An Algorithm for Rendering Generalized Depth of Field Effects Based on Simulated Heat Diffusion

Todd Jerome Kosloff
Brian A. Barsky

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-19

http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-19.html

January 24, 2007

An Algorithm for Rendering Generalized Depth of Field Effects Based on Simulated Heat Diffusion

Todd J. Kosloff*
University of California, Berkeley
Computer Science Division
Berkeley, CA 94720-1776

Brian A. Barsky†
University of California, Berkeley
Computer Science Division and School of Optometry
Berkeley, CA 94720-1776

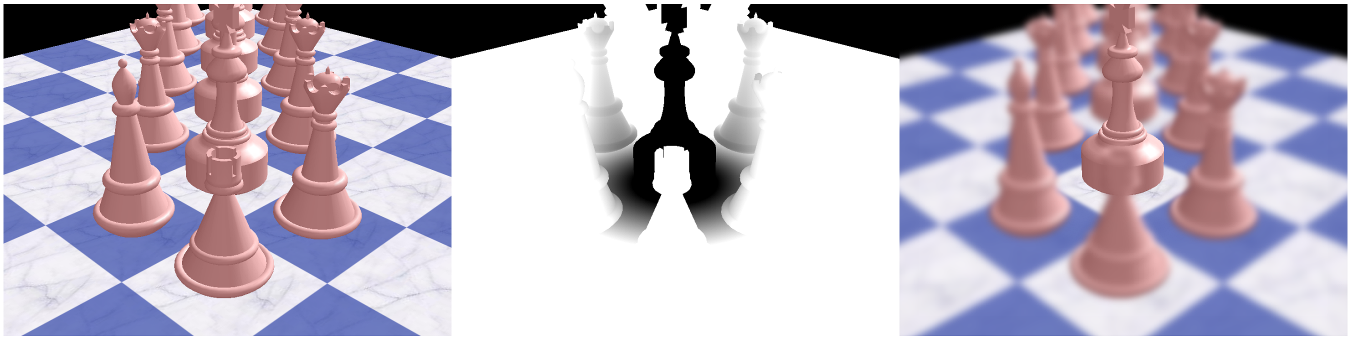

Figure 1: Left: An unblurred chess scene rendered with a pinhole camera model. Middle: A cylindrical blur field. Right: Application of the blur field to the chess scene.

Abstract

Depth of field is the swath through a 3D scene that is imaged in acceptable focus through an optics system, such as a camera lens. Control over depth of field is an important artistic tool that can be used to emphasize the subject of a photograph. In a real camera, the control over depth of field is limited by the laws of physics and by physical constraints. The depth of field effect has been simulated in computer graphics, but with the same limited control as found in real camera lenses. In this report, we use anisotropic diffusion to generalize depth of field in computer graphics by allowing the user to independently specify the degree of blur at each point in three-dimensional space. Generalized depth of field provides a novel tool to emphasize an area of interest within a 3D scene, to pick objects out of a crowd, and to render a busy, complex picture more understandable by focusing only on relevant details that may be scattered throughout the scene. Our algorithm operates by blurring a sequence of nonplanar layers that form the scene. Choosing a suitable blur algorithm for the layers is critical; thus, we develop appropriate blur semantics such that the blur algorithm will properly generalize depth of field. We found that anisotropic diffusion is the process that best suits these semantics.

1 Introduction

Control over what is in focus and what is not in focus in an image is an important artistic tool. The range of depth in a 3D scene that is imaged in sufficient focus through an optics system, such as a camera lens, is called *depth of field* [Erickson and Romano 1999][London et al. 2002][Stroebel et al. 2000]. This forms a swath through a 3D scene that is bounded by two parallel planes which are parallel to the film/image plane of the camera, except in the case of the view camera [Barsky and Pasztor 2004][Merklinger 1993][Merklinger 1996][Simmons 1992][Stone 2004][Stroebel 1999].

