An Image-Based Model for Realistic Lens Systems in Interactive Computer Graphics

Wolfgang Heidrich, Philipp Slusallek, Hans-Peter Seidel
Computer Graphics Group
University of Erlangen
Am Weichselgarten 9
91058 Erlangen, Germany

Phone: +49.9131.859926 Fax: +49.9131.859931
e-mail: {heidrich,slusallek,seidel}@informatik.uni-erlangen.de

Abstract

Many applications, such as realistic rendering, virtual and augmented reality, and virtual studios, require an accurate simulation of real lens and camera systems at interactive rates, including depth of field and geometric aberrations, in particular distortions. Unfortunately, camera models used in Computer Graphics are either too simple to describe these effects or too expensive to simulate for interactive use.

In this paper, we introduce an image-based lens model that is powerful enough to simulate sophisticated properties of real lens systems, yet fast enough for interactive graphics. By exploiting coherence, common graphics hardware can be used to yield high frame rates.

Keywords: Lens Systems, Image-Based Rendering, Hardware Acceleration, Interactive Computer Graphics

1 Introduction

The accurate simulation of properties of complex lens systems, including depth of field and geometric aberrations, in particular distortions, are of high importance to many applications of computer graphics. In the past, most approaches for simulating these properties have been based on off-line rendering methods such as distribution ray-tracing [2, 9].

Efforts for improved lens and camera models for interactive computer graphics have mainly been restricted to the simulation of depth of field [8, 13]. On the other hand, a model that allows for a more accurate simulation of real lens systems would be particularly useful for interactive graphics, because it could not just be used for photorealistic rendering, but also for combining real and synthetic scenes, for example in augmented reality and virtual studios. For these environments it is necessary to simulate real lens systems, so that real-world and synthetic objects can be merged into a single image.

In this paper, we describe an image-based camera model for interactive graphics, which is capable of simulating a variety of properties of real lens systems at high frame rates. Similarly to [9], the model uses the real geometry of lenses and computes an accurate approximation of the exposure on the film plane. However, instead of using ray-tracing, our model approximates the light field [10, 6] between the lens and the film plane. By making use of the coherence in ray-space, common computer graphics hardware can be used for sampling the light field and rendering the final image.

2 Existing Lens Models

We start by briefly reviewing existing camera models, before we describe our new model, and relate it to the older ones.

2.1 Pinhole Model

The pinhole model of a camera is by far the most frequently used camera model in computer graphics. The pinhole camera is a box with a small hole of negligible size in one of its sides. Light falls through this hole and projects an upside-down image onto the film on the opposite side of the box. The situation is depicted in Figure 1.

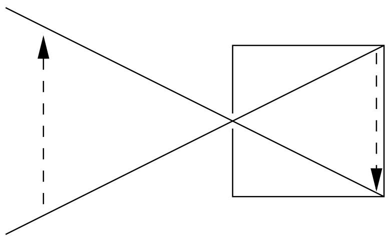

Figure 1: A pinhole camera

The advantage of this model is its simplicity. Because the size of the hole is negligible, the light falling through it can be assumed to be projected through a single point. This projection can be described as a perspective transformation, whose parameters are the dimensions of the film and the distance of the film plane from the hole, along with additional near and far clipping planes (see, for example [4]).

Usually, when dealing with perspective transformations, we do not think of it in terms of a pinhole camera, but as of a perspective projection with a center of projection (COP) and some virtual image plane in front of it. Throughout this paper, we use the term "image plane" for a virtual plane in front of the COP of a perspective projection, while the term "film plane" refers to the plane containing the film in a camera model.

Although it is actually possible to construct a physically working pinhole camera, this is not practical for several reasons. Most importantly, only very little light falls on the film since the hole is so small, and thus the exposure time has to be very long.

2.2 Thin Lens Model

Real lenses have a larger opening, called *aperture*, whose size can no longer be neglected. The simplest model of lenses with circular symmetry and finite aperture is the *thin lens approximation*. This model is used in optics and lens design to describe some of the properties of simple lens systems [1].

