Camera Models and Optical Systems Used in Computer Graphics: Part II, Image-Based Techniques

Brian A. Barsky[1,2,3], Daniel R. Horn[1], Stanley A. Klein[2,3]
Jeffrey A. Pang[1], and Meng Yu[1]

[1] Computer Science Division
[2] School of Optometry
[3] Bioengineering Graduate Group
University of California, Berkeley, California, 94720-1776, USA

http://www.cs.berkeley.edu/optical

Contact author: barsky@cs.berkeley.edu

Abstract. In our companion paper [5], we described the optics underlying camera models that have been used in computer graphics, and presented *object space* techniques for rendering with those models. In this paper, we survey *image space* techniques to simulate these models, and address topics including linear filtering, ray distribution buffers, light fields, and simulation techniques for interactive applications.

1 Introduction

Images generated by common computer graphics algorithms often lack depictions of focus, blur, optical aberrations, and depth of field. In our companion paper [5], published earlier in these proceedings, we described several camera models that have been proposed to address these problems, and presented object space techniques for rendering with those models. In this paper, we present several *image-based* techniques to simulate these camera models. Several of these methods are useful when the 3D geometry of a scene is not available, and they are often more efficient than *object-based* techniques such as ray tracing. We will conclude with a summary of camera models and optical systems used in computer graphics and discuss future directions for research in this area.

2 Image-Based Blur

In 1981, Potmesil and Chakravarty [16] described a linear filtering technique for image-based blur. Their algorithm blurs each pixel with a properly sized blur filter at any given depth. Each pixel is smeared into a disc of a particular radius known as the *circle of confusion* to establish blur. Equation (5) in our companion paper [5] relates the depth at the current pixel in the scene to the diameter of the resulting circle of confusion.

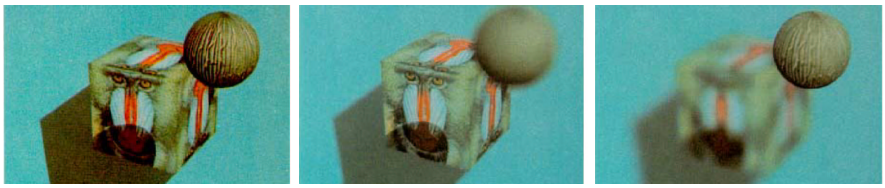

Fig. 1. Image Based Linear Filtering used on an image with a mapped cube and sphere. On the left, the cube appears focused, and on the right the sphere appears focused. (Courtesy of Indranil Chakravarty [16].)

Each point in the circle of confusion is weighted; the function that maps locations in this circle to weights is known as the *intensity distribution function* (IDF) [6, 16]. Although wave optics may be used to compute the precise IDF across the circle of confusion as a wavy Airy disc [6], according to Chen [8], a simple uniform spread of intensity across the circle of confusion can be used when the lens system is not diffraction limited or when the discretization of the pixels is too great. Smearing each pixel with the appropriate circle of confusion at its corresponding depth yields the final image.

On a Prime 750 machine from the early 1980's, a 512x512 image took between 3 and 125 minutes to process depending on the size of the aperture.

To speed the convolution process, Rokita [17] in 1993 presented an argument for approximating convolution with a large circle of confusion by using many convolutions with smaller circles of confusion. A number of faster 3x3 convolution filters can replace any single larger convolution filter. Although this does not result in an exact uniform-intensity circle of confusion, there is not a significant difference to a human observer between images that are generated by the repeated application of the smaller circle and those that are generated by the single application of the larger circle of confusion.

However, there are some deeper problems with Potmesil and Chakravarty's straightforward method of blurring. Blurring with different filters for pixels at different depths results in brightening and has artifacts arising from blurred background objects bleeding into focused foreground objects. When processing an image with a blurred white background and a sharp black foreground object, the final image will be dramatically brighter near the border of the black object as in Figure 4. This effect distorts the colors of the original image, and engenders halos around sharp objects.

The problem stems from the larger issues surrounding occlusion. A lens with a finite aperture allows a camera to view more of the scene than does a pinhole camera as shown in Figure 2, and thus it allows rays from behind an occluded object to arrive at the image plane. However, Potmesil and Chakravarty's algorithm only obtains colors that were present in the original pinhole rendering, and never determines whether a color that was originally occluded is required.

