Pacific Graphics (2012)
C. Bregler, P. Sander, and M. Wimmer (Editors)

Bokeh Rendering with a Physical Lens

Xin Liu and Jon Rokne

Department of Computer Science, University of Calgary, Calgary AB T2N 1N4 Canada

Abstract

Bokeh is important for the realism and aesthetics of graphical rendering, but hard to simulate. In this paper, we propose a novel method that conceptually shoots a 3D display with a physical camera. While a high-quality 3D display is not available, we render the 3D scene layer by layer on a 2D display, and shoot each rendered layer with a physical camera whose focus is adjusted to produce the right amount of blurs. The pure colours and opacities of each layer are extracted by a matting technique and then combined into a single image by alpha blending. The proposed method provides an alternative to bokeh simulation by purely computational algorithms.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer graphics]: Picture/Image Generation—Viewing algorithms

1. Introduction

In photography, the Japanese word *bokeh* refers to the blur, or the aesthetic quality of the blur in out-of-focus areas of a photograph. In computer graphics, the bokeh is traditionally simulated by purely computational algorithms that are either slow or produce unrealistic results. In this paper, we propose a novel method that employs a physical lens to synthesize bokeh effects.

The basic idea of the proposed method is to render a virtual scene on a 3D display and then shoot the 3D display with a physical lens. While a high-quality 3D display is still not available, we construct a *pseudo 3D display* with a 2D display that is visible through a camera. This is done by stratifying the 3D scene and rendering the scene layer by layer against two different backgrounds. We fix a camera in front of the 2D display, as shown in Fig. 1, fix its aperture and shutter speed, and shoot the 2D display with the lens' focus set properly, so that the right amount of blurs is produced. A matting technique is then used to extract the pure colours and opacities of the scene layers from pairs of photographs. Finally, a single image is synthesized by combining all extracted images using alpha blending.

By inserting a physical lens into the rendering pipeline, the proposed method automatically includes various optical effects, which create realistic bokeh. The physical lens "calculates" the complex optical effects with instant real-world physics, and the computer only needs to do several passes of ray tracing and per-pixel image processing to achieve a realistic result. Therefore, the computational complexity is low. The proposed method provides an alternative to bokeh simulation by purely computational algorithms. Its feasibility is shown by preliminary experimental results.

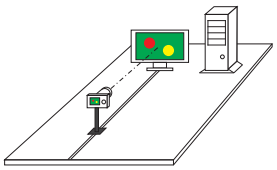

Figure 1: *The hardware settings of the proposed method.*

2. Literature review

Algorithms proposed for bokeh simulation [BHK*03a, BHK*03b, BK08, Dem04] can be classed coarsely into object space and image space algorithms.

Object space algorithms calculate bokeh effects directly from 3D representations of virtual scenes. The *distributed ray tracing* algorithm [CPC84] traces a bunch of rays emanating from each pixel and through points distributed on the lens and calculates the colour of each pixel as the average of the colours of all rays. To improve the efficiency of the distributed ray tracing, Lee et al. [LES09, LES10] used a *depth peeling* technique [Eve01] to compute an intermediate multi-layer 2.5D representation of the scene. They then in-

DOI: 10.2312/PE/PG/PG2012short/023-028

tersected rays with the simpler 2.5D representation instead of the 3D representation. Kolb et al. [KMH95] and Wu et al. [WZH*10] traced rays through a multi-element optical lens to produce a real lens' optical aberrations. Steinert et al. [SDHL11] further incorporated insights from geometrical diffraction theory to simulate advanced photographic phenomena. The *multi-pass rasterization* algorithm [HA90, NDW94] approximates the distributed ray tracing by rendering and averaging multiple pin-hole images using the z buffer and accumulation buffer. With the knowledge of the 3D scene, object space algorithms can correctly calculate visibilities and produce high quality renderings, but they are generally slow.