The fundamental assumption of the thin lens model is that the lens is of negligible thickness. As a consequence, light passing through the lens is refracted only in a single plane, the *principal plane*, and moves in a straight line otherwise.

Light coming from a single point Q in object space is focused in a single point in image space and vice versa. Incident light from object space parallel to the optical axis is focused in the *focal point* F' in image space, while parallel light from image space focuses in the focal point F in object space. Both F and F' lie on the optical axis of the lens. The situation is depicted in Figure 2.

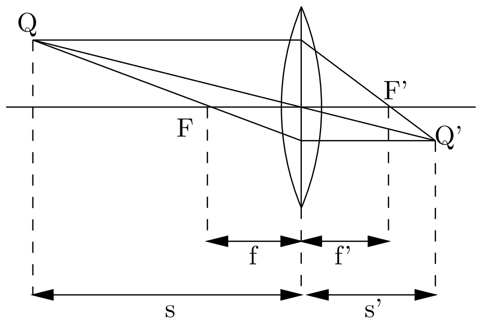

Figure 2: The geometry of a thin lens system

If both object space and image space are in the same medium, the distances f and f' of the focal points from the principal plane are equal, and the distance is called *focal length*. From this property it follows that rays passing through the center of the lens are not bent, but pass straight through the lens system.

With these informations it is possible to construct the image of a scene on a film plane at distance s' from the principal plane, given the aperture and the focal length f. For rendering it is often convenient to specify the distance s of the focal plane (the plane that contains all points that are focussed on the film plane) instead of the focal length. This distance can be easily derived from the well-known relationship

$$\frac{1}{s} + \frac{1}{s'} = \frac{1}{f}.$$

2.3 Rendering Thin Lenses

Typically, rendering of thin lenses is done using distribution ray-tracing [2]. For each sample point on the film, rays are cast through random sample points on the principal plane, and the resulting color values are averaged. Casting of the rays is done by computing the point on the focal plane that corresponds to the point on the film by shooting a ray through the center of the lens. The intersection point is then connected to the chosen sample point on the principal plane.

An alternative approach is to select a fixed set of sample points on the principal plane [8]. All rays passing through a single sample point p_i on the principal plane represent a perspective projection with p_i as the COP (see Figure 3a).

Each of the images generated by these perspective projections represent a 2-dimensional slice of the 4-dimensional light field in front of the lens system, as described in [10, 6]. In the following we call this light field the *scene light field*. Due to the properties of the thin lens model, this slice is identical to a slice of the *camera light field* between the lens system and the film plane, defined by the refracted rays. For the light field, we use the same parameterization as [10]. Each point in this 4-dimensional space corresponds to a ray from the film plane (two parametric directions) to the principal plane (another two parametric directions).

Because each slice of the light field can be computed using a standard perspective projection, computer graphics hardware can be used to render

