Eurographics Symposium on Rendering 2016
E. Eisemann and E. Fiume
(Guest Editors)

Volume 35 (2016), Number 4

Efficient Ray Tracing Through Aspheric Lenses and Imperfect Bokeh Synthesis

Hyuntae Joo[1], Soonhyeon Kwon[1], Sangmin Lee[1], Elmar Eisemann[2] and Sungkil Lee[†1]

[1]Sungkyunkwan University, South Korea
[2]Delft University of Technology, Netherlands

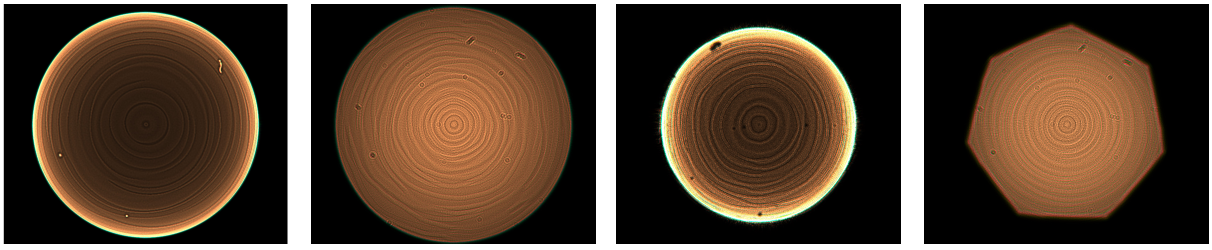

Figure 1: *Examples of bokeh textures generated with our ray-tracing solution.*

Abstract

We present an efficient ray-tracing technique to render bokeh effects produced by parametric aspheric lenses. Contrary to conventional spherical lenses, aspheric lenses do generally not permit a simple closed-form solution of ray-surface intersections. We propose a numerical root-finding approach, which uses tight proxy surfaces to ensure a good initialization and convergence behavior. Additionally, we simulate mechanical imperfections resulting from the lens fabrication via a texture-based approach. Fractional Fourier transform and spectral dispersion add additional realism to the synthesized bokeh effect. Our approach is well-suited for execution on graphics processing units (GPUs) and we demonstrate complex defocus-blur and lens-flare effects.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms

1. Introduction

Optical systems are typically imperfect and not all rays will follow the expected optical path toward the sensor, which leads to optical aberrations [Smi00]. Classical systems make use of many spherical lens elements to reduce this effect. A recent alternative is an *aspheric lens*, whose profile does not correspond to a part of a sphere [KT80]. Their use typically implies fewer optical elements for a similar reduction of aberrations, which, consequently, results in less weight. This property made them a popular choice (e.g., smartphones lenses, pancake lenses, and eyeglasses). Despite their wide-spread use in practice, aspheric lenses have hardly been investigated in computer graphics.

In this paper, we model and simulate aspheric lenses. Because their surface profile is usually nonlinear and non-spherical, there is no general closed-form solution for calculating ray-surface intersections, as exists for spherical lenses [KMH95, SDHL11, HESL11]. Instead, we present an efficient numerical scheme to trace rays through these lens elements. Additionally, aspheric lenses often contain small imperfections due to the fabrication process (e.g., mechanical grinding [LB07]), influencing out-of-focus blur. In particular, it can impact the intensity profile of *bokeh*, the aesthetic appearance of out-of-focus highlights, resulting in "onion-ring bokeh" [ST14]. We simulate this characteristic aspect by using a normal map derived from a virtual grinding process.

The contributions of this paper can be summarized as:

- an aspheric lens model, including imperfections;
- an efficient numerical aspheric-lens intersection method;
- a rendering solution for optical systems.

Our paper is organized as follows. We discuss previous work (Sec. 2) and introduce our aspheric lens model (Sec. 3), including imperfections (Sec. 4). We then cover our rendering algorithm and

† sungkil@skku.edu (corresponding author)

intersection test for aspheric lenses (Sec. 5) before presenting results (Sec. 6) and concluding (Sec. 7).

2. Related Work

This section reviews previous work on simulating optical systems and ray-tracing techniques for parametric surfaces.

2.1. Optical Simulation

Early approaches for optical simulations have relied on simple optical models [Coo86, Che87], but simulating a full optical system has been of growing interest. Many approaches opted for additional realism in terms of geometric distortion, defocus blur, and spectral dispersion. They usually build upon formal data sheets specifying the optical system in terms of its optical elements. The seminal work of Kolb et al. obtained geometric accuracy for distortion and field of view [KMH95]. Steinert et al. included additional aberrations and spectral diversities using Monte-Carlo rendering [SDHL11]. Hullin et al. added reflections and anti-reflective coatings, which play a role for lens flares [HESL11]. They attained physically faithful image quality but were restricted to spherical lenses in order to simplify the ray-surface intersection tests. In this paper, we add support for aspheric lenses.

