An Accurate and Practical Camera Lens Model for Rendering Realistic Lens Effects

Jiaze Wu*†, Changwen Zheng*, Xiaohui Hu* and Chao Li*†
*National Key Laboratory of Integrated Information System Technology
Institute of Software of Chinese Academy of Sciences
†Graduate University of Chinese Academy of Sciences

Abstract

In this paper, an accurate and practical camera lens model is proposed to be applied in realistic rendering of lens-related effects. The optical modeling of this new model is firstly presented from two aspects: lens surface modeling and formulation of ray tracing equations. Then, a number of tunable models for controlling lens properties are introduced and combined together to determine overall imaging performance. An implementation framework for this lens model is presented from two important aspects: its internal working framework and a new rendering pipeline for integrating it into a general ray tracer. In contrast to existing lens models, this new one is characterized by its ability to accurately model the image formation process and its friendly tunable models to control its lens properties. As a consequence, it is capable of simulating complex lens-related effects without too much expertise on lens optics. Finally, multiple rendering experiments are performed to demonstrate the ability and usage of this novel model to simulate a variety of complex lens-related effects.

1. Introduction

Many fields, such as realistic rendering, virtual and augmented reality and virtual studios, require accurate simulation of real physical lenses. However, widely used lens models in computer graphics, including pin-hole, thin lens and thick lens, can not accurately capture optical imaging characteristics of physical lenses, and therefore a large number of lens-related optical effects are absent in computer-synthesized images using these models. More accurate geometric lens models, firstly proposed by Kolb et al. [1] and then extended by Wu et al. [2], are able to model the complex imaging characteristics of physical lenses, but is limited to the spherical and dioptric lenses. However, non-spherical and non-dioptric (catadioptric) lenses are common in reality. In addition, geometric lens models are rarely applied in realistic rendering since they refer to many complex optical principles and are hard to operate for common users.

Motivated by these observations, a realistic and practical camera lens model is proposed in this paper to simultaneously improve the accuracy and practicability. In our new model, multiple lens surface types, such as plane, sphere, ellipsoid, paraboloid and hyperboloid, are supported, and ray tracing equations are deduced for accurately simulating optical behavior of physical lenses. In order to improve usage of our lens model, we introduce a set of models to tune its lens properties based on optical principles of lens design and photography, including focal length, field of view (FOV), diaphragm, normal focusing and selective focusing. For each lens property, one or multiple corresponding controllable parameters are provided. With these parameters, the proposed model is able to be conveniently and friendly operated to render arbitrary and interesting content of a 3D scene with realistic lens-related optical effects.

2. Related work

Existing camera lens models in computer graphics can be roughly divided into two categories: *abstract* lens models and *real* lens models.

The abstract models are just used to describe ideal and perfect image formation process. Pin-hole, thin and thick lens models fall into this category. These models are so far the most commonly used camera lens models in computer graphics, and especially suitable for real-time rendering because of their simplicity. However, due to their low accuracy, this kind of models rarely simulate any optical effects except the depth of field effect [3].

The real models take optical data of physical lenses as input, and precisely depict real and non-ideal image production process by the ray tracing technique. Kolb et al. [1] firstly proposed a geometric lens model for accurately modeling complex real lenses using sequential ray tracing, which is implemented into a path tracer for realistic image synthesis and able to simulate various optical aberrations. In order to improve the rendering performance, Heidrich et al. [4] introduced a novel image-based model, which is built on the geometric description of real lenses. Barsky et al. [5] extended Kolb's lens model to produce the visual effect of the human eye. Wu et al. [2] enhanced Kolb's model to render complex bokeh effect incurred by optical aberrations. Lee et al. [6] rendered the chromatic aberrations of a single lens by calculating different refractive indices for RGB channels, but not for the visual spectrum. In spite of the ability of simulating complex optical effects, these models are hard to operate in realistic rendering. In addition, the

978-0-7695-4497-7/11 $26.00 © 2011 IEEE
DOI 10.1109/CAD/Graphics.2011.18

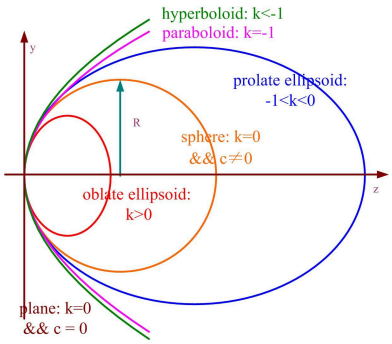


Figure 1. Conic surface types with the same curvature but different conic constants.

wavelength dependency of physical lenses is not considered by these models.

