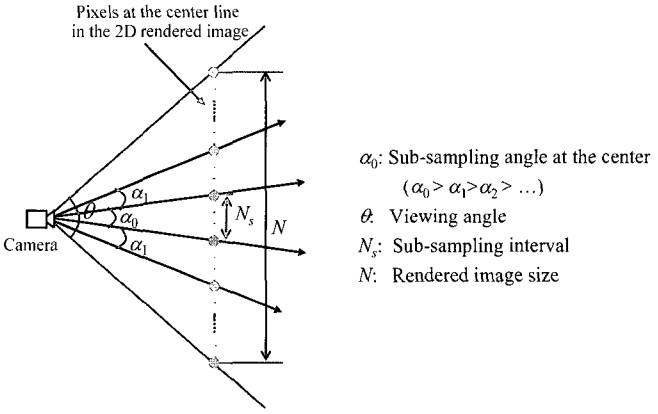


Fig. 2. Side view showing the relationship among the camera, the rendered image, and corresponding sub-sampled rays.

in Fig. 3(a). Here, the voxel sampling interval is set to 1 for simplicity. To examine the meaning of r_v, we may consider Fig. 3(b), In this figure, we assume that A_1 and A_2 denote the transparent sampling points just before the first opaque ones of the two sub-sampled and subsequent rays, R_1 and R_2, respectively, and $d_v(R_1)$ and $d_v(R_2)$ are shorter than r_v. Then, in the examining procedure of six voxels adjacent to A_1 and A_2 for sampling, we easily find that ray R' in the next step can skip up to d_v, because the voxel opacities surrounding A_1 and A_2 are to be zero. (Here, we consider the case that $d_v = d_v(R_1) = d_v(R_2)$, for simplicity.) However, if A_1' and A_2', which are located beyond r_v, are the transparent sampling points just before the first opaque ones of R_1 and R_2 as in Fig. 3(c), a similar decision for depth skipping is not possible because the two voxels between rays R_1 and R_2 are not examined to be transparent. Hence, if those sampling points are located beyond r_v, we consider the diagrams in Fig. 4.

Let us define the block-wise maximum depth, r_b, as follows.

$$r_b = N_b/(2\sin\frac{\alpha_0}{2}), \quad (3)$$

where N_b is the leaf block size as shown in Fig. 4(a). And we assume that sampling points B_1 and B_2 in Fig. 4(b) represent the sampling points just before the first points whose neighbors include opaque voxel(s). If the distances between the camera and points B_1 and B_2 are longer than r_v, we cannot use the voxel-wise depth d_v for skipping in ray R' in the next step. Instead, we adopt the block-wise depth, d_b, which denotes the distance from the camera to the sampling point just before the first opaque leaf block on a ray. Then, we

can guarantee that ray R' between R_1 and R_2 does not meet any opaque voxel until it reaches d_b. (Note that we consider the case that $d_b = d_b(R_1) = d_b(R_2)$, for simplicity.) Based on this observation, we define the depth information, d, for each ray, depending on the voxel-wise depth and the block-wise depth, i.e.,

$$d = \begin{cases} d_v, & d_v \le r_v, \\ d_b, & d_v > r_v \text{ and } d_b \le r_b, \\ r_b, & d_b > r_b. \end{cases} \quad (4)$$

This depth information is used for estimating skipping depths on rays in the next, finer step. Notice that, before the depth estimation, the depth information of the sub-sampled rays of the previous steps should be updated by using their d_v's, which are already available. This is because the depth information, d, depends on r_v and r_b, which a function of α_0, and α_0 is updated at each step by the corresponding sub-sampling interval. Note also that d becomes closer to the real depth value and the ray skipping gets more effective as the step goes on.