Optical imperfections can be classified into intrinsic and extrinsic ones. The former ones include aberrations and flares [HESL11, LE13], the latter ones include issues due to mechanical manufacturing (e.g., grinding and polishing) and real-world artifacts (e.g., dirt or axis misalignment).

One of the pronounced intrinsic imperfections are (spherical) aberrations manifesting in bokeh and can be simulated via ray tracing [LES10, WZH*10, WZHX13]. For higher performance, early models captured the appearance phenomenologically [BW02]. Later approaches relied on efficient filtering; polygonal filtering [MRD12], separable filtering [MH14], low-rank linear filters [McG15], or look-up table [GKK15].

Extrinsic imperfections are not easily incorporated into ideal parametric models [KMH95, WZH*10, SDHL11, HESL11, WZHX13]. One attempt was to use artificial noise for diffraction at the aperture [RIF*09, HESL11], whereas our method is driven by simulating the lens-fabrication process.

2.2. Ray Tracing of Parametric Surfaces

While many modern ray tracers focus on polygonal models [PBMH02, FS05], early approaches often included several parametric surface types, which can be classified in four categories: subdivision [Whi80], algebraic methods [Kaj82, Bli82, Han83, Man94], Bézier clipping [NSK90, EHS05], and numerical techniques [Tot85, JB86, AGM06].

In optics, much work addressed intersections with spherical surfaces but few approaches covered aspheric lenses; typically, tessellation [MNN*10] and Delaunay triangles [OSRM09]) or Newton-Raphson methods [AS52, For66] were applied. Tessellation is costly for sufficient precision, while the Newton-Raphson method is fast but unreliable for complex surfaces. These problems manifest themselves even more clearly when using limited precision floating-point numbers, as is often the case for GPU approaches. In contrast, our solution relies on a bracketing method supported by tight proxy geometry to perform the intersection test, which even works for lower-precision floating point values.

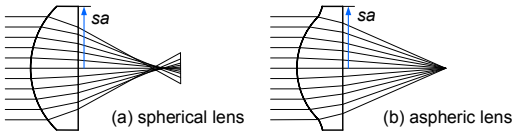

Figure 2: *A spherical lens (a) can cause spherical aberration, while even a single aspheric lens (b) may focus well.*

3. Aspheric-Lens Model

We here introduce our parametric aspheric-lens model. An aspheric lens adds nonlinear deviations to its base curvature c of a spherical surface (Figure 2), leading to a better convergence of peripheral rays on the focus plane. The common parametric form of an aspheric surface with its sagittal profile z on the optical axis (along the z-axis), is expressed in terms of the radial coordinate r from the optical axis as:

$$z(r) := \frac{cr^2}{1+\sqrt{1-(1+\kappa)c^2r^2}} + \sum A_{2i} r^{2i}, \qquad (1)$$

where κ is a conic constant [FTG00] and A_{2i} are coefficients for even-polynomial sections; the coefficients of particular systems can be found in patents or lens design books [Smi05].

The design of an aspheric lens goes through an iterative optimization procedure to minimize aberrations. The conic constant κ is often a consequence of the lens' base shape from which the aspheric lens is derived, hence, the focus lies on optimizing A_{2i}. The design requires high expertise and care in optics. We refer the reader to the textbooks for mathematical properties and practical usages [FTG00, Smi05].

4. Imperfections

In the real world, the fabrication of large spherical/aspheric lenses involves mechanical grinding and polishing [LB07] (Figure 3). The process has usually several phases; for instance, spherical grinding, precision grinding, and polishing are applied in sequence [FTG00]. Similarly, in precision glass molding, a mold is carved, which will then exhibit the imperfections that are passed on to the lens, when it is produced with the mold. The processes are not entirely standardized and although the various phases have been improved over the years, the result is not perfect yet.

The most visible artifacts resulting from the fabrication process are due to a misalignment between grinding and optical axis, grinding asymmetries resulting from the swinging of the tools, and limited grinding precision. A few nanometers of deviation can lead to a visible impact on bokeh. We model the resulting imperfections indirectly by simulating the grinding process itself. Following the fabrication