3. Optical modeling

3.1. Lens surface

In theory, lens surface can be made with arbitrary shape, but lens surface is made by common quadric surfaces for reducing manufacturing cost. For a convex conic surface with the apex at the origin of the coordinate system and the orientation in z-axis, it is represented as

$$z = F(x,y) = \frac{c(x^2+y^2)}{1+\sqrt{1-(1+k)c^2(x^2+y^2)}}, \quad (1)$$

where c refers to the base curvature of the conic surface at its apex, k to a conic constant with the expression $k=-e^2$ and defines its type, and e to its eccentricity. In order to deduce a serial of equations for exact ray tracing, Equation 1 is rewritten in a more useful form:

$$F(x,y,z) = x^2+y^2+z^2-2rz-e^2z^2 = 0. \quad (2)$$

For concave surfaces, r in Equation 2 has to be considered as the negative value of the apical curvature. In general, a plane is characterized by $k=0$ and $c=0$, a sphere by $k=0$ and $c\neq 0$, a paraboloid by $k=-1$, an oblate ellipsoid by $k>0$, a prolate ellipsoid by $-1<k<0$, and a hyperboloid by $k<-1$ (a graphical description in Figure 1).

3.2. Ray tracing

For different purposes, we present two different ray tracing algorithms from lens design field: exact (real) ray tracing and paraxial ray tracing. The exact ray tracing obeys Snell law when dealing with the specular refraction or reflection at the lens surface boundary, is used to transform a 3D scene to a 2D image, and usually cooperates with traditional ray tracing algorithms (for instance path tracing and bidirectional ray tracing) in the 3D scene to finish this transformation work. The paraxial ray tracing, the simplified version of the preceding algorithm, complies with simplified Snell law (the sine of angle in the law is replaced by itself assuming it is small), and is used to calculate basic optical properties of complex lens systems [7]. These optical properties are able to guide users to more precisely control our lens model to render a 3D scene.

3.2.1. Exact ray tracing. The exact ray tracing is performed surface by surface, and several important mathematical and optical calculations must be performed for each lens surface.

Intersection calculation. An incident light ray $\vec{R_0}$ with starting point $\vec{P_0}$ and unit direction $\vec{D_0}$ can be denoted as $\vec{P}=\vec{P_0}+t\vec{D_0}$. The intersection point $\vec{P_1}$ of the ray $\vec{R_0}$ with a lens surface F satisfies the following equation:

$$x_1^2+y_1^2+z_1^2-2rz_1-e^2z_1^2=0,$$

and it can be reformulated as a more useful vector form:

$$\vec{P_1}^2-2r\vec{k}\cdot\vec{P_1}-e^2(\vec{k}\cdot\vec{P_1})^2=0,$$

where $\vec{k}$ is the unit vector of the z axis, and substituting $\vec{P_1}=\vec{P_0}+t\vec{D_0}$ into and then reformulating this equation yields

$$\begin{cases} At^2+2Bt+C=0, \\ A=1-e^2\alpha_0^2, \\ B=\vec{D_0}\cdot\vec{P_0}-r\alpha_0-e^2x_0\alpha_0, \\ C=\vec{P_0}^2-2rx_0-e^2x_0, \end{cases} \quad (3)$$

and finally solving for t yields the desired intersection point.

Normal calculation. With the intersection point determined, we proceed to calculate the normal of the corresponding surface at this point, which is a prerequisite to obtain the direction vector of the refracted light ray in the next step. The unit vector of the normal $\vec{N}$ at arbitrary point on the lens surface can be denoted as

$$\begin{cases} \lambda = \frac{-F'_x}{\sqrt{F'^2_x+F'^2_y+F'^2_z}}, \\ \mu = \frac{-F'_y}{\sqrt{F'^2_x+F'^2_y+F'^2_z}}, \\ \nu = \frac{-F'_z}{\sqrt{F'^2_x+F'^2_y+F'^2_z}}, \end{cases} \quad (4)$$

where F'_x, F'_y and F'_z are the partial derivatives of the lens surface with respect to x, y and z respectively. Substituting the coordinate of the intersection point $\vec{P_1}$ into Equation 4, the unit normal $\vec{N}$ at this intersection point can be denoted as

$$\begin{cases} \lambda = -\frac{x_1c}{\sqrt{1+c^2e^2(x_1^2+y_1^2)}}, \\ \mu = -\frac{y_1c}{\sqrt{1+c^2e^2(x_1^2+y_1^2)}}, \\ \nu = \frac{1-z_1(1-e^2)c}{\sqrt{1+c^2e^2(x_1^2+y_1^2)}}. \end{cases} \quad (5)$$

Refraction calculation. After obtaining the unit normal $\vec{N}$ at the intersection point $\vec{P_1}$ on current lens surface, the

refracted ray $\vec{R_1}$ of the incident ray $\vec{R_0}$ is easily determined according to the following equation:

$$\begin{cases} \vec{R_1} = \frac{n}{n'}\vec{R_0} + \Gamma\vec{N}, \\ \Gamma = \sqrt{1 - \frac{n^2}{n'^2}(1 - (\vec{R_0} \cdot \vec{N})^2)} - \frac{n}{n'}\vec{R_0} \cdot \vec{N}, \end{cases} \tag{6}$$

where n and n' are the refractive indices of the incident and refractive media. Here, the detailed deduction process can be seen from the paper [2].