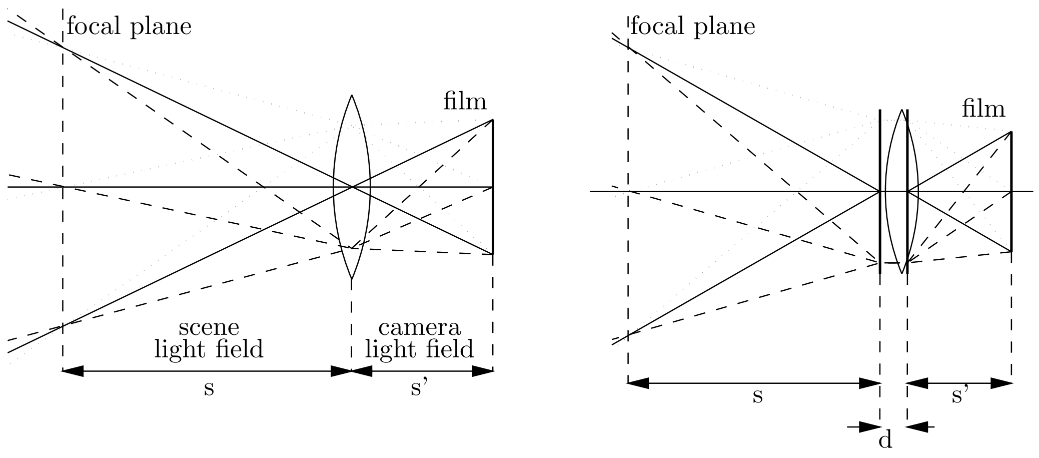

Figure 3: (a) Rendering using the thin lens approximation. The rays from all points on the film through a single sample on the aperture form a perspective transformation, which can be used to render a slice of the light field. (b) Rendering of a thick lens approximation is similar to rendering of thin lenses, except that an additional displacement of the ray is necessary. In both figures rays with the same line style originate from different points on the film, but pass through the same sample point on the lens. All rays with the same line style form a slice of the respective light field.

the slices. Averaging of slices from different sample points p_i to form the final image can be done using an accumulation buffer [8].

It should be noted that the simple averaging of the slices of the light field does not yield the correct exposure on the film, as pointed out by [9]. In Section 3.5 we will show how the algorithm can be extended to compute the correct exposure on the film.

Another approach to rendering depth of field effects of thin lens models is post-filtering as used in [12] and [13]. These methods require a filtering step to be performed in software, which causes high CPU loads and additional data transfers between the graphics board and main memory, and therefore leads to performance penalties. Moreover, post-filtering methods assume that each point in object space is mapped onto a circle of confusion on the image plane. This, however, is not a valid assumption for off-axis points in real lens systems.

2.4 Thick Lens Model

An improved model for circular symmetric lenses, where the thickness cannot be neglected, is the *thick lens model.* It is frequently used in optics for complex lens systems composed of several lenses and allows for a more exact approximation.

In contrast to the thin lens, a thick lens has two principal planes. The (signed) distance d between the two planes is called the thickness of the lens. Rays from object space hit the object-sided principal plane, then move in parallel to the optical axis, until they hit the image-sided principal plane, where they leave the lens (see Figure 3b).

From a rendering point of view, a thick lens can be treated very much like a thin lens, except for the shift in parallel to the optical axis. Both the thin and the thick lens model yield perfectly undistorted images. Real lenses, however, always show aberrations. Although lens designers usually try to minimize aberrations, they are often not negligible. Visualization and simulation of these effects requires a more sophisticated lens model.

2.5 Geometric Lens Model

The geometric lens model is based on the complete geometric description of the lenses together with their index of refraction. The model is evaluated by tracing rays through the true lens geometry, bending it at lens surfaces according to the change in the index of refraction.

The full geometric model correctly simulates all kinds of geometric aberrations, and is the only model that is capable of handling lenses without rotational symmetry, for example bi-focal or progressive addition lenses (PALs) used for eye glasses[11].

This model has been used in [9] to generate accurate simulations of complex lens systems using distribution ray-tracing. Unfortunately, the model is too expensive in terms of rendering time to be used for interactive graphics.

3 An Image-Based Camera Model

In following, we describe a new image-based model for arbitrary lens systems, which is capable of simulating aberrations, based on the geometric description of the lens. Despite this flexibility, it is possible to use computer graphics hardware to render images based on this model. As a result it is well suited to interactive applications requiring high frame rates.

Instead of directly using the full geometry of the lens, our model describes a lens as a transformation of the scene light field in front of the lens into the camera light field between the lens and the film. Every property of a lens system is completely described by such a transformation.

Computer graphics hardware can efficiently generate slices of light fields. Therefore, our model describes a lens system as a mapping from slices of the scene light field into corresponding slices of the camera light field. The mapping consists of two parts: selection of an appropriate slice of the scene light field for a given slice of the camera light field, and a morphing operation correcting for distortions due to the aberrations of the lens system.