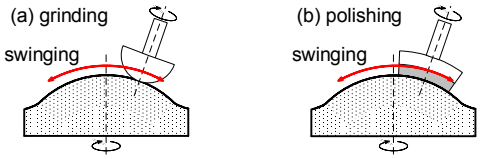

Figure 3: *Simple mechanics of lens fabrication via a grinder.*

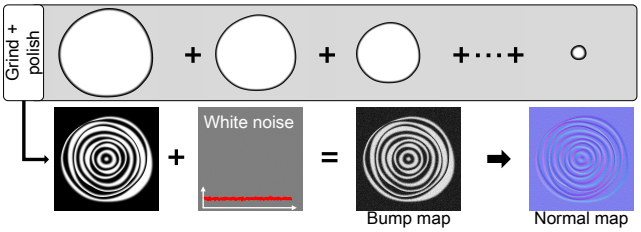

Figure 4: *Example of a grinding pattern generation; parameters are 5000× exaggerated for illustration purposes.*

method, we create grinding patterns by superposing circles with a smooth boundary (reflecting the grinding head) and decreasing radii (or a spiral, depending on the simulated tool). We apply a perturbation via a wave pattern, whose strength is user controlled. Similarly, following real-world examples, the result involves between 50 to 120 rings. Each ring is also drawn with a low-frequency jittering to simulate the swinging. The rings are accumulated to create a height profile and combined with a white noise texture to simulate general imperfections. The final result is a height field from which we derive a normal map. The latter is used during rendering to influence the refractive directions. The process is illustrated in Figure 4.

One more artifact, which can be independent of the production process, are tiny dirt marks. These can become visible with large out-of-focus bokeh. Their appearance can be arbitrary, and we model them as noisy granular details using a noise generator and augment the previously-derived texture. Further, we store this pattern also as an additional color channel to reduce the intensity of intersecting rays.

5. Rendering

Given the lens model, we will, similarly to previous work [KMH95, SDHL11, HESL11], apply a deterministic forward ray tracing to trace the light propagation in an optical system. An overview of the various steps of our approach is given in Figure 5. The incoming rays are generated at the entrance pupil. When the rays encounter a lens element, we test for intersection. If the element is missed or a total reflection occurs, we stop the ray traversal. At the aperture, we use a polygonal texture to define the iris and to test if rays are blocked [HESL11]. Once a ray reaches the sensor, we accumulate its irradiance via additive alpha blending.

To account for imperfections, we apply the normal maps, which are stored as 2D textures for each aspheric surface. Based on the texture lookup, the intersecting ray is jittered. As the normal vectors are based on our virtual grinding patterns, the result will exhibit the typical bokeh with ring patterns, often called “onion-ring” bokeh.

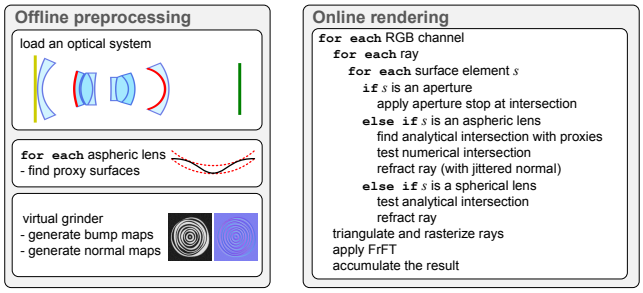

Figure 5: *Overview of the rendering pipeline of our system.*

5.1. Ray-tracing an Aspheric Lens

To test the intersection of a ray and the aspheric surface, we use a *bracketing*-based root-finding method (either the bisection or the false position method). The explicit lens definition (Eq. (1)) is inconvenient for this process, and we instead reformulate it to an implicit representation.

Eq. (1) defines the height-field lens surface given a radial coordinate r, where r is shared by several points located around the optical axis (z-axis). For an arbitrary 3D point $\mathbf{v}$ (with its Cartesian coordinates v_x, v_y, and v_z) this radial coordinate is expressed as $r := (v_x^2 + v_y^2)^{1/2}$. When $\mathbf{v}$ lies on the lens surface, $v_z = z(r)$ is satisfied. This leads to the following implicit formulation:

$$f(\mathbf{v}) := z((v_x^2 + v_y^2)^{1/2}) - v_z, \tag{2}$$

where the sign of $f(\mathbf{v})$ implies if a point lies above or below the aspheric-lens surface and the aspheric-lens surface is the set $\{\mathbf{v} \in \mathbb{R}^3 | f(\mathbf{v}) = 0\}$.

When inserting a ray equation into the implicit formulation, the problem becomes a root-finding process. Derivative-based methods exist, which utilize the base curvature as an initial guess [AS52, For66, Smi00]. However, the use of gradients can be difficult and unreliable, as many GPUs only have limited or no support for double precision. Our solution instead relies on a bracketing method, which does not require to evaluate gradients. Avoiding double-precision computations on GPUs leads to a great performance gain; such preference for bracketing can be found in various GPU approaches [BES12] and is a robust and efficient choice.

The bisection method is the simpler method of the two and initialized with a tight interval. It is iteratively subdivided at its midpoint, yielding two subintervals. The process always continues with the subinterval whose extremities lie on opposite sides of the surface. The false position method [BF01] is similar but assumes in each iteration that the function is linear between the two interval extremities. The root of this linear function defines the position to produce subintervals.

Proxy Geometry The efficiency of our root-finding process depends on the initial interval, which we initialize by determining the ray intersection with a tight surface-englobing proxy geometry. In principle, one could define two planes that englobe the lens surface, but seen that many aspheric lenses are based on a spherical surface, two spherical interfaces usually deliver a tighter fit.