3.2.2. Paraxial ray tracing. If the inclination of a light ray to the optical axis is small enough, the sines and tangents of the various inclination angles in the equations of exact ray tracing may be replaced by the angles themselves (in radians). These equations are then reduced to their first-order approximation (or Gaussian approximation), namely paraxial ray tracing equations. Such paraxial equations are widely used in the lens design field, and provide a very well-established method to analyze the optical properties of complex lens systems, such as object principle plane, image principle plane, object focal plane, image focal plane, object plane, image plane, entrance pupil and exit pupil. For this kind of ray tracing, intersection calculation is fairly simple and only refers to the intersection of an incident ray and a planar surface because the lens surface is assumed to be planar in paraxial region. The normal calculation is obsolete because it is no more the prerequisite of the refraction calculation, which can be determined by a set of more concise equations. In order to obtain the refracted ray in a more simple way, we need to employ a set of important equations [8]:

$$\begin{cases} i = \frac{l-r}{r}u, \\ i' = \frac{n}{n'}i, \\ u' = u + i - i', \\ l' = r + r\frac{i'}{u'}, \end{cases} \tag{7}$$

where u and u' indicate the angles of the incident and refracted rays with respect to the optical axis, i and i' the incident and refracted angles, l and l' the object and image interceptions, and r is the radius of the lens surface. Figure 2 illustrates these equations in a graphical way. Substituting the angles, u and u', by their paraxial relations in terms of the distance from the lens surface to the object and image points, l and l', yields an useful relation:

$$\begin{cases} u = \frac{h}{l}, \\ u' = \frac{h}{l'}, \end{cases} \tag{8}$$

where h is the intersection height of the incident ray with the lens surface. Substituting the first and fourth equations in Equations 7 into their second equation, and simultaneously

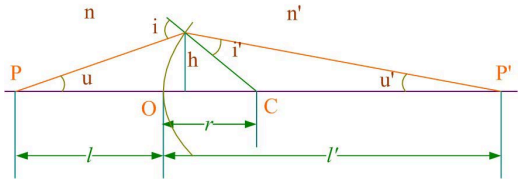


Figure 2. A graphical illustration of paraxial ray tracing for single lens surface.

utilizing Equations 8 yields an important equation:

$$\begin{cases} n'u' = nu + \frac{n'=n}{r}h, \\ u = \frac{\alpha}{\gamma} or \frac{\beta}{\gamma}, \\ u' = \frac{\alpha'}{\gamma'} or \frac{\beta'}{\gamma'}, \end{cases} \tag{9}$$

which is used to calculate the refracted ray.

4. Tunable lens property models

In order to bring our lens model into practical use in realistic rendering, we present multiple controllable models to tune lens properties frequently used in lens optics and photograph, each of whom provides one or multiple tunable parameters. These property models are combined together to drive our lens model to feature the desired optical imaging performance.

Focal length. The focal length is usually fixed when the lens prescription is specified. However, Smith [9] pointed that a lens prescription can be changed to arbitrary desired focal length by scaling all of its dimensions with an appropriate const factor. Based on the optical property, our lens model provides a focal length varying model, $f_s = s_f f$, where s_f is a scale parameter to define the ratio of the desired focal length (f_s) and original focal length (f). When the scale parameter is given, all the dimensions, including radius, thickness and aperture, should be multiplied by the same scale parameter to obtain the expected focal length.

Field of view. Like real photography, the FOV of a lens model can be used to determine the angular extent of a given virtual 3D scene when applied to realistic rendering. Existing lens models usually provide an immediate parameter to control their FOV, but this does not reflect the physical process of FOV variation. In optics, variation of the FOV is mainly dependent on two factors: *film size* and *film distance*. By considering the film dimension while tracing rays originated from the film plane, our lens model provides a tunable FOV model to exactly control the FOV. Once either of both parameters is changed, the FOV will vary in a physically-based way due to use of the exact ray tracing method inside our model.

Diaphragm. The diaphragm of a physical lens acts as an aperture stop, which limits the amount of light entering a lens system. For our lens model, the diaphragm model is

controlled by two different parameters: *f-number* (also called relative aperture or f-stop), which decides the size of the aperture stop, and *blade number*, which indicates the number of the blades constituting the aperture stop. The f-number ($F/\#$) is the ratio of the focal length (f) with respect to the aperture diameter (D), $F/\# = f/D$, and therefore small f-number means large aperture.

Normal focusing. The normal focusing process refers to movement of the focal plane along the optical axis. Movement of the image plane also causes movement of the focal plane. Therefore, the focusing process can be controlled by moving the focal plane or image plane along the optical axis. Our lens model provides a normal focusing model to control the focusing process, which refers two interrelated parameters: *focal distance* (U) and *film distance* (V) . Both parameters satisfy the following thin equation, $\frac{1}{U} + \frac{1}{V} = \frac{1}{f}$.