We represent a slice of the scene light field as a perspective projection of the scene onto a suitable image plane. Thus, a lens is represented as a set of perspective projections and corresponding morphing operators.

Similar to the rendering of thin lenses, the final image is a composite of a number of slices of the camera light field. These slices are defined by a number of sample points p_i on the image-sided surface of the lens system.

3.1 Approximating Lens Systems

The approximation of real lens systems within the new model consists of two steps. For each slice of the camera light field, a corresponding slice in the scene light field has to be selected. Unfortunately, the 2-manifold in the scene light field corresponding to a given slice of the camera light field is not in general a planar slice, and therefore cannot be exactly represented as a perspective projection.

This fact can easily be observed by tracing rays from multiple points on the film through a single point on the lens surface. The rays leaving the system need not intersect in a single common point (which could be used as the COP of a perspective projection, see Figure 4). However, in many cases a perspective projection can be found that is a good approximation to the refracted rays.

In order to find the approximating perspective projection, rays are shot from a regular grid on the film through the point p_i on the image-sided surface of the lens system. Rays are bent as they pass through the lens, yielding a set of rays leaving the lens system on the object side, as shown in Figure 4.

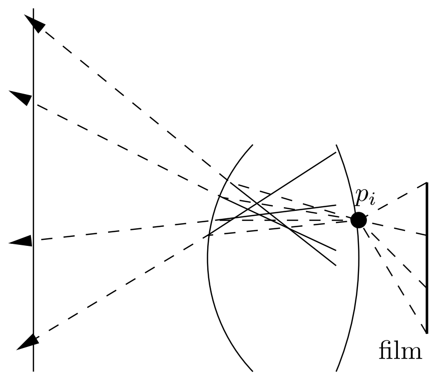

Figure 4: Tracing rays from a grid on the film through a single point p_i on the lens surface yields a set of refracted rays in object space. These do not in general intersect in a single point, and thus an approximate virtual center of projection has to be found.

This set of rays is then approximated with a projective transformation, which requires us to select a virtual image plane, and to find an appropriate center of projection. This is described in Section 3.3. An alternative way of interpreting this approximation is that the corresponding 2-manifold in ray space is linearly approximated with a plane.

The remaining parameters of the perspective transformation, in particular the upper, lower, left and right boundaries on the image plane can then be found by computing the bounding box of the intersections of the rays with the chosen virtual image plane.

Rendering the scene with the computed perspective transformations yields an image containing a slice of the scene light field. At the same time, this also is a distorted slice of the camera light field, with the distortions directly corresponding to the aberrations of the lens. We compensate these distortions using morphing with bilinear interpolation. This is achieved by texture-mapping the slice onto the grid on the film, from which the rays had been shot before. The intersections of the rays with the image plane are used as texture coordinates.

It is important to note that the ray-tracing step, as well as the computation of the COP and the texture coordinates only has to be performed once, as long as the geometry of the lens does not change. Only if the lens system is re-focused, or the aperture of a lens element changes, for example when adjusting an aperture stop, these calculations have to be repeated.

3.2 Hierarchical Subdivision

Lens systems with relatively small aberrations are usually well approximated by a single projective transformation, as described above. For example, this is true for lens systems that have a reasonable thick lens approximation.

However, for lens systems with strong aberrations, an approximation with a single perspective projection introduces a large error. In these cases, the grid on the film can be recursively subdivided and refined using a quad-tree structure. The algorithm is then applied recursively. This corresponds to finding a hierarchical, piecewise linear approximation of the 2-manifold in ray-space.

All quad-tree cells corresponding to a single sample point on the lens form a partition of the film. Combined, they contain the slice of the camera light field required for rendering.

Of course, the hierarchical subdivision introduces an additional rendering cost, since the scene has to be rendered once for every subdivision. Thus, the level of subdivision is a tradeoff between rendering time and image quality.