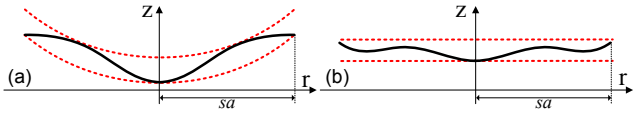

Figure 6: *Example definition of pairs of proxy surfaces: (a) spheres and (b) planes.*

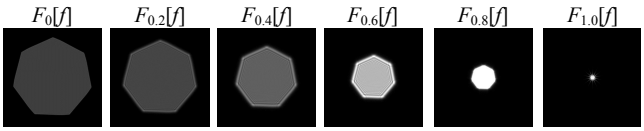

Figure 7: *Evolution of the fractional Fourier transform along its order $\alpha \in [0,1]$ for a heptagonal aperture.*

The proxy geometry is determined in an offline preprocess. First, we construct a spherical interface determined by the surface point on the optical axis and the intersection between the tube and the lens (in many optical data sheets, the corresponding radius is given as *sa*). Next, we find the two tangent positions to the aspheric lens surface along the z-axis. These two configurations define our proxy geometry (Figure 6). The tangent position could be determined via a root finding (inserting their expression in Eq. (2)) but would need to be coupled to an additional test of the lens boundary. In consequence, for simplicity and to support even more general lens surfaces, we use a sample-based procedure. It proved sufficient in practice and tests different configurations by evaluating a dense set of surface points for tangency (all samples positive/negative and one sample close/equal to zero).

5.2. Fractional Diffraction Synthesis

Light patterns (or its lens-flare ghosts) passing through an aperture usually exhibit *ringing* at edges, which is crucial for a realistic appearance of bokeh patterns (Figure 7). This ringing is known to be caused by near-field Fresnel diffraction [Kin75]. This Fresnel diffraction can be formulated to a fractional Fourier Transform (FrFT) [PF94], which uses a fractional order α to control the amount of transition to the full Fourier Transform (FT); see Figure 7. The FrFT $\mathscr{F}_\alpha(u)$ of a signal $g(x)$ is formulated as: $\mathscr{F}_\alpha(u) := \int_{-\infty}^{\infty} \mathscr{K}_\alpha(u,x) g(x) dx$, where the transform kernel $\mathscr{K}_\alpha(u,x)$ is:

$$\mathscr{K}_\alpha(u,x) = \sqrt{1 - j\cot\alpha}\, e^{-2\pi j(\csc\alpha\, ux - \cot\alpha\,(u^2+x^2)/2)}. \qquad (3)$$

We implement a discrete FrFT on the GPU, instead of the CPU (as done by the previous studies [OZK01, HESL11]); see the accompanying supplement for details. Since the phase components of FrFT (see Eq. (3)) shift much faster than FT, it requires a denser sampling to avoid aliasing. The typical implementation relying on fast FT (FFT) is slow and needs considerable zero padding, which may not fit into GPU memory. So, we take a direct discrete sampling approach. Despite its several limitations (e.g., non-unitary and irreversible [PD00]), it is a good GPU fit. In particular, the denser sampling can make use of the built-in bilinear interpolation without explicit upscaling, which greatly improves performance and addresses the major bottleneck of our pipeline.

We choose α empirically (typically, $\alpha \in [0.1, 0.2]$), which gives visually-plausible results. The concrete value is chosen to best match the appearance of a real photograph. The reason being that, albeit conceptually well defined [PF94], an efficient and accurate computation of α is a significant challenge. We would need to not only evaluate per-object αs, but also composite FrFTs of different αs; for multi-lens systems, each bounce would need an additional FrFT evaluation when the aperture is traversed. As the visual differences are typically low, we propose to choose α empirically.

5.3. Anti-Aliased and Spectral Rendering

Our solution supports similar extensions as one by Hullin et al. [HESL11]. To avoid noise, we trace a beam of three rays and rasterize the resulting triangle with interpolation on the sensor. Additionally, we employ multisample antialiasing (MSAA). For spectral rendering, we compute different refractive indices depending on the wavelength [HESL11, SDHL11] and trace the result of each spectral sample separately.

6. Results and Discussions

We implemented our system using the OpenGL API on an Intel Core i7 machine with NVIDIA GTX 980 Ti. Four patented lenses were chosen for our experiments (see Figure 8 for illustrations, patent numbers, and the corresponding output generated by our approach). The red-colored interfaces indicate aspheric elements. Normal maps are applied to the aspheric surfaces only because the manufacturing imperfections are uncommon for spherical surfaces. In combination with the FrFT, the bokeh renderings appear similar to photographs.