Selective focusing. Selective focusing refers to the rotation of the image plane with respect to the lens, unlike the image plane parallel to the lens in the normal focusing. Selective focusing is usually used to direct viewer's attention to a small part of the image while de-emphasizing other parts. For our lens model, selective focusing model is realized by tracing rays from a tilted image plane, and contains two kinds of different rotations: rotating the image plane around its horizontal axis (*tilt angle*) or vertical axis (*swing angle*).

5. Implementation

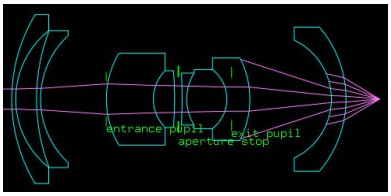
5.1. Lens selection

For our lens model, users can freely select their expected physical lens by specifying a lens file created with a lens prescription, which is widely used in lens design and describes the detailed optical data of a lens system (Figure 3). A large number of lens prescriptions can be freely found in a variety of lens-related textbooks and lens design software. When users want to use a special lens for rendering a scene, they only need to create a lens file according to the corresponding lens prescription. The lens file is passed to and parsed by our developed program and then is used to create our lens model.

5.2. Working framework

Figure 4 illustrates the internal working framework of our lens model, which shows how our model works in practice.It is noticed that the practical running framework can be divided into two stages in order: initialization and ray tracing.

The initialization stage involves several steps in sequence as follows: **1) Specifying lens parameters.** Our lens model provides a set of lens parameters to control its different



(a) 2D profile view

#radius	thickness	glass	diameter	conic	isstop
138.198	3.4	PK1	143	0.0	0
73.801	17.8	air	116	0.0	0
162.285	3.4	FK5	116	0.0	0
73.54	56.2	air	108	0.0	0
76.383	40.1	LAK10	78	0.0	0
33.62	18.2	BAF5	48	0.0	0
-267.237	3.2	air	48	0.0	0
0.0	3.2	air	37.6	0.0	1
-346.38	3.9	BAK4	44	0.0	0
41.93	34.3	SK15	44	0.0	0
-31.48	19.6	SF18	50	0.0	0
-76.359	69.6	air	70	0.0	0
-55.279	12.5	LAKN6	100	0.0	0
-93.624	24.04713	air	122	0.0	0

(b) Lens file

Figure 3. A wide-angle lens obtained from [9].

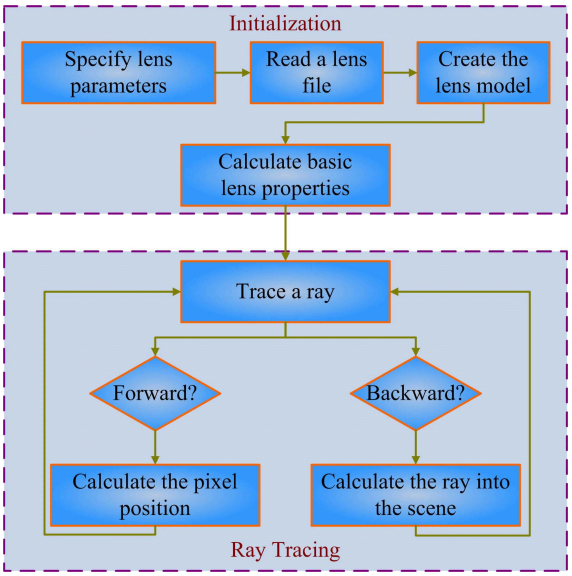


Figure 4. Working framework of our lens model.

lens properties, and these parameters work together as a flexible and convenient interface to drive the model to meet the desired imaging properties. **2) Reading a lens file.** The lens file is the groundwork for accurately modeling various optical behaviors of complex physical lenses. The detailed parameters of real lenses are often unobtainable because of patent protection. However, fortunately, many textbooks [9], [10], in lens design field provide plenty of lenses with detailed optical data. The format of the lens file is illustrated in Figure 3. The lens data is read and parsed from the formatted lens file, and then stored to a special data

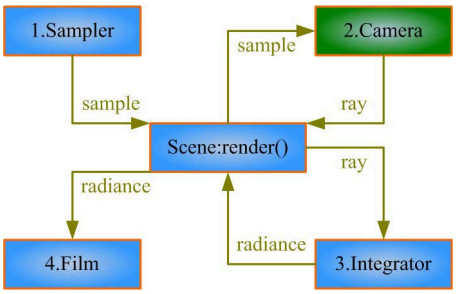


Figure 5. Traditional rendering pipeline for a general physically-based renderer.