The complete pseudo-code for determining the perspective transformations is given below.

```
/* Perspective Projection */
generate rays from film plane
compute virtual COP from the refracted rays
if COP is good enough
  determine lower left and upper right
    of the image by intersecting the
    rays with the image plane
else
  subdivide the grid
  foreach subgrid
    recursive
```

3.3 Computing the Center of Projection

The sets of rays exiting the lens on the object side represent samples of a possibly complicated transformation that characterizes the lens system. The central task in approximating this transformation with a perspective projection is to find a good virtual COP.

This problem is also known in computer vision, where the aberrations of camera lenses have to be removed based on the measured distortions on a calibration grid [7]. The approach proposed by the authors of [7] is to choose the COP so that its distance from all rays is minimal in the least-squares sense.

For our purposes, this approach works well as long as there is no hierarchical subdivision. As soon as subdivision is performed, however, intolerable discontinuities occur between adjacent quad-tree cells. An analysis of the problem shows that the angle between the original rays and the corresponding rays in the perspective projection can be relatively large.

As a solution, we found that minimizing the angle between the original rays, and those generated by the perspective transformation yields much better results. This way, the maximum angle of the approximation could be reduced by a factor of up to 10. This maximum angle is also a good error criterion for terminating the hierarchical subdivision. The minimization process is outlined in the following.

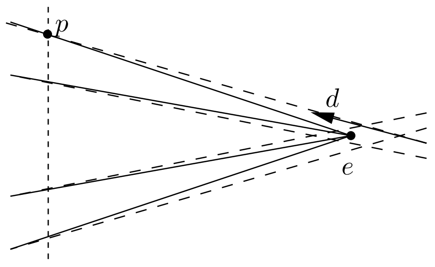

Figure 5: The center of projection is determined by minimizing the angle between the direction d of the original ray, and the vector $p - e$, where p is the intersection point with the image plane.

Given the intersection point p of a ray with the image plane and the normalized direction d of the ray, we would like to find a virtual COP e that minimizes the angle between d and $p - e$ over all rays (see Figure 5). Instead of minimizing this angle, it is also possible to maximize its cosine. The square of this cosine c^2 is given as

$$c^2 = \frac{1}{||p-e||^2} \langle d|p-e\rangle^2 \qquad (1)$$

A least-squares maximum of c is obtained by maximizing the sum $\sum c^2$ over all rays using Newton iteration. To this end, we need the first order Taylor polynomial of the derivative of c^2 around some initial guess e_0 for a COP. It can be easily verified that

$$\frac{\partial}{\partial e_x}\sum c^2 = \frac{2}{a^3}\Big[\sum ab\left(ap_x - bd_x\right) + \sum\left(b^2\left(4f_x^2 - a\right) + ad_x\left(ab - 4bf_x\right)\right)\cdot (e_x - e_{0,x}) + \sum\left(-2ab\left(d_yf_x + d_xf_y\right) + 4b^2f_xf_y + a^2d_xd_y\right)\cdot (e_y - e_{0,y}) + \sum\left(-2ab\left(d_zf_x + d_xf_z\right) + 4b^2f_xf_z + a^2d_xd_z\right)\cdot (e_z - e_{0,z})\Big] + O((e_x - e_{0,x} + e_y - e_{0,y} + e_z - e_{0,z})^2),$$

where $a = ||f||^2$, $b = \langle d|f\rangle^2$ and $f = p - e_0$. Similar formulas can be derived for $\partial c^2/\partial e_y$ and $\partial c^2/\partial e_z$. These three equations define a 3×3 system of linear equations that can be used to compute one step of a Newton iteration.

On the highest level of the subdivision, we use the center of the lens as an initial guess for e_0 in the first step of the iteration. For all other levels of subdivision, we use the virtual COP computed in the previous level as a starting point.

In practice, we have found that two or three iterations are usually sufficient. Again it is important to note that the computation of such an COP only has to be done once as a preprocessing step, or whenever the lens configuration changes.

3.4 Aperture Stops and Vignetting

So far, we assumed that every ray from a point on the film plane through one of the sample points on the lens actually passes through the whole lens system. This, however, is not a valid assumption for many lens systems. Camera lenses usually have an aperture stop. Rays may also be absorbed by other parts of the lens system, especially when they are shot from the outer regions of the film. This effect is called *vignetting*.

