IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 14, 2020, accepted April 29, 2020, date of publication May 14, 2020, date of current version June 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994378

Depth-of-Field Rendering Using Progressive Lens Sampling in Direct Volume Rendering

JISEON KANG[1], JEONGJIN LEE[2], YEONG-GIL SHIN[1], AND BOHYOUNG KIM[3], (Member, IEEE)
[1]Department of Computer Science and Engineering, Seoul National University, Seoul 08826, South Korea
[2]School of Computer Science and Engineering, Soongsil University, Seoul 06978, South Korea
[3]Division of Biomedical Engineering, Hankuk University of Foreign Studies, Seoul 17035, South Korea

Corresponding author: Bohyoung Kim (bkim@hufs.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (MSIT), Development of AR-Based Surgery Toolkit and Applications, under Grant 2017-0-01815, in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT, and Future Planning under Grant 2017R1A2B3011475, in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) under Grant NRF-2019R1A2C1088900, and in part by the Hankuk University of Foreign Studies Research Fund.

ABSTRACT Direct volume rendering is a widely used technique for extracting information from three-dimensional scalar fields acquired by measurement or numerical simulation. However, the translucency of direct volume rendering to express the internal structure of the volume often makes it difficult to recognize the depth of complex structures. In this paper, we propose a new method for applying depth-of-field effects to volume ray-casting to improve the depth perception. A thin lens camera model is used to simulate rays passing through different parts of lens. The proposed method is implemented in the GPU pipeline with no preprocessing, so any acceleration techniques of volume ray-casting can be applied without restrictions. We also propose a multi-pass rendering framework using progressive lens sampling. This new technique uses a different number of lens samples per pixel, depending on the size of the circle of confusion at the point where each ray intersects the volume data. In the experiments with various data, we demonstrated that higher quality images with better depth perception were generated up to 9x faster than the existing depth-of-field method in direct volume rendering.

INDEX TERMS Computer graphics, computers and information processing, image generation, imaging, depth of field effect, direct volume rendering, ray casting, visualization.

I. INTRODUCTION
Volume visualization is used in various applications that require interactive exploration of volume data. Especially in the medical field, it helps doctors to navigate anatomical structure of a patient quickly and accurately with minimal cognitive effort. Medical imaging data, which is usually obtained by physical measurement devices such as CT and MRI scanners, inherently has noise and inhomogeneity. In addition, the organic shapes and spatial relationships between anatomical structures can be often quite complex to recognize, making medical visualiation difficult [1]. In direct volume rendering, the scalar volumetric data is represented in a semi-transparent color so that the inside of the object can be captured. However, overlapping or hidden internal structures still make the spatial comprehension of the data difficult due to translucency [2]. This makes the depth perception of volume visualization more complicated for inhomogeneous data such as medical image data.

Depth-of-field (DOF) is the range of the object's distance through which the eye or camera sees the object in focus. Conversely, as the object is far away from the range, it gradually blurs. This is one of the ways how humans perceive depth, and it has been used in computer graphics to enhance realism or draw attention to certain areas. The DOF effect in ray tracing, which uses stochastic sampling to sample the aperture of a lens to produce a high quality offline image, has been studied and is considered as ground truth [3]. In the surface rendering field, a lot of research is aimed at achieving DOF effects in real time for use in games. Most researches focus on post-processing a pinhole image using blur filters based on its depth map [4], [5]. However, in volume rendering fields,

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

DOF effects have been rarely researched. One well-known DOF research in volume rendering is by Schott *et al.* [6], which applies DOF effects to slice-based volume rendering. The DOF effect can be easily integrated into slice-based volume rendering by applying a blur filter between slices. However, in some cases, this technique produces images with excessive blurring. Except for this study, volume rendering techniques with DOF, especially with respect to volume ray-casting, have been rarely explored. Considering that the volume ray-casting has fewer restrictions and more acceleration methods than slice-based methods [7], the DOF effect applied to the volume ray-casting is more useful in realistically visualizing medical volume data in real time.

In this paper, we introduce a real-time DOF volume ray-casting method providing a realistic visualization. The proposed method is GPU-friendly and easily integrated into the volume ray-casting. It also matches well various acceleration techniques related to the volume ray-casting. We enable additional acceleration through progressive lens sampling, which allows us to obtain DOF images in real time while preserving image quality. The most important advantage of the proposed method is that it is based on reliable DOF image synthesis, ray tracing using lens sampling. The proposed method can be used in medical applications, such as preoperative planning or education with virtual or augmented reality systems [8].

The contributions of this work are:

- A new approach using a thin lens model to simulate realistic depth-of-field effects in direct volume rendering.
- A fast depth-of-field rendering method for GPU-based volume ray casting without any preprocessing or restrictions on its existing acceleration techniques.
- A new acceleration technique that uses multi-pass rendering with progressive lens sampling based on a circle of confusion.

This paper is organized as follows: In Section II, previous works related to DOF are discussed. Section III describes the thin lens model, a camera model for DOF effects used in computer graphics, and volume visualization techniques. Section IV proposes how to integrate the thin lens model into the volume ray-casting and accelerate it using progressive lens sampling. Implementation details are described in Section V and the experimental results are presented in Section VI. Finally, Section VII concludes the paper with future works.

II. RELATED WORKS

Traditional computer graphics use a pinhole camera model that creates focused images at all depths. However, human eyes and real cameras collect light that passes through a finite-size lens. Only objects located within depth of field are in focus and other objects are gradually blurred as they are far away from the focused distance. Since the introduction of a lens and aperture camera model [9], research on DOF effects have been actively conducted. The results thus far are divided into two categories: the image-space approach and the object-space approach.

The image-space approach is post-processing. First a pinhole image is rendered and at the same time the depth map is stored in an additional buffer. Then each pixel of the pinhole image is processed by the blur filter according to the size of the circle of confusion calculated with depth information in the depth map. This approach is very fast and is commonly used to generate immersive scenes in games. The problem is that the calculation time increases rapidly as the aperture diameter of the lens increases. In addition, due to the limitation of approximation using information of surrounding pixels, intensity leakage (i.e., pixel intensities in a focused area flowing into a blurred area) and blurring discontinuity (i.e., discontinuity around blurred boundaries in front of a focal plane) inevitably occur in an region where objects partially overlap [5]. In order to overcome these problems, several methods have been proposed, such as assigning pixel weights [5], [10], dividing a pinhole image into multiple layers based on its depth [4], and storing various information about one pixel [11], [12].

The object-space approach simulates the rays that originate from an object, pass through different parts of the lens, and reach the image plane. Cook *et al.* [3] proposed the *distributed ray tracing* (DRT) that spreads ray samples on the lens to gain DOF effects in ray tracing. This method produces optically correct results, eliminating the problems of the image-space approach. However, a large number of lens samples are required to obtain a high quality image, and the calculation time increases in proportion to the number of samples. Therefore, to improve the rendering speed, a method of using a hardware accumulation buffer [13] and a method selecting a level-of-detail according to the object distance [14] have been proposed. However, it's still difficult to reach real time.