Image space algorithms synthesize bokeh effects from a 2D pin-hole rendering and its companion depth map. The *scatter algorithm* [PC81] scatters each pixel's colour into its neighbours according to the pixel's depth value. To avoid background colours from being blended into mid-ground in-focus pixels, the pixels' colours are generally mixed in front-to-back order [Dem04]. The *gather algorithm* [Rok96, GK07, ZCP07, LKC08, MVL00], on the contrary, blurs each pixel by gathering colours from its neighbours. To mitigate visual artifacts caused by blur discontinuity and intensity leakage, some gather algorithms [KLO06, KB07, KTB09] decompose the pin-hole image into multiple layers according to the pixels' depth values, and then blur each layer separately. Kraus and Strengert [KS07] implemented the multi-layer algorithm on GPUs to achieve interactive performance. Image space algorithms are generally fast, but the results are rather coarse since they do not explicitly model a physical lens and calculate light/lens interactions.

Our algorithm uses the depth peeling technique, belonging to object space algorithms, to render stratified images, which helps it to tackle visibility issues, and then merges the blurred layers like a multi-layer image space algorithm. Unlike the image space algorithms, the opacity of a pixel blurred by a physical lens is not available. We discover the opacity by a matting technique before alpha blending.

3. Pseudo 3D display

Ideally, we would render a virtual scene on a high quality 3D display, and then shoot the display with a physical camera. However, such a display is not available yet with current technology. Therefore, we have to work with an available 2D display. To extend the 2D display in the depth dimension, we stratify the 3D scene and render the scene layer by layer against two different backgrounds. The scene layers are shot by a physical camera (see Sec. 4) and then the photographs are merged into a single 2D image of the scene using matting (see Sec. 5) and alpha blending (see Sec. 6) techniques.

With a thin lens model, an out-of-focus point P is projected into a circle of confusion (COC), as shown in Fig. 2. Supposing that the lens is focused at a distance u_f from the

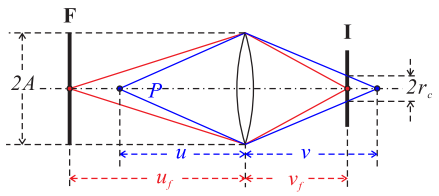

Figure 2: *Computing the circle of confusion.* **I** *stands for image sensor,* **F** *for focal plane.*

lens, we have the following system of equations

$$\begin{cases} \frac{r_c}{A} = \frac{v_f - v}{v} \\ \frac{1}{u_f} + \frac{1}{v_f} = \frac{1}{f} \\ \frac{1}{u} + \frac{1}{v} = \frac{1}{f} \end{cases}, \tag{1}$$

where r_c stands for the COC's radius, A for the aperture's radius, f for the focal length of the lens, v_f for the lens-to-film distance, u for the distance from P to the lens, and v for the distance from P's sharp image to the lens. The aperture's radius can be calculated from the f-number N_f as

$$A = \frac{f}{2N_f}. \tag{2}$$

Solving Equ. (1) we get

$$r_c = \frac{A f u_f}{u_f - f}\left(\frac{1}{u_f} - \frac{1}{u}\right), \tag{3}$$

where $r_c > 0$ if P is behind the focal plane, $r_c = 0$ if P is on the focal plane, and $r_c < 0$ if P is in front of the focal plane.

We stratify the scene into layers according to the object-to-lens distance (depth), such that (1) pixels in each layer has similar blurs, or their r_c values are within a small range, and (2) the r_c values at the *centers* (as defined below) of neighbour layers differ by a constant σ. To facilitate description, we introduce the concept of *continuous layer* l, which is a measurement of the depth u in terms of r_c:

$$l(u) = \frac{r_c}{\sigma} = \frac{A f u_f}{\sigma(u_f - f)}\left(\frac{1}{u_f} - \frac{1}{u}\right). \tag{4}$$