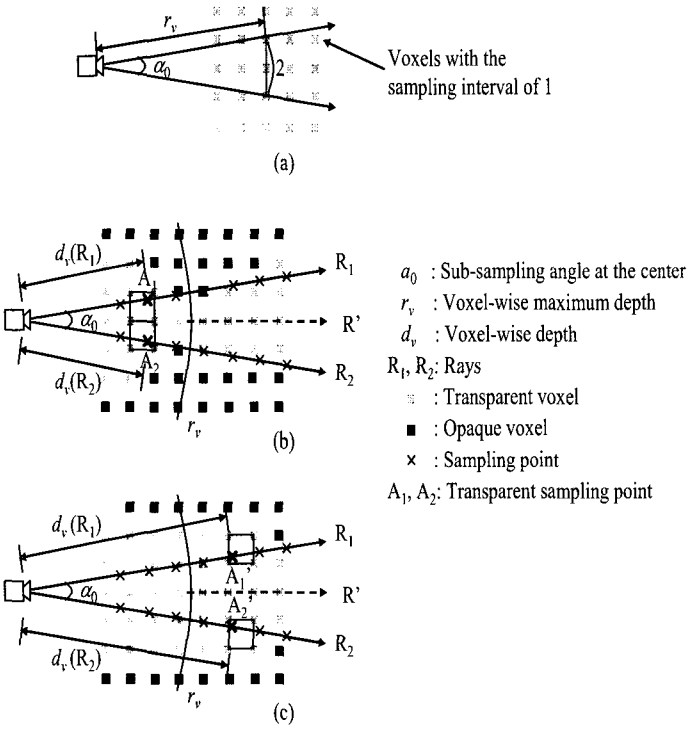


Fig. 3. Definition of the voxel-wise depth d_v and voxel-wise maximum depth r_v.

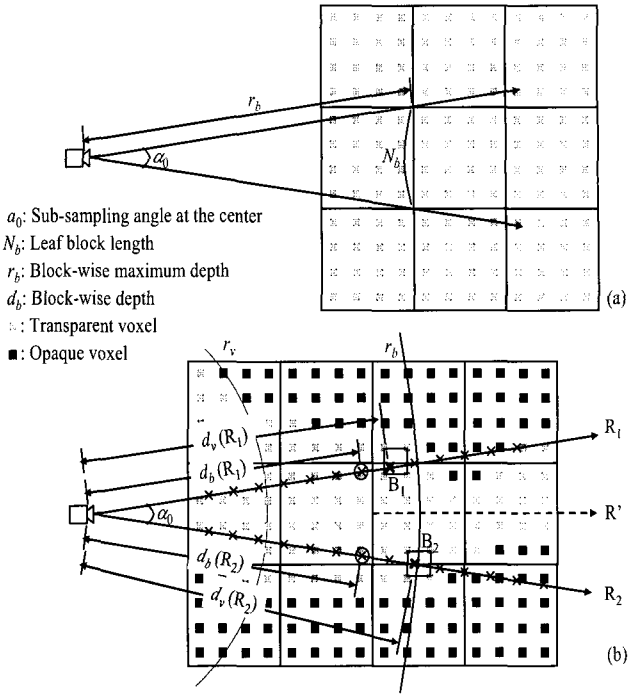


Fig. 4. Definition of the block-wise depth d_b and the block-wise maximum depth r_b.

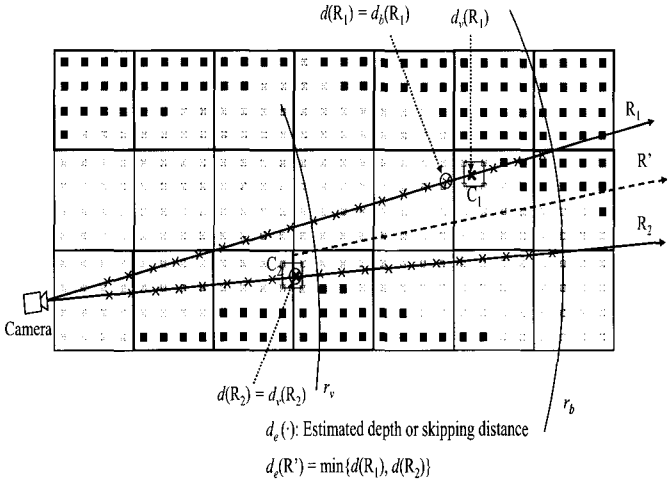


Fig. 5. Procedure to obtain the depth information from a virtual camera.