In a real camera, depth of field is controlled by three attributes: the distance at which the camera lens is focused, the f/stop of the lens, and the focal length of the lens. Professional photographers or cinematographers often manipulate these adjustments to achieve desired effects in the image. For example, by restricting only part of a scene to be in focus, the viewer or the audience automatically attends primarily to that portion of the scene. Analogously, *pulling focus* in a movie directs the viewer to look at different places in the scene, following along as the point of focus moves continuously within the scene.

Creating computer generated imagery can be regarded as simulating the photographic process within a virtual environment. Rendering algorithms in computer graphics that lack depth of field are in fact modeling a pinhole camera model. Without depth of field, everything appears in completely sharp focus, leading to an unnatural, overly crisp appearance. Depth of field effects were first introduced into computer graphics as a means for increasing the realism of computer generated images. Just as in photography, depth of field is also used in computer graphics for controlling what portion of a scene is to be emphasized.

Real depth of field is discussed in Section 3.3. There are significant limitations on what can be achieved by adjusting the focus distance, f/stop, and focal length. It is evident that any given camera lens will have physical constraints that will limit the range of the adjustments. However, even if that were not the case, and we had at our disposal a magical lens that was unlimited in terms of these attributes, there are still many interesting and useful depth of field

*e-mail:koslofto@cs.berkeley.edu
†e-mail:barsky@cs.berkeley.edu

effects that could be imagined, but not realized by any combination of these adjustments, due to the laws of physics.

When we allow our thinking to extend to what is desirable, rather than what we have available, we can imagine useful depth of field effects that are not possible with conventional optics. Consider, for example, a scene consisting of a crowd of people. We want to draw attention to a person towards the front, and another person towards the rear. We have to choose between focusing on the near person or the far person. If a sufficiently small aperture is available, we could perhaps get the entire crowd in focus, but this is contrary to the goal of picking out just the two people we have in mind. Instead, we would like to focus on the near person and the far person, while leaving the middle region out of focus. This cannot be done with a lens. Going even farther with this idea, consider a single row of people, arranged from left to right across the field of view. Since they are all at the same distance from the camera, they will all either be in focus, or all out of focus. There is no way to vary depth of field to focus in on any one of these people more than the others.

In the case of computer graphics, we are still faced with precisely the same limitations as in real photography, because existing depth of field simulation algorithms were intentionally designed to precisely emulate real-world optics. We alleviate these limitations with the first implementation of an effect that we call *generalized depth of field*. In our system, focus is controlled not by the focus distance, f/stop, and focal length settings, but instead by a three-dimensional scalar blur field imposed on a scene. This enables the user to specify the amount of blur independently at every point in space.

Such a system overcomes every limitation on what can be achieved with focus. We can control the amount of depth of field independently from the amount of blur away from the in-focus region. Any number of depths could be made to be in focus, while regions in between those depths will appear blurred. Focus can be made to vary from left to right, rather than from front to back. Even more exotic possibilities are just as easy with our system, such as contorting the in-focus region into the shape of a sphere, or any other geometric shape.

Our algorithm proceeds by first rendering the scene into a set of layers, which do not need to be planar. The 3D world coordinates of each pixel are stored in a position map associated with the layers. The position map is used to connect the user-specified 3D blur field to the layers. As is explained in Section 7.2 , blur values are also required that lie outside the object that is present within a layer. These blur values are extrapolated from the known values obtained through the aforementioned mapping. Having associated blur values with each pixel in the layers and having extrapolated those values, we use the blur values associated with each layer to blur the layers. Finally, the layers are composited from front to back, with alpha blending [Porter and Duff 1984].

One difficulty is that naive spatially varying blur [1] produces artifacts in the form of holes appearing in the middle of objects. To solve this required a careful analysis of the true meaning of the blur field. This meaning, which we refer to as blur semantics, reflects our analysis of what we consider to be the essential characteristics of realistic depth of field that we wish to maintain, despite our goal of creating effects that cannot occur in the real world. The key insight is that the spatially variant blur operation that will satisfy our blur semantics is *anisotropic diffusion*. We therefore blur our layers using a form of repeated convolution that behaves similarly in principle to anisotropic diffusion.