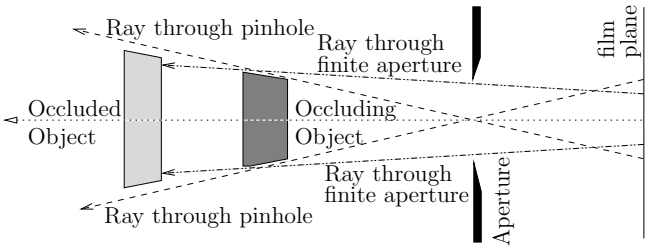

Fig. 2. An object that would be occluded in a scene ray traced through a pinhole appears through a finite aperture.

3 Ray Distribution Buffer

In [18], Shinya proposed a workaround for the inherent occlusion problem. This technique computes the final color of each pixel as an average of the colors deposited by rays entering the lens system from all angles and striking that pixel. For each pixel, the color values used in this computation are stored in a bank of memory known as the *Ray Distribution Buffer* (RDB).

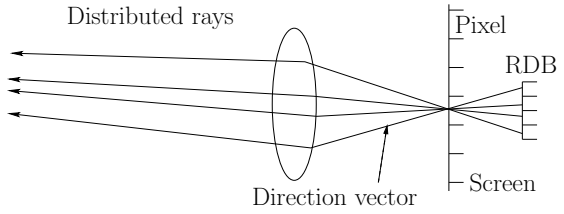

Fig. 3. Each pixel has a sub-pixel Ray Distribution Buffer. An exit direction vector can be calculated for each sub-pixel. (Courtesy of Mikio Shinya. Adapted from [18].)

Shinya's method comprises four stages: First, the image is rendered conventionally with a depth buffer.

The second stage allocates memory for the RDB for each pixel. This memory is filled with many copies of the depth and color of the corresponding pixel in the conventionally rendered image. It is desirable to have rays spreading uniformly across the lens aperture. To achieve this, a direction vector is defined for each of these rays. Each direction vector is stored in an element of the RDB. In a given pixel's RDB, the set of all such direction vectors sprinkles uniformly across the lens aperture.

In the third stage, at each pixel p of the image, the size of the circle of confusion for that pixel is calculated using equation (5) in our companion paper [5]. Then, in the initial image rendered in the first stage, each pixel q_p contained inside the pixel p's circle of confusion is examined. The direction of the ray

Fig. 4. A focused black line occludes a defocused book. Left: The standard linear filtering method results in a translucent occluding object, which is incorrect in this scene. Right: The occlusion problem is solved using the RDB, and the black line is properly opaque. (Courtesy of Mikio Shinya [18].)

emerging from each pixel q_p in the pinhole rendering is calculated; that direction is compared with the directions in the pixel p's RDB, and the corresponding memory slot is located. The depth value in the pixel p's RDB is compared with the pixel q_p's depth value and the lower depth value with its associated color is selected.

After this process is repeated for each pixel, the fourth stage of the algorithm averages the colors of the RDB into the final pixel color. This method successfully overcomes many of the problems with Potmesil and Chakravarty's simpler approach, and the difference is illustrated in Figure 4. Shinya's method requires between 155 and 213 seconds on a 150 MHz IRIS Crimson R4400 depending on the size of the circle of confusion on the majority of the image, as compared with 1321 seconds for the linear filtering method [18].

4 Blurring by Discrete Depth

Barsky et al. [2, 3] developed a method that blurs a scene with precise optics to produce a realistic final image. The method divides a scene into many slices, each slice containing pixels that share the same distance to the camera. Then it blurs each slice of the scene separately, with the appropriate level of blur. Finally the slices are combined for the final image.

The algorithm begins by separating the scene into discrete slices, according to distance from the camera. Note that both size and blur are not linear in distance, but they are approximately linear in diopters. Thus, the slices are not positioned to be equally spaced in depth but rather they are located in equal dioptric increments, ranging from the nearest depth of interest to the farthest.