Our algorithm performs the bokeh synthesis in a two-step process; an offline stage (the virtual grinding to produce the normal map, and the proxy-surface fitting on the CPU) and an online stage on the GPU (ray tracing through the lens system, rasterization into the sensor texture, and the FrFT for starburst streaks). Note that the offline preprocessing is light-weight requiring only 60 ms on average over all shown camera models. Table 1 summarizes the rendering performance during the online stage using 512^2 and 1024^2 resolutions for the entering rays on the entrance pupil and the sensor plane. We measured the result for the synthesis of an achromatic (single wavelength) bokeh resulting from a directional light source.

Overall, our system is capable of real-time performance, even when applying FrFT. Ray tracing and rasterization scale linearly with the complexity of the lens systems and the number of rays, while the FrFT scales with the resolution. FrFT is the most significant bottleneck, but still up to 20× faster than existing Matlab implementations (e.g., 2.8 s at 1024^2). For dispersion, we need to repeat the achromatic processing for each wavelength, which linearly scales with the number of spectral samples; e.g., computing with seven wavelengths takes around 1.5 s at a 1024^2 resolution.

6.1. Defocus Blur and Lens Flares

Similarly to previous work, we can use our model to compute defocus blur and lens flares, but even with our efficient ray-tracing

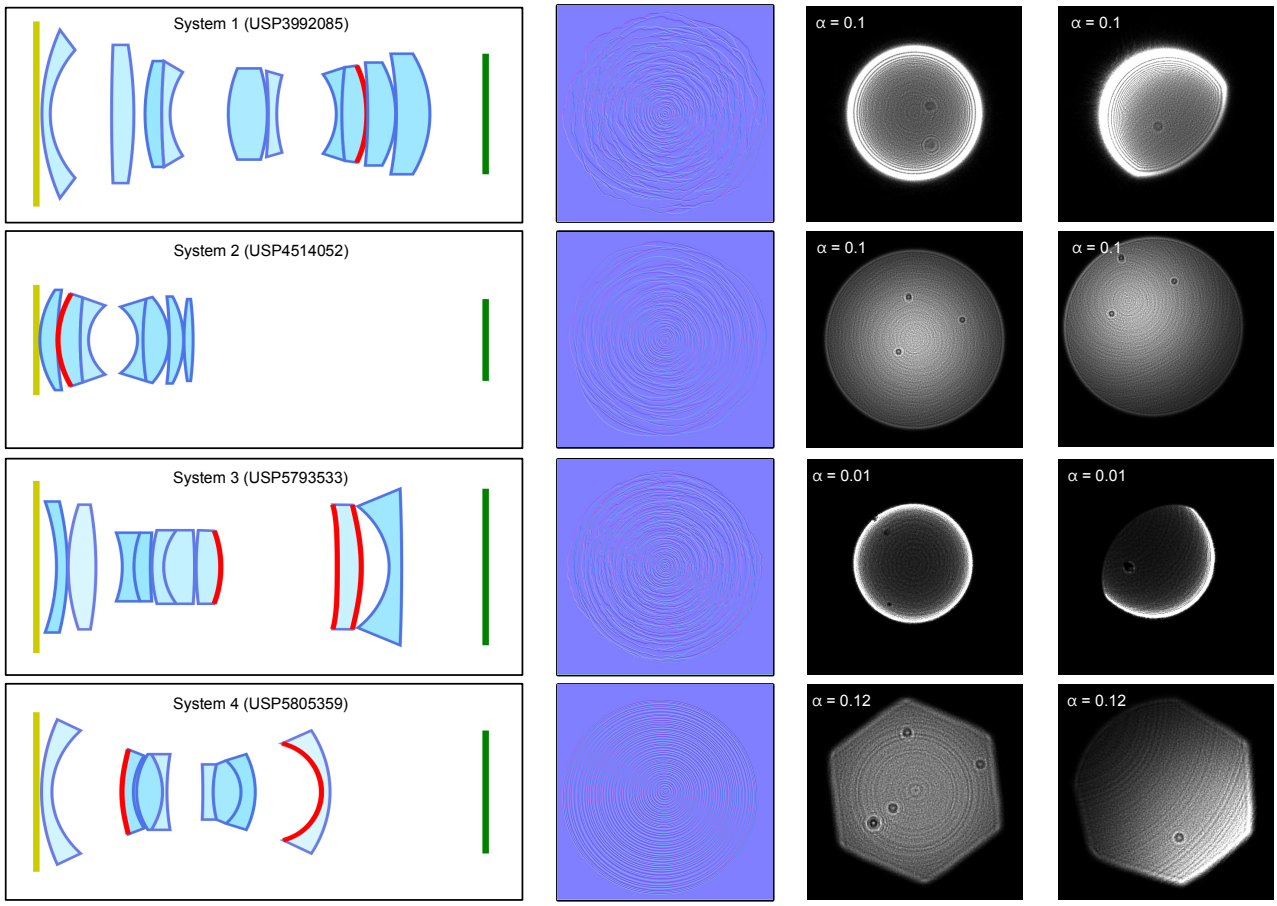

Figure 8: *Examples of lens systems (yellow, red, and green elements for the entrance pupil, aspheric elements, and sensor planes, respectively), normal maps (exaggerated for legibility) and their corresponding bokeh textures.*