2 Motivation for Generalized Depth of Field

2.1 Limitations of Real Depth of Field

At first glance, depth of field may seem like a problem. Less experienced photographers often view the blur in an image as something to be avoided, and consequently try to adjust the focus distance, f/stop, and focal length to have as much of the scene in focus as possible. From this perspective, simulating depth of field in computer generated images enables creating more realistic images by rendering them with imperfections, and thus more like real world photographs.

More positively, depth of field can instead be used deliberately as an artistic tool. If a subject is in front of a busy background, it is desirable to intentionally blur the background so that it will be less distracting and to cause the viewer to attend to the person and not the background.

When using depth of field as a useful tool for selecting only the relevant portions of a scene, there are limitations. If two people are next to each another, it is not possible to focus on just one or the other; both people are at the same distance from the camera, and hence both will be either in focus or out of focus. Alternatively, consider a crowd of many people. Perhaps we would like to use focus to highlight a few people scattered at a few locations throughout the crowd. There is clearly no combination of focus distance, f/stop, and focal length that can achieve this.

When simulating depth of field for computer generated images, the goal has generally been to faithfully replicate the behavior of real cameras, complete with these limitations. We, on the other hand, observe that there is no reason why we must accept these limitations in computer graphics. Rather than control the focus distance, f/stop, and focal length, we allow depth of field to be controlled by a 3D blur field that specifies how much blur should be applied to every point in 3D space.

2.2 Partial Occlusion

Since a lens has a finite aperture, that which is imaged at a any given point on the film/image plane is the aggregate of the light emerging from every point on the lens. Thus, the lens can be considered as viewing the scene from many different points of view, simultaneously. Specifically, light rays impinging at the center of the lens may be emanating from foreground objects that occlude background objects whereas other rays arriving at a point on the lens far from its center may bypass the occluder. Thus, the background is visible from some points on the lens, and not from others. Therefore, there are single points on the image where a given object is *partially occluded*. This can be seen when objects that are out-of-focus have soft, semi-transparent edges. Partial occlusion is a vital aspect in the appearance of depth of field.

A computer simulation of depth of field that lacks partial occlusion does not faithfully reproduce the appearance of real photographs. Consider, for example, a small object that is close to the camera with a large aperture. Partial occlusion would be very noticeable because it would apply to the entire object and not just to the edge. In such cases, the lack of partial occlusion is completely unacceptable.

[1] a straightforward convolution with a spatially varying kernel radius

3 Background

3.1 Basic Lens Parameters

The focal length of a lens, denoted by f, is the distance the lens must be from the film/image plane to focus incoming parallel rays to a point on that plane. The aperture is the physical opening of the lens, which determines the quantity of light that will enter the optical system. We denoted the aperture diameter by a_{diam}. The f/stop (or F number) is the ratio of the focal length to the diameter of the aperture of the lens, $\frac{f}{a_{diam}}$. Note that the aperture diameter is not completely specified by the f/stop. Indeed, when changing the focal length on a zoom lens, the f/stop varies in many amateur lenses, but the aperture varies in in more professional zoom lenses.

3.2 Circle of Confusion

Blur arises when a point in the 3D scene is not imaged as a point, but as a disk, which is called the *circle of confusion*. The amount of blur for a given depth can be described by the diameter of the circle of confusion for that depth.

Consider a point at a distance d_{focus} in front of the lens that we wish to be in focus. Using the thin lens approximation [Jenkins and White 1976], the distance d'_{focus} behind the lens where the film/image plane would have to be located can be derived [Barsky et al. 2003a]:

$$d'_{focus} = \frac{f * d_{focus}}{d_{focus} - f} \qquad (1)$$

where f is focal length.