All of the DOF rendering studies presented earlier are surface rendering techniques. For volumetric data, there is a similar study in angiography visualization that blurs out portions exceeding a certain threshold distance [15]. This method borrows DOF concepts to improve depth perception without simulating the actual DOF effect. Schott *et al.* [6] is almost the only and representative way to implement DOF effects for volume visualization. Schott's method is based on slice-based volume rendering and integrates blur filters into the slice compositing process, like the image-space approach. Volume slice sets are divided into front and rear subsets of the focal plane and composited separately. As each subset is synthesized sequentially, incremental filtering is performed to sample and accumulate previous composite results. The advantage of this method is that it requires only minor modifications to the conventional slice-based volume rendering without preprocessing. However, as the compositing stage progresses, blur is repeatedly applied to already-synthesized slices, which could result in exaggerated blurring. Regarding another volume visualisation technique, i.e., volume ray-casting popularly known as GPU-based

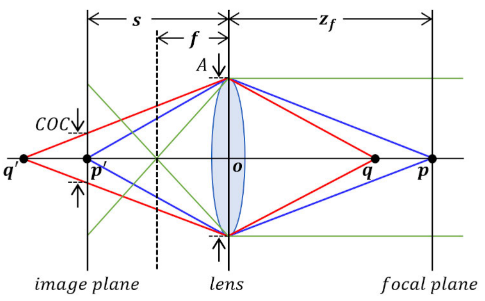


FIGURE 1. **Illustration of the thin lens model. Point *p* is on the focal plane at distance z_f and blue rays from point *p* gather at point *p′* on the image plane. Red rays starting at point *q* out of the focal plane form a blurry circle on the image plane (*circle of confusion*, COC) at distance *s* [16].**

volume rendering, to the best of our knowledge the DOF research has not been proposed.

III. FUNDAMENTALS

A. THE THIN LENS MODEL

The way to create DOF effects in computer graphics is to use the thin lens model instead of the conventional pinhole camera model. The thin lens model assumes that the lens is infinitely thin, which means that the refraction is ignored inside the lens. Equation (1) is the thin lens equation.

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{z_f}, \tag{1}$$

where f is the focal length, s is the distance from the lens to the image plane, and z_f is the distance from the lens to the focused object.

Fig. 1 shows the path of rays in the thin lens model. Rays starting at the point p on the focal plane pass through the lens and form the point p' on the image plane. Rays starting at the point q outside the focal plane gather at the point q' at a place other than the image plane [17]. On the image plane, it appears as a blurry circle called as the *circle of confusion* (COC). The DOF is defined as a range of distances from the lens to an object where the COC is so small that objects appear sharply. The diameter $c(z)$ of COC, which is the diameter of COC for the point at the distance z, is calculated using (2) [6].

$$c(z) = \left| A\frac{f(z_f - z)}{z(z_f - f)} \right|, \tag{2}$$

where z is the distance from the lens to the object and A is the diameter of the aperture.

B. VOLUME VISUALIZATION

Volume visualization provides insight to users by displaying volumetric data in two-dimensional images. Unlike surface rendering where objects are represented as surfaces with material properties, volume rendering deals with the structure or unstructured three-dimensional data and shows both the interior of the material and the boundary between materials [18], [19]. Volume rendering algorithms can be grouped into two categories: indirect volume rendering and direct volume rendering.

Indirect Volume Rendering (IVR) techniques use geometric primitives (most commonly triangles) to approximate surfaces contained in volumetric data, which can be rendered using conventional graphics accelerator hardware. The marching cubes algorithm [20] is mainly used for isosurface extraction. Since only a surface representation is used, much of the information contained in the data may be lost, and an excessive amount of geometric primitives may be required for proper surface approximation [21].

Direct volume rendering (DVR) techniques create a result image directly from relevant voxels of the volumetric data. Slice-based (or texture-based) volume rendering is performed by slicing the 3D texture block that stores the volumetric data into proxy planes oriented parallel to the view plane [21]. Volume ray-casting [21], [22] tracks rays from camera into the volume, computing the volume-rendering integral along these rays. Volume ray-casting is more widely used than slice-based approaches because rays can be handled independently from each other, allowing for several optimizations strategies [7].

The DOF rendering in DVR proposed in this paper is based on volume ray-casting. The proposed method is compared with DRT [3], which is considered ground truth, to see that it synthesizes the appropriate depth-of-field effects. DRT is a ray tracing technique for surface rendering, but can be applied to volume rendering by borrowing the concept of sampling a lens and generating multiple rays from one pixel. In addition, the proposed method are compared to the existing DOF method [6] based on slice-based volume rendering of DVR, regarding the image quality and rendering performance.

IV. PROGRESSIVE DEPTH-OF-FIELD VOLUME RAY-CASTING

In this section, we present a rendering method to create DOF effects in volume ray-casting. Our method performs three-pass rendering with progressive lens sampling to use different numbers of lens samples per pixel. In the first render pass, DOF effects are generated with the minimum number of lens samples. And in the second and third render passes, the aliasing is suppressed by adding lens samples to some pixels where the COC size is expected to be large.

The proposed method works on the GPU pipeline without any preprocessing. Phong shading, perspective projection, and *early ray termination* (ERT) [22] are applied by default.

A. GEOMETRY SETUP

The basic idea of volume ray-casting is to generate a ray towards the volume for each pixel and calculate the volume-rendering integral along the ray [7]. There are a variety of optimization techniques that take advantage of the independent characteristics of the rays, which can be directly used in reducing the time required for calculating time-consuming DOF effects. To integrate DOF effects into

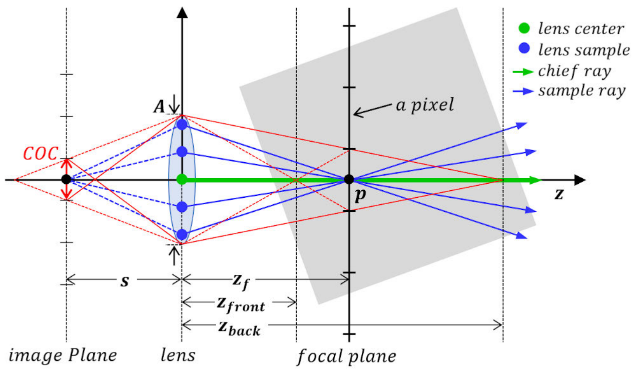


FIGURE 2. Volume ray-casting based on the thin lens model for one pixel value. First we find the intersection of the focal plane and the chief ray (point *p*). Then trace sample rays from lens samples to the intersection point *p*. The final pixel color is the average of the volume-rendering integrals of all the sample rays. The distance interval [z_{front}, z_{back}] where COC is less than or equal to the pixel size is for three-pass progressive DOF rendering.

volume ray-casting, we simply place a thin lens at the camera position and add a focal plane as shown in Fig. 2.

In the real pinhole camera, the image plane and objects are located on the opposite sides relative to the pinhole. In computer graphics, on the other hand, the image plane is placed between the camera and objects to prevent the resulting image from flipping. Like the real pinhole camera, the thin lens model also has the image plane and objects in opposite directions relative to the lens. The application of the lens equation (1) can be confusing when the image plane is on the same side as the objects. Therefore, it is better to use the image plane only conceptually, and think of the focal plane acting as the image plane to generate rays per pixel [16].

Fig. 2 shows how the thin lens model generates rays for one pixel in image plane. The focal plane is placed at the focal distance z_f, set by user input, perpendicular to the optical axis. DOF volume ray-casting begins by generating a chief ray per pixel in the same way as the conventional volume ray-casting (a green line in Fig. 2). After finding the position p where the chief ray intersects the focal plane, sample rays start from lens samples (blue dots on lens in Fig. 2) and then pass through the focal point p (blue lines in Fig. 2). All sample rays compute the volume-rendering integral individually, and then all the volume-rendering integrals of all sample rays are averaged to determine the final pixel color.

GPU-based volume ray-casting uses front-to-back compositing with ERT, so the more opaque is the volume, the faster the rendering ends. In the focus range [z_{front}, z_{back}] where the COC is smaller than the pixel size (red lines in Fig. 2), all sample rays pass through the volume along roughly the same path, which is similar to the path of the chief ray. Therefore, if the boundary of the volume at which the sample rays start is in this range, it produces almost the same result as the conventional volume ray-casting. This means that we can apply different numbers of lens samples for each pixel, depending on whether the ray's starting point falls within the range [z_{front}, z_{back}] or not. More details are covered in the next section.