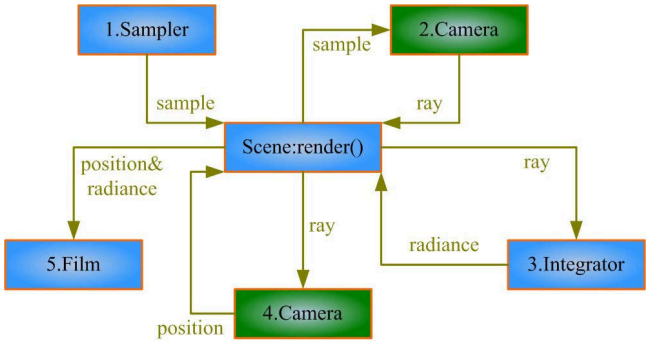


Figure 6. New rendering pipeline for integrating our lens model.

structure for the following model creation. **3) Creating the lens model.** Taking the above special data structure as input, each lens surface is created from left to right, shown in the 2D profile view of Figure 3, and then the entire lens system model is established for the ray tracing executed later. **4) Calculating basic lens properties.** After the lens model is created, the paraxial ray tracing introduced in Section 3.2 is used to calculate their basic lens properties, such as object principle plane, image principle plane, object focal plane, image focal plane, object plane, image plane, entrance pupil and exit pupil.

The ray tracing stage refers to cooperation between the ray tracing algorithm for this lens model and the ray tracing algorithms in a 3D scene for a general renderer, which is discussed in the following text.

5.3. Rendering pipeline

In order to make our lens model support realistic rendering, a widely distributed rendering software in computer graphics, LuxRender [11], is chosen as the rendering platform, where our model is integrated. LuxRender is a physically-based and unbiased rendering engine in computer graphics community, which is extended from PBRT [12], another widely used rendering engine by researchers. Figure 5 illustrates the basic rendering pipeline of LuxRender, which consists of four basic components, namely *sampler*, *camera*, *integrator* and *film*. The *render* function plays a central role to bring these four components together. From this figure, traditional rendering pipeline is roughly divided into four stages: 1) Sampler generates a sample using a variety of sampling algorithms and passes it to the camera; 2) Camera generates an initial ray using the sample from the sampler, and sends it to the integrator; 3) Integrator receives the initial ray from the camera, traces it in the 3D scene by generating a corresponding light path, calculates the radiance carried along this light path, and then forwards the radiance to the film; 4) Film records the radiance calculated from the integrator. This rendering pipeline is executed iteratively for each pixel of an image until specified conditions are met, for instance the maximum value of sample number per pixel is arrived.

Our lens model is implemented as a part of the camera, and integrated into the traditional rendering pipeline, but leads to low efficiency under spectral rendering environment. Aiming at this problem, a new rendering pipeline is presented to support more efficient spectral rendering, which is shown in Figure 6. The new pipeline is composed of five stages: 1) Sampler generates a sample like the first stage in the old rendering pipeline; 2) Camera generates an initial ray between the entrance pupil and focal plane using the sample from the sampler, and sends it to the integrator; 3) Integrator receives the initial ray from the camera, traces it in the 3D scene by generating a corresponding light path, calculates the radiance carried along the light path, and then forwards the radiance and initial ray to the camera; 4) Camera inversely traces the initial ray in our lens model to obtain the pixel position which the radiance contributes to, and then passes the radiance and pixel position to the film; 5) Film records the radiance at the specified pixel position.

5.4. Graphical interface

As an example of applying our lens model into practice, we have successfully integrated it into a 3D modeling software, Blender [13]. We develop a graphical user interface (Figure 7) using Python language, and this interface is built on and integrated into the Blender exporter of LuxRender [11]. With this updated exporter, we can immediately invoke LuxRender to render realistic lens effects after creating the 3D scene in Blender. With this GUI, we can conveniently modify model parameters to obtain desired optical imaging properties. The 3D scene can be created at arbitrary measurement unit in Blender, but our model is built in millimeters. Therefore, to compensate the inconsistency, the GUI provides three scale parameters for the transformation from the world space of the 3D scene to the camera space of our model.

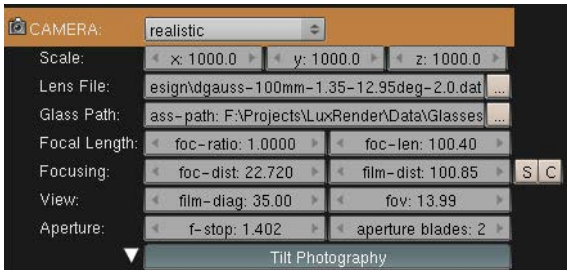


Figure 7. A graphical user interface for operating our lens model.