Table 1: *Performance of bokeh rendering measured in frame time (ms) for the four lens systems with respect to the number of rays (ray tracing and raster) and resolution (FrFT).*

System ID	Ray tracing		Raster		FrFT	
	512^2	1024^2	512^2	1024^2	512^2	1024^2
1	0.77	2.43	0.26	0.59	26.93	199.14
2	1.14	3.97	0.43	0.79	28.04	202.30
3	0.87	2.76	0.24	0.59	27.18	202.11
4	0.65	2.10	0.29	0.62	27.17	201.18

solution, it would be very costly to apply our approach for full depth-of-field or lens-flare rendering on entire scenes. It would require a large number (e.g., thousands) of lens samples, similarly to glossy highlights in Monte-Carlo rendering and there are no good solutions yet for this particular case, such as adaptive sampling or interpolation methods. Consequently, the ray tracing scheme and the fractional Fourier transforms can become bottlenecks. One possibility for higher efficiency is to generate a bokeh texture with our system, which is then used in a simplified simulation.

Figure 9 illustrates an example of thin-lens depth-of-field blur using accumulation buffering [HA90] in combination with a precomputed bokeh texture. Each rendered view is modulated by the intensity stored in the bokeh texture. We used a 256^2 resolution in the accompanying video and 512^2 in the figure. In practice, 256^2 usually suffices. The accumulation buffering remains a costly process (e.g., 100 s for 256^2 samples, even though a single sample only takes 1.5 ms) but it would be possible to rely on faster methods [LES09, LES10].

Similarly, we can approximate lens flare. Real-time lens-flare rendering typically uses the same type of pre-defined texture, which illustrates the light rays traversing the aperture. We used a recent sprite-based lens-flare rendering [LE13] for demonstration purposes (Figure 10). It is based on the paraxial linear approximation to locate the flares in real time.

7. Conclusion and Discussions

We presented an efficient ray-tracing technique for parametric aspheric lenses. Our solution includes the simulation of mechanical imperfections, which builds upon the lens-fabrication process and

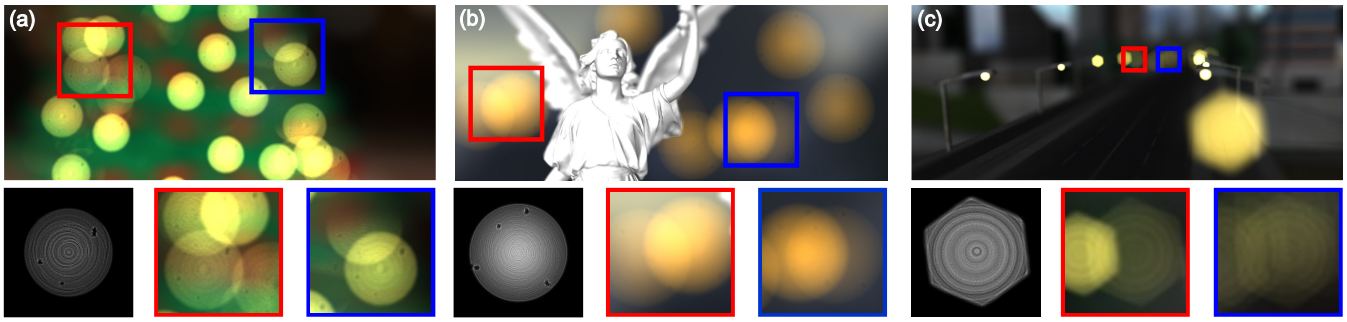

Figure 9: *Examples of defocus blur [HA90] using bokeh textures generated with our solution.*

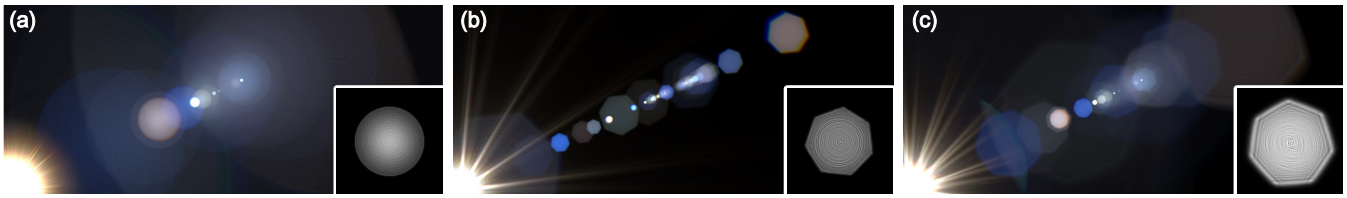

Figure 10: *Examples of a texture sprite-based lens-flare rendering [LE13] to which our bokeh textures applied.*

supports various effects. It is possible to render high-quality images and to generate realistic bokeh effects for use in real-time lens-flare or defocus algorithms.