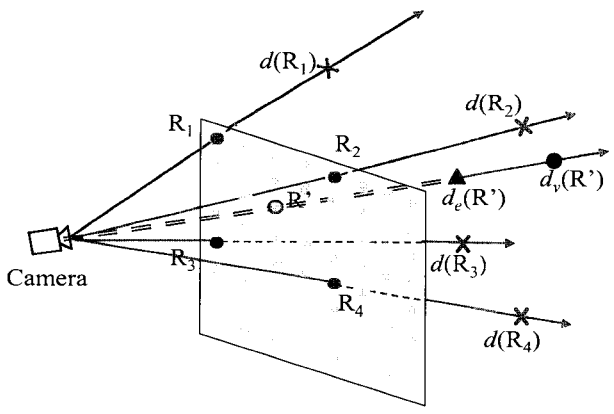
Rendering Refinement based on Depth Information

For the rendering refinement, the algorithm reduces the sub-sampling interval by half, and performs ray casting only for newly added pixels (or rays) to

determine the pixel values and corresponding depth information. Here, the depth information obtained by equation (4) in the previous step is utilized in determining pixel values to reduce the processing time. As mentioned above, in Figs. 3 and 4, we illustrated instances where the d_v values were almost the same for subsequent rays. Fig. 5 shows a more general case, where the viewing angle is tilted and the d_v values are obviously different. As mentioned in the previous section, criteria r_v and r_b used in equation (4) are still applicable for a tilted viewing angle, because α_0 is always larger than α_1, α_2, ... Since voxel-wise depths, d_v for the two rays R_1 and R_2, are quite different in the example of Fig. 5, depth information, $d(R_1)$ and $d(R_2)$, based on equation (4) are also different. In this case, we conservatively choose the estimated skipping distance of R' as their minimum value. Thus far, in describing the proposed ray skipping scheme based on the depth information, we have adopted simplified 2D diagrams, as in Figs. 3, 4 and 5. If we expand this concept to an actual 3D case, as in Fig. 6, the estimated depth (or the skipping distance) of R' can be described as follows.

$$d_e(R') = \min\{\, d(R_1), d(R_2), d(R_3), d(R_4) \,\}. \quad (5)$$

Using equation (5), a new refining ray begins sampling (or ray casting) after $d_e(\cdot)$ from the camera. Thereby, we can skip the transparent space efficiently and start the rendering near the interior surface. Also the updated voxel-wise depth value d_v of the new ray and the corresponding depth information d are subsequently determined for the next step. This rendering refinement step is performed recursively until a full-size rendering image is acquired (see Fig. 1).



$d(\cdot)$: Depth information for the corresponding ray

$d_e(R') = \min\{d(R_1), d(R_2), d(R_3), d(R_4)\}$

Fig. 6. Depth estimation and utilization to reduce the rendering time.

Parameters to be Considered

The initial rendering time and the efficiency of depth information, which affect the total rendering time, depend on parameters such as the number of total rays, the sub-sampling factor, and leaf block size. If the sub-sampling interval for the initial rendering is large, the initial rendering time can be reduced. However, a longer rendering refinement time will be needed, because the number of refinement steps increases and estimated depths for ray skipping becomes shorter in the earlier steps. Meanwhile, for a small size of leaf block, the initial rendering time becomes shorter. However, the interval where the block-wise depth can be used ($(r_b - r_v)$ in Fig. 4), decreases. Thereby, the depth estimation range also decreases, potentially increasing the total rendering time. Therefore, a proper selection of parameter values is important for effectively reducing the rendering time. In this paper, we experimentally determine the parameters as described in the following section.