The use of a scale that is linear in diopters ensures that blur will change linearly with the number of depth filters so that large distances are not covered with superfluous blur filters nor a paucity of them.

Humans can discriminate a change of approximately 1/8 diopter under optimal conditions; therefore, pixels within a slice may share a level of blur. Each slice is thus blurred with a single *blur filter*, calculated from the IDF as described in Section 2. Fast Fourier Transforms can be used in a relatively efficient method for blurring a slice.

Additionally, Barsky et al. [4] developed an alternate mechanism to obtain an IDF by measuring a physical optical system. An instrument that measures

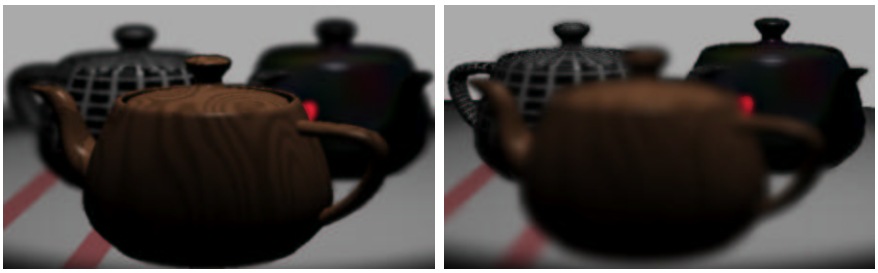

Fig. 5. Image-based blurring by discrete depths [2, 3]. In the left image the foreground is in focus with the background blurred, and in the right image the background is in focus with the foreground blurred.

the path of light directed through an optical system is used to construct an IDF for any given collection of lenses or a patient's eye.

After each slice has been blurred, they are composited. To handle discontinuities between the blur filters at different depths, linear interpolation between the two resulting images is used to eliminate sharp transitions between depth filters.

The algorithm requires 156 seconds to compute blur across 11 depth slices for the teapot image in Figure 5 on a Apple G3 600 MHz iBook, which is significantly faster than distributed ray tracing on modern hardware.

5 Light Field Techniques

Light field rendering [13] and lumigraph systems [9] are alternative general methods to represent the visual information in a scene from a collection of reference images. Both techniques were proposed in 1996, and they share a similar structural representation of rays. Unlike model-based rendering, which uses the geometry and surface characteristics of objects to construct images, light field rendering, as an example of image-based rendering techniques, resamples an array of the reference images taken from different positions and generates new views of the scene. These images may be rendered or captured from real scenes.

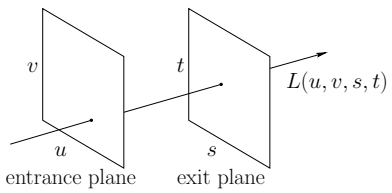

Fig. 6. 4D light field point. (Courtesy of Marc S. Levoy. Adapted from [13] © 1996 ACM, Inc. Reprinted by permission.)

In 1995, McMillan and Bishop [15] proposed *plenoptic modeling*, based on the *plenoptic function*, which was introduced by Adelson and Bergen [1]. A plenoptic function is a five-dimensional function that describes the entire view from a given point in space. The five dimensions comprise three parameters indicating the position of the point, as well as a vertical and horizontal angle indicating the direction of the ray from the particular point. Originally, the function also contained wavelength and time variables. Treating time as constant and light as monochromatic, the modeling ignores these two parameters. A *light field*, described by Levoy and Hanrahan [13], is a 4D reduced version of the plenoptic function. Light field rendering assumes the radiance does not change along a line. Lines are parameterized by two coordinates of intersections of the lines on each of two parallel planes in arbitrary positions: the entrance plane and the exit plane. Figure 6 illustrates this parameterization. The system takes as input an array of 2D images and constructs a 4D light field representation. It regards the camera plane and the focal plane as the entrance and exit planes of the two-plane parameterization, respectively. Each image is taken from a particular point on the camera plane and stored in a 2D array (Figure 7). Thus, the light field is the composition of the slices of the 2D images. Once the light field is generated, the system can render a new image by projecting the light field onto the image plane, given a camera position and its orientation.