Reversely, we can calculate the depth u from the continuous layer l as

$$u(l) = \frac{A f u_f}{Af - l\sigma(u_f - f)}. \tag{5}$$

A (discrete) layer $\hat{l}$ spans the depth range $\left[u\left(\hat{l}-0.5\right), u\left(\hat{l}+0.5+\dot{l}\right)\right]$, for $\hat{l} = \cdots, -2, -1, 0, 1, 2, \cdots$, where $\dot{l}$ is a user designated constant. The depth at $u\left(\hat{l}\right)$ is called the *center* of layer $\hat{l}$, the depths at $u\left(\hat{l}-0.5\right)$ and $u\left(\hat{l}+0.5\right)$ are called the *front* and *back* ends of the layer,

and the depth range $\left[u\left(\hat{l}+0.5\right), u\left(\hat{l}+0.5+\dot{l}\right)\right)$ is called the *transitive* range. The scene layer between the front and back ends are rendered as it is; the scene layer in the transitive range is dissolved into the background linearly to the l value. For layer $\hat{l}$, the rendered colour is therefore

$$c = \begin{cases} c_s & \text{if } \hat{l}-0.5 \le l(u) < \hat{l}+0.5 \\ \alpha c_s + (1-\alpha)c_k & \text{if } \hat{l}+0.5 \le l(u) < \hat{l}+0.5+\dot{l} \\ c_k & \text{otherwise} \end{cases}, \tag{6}$$

where $\alpha = \left(\dot{l} - l(u) + \hat{l} + 0.5\right)/\dot{l}$, c_s is the scene colour, and c_k is the background colour. Figure 3 illustrates the scene stratification conceptually, where solid blue stands for the pure scene colour, solid white for the pure background colour, and bright blue for a mixture of scene and background colours.

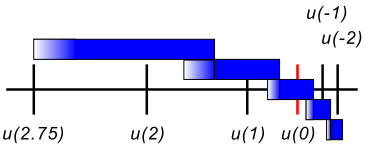

Figure 3: *A conceptual illustration of scene layers.*

The mixture of the scene colour and the background colour will be translated into translucency by matting, as explained in Sec. 5. The closer the mixed colour is to the background, the more transparent a pixel will be. When alpha blending, as explained in Sec. 6, is used to merge the layers, a front layer will be mixed with its immediate back layer in the transitive range, with the front layer fading out and the back layer fading in. In this manner, smooth transitions between layers can be created. We did not dissolve the front end, because this could cause severe visibility errors.

4. Shooting screen

In the proposed method, bokeh is produced by a physical lens. To this end, we shoot the screen with a camera fixed in front of the screen, whose optical axis is perpendicular to the screen, as shown in Fig. 1. This hardware is easy to set up, because no large mechanical moving parts are required. We shoot all scene layers with the same focal length, aperture and shutter speed to obtain consistent bokeh effects in a single synthesized image. Different object distances, which produce different degrees of blurs, are simulated by adjusting the lens' focus. In this way, a wide range of foreground and background bokeh from near to the lens to infinity can be simulated by turning the focus ring of the lens by a limited angle.

The foreground bokeh (bokeh of objects in front of the focal plane) are produced by focusing the lens behind the screen, and the background bokeh (bokeh of objects behind the focal plane) are produced by focusing the lens in front of the screen. For a layer $\hat{l}$ displayed on a screen, the physical lens' focus is adjusted, so that the COC of a point on the screen has the same size as the COC of a point at the layer center produced by the desired virtual camera. That is

$$\mathbf{r_c} = r_c\left(u\left(\hat{l}\right)\right). \tag{7}$$

Here and in the sequel, bold fonts are used to denote measurements in the physical world. We use the approximate thin lens model to solve for the focus $\mathbf{u_f}$ that produces the COC size given by Equ. (7). As shown in Fig. 4, once a camera is fixed in front of the screen $\mathbf{S}$, the distance $\mathbf{d_{IS}}$ from the image sensor $\mathbf{I}$ to the screen $\mathbf{S}$ can be measured with a ruler. Letting $\mathbf{u}$ denote the distance from the lens to the screen, we have

$$\frac{1}{\mathbf{u_f}} + \frac{1}{\mathbf{d_{IS}} - \mathbf{u}} = \frac{1}{\mathbf{f}}. \tag{8}$$