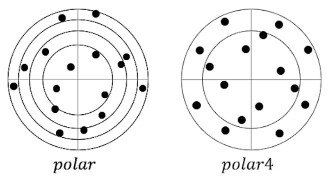


FIGURE 3. Disk sampling strategies. Left: polar mapping. Right: polar4 mapping.

B. PROGRESSIVE LENS SAMPLING

To get a high-quality DOF rendering without aliasing, we need to use a sufficient number of lens samples. However, the calculation time increases in proportion to the number of lens samples. Therefore, it is necessary to select an appropriate sampling technique that uses a small number of lens samples without compromising the image quality. In a recent study, Christensen [23] proposed a method for effectively sampling a disk light source in ray tracing. Referring to Christensen's paper, we choose the combination of *polar4* mapping and Owen-scrambled Sobol' (0,2) sequence for progressive lens sampling. This combination provides the best results among progressive disk sampling strategies where it is important to distribute samples randomly and evenly over unit circles.

Polar mapping is the most commonly used disk sampling that converts a point (u, v) on a unit square to a unit circle with $r = \sqrt{u}$ and $\phi = 2\pi v$. But, it creates rather irregular shapes with poor aspect ratios. To overcome this problem, *polar4* mapping is proposed with some changes in *polar* mapping. In the *polar4* mapping, samples are mapped from the unit square only to the quarter-circle wedge in the first quadrant. Then, the samples in the first quadrant are rotated by 90, 180, and 270 degrees, adding new samples to the remaining quadrants. As shown in Fig. 3, the samples of the *polar4* mapping (right) are more evenly distributed throughout the circle than those of the *polar* mapping (left).

Sample patterns are divided into two categories: sample sets and sample sequences. Sample sets consist of finite unordered sample points, and sample sequences consist of infinite ordered sample points. Progressive (hierarchical) sample sequences are well distributed with any prefix of the entire sequence [24]. Christensen's paper [23] confirmed that Owen-scrambled Sobol' (0,2) [25] is one of the best progressive sample sequences showing even distribution when used with the *polar4* mapping. So we generate a progressive sample sequence with Owen-scrambled Sobol' (0,2) and split it into three subsequences for three-pass rendering. The length of each subsequence is set to the number of samples accumulated up to the previous pass, which makes each render pass use twice the number of samples used up to the previous pass. In Fig. 4, the samples newly used in each render pass are shown as filled circles, and the samples already used in the previous passes are shown as open circles. Each render pass adds new samples to be distributed well including the previous ones, producing a natural resulting image even if the final render pass is different for each pixel.

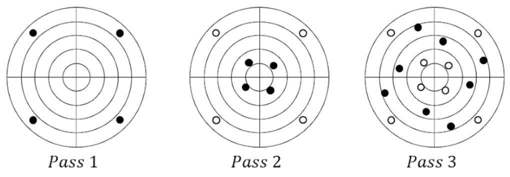


FIGURE 4. Progressive lens sample sequence of three-pass rendering. 16 samples generated with Owen-scrambled Sobol' (0,2) are mapped to a unit circle using polar4 mapping. Filled circles represent the sample points newly added in each render pass. Open circles represent the sample points used in the previous pass.

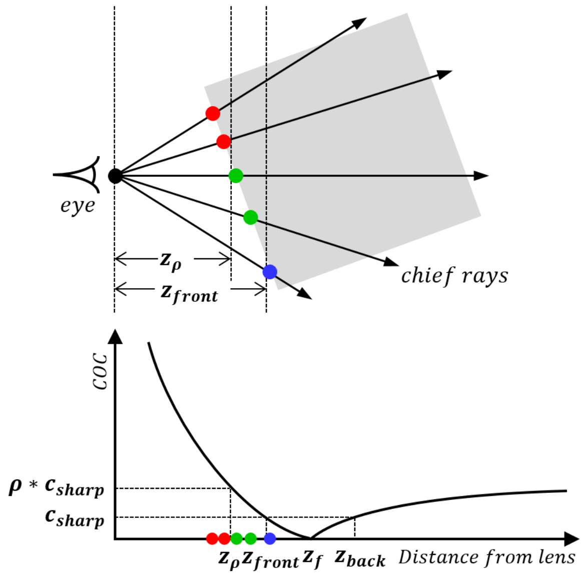


FIGURE 5. COC-based progressive rendering determines the number of render passes, based on the starting point of the chief ray in the COC graph (bottom): blue points go through one pass, green points do two passes, and red points do three passes.

Fig. 5 shows how to determine the final render pass for each pixel based on the COC. $[z_{front}, z_{back}]$ is the DOF where the COC size is smaller than the pixel size c_{sharp}. This range can be obtained using (3) and (4). The pixel size, denoted by c_{sharp} is calculated using (5).

$$z_{front} = \frac{Afz_f}{Af + c_{sharp}(z_f - f)} \tag{3}$$

$$z_{back} = \frac{Afz_f}{Af - c_{sharp}(z_f - f)} \tag{4}$$

$$c_{sharp} = \frac{2}{h} \times s \times \tan(\frac{FOV}{2}), \tag{5}$$

where h is the height of viewport, s is the distance from lens to the image plane, and FOV is user's field of view.

As the object moves far away from the focal distance z_f, the size of the COC increases (see the COC graph in Fig. 5). The size of the COC increases dramatically, especially when the object moves toward the eye from the focal distance. This indicates that as the object before the DOF is closer to the eye, the more aliasing will appear if a sufficient number of lens samples are not used. Therefore, we add a distance boundary z_ρ using (6) in front of z_{front} to set more lens samples near the eye. If the starting point of each pixel's chief ray is farther than z_{front}, only one pass is rendered. But if the starting point of the chief ray is between $[z_\rho, z_{front}]$, it renders up to two passes; and if less than z_ρ, it renders three passes. ρ in (6) is a constant that determines the level of progressive renderings. If the COC is greater than ρ times the pixel size, the rendering goes up to three passes. $\rho(\geq 1)$ is empirically set to obtain an image without aliasing, depending on the data context and the transfer function used (described in detail in Section VI).

$$z_\rho = \frac{Afz_f}{Af - \rho \times c_{sharp}(z_f - f)} \tag{6}$$

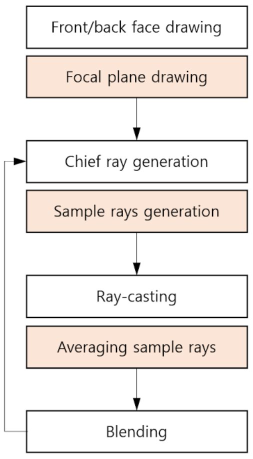


FIGURE 6. Rendering pipeline of the progressive DOF ray-casting. Processes newly added to the conventional volume ray-casting are shaded.

V. IMPLEMENTATION DETAIL

GPU-based volume ray-casting stores the entire volume data in a 3D texture and performs the ray-casting in the fragment shader for each pixel [26]. Fig. 6 shows a block diagram of progressive DOF volume ray-casting proposed in this paper, which extends the GPU-based volume ray-casting. The whole process consists of four steps, with the additional processes for DOF effects shaded in Fig. 6.

In the first step, we create a *volume bounding box* (VBB) of which vertices are the 3D texture coordinates of the volume. Then in the viewing direction of the camera, the front and back faces of VBB are rendered and stored into 2D textures, respectively. This process utilizes the hardware rasterization of the GPU to quickly calculate the start and end positions of chief rays. In addition to this, the focal plane that fills the entire screen is rendered and stored into a 2D texture. At this point, each vertex color of the focal plane is set to the focal distance given by the user, expressed in 3D texture coordinates of the volume.