A point that is not at the the distance d_{focus} in front of the lens will thus not be imaged as a point at the distance d'_{focus} behind the lens where the film/image plane is, but instead would form a point at the distance d_{image} behind the lens. Thus, on the film/image plane, it would be imaged as a circle of confusion having a diameter denoted by c_{diam}. Using similar triangles, this can be calculated [Barsky et al. 2003a]:

$$c_{diam} = a_{diam} * \frac{d_{image} - d'_{focus}}{d_{image}} \qquad (2)$$

where a_{diam} is the aperture diameter. From this, we see that the circle of confusion diameter is intrinsically independent of focal length, although focal length would enter into the equation if it were recast in terms of f/stop rather than aperture diameter.

3.3 Real Depth of Field

In an optical system such as a camera lens, there is plane in the 3D scene, located at the focus distance, that is rendered at optimal sharpness. There is a swath of volume through the scene that is rendered in reasonable sharpness, within a permissible circle of confusion, to be exact. This region of acceptable focus is delineated by near and far planes. However, this region is not centered about the focus distance; rather, the near plane is closer to the plane of perfect focus than is the far plane.

The particular distance that is imaged in perfect focus can be selected by moving the lens towards or away from the film/image plane. Changing the focus distance will have a concomitant effect on the amount of depth of field, that is, the size of the swath. Specifically, for a given f/stop and focal length, focusing at a distance that is close to the camera provides only a narrow range of depths being in focus, with the amount of depth of field increasing in a nonlinear fashion as the focus distance is increased, and conversely.

The size of the aperture also affects the amount of depth of field. The infinitesimal aperture of a pinhole camera has infinite depth of field, and this decreases as the aperture increases, and conversely. An important subtlety overlooked by most photographers is that for a fixed focus distance, the amount of depth of field is completely determined by the aperture. The confusion arises because camera lenses do not afford direct control over the aperture, but instead provide f/stop adjustment. Consequently, for a fixed f/stop, changing the focal length implies changing the aperture size which affects the amount of depth of field. Of course, for a fixed focal length lens, this distinction does not matter.

Thus, for a fixed f/stop and fixed focus distance, changing the focal length of the lens (zooming in or out) will again affect the depth of field, since the aperture changes. In particular, increasing the focal length (zooming in) with a constant f/stop decreases the depth of field, because the aperture increases, and conversely.

Less widely discussed is the behavior of blur outside the region of acceptable focus. Objects that are at an increasing distance from the focus distance become increasingly blurred at a rate related to the aperture size. For very large aperture size, not only is there a very narrow range of depths that are imaged in focus, but the out-of-focus areas have an extreme amount of blur. This arises frequently in the case of macro photography where the subject is very close to the camera. Although these relationships are well understood by experienced photographers, they are not obvious to novices. Furthermore, no amount of experience in photography can enable an escape from the limitations dictated by the laws of optics.

3.4 Point Spread Function

A Point Spread Function (PSF) plots the distribution of light energy on the image plane based on light that has emanated from a point source and has passed through an optical system. Thus it can be used as an image space convolution kernel.

4 Related Work

The first simulation of depth of field in computer graphics was developed by Potmesil and Chakravarty who used a postprocessing approach where a single sharp image is blurred using a spatially variant convolution. Postprocessing a single image can lead to occlusion artifacts. One such artifact occurs when background pixels are spread onto foreground pixels. This should not happen; the foreground should occlude the background. Shinya solved this with a ray distribution buffer, or RDB. This approach, rather than blindly averaging adjacent pixels, stores pixels in a z-buffered RDB as they are being averaged. The z-buffer in the RDB ensures that near pixels will occlude far pixels during the averaging process. Rokita [Rokita 1996] achieved depth of field at rates suitable for virtual reality applications by repeated convolution with 3×3 filters and also provided a survey of depth of field techniques [Rokita 1996].