SIMULATION RESULTS AND DISCUSSION

For simulation, we use a bronchus CT data set of 256 × 256 × 283 with voxel dimensions of 0.703 × 0.703 × 1.0 mm^3, and a colon data set of 512 × 512 × 127 with voxel dimensions of 0.553 × 0.553 × 1.0 mm^3. The size of output image is set to 256 × 256 for a viewing angle of 90°. Experiments are performed on a PC equipped with an AMD 800MHz CPU.

To determine proper parameters experimentally, we examine the rendering time depending on the initial sub-sampling factor and leaf block size, and described in Tables 1 and 2 for the bronchus and colon images, respectively. Based on these results, we set the initial sub-sampling interval to 4 and the leaf block size to 4 × 4 × 4 for both images. Note that rendering time changes depending on parameters in the initial and refinement stages follow our discussion described in the previous sub-section.

In Tables 3, rendering times of the proposed and various existing algorithms are compared for the bronchus and colon images, respectively. The rendering time is the average value for 40 images obtained during navigation following a predefined path. For fair comparison among the algorithms, we use the same rendering software besides the skipping part. 'Ideal skipping' in Table 3 shows the rendering time obtained by assuming that we know real depth values for all rays. 'Brute-force' shows the rendering time when the rendering is done for regularly sampled rays without skipping. 'Potential-field-assisted skipping' denotes the time obtained by assuming that the distance to the interior surface of the organ is known for every voxel [11]. Finally, 'Block-based skipping' is the rendering time when the volume is divided into blocks, and transparency flags are set to blocks [12].

Table 1. Rendering time (msec) depending on sub-sampling factor and leaf block size for a bronchus image.

	Initial sub-sampling factor								
	2			4			8		
Leaf Block size	Initial rendering	Rendering refinement	Total	Initial rendering	Rendering refinement	Total	Initial rendering	Rendering refinement	Total
4 × 4 × 4	290	358	648	74	463	537	19	558	577
8 × 8 × 8	368	353	721	95	468	563	24	590	614

Table 2. Rendering time (msec) depending on sub-sampling factor and leaf block size for a colon image.

	Initial sub-sampling factor								
	2			4			8		
Leaf Block size	Initial rendering	Rendering refinement	Total	Initial rendering	Rendering refinement	Total	Initial rendering	Rendering refinement	Total
4 × 4 × 4	339	292	631	80	390	470	22	556	578
8 × 8 × 8	390	298	688	98	418	516	26	711	737

Table 3. Rendering time for the bronchus and colon images. In the proposed algorithm, the initial sub-sampling interval of 4 and leaf block size of 4 × 4 × 4 are used.

	Bronchus image		Colon image	
Method	Rendering time (ms)	Rendering time (ratio to ideal skipping)	Rendering time (ms)	Rendering time (ratio to ideal skipping)
Ideal skipping	396	1.00	365	1.00
Brute-force	1897	4.79	2352	6.44
Potential-field-assisted skipping [11]	1229	3.10	1027	2.81
Block-based skipping [12]	1381	3.39	1552	4.25
Proposed	493	1.24	431	1.18

Relative ratios of rendering time to 'Ideal skipping' for the bronchus and colon images are illustrated in Fig. 7 and Fig. 8, respectively. Here, notice that all the rendered images obtained from the algorithms above are identical. From these experimental data, the proposed algorithm provides the fastest rendering speed. Figs. 9 and 10 show an exact depth image in 'Ideal skipping', depth image estimated in our algorithm, and rendered image that is common, for a bronchus and colon, respectively. It should be noted that there is only a slight difference between the two depth images. And it makes the rendering speed of the proposed algorithm similar to the one of 'Ideal skipping'. Additional memory for the proposed efficient skipping, is used to save the block classification result and the voxel-wise depth, block-wise depth, and depth information for the 2D rendered image pixels of $N/2 \times N/2$. This memory requirement is minor compared with

the original 3D volume data for rendering.