At the beginning of the second step, we check whether the starting position of the chief ray is within the range of the corresponding render pass in Section IV-B. In this step, we compute the chief ray by extracting the start coordinates from the VBB front texture and the end coordinates from the VBB back texture. Then, after obtaining the focal point

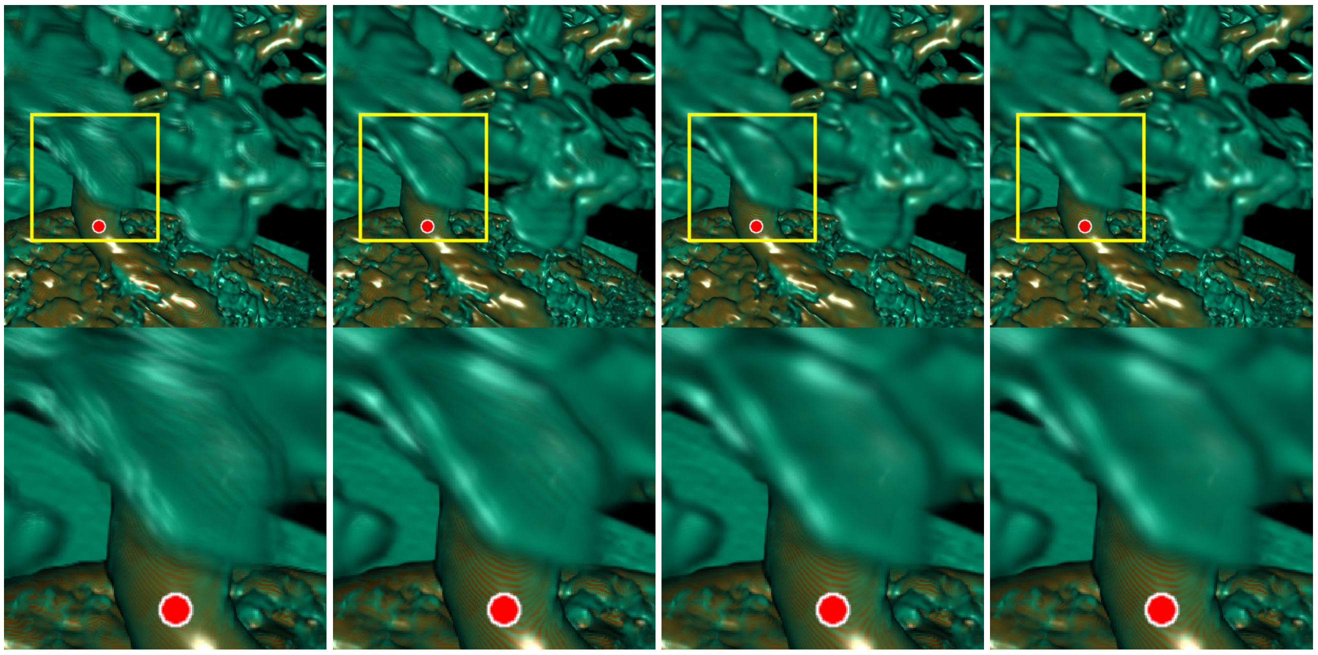

(a) 4 samples (1-pass), 134.74 fps (b) 8 samples (1-pass), 52.27 fps (c) 16 samples (1-pass), 13.90 fps (d) 16 samples (3-pass), 30.19 fps

FIGURE 7. Renderings of bonsai data set (256 × 256 × 256 voxels) generated by DOF volume ray-casting. (a), (b), and (c) 1-pass DOF volume ray-casting with 4, 8, and 16 lens samples, respectively. (d) 3-pass DOF volume ray-casting with 16 progressive lens samples. The bottom row shows a close-up image of the yellow square region in the top row. The red dot indicates the point in focus. In each image, gradients were computed on the fly.

Algorithm 1 DOF Ray-Casting Pipeline

```
Data: Volume, TransferFunction, FocalPlane
1  $chiefRay \leftarrow$ a primary DVR ray
2  $p \leftarrow$ intersection between $chiefRay$ and FocalPlane
3  $sampleRays[N] \leftarrow$ from $lensSamples[N]$ passing through $p$
4  while a marching position $x$ of $chiefRay$ is in $Volume$ do
5      $accumulated_a \leftarrow 0$
6      foreach $s = 0$ to $N$ do
7          $(intensity, gradient) \leftarrow$ Texture3D$(Volume, x)$
8          $shaded_{rgba} \leftarrow$ Shading$(TransferFunction, intensity, gradient)$
9          $composited[s]_{rgba} \leftarrow$ Compositing$(shaded_{rgba})$
10         $accumulated_a += composited[s]_a$
11         march $sampleRays[s]$
12     end
13     if $accumulated_a >= (N - 0.1)$ then
14         goto line 18
15     end
16     march $chiefRay$
17 end
18 $fragColor \leftarrow$ mean of $composited[N]_{rgba}$ values
```

coordinates from the focal plane texture, sample rays are calculated starting from the lens samples and passing through the focal point. The lens samples in Fig. 4 are used according to the current render pass.

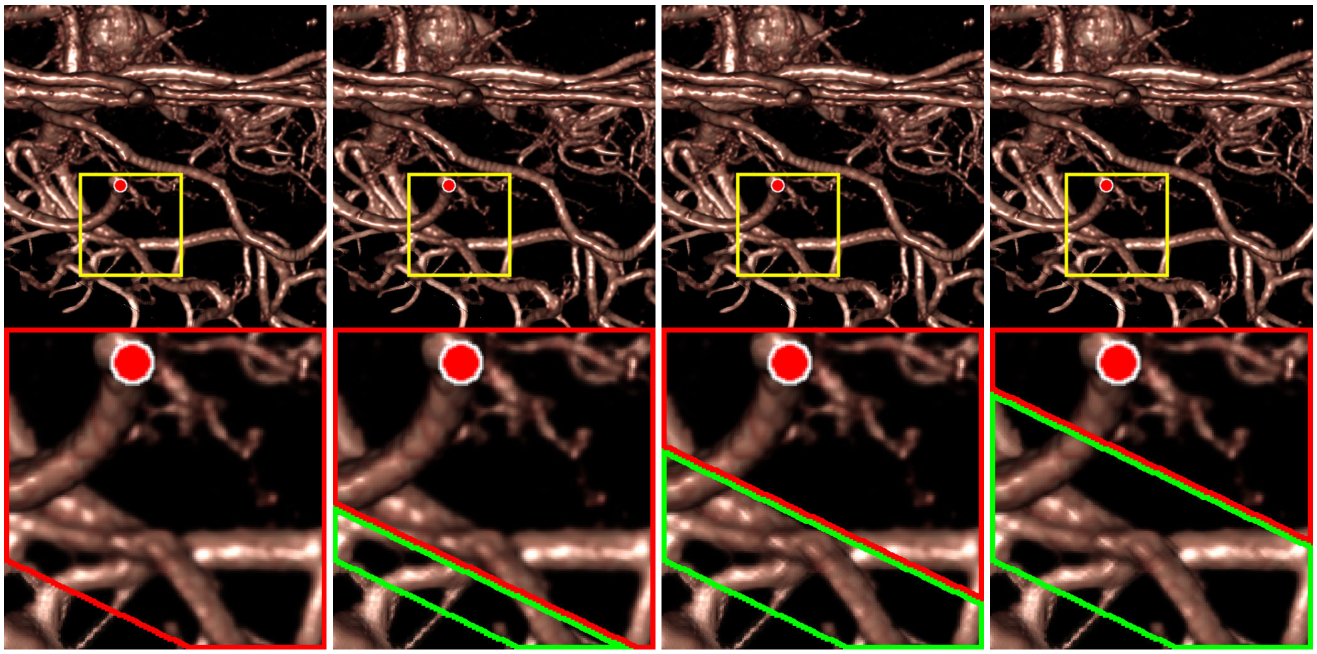
In the third step, the volume ray-casting of each sample ray is performed. When sample rays are cast, their volume-rendering integrals are calculated and accumulated individually. If the chief ray is out of the volume or the opacity of all the sample rays saturates, the ray-casting loop stops and the integral results of all the sample rays are averaged. Note that the chief ray is only used for geometry setup and is not used to calculate the resulting color.