One of the major challenges of this technique is to sample the light field. Since the light field is discrete, only rays from the reference images are available. When resampling the light field to form a new image, we must approximate rays that are absent from the input images. This approximation may introduce artifacts. Different methods have been suggested for interpolating rays to address the problem. There is also some research focusing on computing the minimum sampling rate of the light field required to reduce the artifacts, such as Lin et al. [14] and Chai et al. [7]. Levoy et al.'s approach to interpolate a ray combines the rays that are close to the desired ray using a 4D low pass filter. However, this integration of rays is based on the assumption that the entire scene lies on a single and fixed exit plane. Therefore, a scene that has a large range of depth will appear blurred.

The lumigraph system, which was introduced by Gortler et al. [9], proposed a different approach to reduce the artifacts arising from the sampling problem. In the algorithm, the light field is alternatively called the lumigraph. The system first approximates the depth information of the scene. Next, instead of using a single exit plane as is the case in Levoy et al. [13], the lumigraph constructs a set of exit planes at the intersections of the rays and the objects, and then projects the rays to the corresponding planes. A basis function based on the depth information is associated with each 4D grid point in the light field. The color value of the desired ray is a linear combination of the color values of the nearby 4D points weighted by the corresponding basis functions. The lumigraph system reduces the artifacts by applying depth information; however, such information can be difficult to acquire from images captured from real scenes.

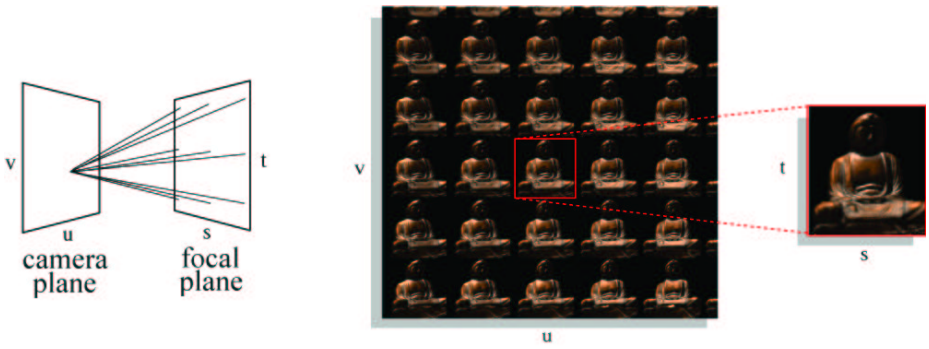

Fig. 7. The visualization of a light field. Each image in the array represents the rays arriving at one point on the camera plane from all points on the focal plane, as shown at left. (Courtesy of Marc S. Levoy. Adapted from [13] © 1996 ACM, Inc. Reprinted by permission.)

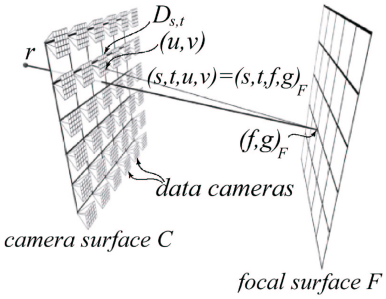

Fig. 8. Isaksen's parameterization uses a camera surface C, a collection of data cameras $D_{s,t}$ and a dynamic focal surface F. Each ray (s, t, u, v) intersects the focal surface F at $(f, g)_F$ and is therefore named $(s, t, u, v)_F$. (Courtesy of Aaron Isaksen. Adapted from [11].)

In 2000, Isaksen [11, 12] developed the dynamically reparameterized light field. This technique extended the light field and lumigraph rendering methods to enable a variety of aperture sizes and focus settings without requiring geometric information of the scene. The system associates each point on the camera plane with a small camera, called a *data camera* (Figure 8). A image is taken from each data camera at the corresponding point. To reconstruct a ray in the light field, the system maps a point on the focal surface onto several data cameras. This results in a ray for each selected data camera for the corresponding point on the focal surface. A mapping function is defined to decide through which data cameras the rays pass. Then the system applies a filter to combine the values of the rays from all data cameras. To vary the aperture sizes, the system changes the shape of the aperture filter. This changes the number of data cameras through which the rays pass. The aperture filter is also used to apply depth information.