Solving for $\mathbf{u}$, we get

$$\mathbf{u} = \mathbf{d_{IS}} - \frac{\mathbf{f u_f}}{\mathbf{u_f} - \mathbf{f}}. \tag{9}$$

Replacing u in Equ. (3) with Equ. (9) and with some manipulations we get a quadric equation:

$$\mathfrak{A}\mathbf{u_f}^2 + \mathfrak{B}\mathbf{u_f} + \mathfrak{C} = 0, \tag{10}$$

where

$$\begin{cases} \mathfrak{A} = \mathbf{r_c d_{IS}} - \mathbf{r_c f} + \mathbf{A f} \\ \mathfrak{B} = \mathbf{r_c f}^2 - 2\mathbf{r_c d_{IS} f} - \mathbf{A d_{IS} f} \\ \mathfrak{C} = \mathbf{r_c d_{IS} f}^2 + \mathbf{A d_{IS} f}^2 \end{cases}. \tag{11}$$

Solving Equ. (10), we get two solutions, one of which is valid for the focus that produces COCs with radius $\mathbf{r_c}$:

$$\mathbf{u_f} = \frac{-\mathfrak{B} + \sqrt{\mathfrak{B}^2 - 4\mathfrak{A}\mathfrak{C}}}{2\mathfrak{A}}. \tag{12}$$

The other solution that is very close to the lens is physically invalid.

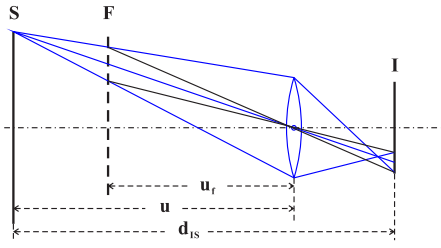

Figure 4: *Computing the focus producing a given size of COC.* **I** *stands for image sensor,* **F** *for focal plane, and* **S** *for screen.*

An accurately in-focus photograph of the screen generally suffers from *moiré patterns*. When shooting the in-focus layer 0, we defocus the lens slightly to an extent that removes

the moiré patterns but does not impair the sharpness of the photograph.

As shown in Fig. 5, if the camera is refocused on a plane **F** in front of the screen to obtain background bokeh by moving the lens away from the image sensor, the screen's image is magnified compared with the standard settings. Similarly, if the camera is refocused on a plane behind the screen to obtain foreground blurs by moving the lens near to the image sensor, the screen's image is minified. We must compensate for the change of the lens' magnification, so that different layers can be combined seamlessly.

When the camera is focused on the screen, the object/screen-to-lens distance is called the *standard object distance* $\mathbf{u_s}$, and the image-sensor-to-lens distance is called the *standard image distance* $\mathbf{v_s}$, as shown in Fig. 5. Using the thin lens equation, we have

$$\begin{cases} \frac{1}{\mathbf{u_s}} + \frac{1}{\mathbf{v_s}} = \frac{1}{\mathbf{f}} \\ \mathbf{u_s} + \mathbf{v_s} = \mathbf{d_{IS}} \end{cases}. \tag{13}$$

Solving the above equation we get

$$\begin{cases} \mathbf{u_s} = \frac{\mathbf{d_{IS}} + \sqrt{\mathbf{d_{IS}}^2 - 4\mathbf{f}\mathbf{d_{IS}}}}{2} \\ \mathbf{v_s} = \frac{\mathbf{d_{IS}} - \sqrt{\mathbf{d_{IS}}^2 - 4\mathbf{f}\mathbf{d_{IS}}}}{2} \end{cases}. \tag{14}$$