Our method improves existing optical simulations, which can be used to compute precise out-of-focus blur. Ray tracing for a single lens sample is still light-weight, but strong highlights need many lens samples in order to avoid noise. One viable alternative for real-time purposes, which would sacrifice some accuracy, would be a polygonal beam tracing and adding imperfections in a postprocess.

While our system supports artistic parameter choices, it would be interesting to offer a way to restrict the fabrication parameters to realistic values. We would also like to explore how to directly derive the parameters from photos.

Acknowledgments

The Christmas Tree and Lucy models are provided through the courtesy of cgaxis.com and Stanford 3D scanning repository, respectively. This work was supported in part by the Mid-career and Global Frontier (on Human-centered Interaction for Coexistence) R&D programs through the NRF grants funded by the Korea Government (MSIP) (Nos. 2015R1A2A2A01003783, 2012M3A6A3055695) and the Intel Visual Computing Institute at Saarland University and the EU Project Harvest4D. Correspondence concerning this article can be addressed to Sungkil Lee.

References

[AGM06] ABERT O., GEIMER M., MULLE S.: Direct and fast ray tracing of NURBS surfaces. In *IEEE Symp. Interactive Ray Tracing* (2006), IEEE, pp. 161–168. 2

[AS52] ALLEN W. A., SNYDER J. R.: Ray tracing through uncentered and aspheric surfaces. *J. Optical Society of America 42*, 4 (1952), 243–249. 2, 3

[BES12] BABOUD L., EISEMANN E., SEIDEL H.-P.: Precomputed safety shapes for efficient and accurate height-field rendering. *IEEE Trans. Visualization and Computer Graphics 18*, 11 (2012), 1811–1823. 3

[BF01] BURDEN R. L., FAIRES J. D.: *Numerical analysis*. Brooks/Cole, 2001. 3

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Trans. graphics 1*, 3 (1982), 235–256. 2

[BW02] BUHLER J., WEXLER D.: A phenomenological model for bokeh rendering. In *ACM SIGGRAPH Abstracts and Applications* (2002), ACM, pp. 142–142. 2

[Che87] CHEN Y. C.: Lens effect on synthetic image generation based on light particle theory. *The Visual Computer 3*, 3 (1987), 125–136. 2

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Tran. Graphics 5*, 1 (1986), 51–72. 2

[EHS05] EFREMOV A., HAVRAN V., SEIDEL H.-P.: Robust and Numerically Stable Bézier Clipping Method for Ray Tracing NURBS Surfaces. In *Proc. Spring Conf. Computer Graphics* (2005), ACM, pp. 127–135. 2

[For66] FORD P.: Aspheric ray trace. *J. Optical Society of America 56*, 2 (1966), 209–211. 2, 3

[FS05] FOLEY T., SUGERMAN J.: KD-tree acceleration structures for a GPU raytracer. In *Proc. Graphics hardware* (2005), pp. 15–22. 2

[FTG00] FISCHER R. E., TADIC-GALEB B.: *Optical system design*. McGraw-Hill, 2000. 2

[GKK15] GOTANDA Y., KAWASE M., KAKIMOTO M.: Real-time rendering of physically based optical effects in theory and practice. In *ACM SIGGRAPH 2015 Courses* (2015), ACM, p. 23. 2

[HA90] HAEBERLI P., AKELEY K.: The accumulation buffer: hardware support for high-quality rendering. *ACM Computer Graphics 24*, 4 (1990), 309–318. 5, 6

[Han83] HANRAHAN P.: Ray tracing algebraic surfaces. *ACM Computer Graphics 17*, 3 (1983), 83–90. 2

[HESL11] HULLIN M., EISEMANN E., SEIDEL H.-P., LEE S.: Physically-Based Real-Time Lens Flare Rendering. *ACM Trans. Graphics 30*, 4 (2011), 108:1–9. 1, 2, 3, 4

[JB86] JOY K. I., BHETANABHOTLA M. N.: Ray tracing parametric surface patches utilizing numerical techniques and ray coherence. *ACM Computer Graphics 20*, 4 (1986), 279–285. 2

[Kaj82] KAJIYA J. T.: Ray Tracing Parametric Patches. *ACM Computer Graphics 16*, 3 (1982), 245–254. 2

[Kin75] KINTNER E. C.: Edge-ringing and fresnel diffraction. *Optica Acta: International Journal of Optics 22*, 3 (1975), 235–241. 4