In the final step, the resulting colors from the previous and current render passes are averaged and stored in the render buffer. The same process is repeated going back to the second stage until the final render pass is performed.

VI. EXPERIMENTAL RESULTS

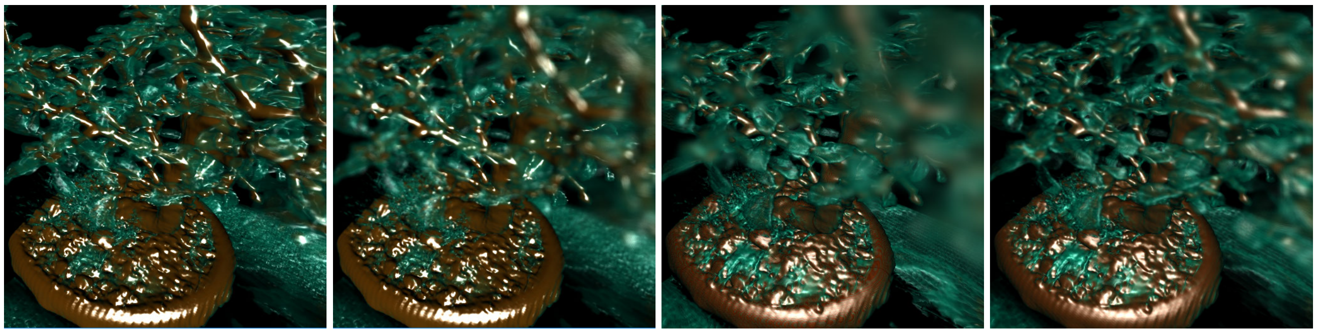
The proposed method and the slice-based DOF volume rendering [6] were implemented using OpenGL/GLSL on a personal computer with an Intel i7-770K 4.2GHz CPU (32GB RAM) and NVIDIA GeForce GTX 1080 GPU. DRT [3] images were rendered using the Intel OSPRay [27], [28] ray tracing engine on the same environment. All images were rendered at 512×512 resolution and a simple one-dimensional transfer function widely used for each volume data was applied. DOF effects of the slice-based DOF method [6] were generated by taking 2×2 samples when blurring during slice traversal.

Fig. 7 shows images of the bonsai data set rendered by DOF volume ray-casting with various sample counts. The focus is on the tree trunk so that the leaves closer to the camera are blurred. As the number of lens samples increases, anti-aliasing effects occur, resulting in a more natural blur (Fig. 7 (a)-(c)). However, the frames per second (fps)



(a) $\rho = 1.0$, 25.57 fps (b) $\rho = 1.5$, 25.97 fps (c) $\rho = 2.0$, 26.47 fps (d) $\rho = 2.5$, 27.05 fps

FIGURE 8. Renderings of aneurysm data set (256 × 256 × 256 voxels) generated by progressive DOF volume ray-casting while varying the parameter ρ. The bottom row shows a close-up image of the yellow square region in the top row. The areas enclosed by the green and red borders in the second row represent up to 2-pass renderings (8 samples total) and up to 3-pass renderings (16 samples total), respectively. As the value of ρ increases from left to right, some areas rendered with 3-pass rendering change to 2-pass rendering. The red dot indicates the point in focus. In each image, gradients were computed on the fly.

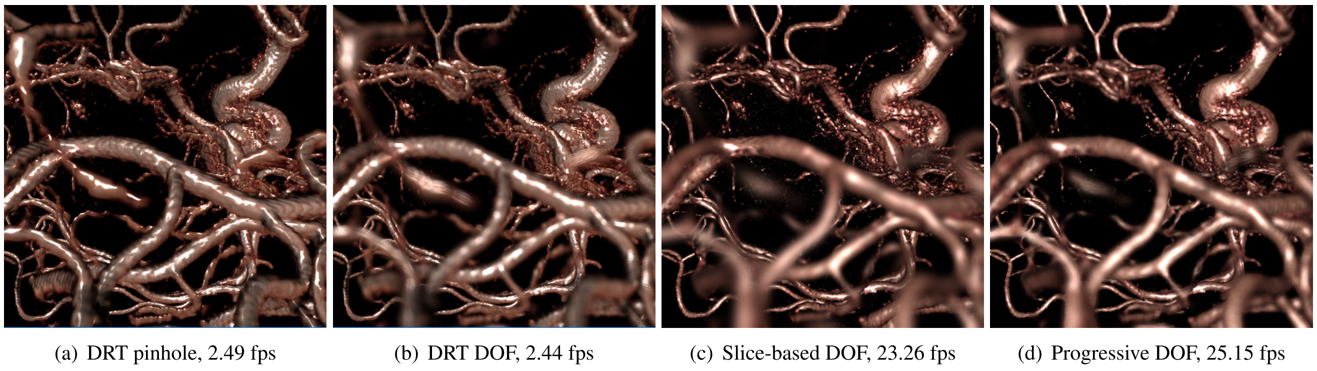


(a) DRT pinhole, 2.92 fps (b) DRT DOF, 2.42 fps (c) Slice-based DOF, 21.42 fps (d) Progressive DOF, 22.44 fps

FIGURE 9. Renderings of bonsai data set (256 × 256 × 256 voxels) generated by (a) distributed ray tracing with pinhole camera, (b) distributed ray tracing with DOF [3], (c) slice-based DOF volume rendering [6] with the COC angle $\alpha = 88°$ and 2 × 2 samples taken during the incremental filtering, and (d) progressive DOF volume ray-casting with the aperture size $A = 0.03$. The focus in the images, except for (a), is on the bottom of the tree trunk and the depth of field is similar. In each image, gradients were computed on the fly.

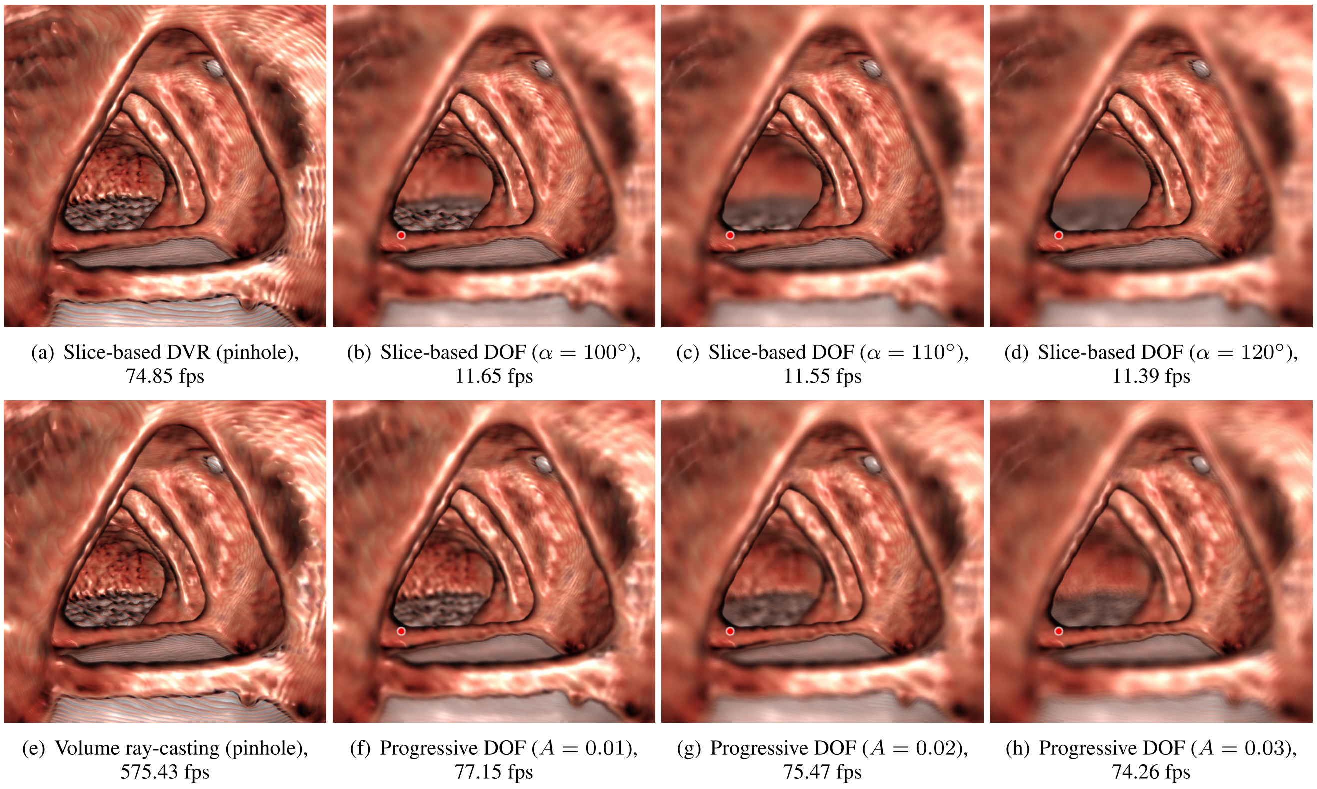
drops inversely proportional to the number of lens samples. As shown in Fig. 7 (c) and (d), our DOF ray-casting using progressive lens sampling (Fig. 7 (d)) exhibits image quality similar to the rendering with 16 lens samples (Fig. 7 (c)) while providing over 2x faster rendering.