Let **S** denote the size of an object on the screen, and **s** the

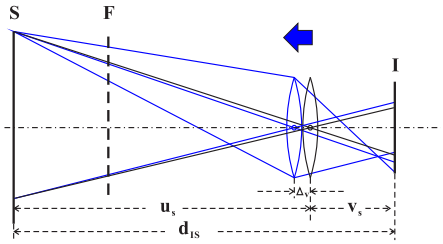

Figure 5: *The change of magnification on refocusing.* **I** *stands for image sensor,* **F** *for focal plane, and* **S** *for screen.*

size of its image when the camera is focused on the screen. We have

$$\frac{\mathbf{s}}{\mathbf{S}} = \frac{\mathbf{v_s}}{\mathbf{u_s}}. \tag{15}$$

When the camera is refocused at $\mathbf{u_f}$ according to Equ. (12), the new image-sensor-to-lens distance $\mathbf{v_f}$ can be computed as

$$\mathbf{v_f} = \frac{\mathbf{u_f}\mathbf{f}}{\mathbf{u_f} - \mathbf{f}}. \tag{16}$$

Let $\Delta\mathbf{v} = \mathbf{v_f} - \mathbf{v_s}$, and then $\Delta\mathbf{v} < 0$ when the focus is behind the screen and $\Delta\mathbf{v} > 0$ when the focus is in front of the screen. We consider the intersection between the image sensor and the line passing through a point on the screen and the optical center of the lens to be the *center* of the point's blurred image. Letting $\mathbf{s_f}$ denote the size of the same object's image after refocusing, we have

$$\frac{\mathbf{s_f}}{\mathbf{S}} = \frac{\mathbf{v_s} + \Delta_\mathbf{v}}{\mathbf{u_s} - \Delta_\mathbf{v}}. \tag{17}$$

Comparing Equations (15) and (17), the images can be combined seamlessly if the images are scaled by the compensating factor

$$\lambda = \frac{\mathbf{v_s}(\mathbf{u_s} - \Delta_\mathbf{v})}{\mathbf{u_s}(\mathbf{v_s} + \Delta_\mathbf{v})}. \tag{18}$$

An optical lens has a larger *field of view* (FOV) than a pin-hole at the optical center. The extra FOV of the physical camera can be covered by enlarging the virtual pin-hole camera's FOV slightly, so that the photographs are correct around their margins.

5. Matting

According to Smith and Blinn's theory [SB96], the per-pixel pure colour and opacity of a translucent object can be solved from photographs of the object against two known backgrounds, if the background colours have a non-zero distance at every pixel. This *matting* technique was derived from the *blue screen matting* that is widely used in film and video industries. In this paper, we deliberately blend renderings with known constant colours by computational algorithms (see Sec. 3) and a physical lens (see Sec. 4) to trace transparencies. Dark blue and dark green are currently used, although the choice of backgrounds is worth further investigation. We then employ the matting technique to solve for the pure scene colours and opacities.

Formally, each photograph pixel $C_p = [R_p, G_p, B_p]^T$ retrieved from the camera is viewed as a composite of the pure scene colour, $C_s = [R_s, G_s, B_s]^T$, which is pre-multiplied by the opacity α_s, and the known background colour $C_k = [R_k, G_k, B_k]^T$:

$$C_p = C_s + (1 - \alpha_s) C_k. \tag{19}$$

Each scene layer is rendered against two different constant backgrounds with colours C_{k_1} and C_{k_2}, respectively. We then take a picture for each of the renderings. The per-pixel colour of the photographs are denoted by C_{p_1} and C_{p_2}, respectively. Combining the two sets of equations from the two photographs, and expressing them in a matrix form, we get an over-determined system of linear equations:

$$\begin{bmatrix} 1 & 0 & 0 & -R_{k_1} \\ 0 & 1 & 0 & -G_{k_1} \\ 0 & 0 & 1 & -B_{k_1} \\ 1 & 0 & 0 & -R_{k_2} \\ 0 & 1 & 0 & -G_{k_2} \\ 0 & 0 & 1 & -B_{k_2} \end{bmatrix} \begin{bmatrix} C_s \\ \alpha_s \end{bmatrix} = \begin{bmatrix} R_{p_1} - R_{k_1} \\ G_{p_1} - G_{k_1} \\ B_{p_1} - B_{k_1} \\ R_{p_2} - R_{k_2} \\ G_{p_2} - G_{k_2} \\ B_{p_2} - B_{k_2} \end{bmatrix}. \tag{20}$$