According to the obtained data, it is demonstrated that the proposed method can provide prospective results for fast perspective volume rendering in a virtual endoscopy. The proposed algorithm reduces the rendering time by 74% for the bronchus image and by 81% for the colon, compared with the brute-force ray casting scheme. This rendering time is about 2.5 times faster than the other compared fast algorithms. Our algorithm makes three major contributions. First, two types of depth criteria are introduced and efficiently used without affecting the image quality. Second, the algorithm can be easily applied for progressive rendering. In addition, the depth information is refined during the progressive process so that more efficient voxel skipping may be possible. Finally, contrary to other existing algorithms, our approach does not require any additional hardware or time consuming pre-processing.

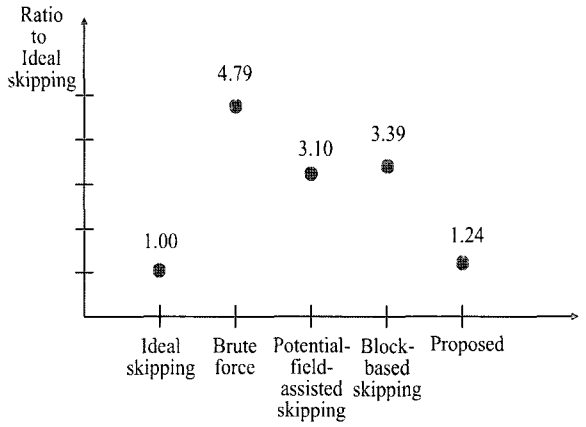


Fig. 7. Relative ratios of rendering time to 'Ideal skipping' for a bronchus image data set.

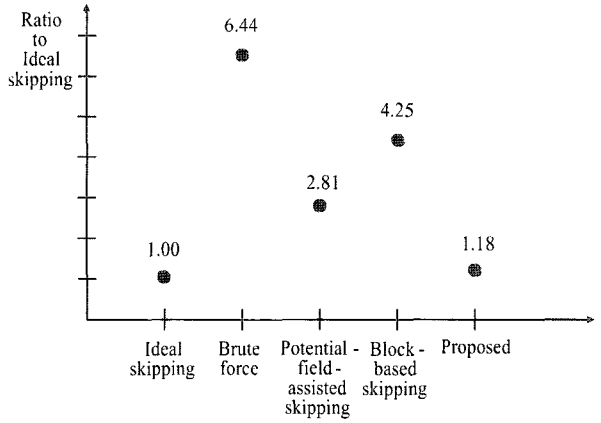


Fig. 8. Relative ratios of rendering time to 'Ideal skipping' for a colon image data set.

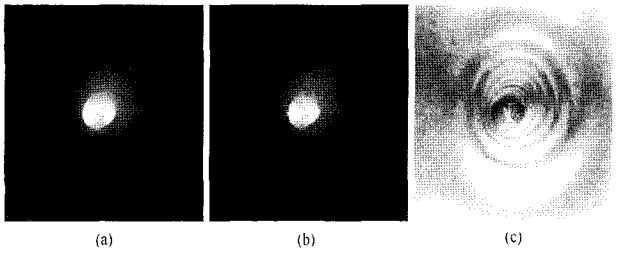


Fig. 9. (a) Exact depth image for ideal skipping, (b) depth image estimated from the proposed method, and (c) the corresponding rendered result, for a bronchus image data set.

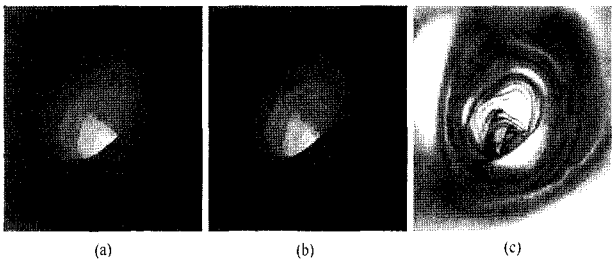


Fig. 10. (a) Exact depth image for ideal skipping, (b) depth image estimated from the proposed method, and (c) the corresponding rendered result, for a colon image data set.

CONCLUSIONS

We propose a fast and progressive volume rendering algorithm for a virtual endoscopy. By using the estimated depths in each step, we can efficiently skip empty spaces and only perform rendering near the interior surface. Thereby, the ray casting time can be drastically saved, without any image quality degradation. We also note in the proposed algorithm that the depth information is refined simultaneously during the ray casting for volume rendering. Experimental data shows that the proposed algorithm provides much faster performance compared with the existing algorithms.