Cook introduced distributed raytracing [Cook et al. 1984], the first multisampling approach to depth of field. Distributed ray tracing solves many problems, including soft shadows and glossy reflections, as well as depth of field. Several rays are traced for each pixel to be rendered. Each ray originates from a different point on the aperture, and the rays are oriented such that they intersect at the plane of sharp focus.

Dippe and Wold analyzed the stochastic sampling of distributed ray tracing [Dippe and Wold 1985], and concluded that random distributions of rays are superior to uniform distributions. They suggest the Poisson distribution as being a suitable random distribution. Lee, Redner and Uselton applied statistical analysis and stratified sampling to determine how many rays need to be trace for a given pixel to adequately capture the variation [Lee et al. 1985].

Haeberli and Akeley introduced the accumulation buffer [Haeberli and Akeley 1990], which is now present on all modern graphics hardware. The accumulation buffer allows several renders of a single scene to be accumulated, or averaged. The various effects that distributed raytracing achieves can be achieved with the accumulation buffer. In particular, depth of field is attained by rendering and averaging several images with the pinhole model. For each image, the camera is moved to a different point on the aperture, and the view frustum is skewed such that the depth that is in focus remains constant.

The depth of field algorithms discussed above represent a generic, simplified camera. Real world camera lenses suffer from distortion and the precise image varies according to the precise shape of the individual lens. Kolb et al. performed distributed ray tracing through detailed lens models corresponding to actual lenses [Kolb et al. 1995]. The resulting images exhibit the appropriate distortion and blur inherent in these lenses.

Heidrich et al. calculated the light field between the lens and film/image plane as a transformation of the scene light field [Heidrich et al. 1997]. They then can sample various slices of the light field to generate views of the scene from different points on the aperture, and average those to get depth of field effects. This technique has an important advantage over most others, in that it incorporates distortions caused by the lens, rather than only capturing blur.

Kosara, Miksch, and Hauser were the first to suggest that by departing from the physical laws governing depth of field, blur can become a much more useful tool for indicating the relevant parts of a scene [Kosara et al. 2001]. They call their system semantic depth of field, indicating that blur is a meaningful design element, rather than merely an artifact of the way lenses work. Their system operates in a similar manner to Scofield's. That is, objects are rendered into buffers, the buffers are blurred according to the relevance of the objects, then the buffers are composited. This approach, while fast and simple, operates at an object-level granularity. There is no way to blur only half of an object, nor can blur vary smoothly across the surface of an object. Thus it is impossible to have a window of focus smoothly move across a scene. We, on the other hand, support a fully arbitrary blur field, where every point in 3D space can be blurred or not blurred as desired.

Ng devised an actual camera that captures the four-dimensional light field between the lens and film/image plane [Ng 2005]. An array of microlenses distributes the light that would ordinarily fall on a single pixel from all directions instead onto a collection of directional samples that when summed together yield the original pixel. Ng's light field exposures can be processed after the fact to obtain photographs whose aperture and focus distance can be anything within a range, rather than the single aperture and focus distance that must be chosen before taking a photograph with an ordinary camera.

In Synthetic Aperture Confocal Imaging, a camera photographs a scene through a collection of carefully spaced mirrors [Levoy et al. 2004]. The images from each mirror can be considered as a sample from a large synthetic aperture. Due to the large aperture, objects that are off the focal plane are extremely blurred and quite transparent. The principle of confocal microscopy can be extended to non-microscopic objects by combining this approach with similarly focused light from a synthetic aperture projector. That is, specific points in 3D space can be imaged, and occluding objects are made both dark and blurred. This enables only the part of the scene that is of interest to be seen, even if it is partially occluded.

The Stanford camera array is a wall of cameras that operate in parallel [Wilburn et al. 2005]. The array outputs a collection of photographs that vary slightly in their viewpoints. Wilburn et al. present a variety of uses for this array, one of which relates to depth of field, as follows: Averaging the pictures coming from each camera in the array simulates one single camera with a very large aperture. This large synthetic aperture results in extremely shallow depth of field. With such a large aperture, partial occlusion becomes very interesting. Foreground objects are so severely out of focus that they practically vanish, allowing occluded background objects to become visible.