Solving the above system of equations with the method of

least squares, we get

$$\begin{cases} \alpha_s = 1 - \frac{(C_{p_1} - C_{p_2})^T (C_{k_1} - C_{k_2})}{||C_{k_1} - C_{k_2}||^2} \\ C_s = \frac{(Cp_1 + Cp_2) - (1-\alpha_s)(C_{k_1} + C_{k_2})}{2} \end{cases}. \tag{21}$$

6. Image blending

After the matting operations, we get M images with per-pixel opacities, which are to be combined by alpha blending. Let C_i and α_i denote the per-pixel colour and opacity in the ith image, for $i = 0, 1, \cdots, M-1$. (To facilitate the description, we number the images of the front-most layer with 0, instead of a negative number as in Sec. 3.) The accumulated colour and opacity combing all layers are computed as

$$\begin{cases} C_a = C_0 + C_1(1-\alpha_0) + \cdots \\ \qquad + C_{M-1} \prod_{i=0}^{M-2} (1-\alpha_i) \\ \alpha_a = \alpha_0 + \alpha_1(1-\alpha_0) + \cdots \\ \qquad + \alpha_{M-1} \prod_{i=0}^{M-2} (1-\alpha_i) \end{cases}. \tag{22}$$

We finally normalize the colour with the opacity, and compute the per-pixel colour of the synthesized image as

$$C = \frac{C_a}{\alpha_a}. \tag{23}$$

7. Computational complexity

We analyze the computational complexity of the proposed method in the context of ray tracing. Supposing that a scene is stratified into M layers, and the rendering contains N_p pixels, the proposed method needs to ray trace the scene in M passes for each of the N_p pixels, each pass starting from a farther distance. The rendering phase has a time complexity of $O(M \cdot N_p)$. The image processing phase uses per-pixel algorithms that are linear to the number of layers, so its time complexity is $O(M \cdot N_p)$. Hence, the total computational complexity is $O(M \cdot N_p)$. Considering that the per-pixel image processing is generally much faster than ray tracing, the proposed method needs approximately M times the computations required for rendering a pin-hole image.

8. Experiments

We validated the proposed method conceptually with devices available in our lab, including an Apple® Cinema HD display and a Nikon® D70 camera with a Nikkor® 50 mm f/1.8D lens. The display provides 0.258 mm pixel pitch, 200 cd/m^2 brightness and 350:1 contrast ratio. The camera is equipped with a 23.7×15.6 mm^2 APS-C CCD sensor, providing 3008×1200 effective pixels. The lens uses a classical double Gauss 5 groups/6 elements structure. The camera was mounted in front of the display. We bound a short plastic ruler on the focus ring of the lens, which provided a reference scale. A correspondence between the scale and the object-to-sensor distance was built up with a pre-calibration process. To that end, we put a planar target at precisely measured distances from the image sensor mark of the camera, and focused the lens on the target using the camera's auto-focus function at each distance. The corresponding scales and object-to-sensor distances are recorded in a table. With the table, a desired focus can be achieved by manually turning the focus ring to the correct scale. (The system could be automated by driving the focus ring with a computer controlled stepper motor, which was left for future work.) We used a program to render the layered scene on the screen, which also provided instructions on how to adjust the lens' focus. We then shot the screen with the lens' focus adjusted manually. Finally, the photographs were transferred to the computer via USB and processed with another program to synthesize an image with bokeh effects.