6. Rendering results

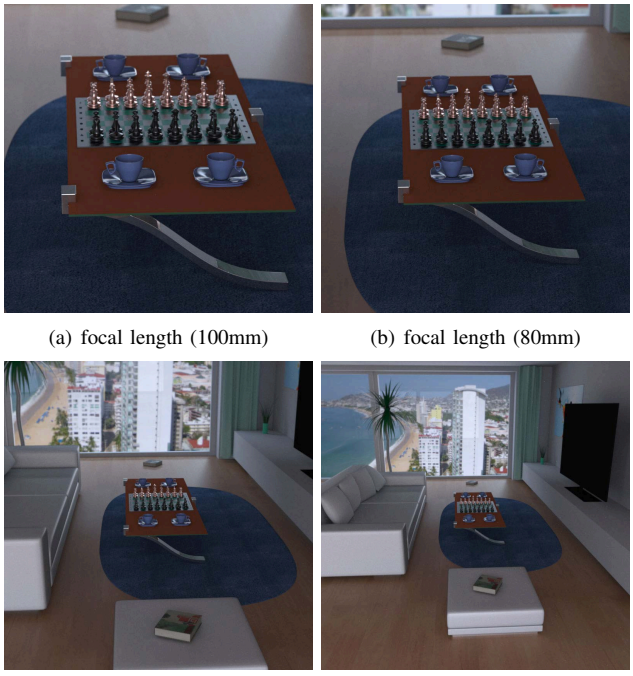
We performed several sets of different rendering experiments to demonstrate the practicability and accuracy of our lens model, and the corresponding rendering results are displayed and discussed in the following part of this section. All the lenses used in rendering experiments are obtained from the lens design textbooks [9].

FOV variation. Figure 8 illustrates gradually increasing FOV by tuning the film size and focal length, and a wide-angle lens (F/3.4) is used with the f-number equal to F/8.0. From Figure 8 (a) to (b), the film size (diagonal) is fixed to 30mm, but the focal length is changed from 100mm to 80mm, which means the focal length ratio changes from 0.8 to 0.5. From Figure 8 (c) to (d), the focal length is fixed to 60mm, but the film size decreases from 50mm to 80mm.

Normal focusing. Figure 9 shows a gradually focusing process from near to far by tuning the focus distance, and a double-Gauss lens (F/1.35) is used with the aperture fully open and the focal length of about 100mm. When the focus distance is set to 2000mm, the book in the foreground is sharply focused. Then changed to 3000mm, and the chesses are in clear focus. If the lens model is focused at 5000mm, the book near the window is very clear. After the focus distance is increased to 10000mm, everything in the room are blurry, but the landscape out of the window is fairly clear. Although not referred, the film distance is similarly used to realize the same focus processing according to the normal focusing model.

Selective focusing. Figure 10 illustrates a variety of selective focusing effects using two different lenses, namely double-Gauss (F/2.0) and retro-focus (F/3.0), and both lenses are set with full aperture. For each lens, the film plane is swung clockwise (a vertical and clockwise rotation around itself) by 10°and 20°. Although the objects in the scene are in the same depth, but only the middle one is sharply focused and others are blurred in different degrees. It is noticed that various defocused effects appearing on the left and right parts of the images.

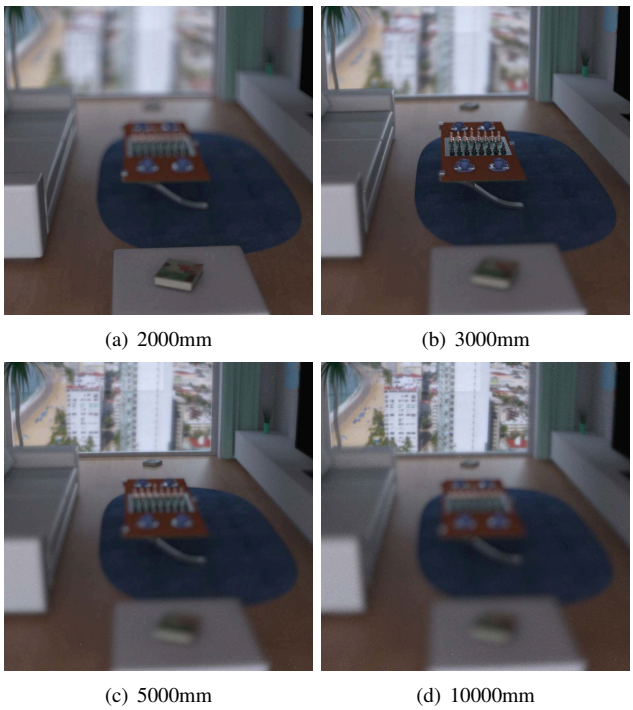
Aperture variation. Figure 11 shows a variety of lens



(a) focal length (100mm) (b) focal length (80mm)

(c) film size (50mm) (d) film size (80mm)

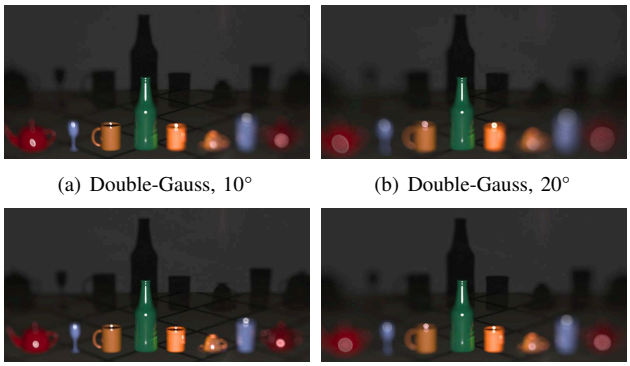
Figure 8. Change of the rendered coverage by modifying the film size and focal length.