Thus, it is necessary to deal with occasions when a ray cast from one of the grid points on the film does not pass through the lens. In this case the grid has to be adjusted so that every ray passes through.

First, rays are cast from all grid points on the outer border of the grid. If one of these rays does not pass through the lens, bisection between the grid point an the center of the grid is used to move the grid point closer to the optical axis. Then the inner points of the grid are placed inside this border at approximately equal distances.

3.5 Radiometry

In the above description of both our algorithm and the algorithm for rendering thin and thick lenses, the final image has been generated by simply averaging the 2-dimensional slices of the light field. It has been pointed out in [9], that this is actually not a correct model for the exposure on the film.

Exposure is the integral of irradiance incident on the film over time. Assuming that a scene is static, that is, that the irradiance is constant over the exposure time, and that the shutter opens and closes instantaneously, the exposure is simply the irradiance times the exposure interval. Following the notation in [9], the irradiance $E(x')$ at a point x' on the film is given as

$$E(x') = \int_{x''} L(x'', x')\frac{\cos\theta'\cos\theta''}{||x'' - x'||^2}dA'', \qquad (2)$$

where x'' is a point on the image-sided surface of the lens, and dA'' is the differential area around this point. θ' and θ'' are the angles between the line connecting x' and x'', and the normal vectors in x' and x'', respectively. This integral is approximated as a discrete sum over the radiance contributed by the sample points on the image-sided surface of the lens:

$$E(x') = \frac{A}{n}\sum_{i=1}^{n} L(x_i'', x')\frac{\cos\theta_i'\cos\theta_i''}{||x_i'' - x'||^2}, \qquad (3)$$

where A is the area of the lens.

The term $L(x_i'', x')$ can be reconstructed from the camera light field for every sample point $x_i'' = p_i$ and every point x' on the film. The function $W_i(x') := \frac{\cos\theta_i'\cos\theta_i''}{||x_i''-x'||^2}$ is usually smooth and does not vary too much over the film plane. Thus, it is sufficient to evaluate this function at discrete points on the film, and then use linear interpolation in between.

Therefore, the irradiance as given in Equation 3 can be rendered by alpha-blending every slice $L(x_i'', x')$ with a Gouraud-shaded grid of textured polygons having $W_i(x')$ as the alpha values at the vertices.

The combination of this technique with the methods from previous sections results in the following preprocessing steps for rendering:

```
/* **Preprocessing** */
choose sample points $p_i = x_i''$
**foreach** sample point $p_i$ on the lens
  generate grid points $x_i'$ on the film
  generate perspective projection from
```

```
    the rays through $p_i$
  generate grid $W_i$ of polygons with
    alpha values $W_i(x'_j)$
```

After preprocessing, the following steps have to be performed in order to render every single frame:

```
/* **Rendering** */
clear the accumulation buffer
**foreach** sample point $p_i$ on the lens
  **foreach** projection $j$
    render the scene
    store the image as a texture $T_{i,j}$
**foreach** sample point $p_i$ on the lens
  **foreach** texture $T_{i,j}$ belonging to $p_i$
    map $T_{i,j}$ onto a 2D grid on the film
  alpha-blend the image with grid $W_i$
  add the image to the accumulation buffer
```

4 Results

We have tested our new lens model with several different lens systems. In particular, we have used an achromatic doublet as shown in the top row of Figure 6, as well as a simple bi-convex lens, shown in the bottom row.

The achromatic doublet is a real lens system that has been developed and used in real cameras in the 1920s [3]. The top left of Figure 6 shows an image rendered with the algorithms described in this paper, while the top right shows reference images of the same scene rendered using distribution ray-tracing.

In both images the barrel distortions of the lens are obvious. Moreover, in both images the outer regions of the film are blurred due to aberrations of the lens. Our method is the first one to allow for the simulation of this effect at interactive rates.