Fig. 8 shows the change in image quality and frame rate as the constant ρ in (6) changes. The yellow square in each image in the first row contains pixels whose maximum render pass varies greatly. In the second row images, the area enclosed by the red border represents the pixels rendered in up to three passes using the maximum number of lens samples (16 samples). The area enclosed by the green border contains the pixels rendered in up to two passes (8 samples). If ρ is 1.0 (Fig. 8 (a)), pixels whose COC size is larger than the pixel size are rendered in up to three passes. $\rho = 2.0$ (Fig. 8 (c)) means that three passes are performed on pixels whose COC size is larger than twice the pixel size, and two passes are performed on pixels whose COC size is larger than the pixel size and less than twice the pixel size. As the value of ρ increases from left to right, some areas rendered with 3-pass rendering change to 2-pass rendering. The frame rate slightly increases with varying ρ, but the difference in image quality is unnoticeable. This is due to the fact that the number of lens samples is determined relative to the volume boundary as shown in Fig. 5, whereas the voxels affecting the result image are concentrated near the center of the volume. Therefore, we fixed the ρ to a specific



(a) DRT pinhole, 2.49 fps (b) DRT DOF, 2.44 fps (c) Slice-based DOF, 23.26 fps (d) Progressive DOF, 25.15 fps

FIGURE 10. Renderings of aneurysm data set (256 × 256 × 256 voxels) generated by (a) distributed ray tracing with pinhole camera, (b) distributed ray tracing with DOF [3], (c) slice-based DOF volume rendering [6] with the COC angle $\alpha = 70°$ and 2 × 2 samples taken during the incremental filtering, and (d) progressive DOF volume ray-casting with the aperture size $A = 0.025$. The focus in the images, except for (a), is on the thickest blood vessel at the top right of the images and the depth of field is similar. In each image, gradients were computed on the fly.



(a) Slice-based DVR (pinhole), 74.85 fps (b) Slice-based DOF ($\alpha = 100°$), 11.65 fps (c) Slice-based DOF ($\alpha = 110°$), 11.55 fps (d) Slice-based DOF ($\alpha = 120°$), 11.39 fps

(e) Volume ray-casting (pinhole), 575.43 fps (f) Progressive DOF ($A = 0.01$), 77.15 fps (g) Progressive DOF ($A = 0.02$), 75.47 fps (h) Progressive DOF ($A = 0.03$), 74.26 fps

FIGURE 11. Renderings of colon CT data set (512 × 512 × 672 voxels). (a) Slice-based volume rendering with pinhole camera. (b), (c), and (d) Slice-based DOF volume rendering [6] with various COC angles α. (e) Volume ray-casting with pinhole camera. (f), (g), and (h) Progressive DOF volume ray-casting using various sizes of the lens aperture A. The DOF images, except for (a) and (e), have the same focal distance $z_f = 0.185$, and have a similar depth of field in the same column. The red dot indicates the point in focus. In each image, gradients were computed on the fly.

value of 1.4 because it does not noticeably affect the resulting image.

Fig. 9 shows the results when each DOF rendering method has the same depth-of-field for the bonsai data set. Fig. 9 (b) created by DRT [3] is considered as ground truth. Unlike Fig. 9 (a) that focuses on all distances, the leaves close to the camera appear blurry because the tree trunk is in focus. As shown in Fig. 9 (d), the proposed method provides as natural DOF blurring in out-of-focus leaves as DRT DOF (Fig. 9 (b)) with about 9x faster rendering speed. In contrast, in Fig. 9 (c) generated by slice-based DOF volume rendering [6], most of the out-of-focus leaves are over-blurred. As authors mentioned in their paper [6], this is due to the incremental filtering in which the first slice continuously blurred during the slice traversal. The bonsai data set contains many transparent voxels (green leaves), which reduces the effect of ERT in ray-casting method, so the rendering speed improvement of our proposed progressive method (Fig. 9 (d)) is not noticeable compared to the slice-based method [6] (Fig. 9 (c)).