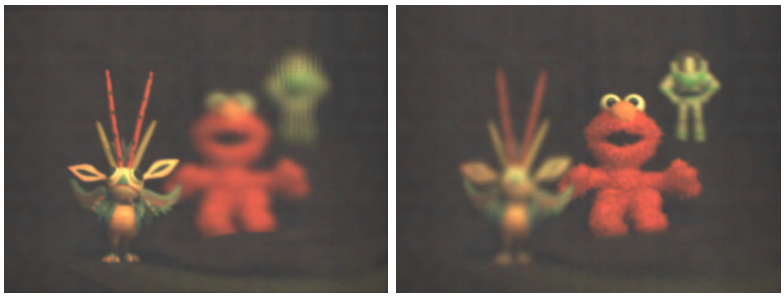

Fig. 9. By changing the shape of the focal surface, the same image can be rendered with varying parts in focus. (Courtesy of Aaron Isaksen [11].)

The rays from several data cameras pass through the aperture and intersect on the focal surface. If the intersection is near the surface of the virtual object, the ray will appear in focus; otherwise, it will not. On the other hand, the range of the depth of field is limited by the number of data cameras on the camera surface. A deeper scene requires more data cameras on the camera surface. Similar to varying aperture sizes, the system changes the focal surface dynamically. The sampling results in the same view of the scene with different focus, depending on which focal surface is chosen. Figure 9 shows the results of the same image, rendered with different focal surfaces. The light field input images are captured by an Electrim EDC1000E CCD camera. Then, the new images are rendered by PC rasterizers for Direct3D 7. The real-time renderer allows users to change the location of the camera and the focal plane, and the aperture size in real time.

An important advantage of light field rendering techniques is the capability of generating images without a geometric representation. Instead, the geometric information is implicitly obtained from a series of reference images, taken from the surrounding environment. Furthermore, rather than simply mapping rays to pixel colors, these techniques map rays to rays and apply a prefiltering process, which integrates over collections of nearby rays.

6 Realistic Lens Modeling in Interactive Computer Graphics

Heidrich et al. [10] introduced an image-based lens model, which represents rays as a light field and maps slices of the scene light field into corresponding slices of the camera light field, instead of using ray tracing. This technique simulates the aberration of a real lens and achieves hardware acceleration at the same time. The 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. Due to the aberration of the lens, rays from a grid on the film do not intersect in a single point on the image-sided surface of the lens system. In this case, the mapping cannot be described as a single perspective projection. A virtual center of projection (COP) is approximated for each grid on the film. The system

calculates the angle between the original rays and those generated by the perspective transformation through the corresponding calculated virtual COP. If the angle is small enough, the system continues the approximation on the next grid; otherwise, the grid is subdivided and the system calculates a virtual center of projection and repeats the same process for each subgrid. Eventually, all slices of the camera light field, are combined with weights relative to the radiometry of the incoming rays to form the final image. In addition to realistic modeling of the lens, another important contribution of this technique is its application in interactive computer graphics. Light fields, which are composed of 2D images, can be easily generated by computer graphics hardware at high speeds. Once the light field is sampled, a final image is ready to be rendered. Heidrich's experiments generate images on a RealityEngine2 using 10 sample points on the lens with approximately 14 frames per second. The grid size on the film plane is 10 x 10 polygons.

7 Summary and Future Directions

This paper covered the theory underlying lens systems that add realistic vision and camera effects to a scene. There are both object space and image space techniques, and they have different tradeoffs with respect to blur quality, speed, and realism. Future directions of research may involve extending Potmesil and Chakravarty's technique to improve object identification and occlusion detection when adding blur to a prerendered image. Likewise, modifying Heidrich et al.'s technique for rendering a lens with arbitrary geometry would result in a significantly faster technique for rendering fish-eye distortions and other distortions that cannot be approximated with a thick lens. Other directions of future research could include automatically determining how many rays or lens sample points are necessary for a given image quality in the Kolb et al. and Heidrich et al. techniques, respectively.