(a) 2000mm (b) 3000mm

(c) 5000mm (d) 10000mm

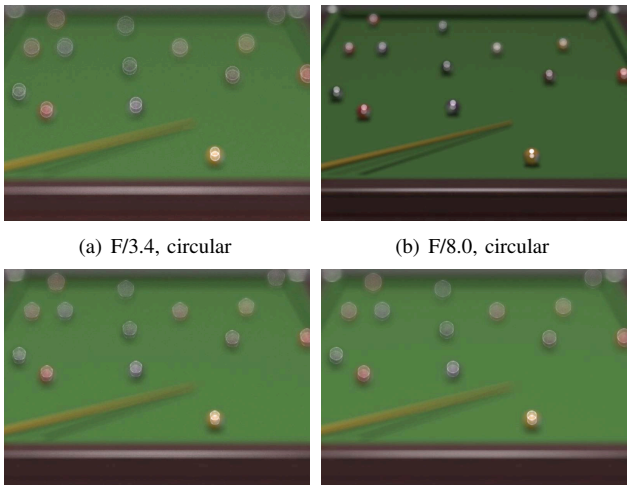
Figure 9. Varying focusing effects with the focal distance gradually increased.



(a) Double-Gauss, 10° (b) Double-Gauss, 20°

(c) Retro-focus, 10° (d) Retro-focus, 20°

Figure 10. Selective focusing effects of different lenses by clockwise swinging the film plane (a vertical and clockwise rotation around itself).



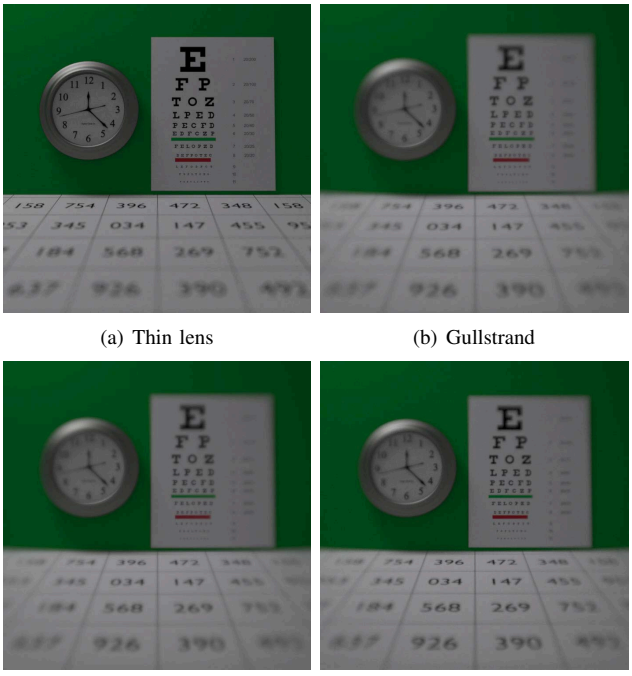
(a) F/3.4, circular (b) F/8.0, circular

(c) F/3.4, pentagonal (d) F/3.4, hexagonal

Figure 11. Various lens effects by changing aperture size and shape.

effects by changing aperture size and shape, and a wide-angle lens (F/3.4) is used with the focal length of about 100mm. For Figure 11(a), the aperture is fully open, and it is noticed that the interesting light intensity distribution within the COC appears, which is caused by the spherical aberration. When the aperture is turned down to F/8.0, the spherical aberration will be absent and consequently the distribution within the COC is more uniform, shown in Figure 11(b). For Figure 11 (c), the pentagonal aperture (5 blades), instead of circular aperture, is used and we can observe the pentagonal COC unlike the circular COC in Figure 11 (a). For Figure 11 (d), the number of the blades is increased to 6, and consequently the hexagonal COC appears in the image.

Vision simulation. Our lens mode can also be used to simulate the visual perception of the human eye. Figure 12



(a) Thin lens (b) Gullstrand

(c) LeGrand (d) Kooijman

Figure 12. Visual perception of different eye models under the unaccommodated (fully relaxed) state.

illustrates the visual results of three representative schematic eye models, namely Gullstrand [14], Le Grand [15] and Kooijman [16], the former two models only consist of spherical surfaces and the last model contains aspherical surfaces. In addition, the thin lens model is also used for reference. It is noticed that the thin lens model incurs blurring only in depth, which is characteristic of depth of field of the human eye. However, it fails to create horizontally or vertically gradual blurring, which is featured by optical aberrations of the human eye. In contrast, these schematic eye models are designed to attempt to accurately predict the aberrations inherent in the human eye, and therefore able to produce deep, horizontal and vertical blurring effects. However, it is seen that the results produced by Gullstrand and LeGrand models are much blurrier than Kooijman model because spherical surfaces introduce overmuch aberrations in contrast to aspherical surfaces.