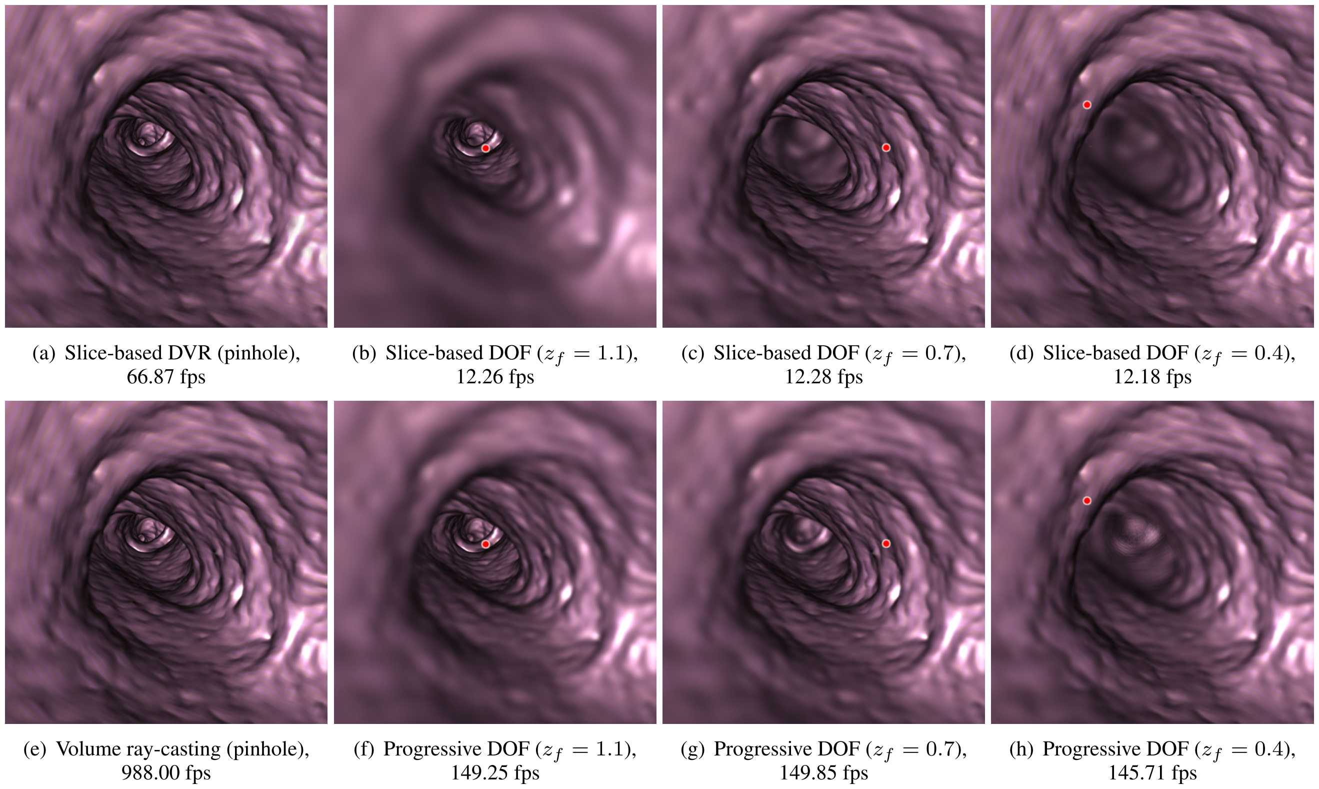

(a) Slice-based DVR (pinhole), 66.87 fps
(b) Slice-based DOF ($z_f = 1.1$), 12.26 fps
(c) Slice-based DOF ($z_f = 0.7$), 12.28 fps
(d) Slice-based DOF ($z_f = 0.4$), 12.18 fps
(e) Volume ray-casting (pinhole), 988.00 fps
(f) Progressive DOF ($z_f = 1.1$), 149.25 fps
(g) Progressive DOF ($z_f = 0.7$), 149.85 fps
(h) Progressive DOF ($z_f = 0.4$), 145.71 fps

FIGURE 12. Renderings of bronchus CT data set (512 × 512 × 768 voxels). (a) Slice-based volume rendering with pinhole camera. (b), (c), and (d) Slice-based DOF volume rendering [6] with various focal distances z_f. (e) Volume ray-casting with pinhole camera. (f), (g), and (h) Progressive DOF volume ray-casting with various focal distances z_f. The images in each column have the same focal distance z_f and a similar depth of field. The red dot indicates the point in focus. In each image, gradients were computed on the fly.

Fig. 10 shows the results of each DOF rendering method for the aneurysm data set. Due to the data characteristics that the voxels of interest are mapped to almost one color, it is difficult to recognize the anteroposterior relationship between complex vessels. However, DOF images (Fig. 10 (b)-(d)) created with focus on the thickest blood vessel at the top right of the screen enhance depth perception compared to the image using pinhole cameras (Fig. 10 (a)). In Fig. 10 (c) generated by the slice-based method [6], there are problems that the blood vessel at the bottom of the screen is excessively blurred and the thin blood vessel in the center of the screen seems to focus. On the other hand, in the Fig. 10 (d) created by the proposed method, the blood vessels are properly blurred with distance. Our progressive DOF method produces as realistic DOF effects as DRT DOF (Fig. 10 (b)) with considerably faster (about 10x) rendering speed.

Fig. 11 is the best case demonstrating the advantages of the proposed method, where the long tube shape of the colon shows DOF effects well. The focal distance is the same for all images except the pinhole images, and the focus is on the protruding portion between the small pouches and the haustra in the center of the screen. In the first row, the DOF images (Fig. 11 (b)-(d)) were created using the slice-based DOF method [6] with incremental filtering of different COC angle α. In the second row, the DOF images (Fig. 11 (f)-(h)) were created using the proposed method with a thin lens of which aperture size A was adjusted to have the same depth-of-field as in the slice-based DOF image in the first row of the same column. In the images of the proposed method, the white implant and the lining of colon are blurred gradually as they move away from focus. And, as the lens aperture increases from (f) to (h) in Fig. 11, blurring in out-of-focus areas naturally increases. In contrast, in the images of the slice-based DOF method, the out-of-focus areas appear to be somewhat discontinuously blurred as the COC angle increases from (b) to (d) in Fig. 11. The camera in Fig. 11 is placed inside the volume to virtually navigate the interior of the colon, delivering the virtual colonoscopy images. Because this virtual endoscope creates an image that fills the entire screen, rays at each pixel are rapidly saturated in the ray-casting technique. On the other hand, in the slice-based rendering, a lot of slices are still required to be incrementally filtered for the rendering of enlarged small region, resulting in significantly reduced rendering speed. Regarding the rendering speed, the progressive DOF method is up to 6x faster than the slice-based DOF method [6].

Fig. 12 shows images of a bronchoscope, another example of a virtual endoscope. The images in the first row (Fig. 12 (a)-(d)) were rendered by the slice-based method using pinhole camera and Schott [6]'s incremental filtering. The focal distance is shortened from (b) to (d) in Fig. 12: the focus in (b) is on the end of the left primary bronchus, and the focus in (d) is near the camera where two primary bronchi begin to branch off. When trying to set the depth-of-field to a short range, the slice-based DOF method [6] tended to cause excessive blurring in out-of-focus areas. In the second row, the images (Fig. 12 (e)-(h)) were rendered by volume ray-casing using pinhole camera and our progressive lens sampling. The focal distance of each image is the same as in the slice-based DOF image in the corresponding column. Comparing corresponding image pairs of the same depth-of-field, the proposed method produces an image that improves depth perception approximately 7 to 9x faster than the slice-based DOF method [6]. In Fig. 12 (h), aliasing appears slightly in the region near the end of the left primary bronchus, which is caused by the lack of samples at a distance farther than the focal distance. This is because, as mentioned in Section IV-B, the algorithm was designed with a focus on the rapid change of the COC size in front of the focal plane. Increasing the number of lens samples used in the first render pass can solve this problem; however, this would slow down the rendering speed.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose an effective depth-of-field rendering method in direct volume rendering. The proposed method integrates the thin lens model into GPU-based volume ray-casting to generate DOF effects. It extends the volume ray-casting pipeline by adding some processes that simulate rays passing through the thin lens and calculate the volume-rendering integrals. For acceleration, we apply ERT, which is basically used in the volume ray-casting, to our DOF ray-casting. Also, we suggest a 3-pass rendering using progressive lens sampling for an additional acceleration. As the ray's starting point is closer to the camera, each pixel value is updated with more lens samples during the three render passes. The proposed method provided DOF effects of better quality than the existing DOF volume rendering [6], and in some cases, showed a frame rate up to 9x higher.

As demonstrated in experimental results, the proposed method allows a real-time rendering in most cases. Especially when the camera position moves inside the volume, the DOF effects greatly help to identify the context of complex structures at fairly high frame rates, without any excessive blur of the out-of-focus regions. In addition, the proposed method allows natural control over the blurring of out-of-focus areas and thus it can be used to visualize medical images where the context around the region of interest is very important. Furthermore, unlike Schott's method [6], which can focus only within the volume, the proposed method has the advantage that the focus position can be placed anywhere.

In the future, we are planning to exploit various acceleration techniques such as deferred rendering and empty space skipping. These techniques could be used to improve our three-pass rendering using progressive lens sampling because they can find the voxels closest to the camera faster. Also, as all rays in our DOF ray-casting are independent, we are considering to implement our method on modern graphics cards that support much faster ray casting.

ACKNOWLEDGMENT

The ICT at Seoul National University provides research facilities for this study.