The hardware-rendered image was generated on a RealityEngine2 using 10 sample points on the lens with a frame rate of approximately 14 frames per second. A single perspective projection was used. The ray-traced image uses distribution ray-tracing with 10 samples per pixel, and took roughly 4.5 minutes for a 256×256 resolution on an Onyx with an R8000 processor. Note that the ray-traced image does not contain indirect illumination or shadows to allow for performance comparisons between the two algorithms.

The bottom row of Figure 6 shows the same scene rendered using a simple, uncorrected bi-convex lens. The image demonstrates the barrel distortion of the lens as well as significant depth of field effects. The scene was again rendered with 10 sample points, but this time the film plane was subdivided. As a criterion for the subdivision, we enforced a maximum angle of 0.05 degrees between an original ray and the corresponding approximated ray as described in Section 3.3. This resulted in a total of 40 perspective projections to be rendered for one frame. As a result, the frame rate decreased by approximately a factor of 4 to roughly 3 frames per second.

For the grid on the film plane we used an initial grid of 10×10 polygons. Higher grid sizes result in a higher number of rays that have to be traced in the preprocessing step. Lower grid resolutions require more subdivisions, which reduces the frame rate since more projections have to be rendered per frame.

5 Conclusion

In this paper, we have presented a novel image-based model for lens systems. The model allows for using graphics hardware for rendering, and is therefore well-suited for high-quality, interactive rendering.

Although the model is capable of simulating a larger variety of lens properties than previous models, there are some aspects of real lenses that cannot be handled. Most importantly, chromatic aberrations and frequency dependent refraction coefficients are not simulated, since contemporary computer graphics hardware only supports RGB rendering.

Another limitation of our model is the assumption of instantaneous shutter opening and a constant irradiance during the whole exposure time. More complex shutter functions and motion blur could be implemented by averaging multiple images over time. This, however would be too costly to achieve interactive frame rates on contemporary hardware.

In summary, our model adds a significant amount of realism to the simulation of lens systems in interactive graphics. Although it is not capable of simulating every property of real lenses, it constitutes a good compromise between quality of simulation and rendering performance.

References

[1] Max Born and Emil Wolf. *Principles of Optics*. Pergamon Press, Oxford, 1993.

[2] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, pages 134–145, July 1984.

[3] Johannes Flügge. *Das photographische Objektiv*. Springer Wien, 1955.

[4] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics, Principles and Practice, Second Edition*. Addison-Wesley, Reading, Massachusetts, 1990. Overview of research to date.