7. Conclusion

We have implemented a valuable accurate camera lens model to improve the accuracy of the mapping process from the 3D scene to the 2D image surface, and its usage in rendering realistic optical effect caused by complex lens systems. With this new model, the imaging properties of physical lenses can be accurately simulated by modeling their different aspects, for instance lens surface modeling and

derivation of ray tracing equations. In order to make it easier to be operated in realistic rendering, this lens model provides multiple controllable models to tune the lens properties based on principles of lens design and photography. Aiming at the unsuitability to integrate this new model into existing ray tracing rendering pipeline, we have presented a new rendering pipeline to solve it.

Despite the ability to produce many complex lens-related optical effects, the proposed lens model fails to simulate some other optical phenomena, such as lens flare, which is caused by multiple back-and-forth refections between the lens surfaces. In optics, this kind of multiple reflections is incurred by Fresnel reflection and scattering occurring in physical lenses. Therefore, we will try to implement a more accurate and realistic lens model to account for more lens-related effects by modeling more complex optical behaviors in physical lenses. We also intend to employ adaptive sampling techniques [17]–[19] to accelerate synthesis of lens-related effects. In addition, it is possible to achieve both interactive and realistic rendering of complex lens effects by simplifying the proposed model in combination with the recent interactive ray tracing engine, OptiX [20].

Acknowledgements

We thank the LuxRender community and anonymous providers for the original version of the hotel room scene.

References

[1] C. Kolb, D. Mitchell, and P. Hanrahan, "A realistic camera model for computer graphics," in *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, Los Angeles, 1995, pp. 317–324.

[2] J. Wu, C. Zheng, X. Hu, Y. Wang, and L. Zhang, "Realistic rendering of bokeh effect based on optical aberrations," *The Visual Computer*, vol. 26, no. 6, pp. 555–563, 2010.

[3] M. Potmesil and I. Chakravarty, "A lens and aperture camera model for synthetic image generation," in *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, Dallas, 1981, pp. 297–305.

[4] W. Heidrich, P. Slusallek, and H. P. Seidel, "An image-based model for realistic lens systems in interactive computer graphics," in *Proceedings of Graphics Interface*, Kelowna, 1997, pp. 68–75.

[5] B. A. Barsky, B. P. Chen, A. C. Berg, M. Moutet, D. D. Garcia, and S. A. Klein, "Incorporating camera models, ocular models, and actual patient eye data for photo-realistic and vision-realistic rendering," 2003.

[6] S. Lee, E. Eisemann, and H.-P. Seidel, "Real-time lens blur effects and focus control," *ACM Transactions on Graphics (Proceedings of the SIGGRAPH conference)*, vol. 29, no. 3, pp. 1–7, 2010.

[7] R. E. Fischer, B. Tadic-Galeb, and P. R. Yoder, *Optical System Design*, 2nd ed. New York: McGraw-Hill, 2008.

[8] M. Born and E. Wolf, *Principles of optics*, 7th ed. Cambridge: Cambridge University Press, 1999.

[9] W. J. Smith, *Modern lens design*. New York: McGraw Hill, 1992.

[10] M. Laikin, *Lens Design*, 3rd ed. New York: Marcel Dekker, 2001.

[11] LuxRender. (2010, 10) Luxrender: Gpl physically based renderer. [Online]. Available: www.luxrender.net

[12] M. Pharr and G. Humphreys, *Physically based rendering: from theory to implementation*. San Francisco: Morgan Kaufmann, 2004.

[13] Blender. (2008-06-04) Blender: A free open source 3d content creation suite. Blender Foundation. [Online]. Available: www.blender.org

[14] A. Gullstrand, *Appendix in H. von Helmholtz*, 3rd ed. Physiologique Optic, 1909.

[15] Y. Le Grand and S. G. El Hage, "Physiological optics." Berlin: Springer-Verlag, 1980.

[16] A. C. Kooijman, "Light distribution on the retina of a wide-angle theoretical eye," *Journal of the Optical Society of America*, vol. 73, no. 11, pp. 1544–1550, 1983.

[17] C. Soler, K. Subr, F. Durand, N. Holzschuch, and F. Sillion, "Fourier depth of field," *ACM Transactions on Graphics*, vol. 28, no. 2, p. 18, 2009.

[18] T. Hachisuka, W. Jarosz, R. P. Weistroffer, and K. Dale, "Multidimensional adaptive sampling and reconstruction for ray tracing," *ACM Transactions on Graphics(Proceedings of the ACM SIGGRAPH conference)*, vol. 27, no. 3, p. 33, 2008.

[19] R. S. Overbeck, C. Donner, and R. Ramamoorthi, "Adaptive wavelet rendering," *ACM Transactions on Graphics(Proceedings of the ACM SIGGRAPH Asia conference)*, vol. 28, no. 5, p. 140, 2009.

[20] NVIDIA. (2011) Nvidia® optix ™ ray tracing engine. [Online]. Available: http://developer.nvidia.com/object/optix-home.html