REFERENCES

[1] B. Preim, A. Baer, D. Cunningham, T. Isenberg, and T. Ropinski, "A survey of perceptually motivated 3D visualization of medical image data," *Comput. Graph. Forum*, vol. 35, no. 3, pp. 501–525, Jun. 2016, doi: 10.1111/cgf.12927.

[2] R. Englund and T. Ropinski, "Quantitative and qualitative analysis of the perception of semi-transparent structures in direct volume rendering: Analysis of perception in DVR," *Comput. Graph. Forum*, vol. 37, no. 6, pp. 174–187, Sep. 2018, doi: 10.1111/cgf.13320.

[3] R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," *ACM SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 137–145, Jul. 1984, doi: 10.1145/964965.808590.

[4] M. Kraus and M. Strengert, "Depth-of-Field rendering by pyramidal image processing," *Comput. Graph. Forum*, vol. 26, no. 3, pp. 645–654, Sep. 2007, doi: 10.1111/j.1467-8659.2007.01088.x.

[5] S. Lee, G. Jounghyun Kim, and S. Choi, "Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 3, pp. 453–464, May 2009, doi: 10.1109/tvcg.2008.106.

[6] M. Schott, A. V. Pascal Grosset, T. Martin, V. Pegoraro, S. T. Smith, and C. D. Hansen, "Depth of field effects for interactive direct volume rendering," *Comput. Graph. Forum*, vol. 30, no. 3, pp. 941–950, Jun. 2011, doi: 10.1111/j.1467-8659.2011.01943.x.

[7] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf, *Real-Time Volume Graphics*. Wellesley, MA, USA: A K Peters, Ltd., 2006.

[8] K. Abhari, S. H. John Baxter, E. S. Chen, A. R. Khan, C. Wedlake, T. Peters, R. Eagleson, and S. Ribaupierre, "The role of augmented reality in training the planning of brain tumor resection," in *Augmented Reality Environments for Medical Imaging and Computer-Assisted Interventions*. Berlin, Germany: Springer, 2013.

[9] M. Potmesil and I. Chakravarty, "A lens and aperture camera model for synthetic image generation," *ACM SIGGRAPH Comput. Graph.*, vol. 15, no. 3, pp. 297–305, Aug. 1981, doi: 10.1145/965161.806818.

[10] T. Zhou, J. X. Chen, and M. Pullen, "Accurate depth of field simulation in real time," *Comput. Graph. Forum*, vol. 26, no. 1, pp. 15–23, Mar. 2007, doi: 10.1111/j.1467-8659.2007.00935.x.

[11] S. Lee, E. Eisemann, and H.-P. Seidel, "Depth-of-field rendering with multiview synthesis," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–6, Dec. 2009, doi: 10.1145/1618452.1618480.

[12] K. Selgrad, C. Reintges, D. Penk, P. Wagner, and M. Stamminger, "Real-time depth of field using multi-layer filtering," presented at the I3D, Feb. 2015.

[13] P. Haeberli and K. Akeley, "The accumulation buffer: Hardware support for high-quality rendering," *ACM SIGGRAPH Comput. Graph.*, vol. 24, no. 4, pp. 309–318, Sep. 1990, doi: 10.1145/97880.97913.

[14] Y. Jeong, K. Kim, and S. Lee, "Real-time defocus rendering with level of detail and sub-sample blur," *Comput. Graph. Forum*, vol. 32, no. 6, pp. 126–134, Sep. 2013, doi: 10.1111/cgf.12075.

[15] T. Ropinski, F. Steinicke, and K. Hinrichs, "Visually supporting depth perception in angiography imaging," in *Smart Graphics*. Berlin, Germany: Springer, 2006, pp. 93–104.

[16] S. Marschner and P. Shirley, *Fundamentals of Computer Graphics*, 4th ed. New York, NY, USA: A K Peters, 2015, pp. 303–316.

[17] K. Suffern, *Ray Tracing from the Ground Up*. Wellesley, MA, USA: A K Peters, 2016, pp. 167–180.

[18] M. Meißner, H. Pfister, R. Westermann, and C. M. Wittenbrink, "Volume visualization and volume rendering techniques," presented at the Eurographics, 2000.

[19] R. A. Drebin, L. Capenter, and P. Hanrahan, "Volume rendering," *ACM Siggraph Comput. Graph.*, vol. 22, no. 4, pp. 65–74, Jan. 1988.

[20] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM Siggraph Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987, doi: 10.1145/37401.37422.

[21] C. D. Hansen and C. R. Johnson, *The Visualization Handbook*. Burlington, MA, USA: Elsevier, 2011, pp. 125–258.

[22] J. Kruger and R. Westermann, *Acceleration Techniques for GPU-Based Volume Rendering*. Washington, DC, USA: VIS, 2003, pp. 287–292.

[23] P. Christensen, "Progressive sampling strategies for disk light sources," Pixar Animation Studios, Emeryville, CA, USA, Tech. Rep., Sep. 2018.

[24] P. Christensen, A. Kensler, and C. Kilpatrick, "Progressive multi-jittered sample sequences," *Comput. Graph. Forum*, vol. 37, no. 4, pp. 21–33, Jul. 2018, doi: 10.1111/cgf.13472.

[25] J. Matoušek, "On theL2-discrepancy for anchored boxes," *J. Complex.*, vol. 14, no. 4, pp. 527–556, Dec. 1998, doi: 10.1006/jcom.1998.0489.

[26] M. Hadwiger, P. Ljung, C. R. Salama, and T. Ropinski, "Advanced illumination techniques for GPU-based volume Raycasting," presented at the ACM SIGGRAPH 2009.

[27] I. Wald, G. Johnson, J. Amstutz, C. Brownlee, A. Knoll, J. Jeffers, J. Gunther, and P. Navratil, "OSPRay—A CPU ray tracing framework for scientific visualization," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 931–940, Jan. 2017.

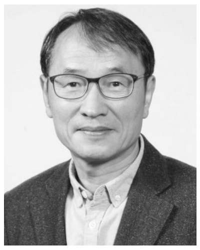
[28] *Intel OSPRay: The Open, Scalable, and Portable Ray Tracing Engine*. Accessed: Aug. 14, 2020. [Online]. Available: https://www.ospray.org

JISEON KANG received the B.S. degree in computer science and engineering from Ewha Womans University, Seoul, South Korea, in 2009. She is currently pursuing the Ph.D. degree in computer science and engineering with Seoul National University, Seoul.

From 2007 to 2008, she was an Intern with the LG Innotek R&D Center, South Korea, where she was a Research Engineer, from 2009 to 2011. Her research interests include volume visualization, computer graphics, and medical imaging.

JEONGJIN LEE received the B.S. degree in mechanical engineering and the M.S. and Ph.D. degrees in computer science and engineering from Seoul National University, Seoul, South Korea, in 2002 and 2008, respectively.

He is currently an Associate Professor with the School of Computer Science and Engineering and the Director of the Deep Imaging Laboratory, Soongsil University, Seoul. His research interests include deep learning, AI-based mobile healthcare, interactive vision techniques in augmented reality, and biomedical image processing.



YEONG-GIL SHIN received the B.S. and M.S. degrees in computer science from Seoul National University, Seoul, South Korea, and the Ph.D. degree in computer science from the University of Southern California, Los Angeles, CA, USA, in 1990.

He is currently a Professor with the Department of Computer Science and Engineering and the Director of the Computer Graphics and Image Processing Laboratory, Seoul National University. His research interests include computer graphics, volume visualization, medical imaging, and image processing.

BOHYOUNG KIM (Member, IEEE) received the B.S. and M.S. degrees in computer science and the Ph.D. degree in computer science and engineering from Seoul National University, Seoul, South Korea, in 2001.

She is currently an Associate Professor with the Division of Biomedical Engineering, Hankuk University of Foreign Studies, South Korea. Her research interests include computer graphics, volume visualization, medical imaging, and information visualization.