The results on simulating the bokeh effects of a room-sphere scene and a Christmas-dog scene are shown in Fig. 6 and Fig. 7, respectively. For both scenes, we set the destination image's resolution to 1203×800 pixels, the COC radius incremental step σ to 5 pixels, the transitive range parameter $\dot{l}$ to 0.25, and the f-stop of the physical lens to f/1.8. For the room-sphere scene, we targeted at simulating the bokeh effects of three virtual lenses of 50 mm f/1.8, 50 mm f/2.8, and 35 mm f/1.4, all focusing at 1450 mm, approximately on the second nearest ball. The scene was stratified accordingly into 7, 5, and 5 layers. The results are shown in Fig. 6 (b), (c), and (d). The synthesized images show realistic bokeh effects created by a physical lens, including blurs and optical aberrations. The depth-of-field of a photograph increases as the lens' f-number increases and/or the lens' focal length decreases. This phenomenon is correctly revealed by the synthesized images. For the Christmas-dog scene, we targeted at simulating the bokeh effects of a virtual 50 mm f/1.8 lens focusing at 2200 mm, approximately on the dog's face. The scene was stratified into six layers accordingly. The synthesized image is shown in Fig. 7 (b). The highlights show strong spherical and coma aberrations. The blurred regions show somewhat complex patterns. The foreground leaves were blurred by the lens, and its boundaries became translucent after matting, which allowed the back layers, rendered in separate passes, to be partially visible. The visibility issue was well tackled for this Christmas-dog scene. The quality of the results is limited by our experimental conditions. Whereas, the results show that the proposed algorithm is robust w.r.t. coarse device calibrations.

9. Conclusion

In this paper, we have proposed a novel method for bokeh simulation that conceptually shoots a 3D display with a physical lens. While a high quality 3D display is not available, we have constructed a pseudo 3D display by stratifying

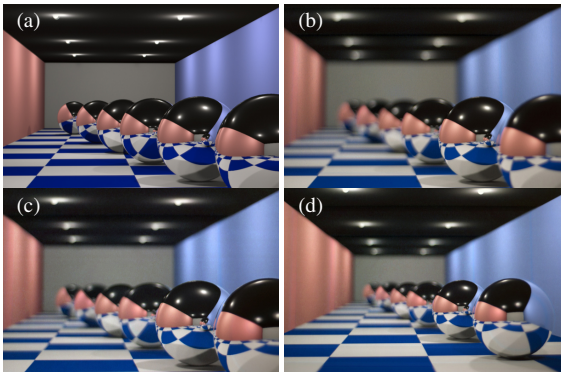

Figure 6: *The pin-hole rendering (a) and the synthesized images with bokeh effects of a 50 mm f/1.8 lens (b), a 50 mm f/2.8 lens (c) and a 35 mm f/1.4 lens (d) of the sphere-room scene.*

Figure 7: *The pin-hole rendering (a) and the synthesize image with bokeh effects of a 50 mm f/1.8 lens (b) of the Christmas-dog scene.*

the 3D scene and shooting the 2D renderings of the scene layers with a real camera over time. The image with bokeh effects is synthesized by matting and alpha blending. With a physical lens, the proposed method produces realistic bokeh effects with a low computational complexity. The feasibility of the proposed method has been shown by preliminary experimental results.

Acknowledgments

This work is supported by Alberta Innovates – Technology Futures and Alberta Advanced Education & Technology.

References

[BHK*03a] BARSKY B., HORN D., KLEIN S., PANG J., YU M.: Camera models and optical systems used in computer graphics: part i, object-based techniques. In *Proc. of Computational Science and Its Applications* (2003), pp. 246–255. 1

[BHK*03b] BARSKY B., HORN D., KLEIN S., PANG J., YU M.: Camera models and optical systems used in computer graphics: part ii, image-based techniques. In *Proc. of Computational Science and Its Applications* (2003), pp. 256–265. 1