Acknowledgments

The authors would like to thank Adam W. Bargteil, Billy P. Chen, Shlomo (Steven) J. Gortler, Aaron Isaksen, and Marc S. Levoy for their useful discussions as well as Mikio Shinya and Wolfgang Heidrich for providing figures.

References

1. E.H. Adelson and J.R. Bergen. *Computational Models of Visual Processing.* The MIT Press, Cambridge, Mass., 1991.
2. Brian A. Barsky, Adam W. Bargteil, Daniel D. Garcia, and Stanley A. Klein. Introducing vision-realistic rendering. In Paul Debevec and Simon Gibson, editors, *Eurographics Rendering Workshop*, pages 26–28, Pisa, June 2002.
3. Brian A. Barsky, Adam W. Bargteil, and Stanley A. Klein. Vision realistic rendering. Submitted for publication, 2003.

4. Brian A. Barsky, Billy P. Chen, Alexander C. Berg, Maxence Moutet, Daniel D. Garcia, and Stanley A. Klein. Incorporating camera models, ocular models, and actual patient eye data for photo-realistic and vision-realistic rendering. Submitted for publication, 2003.
5. Brian A. Barsky, Daniel R. Horn, Stanley A. Klein, Jeffrey A. Pang, and Meng Yu. Camera models and optical systems used in computer graphics: Part I, Object based techniques. In *Proceedings of the 2003 International Conference on Computational Science and its Applications (ICCSA'03)*, Montréal, May 18–21 2003. Second International Workshop on Computer Graphics and Geometric Modeling (CGGM'2003), Springer-Verlag Lecture Notes in Computer Science (LNCS), Berlin/Heidelberg. (These proceedings).
6. Max Born and Emil Wolf. *Principles of Optics*. Cambridge University Press, Cambridge, 7th edition, 1980.
7. Jin-Xiang Chai, Xin Tong, and Shing-Chow Chan. Plenoptic sampling. In Kurt Akeley, editor, *Proceedings of ACM SIGGRAPH 2000*, pages 307–318, New Orleans, July 23–28 2000.
8. Yong C. Chen. Lens effect on synthetic image generation based on light particle theory. *The Visual Computer*, 3(3):125–136, October 1987.
9. Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In Holly Rushmeier, editor, *ACM SIGGRAPH 1996 Conference Proceedings*, pages 43–54, New Orleans, August 4–9 1996.
10. Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. An image-based model for realistic lens systems in interactive computer graphics. In Wayne A. Davis, Marilyn Mantei, and R. Victor Klassen, editors, *Proceedings of Graphics Interface 1997*, pages 68–75. Canadian Human Computer Communication Society, May 1997.
11. Aaron Isaksen. Dynamically reparameterized light fields. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass., November 2000.
12. Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In Kurt Akeley, editor, *Proceedings of ACM SIGGRAPH 2000*, pages 297–306, New Orleans, July 23–28 2000.
13. Marc Levoy and Pat Hanrahan. Light field rendering. In Holly Rushmeier, editor, *ACM SIGGRAPH 1996 Conference Proceedings*, pages 31–42, New Orleans, August 4–9 1996.
14. Zhouchen Lin and Heung-Yeung Shum. On the numbers of samples needed in light field rendering with constant-depth assumption. In *Computer Vision and Pattern Recognition 2000 Conference Proceedings*, pages 588–579, Hilton Head Island, South Carolina, June 13–15 2000.
15. Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *ACM SIGGRAPH 1995 Conference Proceedings*, pages 39–46, Los Angeles, August 6–11 1995.
16. Michael Potmesil and Indranil Chakravarty. Synthetic image generation with a lens and aperture camera model. *ACM Transactions on Graphics*, 1(2):85–108, April 1982. (Original version in ACM SIGGRAPH 1981 Conference Proceedings, Aug. 1981, pp. 297-305).
17. Przemyslaw Rokita. Fast generation of depth-of-field effects in computer graphics. *Computers & Graphics*, 17(5):593–595, September 1993.
18. Mikio Shinya. Post-filtering for depth of field simulation with ray distribution buffer. In *Proceedings of Graphics Interface '94*, pages 59–66, Banff, Alberta, May 1994. Canadian Information Processing Society.