[KMH95] KOLB C., MITCHELL D., HANRAHAN P.: A realistic camera model for computer graphics. In *Proc. ACM SIGGRAPH* (1995), ACM, pp. 317–324. 1, 2, 3

[KT80] KINGSLAKE R., THOMPSON B.: *Applied optics and optical engineering*. Academic, 1980. 1

[LB07] LEE E.-S., BAEK S.-Y.: A study on optimum grinding factors for aspheric convex surface micro-lens using design of experiments. *Machine Tools and Manufacture 47*, 3 (2007), 509–520. 1, 2

[LE13] LEE S., EISEMANN E.: Practical Real-Time Lens-Flare Rendering. *Computer Graphics Forum (Proc. EGSR'13) 32*, 4 (2013), 1–6. 2, 5, 6

[LES09] LEE S., EISEMANN E., SEIDEL H.-P.: Depth-of-field rendering with multiview synthesis. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 134:1–134:6. 5

[LES10] LEE S., EISEMANN E., SEIDEL H.-P.: Real-Time Lens Blur Effects and Focus Control. *ACM Trans. Graphics (Proc. SIGGRAPH'10) 29*, 4 (2010), 65:1–7. 2, 5

[Man94] MANOCHA D.: Solving systems of polynomial equations. *Computer Graphics and Applications 14*, 2 (1994), 46–55. 2

[McG15] MCGRAW T.: Fast bokeh effects using low-rank linear filters. *The Visual Computer 31*, 5 (2015), 601–611. 2

[MH14] MOERSCH J., HAMILTON H. J.: Variable-sized, circular bokeh depth of field effects. In *Proc. Graphics Interface* (2014), pp. 103–107. 2

[MNN*10] MORITA S.-Y., NISHIDATE Y., NAGATA T., YAMAGATA Y., TEODOSIU C.: Ray-tracing simulation method using piecewise quadratic interpolant for aspheric optical systems. *Applied optics 49*, 18 (2010), 3442–3451. 2

[MRD12] MCINTOSH L., RIECKE B. E., DIPAOLA S.: Efficiently Simulating the Bokeh of Polygonal Apertures in a Post-Process Depth of Field Shader. *Computer Graphics Forum 31*, 6 (2012), 1810–1822. 2

[NSK90] NISHITA T., SEDERBERG T. W., KAKIMOTO M.: Ray Tracing Trimmed Rational Surface Patches. *ACM Computer Graphics 24*, 4 (1990), 337–345. 2

[OSRM09] ORTIZ S., SIEDLECKI D., REMON L., MARCOS S.: Three-dimensional ray tracing on delaunay-based reconstructed surfaces. *Applied optics 48*, 20 (2009), 3886–3893. 2

[OZK01] OZAKTAS H. M., ZALEVSKY Z., KUTAY M. A.: *The fractional Fourier transform with applications in optics and signal processing*. Wiley, 2001. 4

[PBMH02] PURCELL T. J., BUCK I., MARK W. R., HANRAHAN P.: Ray tracing on programmable graphics hardware. *ACM Trans. Graphics 21*, 3 (2002), 703–712. 2

[PD00] PEI S.-C., DING J.-J.: Closed-form discrete fractional and affine fourier transforms. *IEEE Trans. Signal Processing 48*, 5 (2000), 1338–1353. 4

[PF94] PELLAT-FINET P.: Fresnel diffraction and the fractional-order fourier transform. *Optics Letters 19*, 18 (1994), 1388–1390. 4

[RIF*09] RITSCHEL T., IHRKE M., FRISVAD J. R., COPPENS J., MYSZKOWSKI K., SEIDEL H.-P.: Temporal Glare: Real-Time Dynamic Simulation of the Scattering in the Human Eye. *Computer Graphics Forum 28*, 2 (2009), 183–192. 2

[SDHL11] STEINERT B., DAMMERTZ H., HANIKA J., LENSCH H. P.: General spectral camera lens simulation. *Computer Graphics Forum 30*, 6 (2011), 1643–1654. 1, 2, 3, 4

[Smi00] SMITH W. J.: *Modern Optical Engineering*. McGraw-Hill, 2000. 1, 3

[Smi05] SMITH W. J.: *Modern lens design*. McGraw-Hill, 2005. 2

[ST14] SIVOKON V. P., THORPE M. D.: Theory of bokeh image structure in camera lenses with an aspheric surface. *Optical Engineering 53*, 6 (2014), 065103–065103. 1

[Tot85] TOTH D. L.: On Ray Tracing Parametric Surfaces. *ACM Computer Graphics 19*, 3 (1985), 171–179. 2

[Whi80] WHITTED T.: An improved illumination model for shaded display. *Comm. of the ACM 23*, 6 (1980), 343–349. 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

[WZHX13] WU J., ZHENG C., HU X., XU F.: Rendering realistic spectral bokeh due to lens stops and aberrations. *The Visual Computer 29*, 1 (2013), 41–52. 2