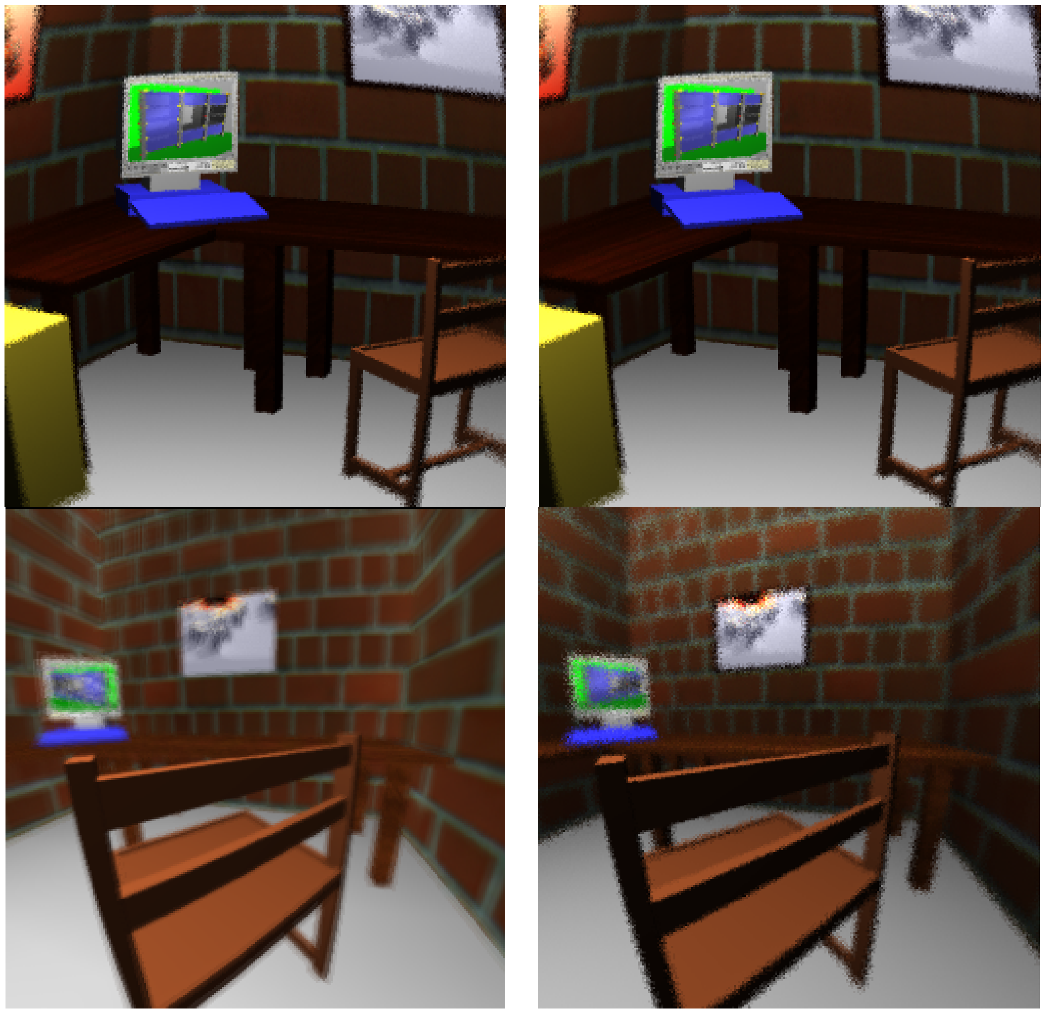

Figure 6: Two examples for lens systems: an achromatic doublet in the top row, and a biconvex lens in the bottom row. The images in the left column have been rendered using the algorithm described in this paper, while distribution ray-tracing has been used for the images on the right.

[5] Andrew Glassner. *Principles of Digital Image Synthesis*, volume 2. Morgan Kaufmann, 1995.

[6] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 43–54, August 1996.

[7] Keith D. Gremban, Charles E. Thorpe, and Takeo Kanade. Geometric camera calibration using systems of linear equations. In *1988 IEEE International Conference on Robotics and Animation*, volume 1, pages 562–567, 1988.

[8] Paul E. Haeberli and Kurt Akeley. The accumulation buffer: Hardware support for high-quality rendering. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, pages 309–318, August 1990.

[9] Craig Kolb, Don Mitchell, and Pat Hanrahan. A realistic camera model for computer graphics. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 317–324, August 1995.

[10] Marc Levoy and Pat Hanrahan. Light field rendering. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 31–42, August 1996.

[11] J. Loos, G. Greiner, H.-P. Seidel, P. Slusallek, and E. Wirsching. Advanced spectacle lens design by combining wavefront tracing and variational design. Technical report, Computer Graphics Group, University of Erlangen, 1995. Technical Report 3/1995.

[12] M. Potmesil and I. Chakravarty. A lens and aperture camera model for synthetic image generation. In *Computer Graphics (SIGGRAPH '81 Proceedings)*, pages 297–305, August 1981.

[13] M. Shinya. Post-filtering for depth of field simulation with ray distribution buffer. In *Proceedings of Graphics Interface '94*, pages 59–66. Canadian Information Processing Society, May 1994.

[14] O. N. Stavroudis. *The Optics of Rays, Wavefronts and Caustics*, volume 38 of *Pure and Applied Physics*. Academic Press, 1972.