[BK08] BARSKY B., KOSLOFF T.: Algorithms for rendering depth of field effects in computer graphics. In *Proc. of the 12th WSEAS International Conference on Computers* (2008), pp. 999–1010. 1

[CPC84] COOK R., PORTER T., CARPENTER L.: Distributed ray tracing. In *Proc. of ACM SIGGRAPH* (1984), vol. 18, pp. 137–145. 1

[Dem04] DEMERS J.: Depth of field: A survey of techniques. *GPU Gems 1* (2004), 375Ú390. 1, 2

[Eve01] EVERITT C.: Interactive order-independent transparency. *White paper, nVIDIA 2*, 6 (2001), 7. 1

[GK07] GÖRANSSON J., KARLSSON A.: Chapter 28: Practical post-process depth of field. *GPU Gems 3* (2007), 583–606. 2

[HA90] HAEBERLI P., AKELEY K.: The accumulation buffer: Hardware support for high-quality rendering. In *Proc. of ACM SIGGRAPH* (1990), vol. 24, pp. 309–318. 2

[KB07] KOSLOFF T., BARSKY B.: An algorithm for rendering generalized depth of field effects based on simulated heat diffusion. In *Proc. of the International Conference on Computational Science and Its Applications* (2007), Springer-Verlag, pp. 1124–1140. 2

[KLO06] KASS M., LEFOHN A., OWENS J.: *Interactive depth of field using simulated diffusion on a gpu*. Tech. rep., Pixar Animation Studios, 2006. 2

[KMH95] KOLB C., MITCHELL D., HANRAHAN P.: A realistic camera model for computer graphics. In *Proc. of ACM SIGGRAPH* (1995), pp. 317–324. 2

[KS07] KRAUS M., STRENGERT M.: Depth-of-field rendering by pyramidal image processing. *Computer Graphics Forum 26*, 3 (2007), 645–654. 2

[KTB09] KOSLOFF T., TAO M., BARSKY B.: Depth of field postprocessing for layered scenes using constant-time rectangle spreading. In *Proc. of Graphics Interface* (2009), pp. 39–46. 2

[LES09] LEE S., EISEMANN E., SEIDEL H.: Depth-of-field rendering with multiview synthesis. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 1–6. 1

[LES10] LEE S., EISEMANN E., SEIDEL H.: Real-time lens blur effects and focus control. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 1–7. 1

[LKC08] LEE S., KIM G., CHOI S.: Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. *IEEE Transactions on Visualization and Computer Graphics* (2008), 453–464. 2

[MVL00] MULDER J., VAN LIERE R.: Fast perception-based depth of field rendering. In *ACM Symposium on Virtual Reality Software and Technology* (2000), ACM, pp. 129–133. 2

[NDW94] NEIDER J., DAVIS T., WOO M.: Chapter 10: The framebuffer. *OpenGL Programming Guide* (1994), 202–224. 2

[PC81] POTMESIL M., CHAKRAVARTY I.: A lens and aperture camera model for synthetic image generation. In *Proc. of ACM SIGGRAPH* (1981), vol. 15, pp. 297–305. 2

[Rok96] ROKITA P.: Generating depth-of-field effects in virtual reality applications. *IEEE Computer Graphics and Applications 16*, 2 (1996), 18–21. 2

[SB96] SMITH A., BLINN J.: Blue screen matting. In *Proc. of ACM SIGGRAPH* (1996), pp. 259–268. 4

[SDHL11] STEINERT B., DAMMERTZ H., HANIKA J., LENSCH H. P. A.: General spectral camera lens simulation. *Computer Graphics Forum 30* (2011), 1643–1654. 2

[WZH*10] WU J., ZHENG C., HU X., WANG Y., ZHANG L.: Realistic rendering of bokeh effect based on optical aberrations. *The Visual Computer 26*, 6-8 (2010), 555–563. 2

[ZCP07] ZHOU T., CHEN J., PULLEN M.: Accurate depth of field simulation in real time. *Computer Graphics Forum 26*, 1 (2007), 15–23. 2