REFERENCES

[1] Y. Lee, P.H. Lin, C.H. Lin, Y.N. Sun, and X.Z. Lin, *"Interactive 3-D Virtual Colonoscopy System"* , IEEE Trans. Information Technology in Biomedicine, Vol. 3, No. 2, pp. 139-150, 1999.

[2] L. Hong, A. Kaufman, Y. Wei, A. Viswambharn, M. Wax, and Z. Liang, *"3D Virtual Colonoscopy"*, Proc. IEEE Symp. Frontier in Biomedical Visualization, pp. 26-32, 1995.

[3] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He, *"Virtual Voyage: Interactive Navigation in the Human Colon"*, Proc. ACM SIGGRAPH '97, pp. 27-34, 1997.

[4] W.E. Loresen, and H.E. Cline, *"Marching Cubes: A High*

Resolution 3D Surface Construction Algorithm", Computer Graphics, Vol.21, No.4, pp. 163-169, 1987.

[5] R. Hietala, and J. Oikarinen, *"A Visibility Determination Algorithm for Interactive Virtual Endoscopy"*, Proc. IEEE Visualization 2000, pp. 29-36, 2000.

[6] S. You, L. Hong, M. Wan, K. Junyaprasert, A. Kaufman, S. Muraki, Y. Zhou, M. Wax, and Z. Liang, *"Interactive Volume Rendering for Virtual Colonoscopy"*, Proc. IEEE Visualization '97, pp. 433-436, 1997.

[7] M. Levoy, Display of Surface from Volume Data, IEEE Computer Graphics and Applications, pp. 29-37, 1988.

[8] K. Mueller, N. Shareef, J. Huang, and R. Crawfis, *"High-Quality Splatting on Rectilinear Grids with Efficient Culling of Occluded Voxels"*, IEEE Trans. Visualization and Computer Graphics, Vol. 5, No. 2, pp. 116-134, 1999.

[9] J.K. Udupa, and D. Odhner, Shell Rendering, IEEE Computer Graphics and Ap1plications, pp. 58-67, 1993.

[10] P. Lacroute, and M. Levoy, *"Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation"*, Proc. ACM SIGGRAPH '94, pp. 451-458, 1994.

[11] M. Wan, Q. Tang, A. Kaufman, and Z. Liang, *"Volume Rendering Based Interactive Navigation within the Human Colon"*, Proc. IEEE Visualization '99, pp. 397-401, 1999.

[12] M. Levoy, *"Efficient Ray Tracing of Volume Data"*, ACM Trans. Graphics, Vol. 9, No. 3, pp. 245-261, 1990.

[13] M. Levoy, *"Volume Rendering by Adaptive Refinement"*, Visual Computer, Vol. 6, No. 1, pp. 2-7, 1990.

[14] M. Wan, A. Sadiq, and A. Kaufman, *"Fast and Reliable Space Leaping for Interactive Volume Rendering"*, Proc. IEEE Visualization, pp. 195-202, 2002.

[15] D. Silver, H. Sundar, and N. Gagvani, *"Shape Based Culling for Volume Graphics"*, Proc. 13th Eurographics Workshop on Rendering, Poster Session Presentation, 2002.

[16] A. Vilanova, E. Groller, and A. Konig, *"Cylindrical Approximation of Tubular Organs for Virtual Endoscopy"*, Proc. Computer Graphics and Imaging, pp. 283-289, 2000.

[17] R. Yagel, and Z. Shi, *"Accelerating Volume Animation by Space-Leaping"*, Proc. IEEE Visualization, pp. 62-69, 1993.

[18] M.L. Brady, K.K. Jung, H.T. Nguyen, and T.P.Q. Nguyen, *"Interactive volume navigation"*, IEEE Trans. Visualization and Computer Graphics, Vol. 4, No. 3, 1998.