Depth of field is relevant in all optical systems, not only camera lenses. The human eye also has a finite aperture (the pupil) and can focus at different depths by means of accommodation of the internal crystalline lens. The optical characteristics of human eyes are slightly different for each person, resulting in different point spread functions from person to person. Barsky introduced vision realistic rendering to the computer graphics community [Barsky 2004], which uses actual data scanned from human eyes using a wavefront aberrometer. The aberrometer samples a wavefront, which is interpolated using the Zernike polynomial basis. The wavefront is then converted into a set of point spread functions, which are then used to blur different parts of an image, using a postprocessing method.

This postprocessing method involves splitting the scene into layers where each layer has a constant depth. Each layer can then be blurred in the frequency domain using an FFT. The blurred layers are then composited. This method faces a challenge; if care is not taken, artifacts in the form of black bands can appear where one layer meets another layer. This is particularly a problem for a single object that spans a range of depths, such as a table viewed in perspective. Various solutions to this problem are discussed in detail by Barsky et al. [Barsky et al. 2003c][Barsky et al. 2005].

Isaksen developed a new parameterization for light fields that enables controllable depth of field [Isaksen et al. 2000]. It is especially relevant here that their parameterization includes a focal surface that does not necessarily lie on a plane. Therefore, they can control focus in ways beyond what a real camera can produce. They do not completely generalize depth of field, however; focal surfaces are merely a subset of what can be achieved with our arbitrary blur fields.

Krivanek developed a very fast method for rendering point cloud models with depth of field via splatting [Krivanek et al. 2003]. This elegant method simply replaces each splat footprint with a Gaussian blob whose standard deviation increases with circle of confusion. Rendering these large splats does not lead to a slowdown, as when the splats are large, it is possible to render using a simplified version of the model comprising fewer points without an apparent loss in quality.

Perona and Malik used anisotropic diffusion to blur images in a nonuniform fashion [Perona and Malik 1994][Perona and Malik 1988]. They control the anisotropy in such a way that details are blurred away, leaving only the overall shapes of the objects in an image. Their purpose in doing so is for performing edge detection. We also use a form of anisotropic diffusion for blurring images in a nonuniform fashion, but we control the anisotropy via nonplanar slices of the user-specified 3D blur field.

We are aware of one prior instance of anisotropic diffusion being used to postprocess images to yield depth of field [Bertalmio et al. 2004]. Bertalmio et al. used anisotropy to alleviate intensity leakage, that is, the occlusion problem that can alternatively be addressed by the ray distribution buffer. They did not, however, address the more difficult partial occlusion problem that we address.

Scofield presents a fast and simple depth of field algorithm that, like ours, postprocesses layers, then composites the layers from back to front [Scofield 1994]. His method, like ours, uses layers to solve the partial occlusion problem. Scofield's technique assumes that each layer lies on a plane parallel the film/image plane. That is, every pixel on a given layer shares the same depth. For a physically realistic camera model, blur varies only with depth. Consequently, blur is uniform within a given layer even though it is nonuniform within a scene. When blur does not change from point to point, blur becomes a straightforward convolution. Convolution in the spatial domain is equivalent to multiplication in the frequency domain, so a Fourier transform is useful for performing convolution. Thus, the blurring can be performed by an FFT, resulting in a very fast technique. Our technique differs from Scofield's insofar as our layers are nonplanar, which leads to spatially varying blur even within a single one of our layers.

Whereas cameras with the film/image plane parallel to the lens plane follow the conventional rules for predicting circle of confusion, view cameras do not follow these rules [Shaman 1978][Stroebel 1999][Simmons 1992][Stone 2004][Merklinger 1993][Merklinger 1996]. A view camera can be adjusted via tilts and swings to have its film/image and lens planes at arbitrary angles relative to one another. This results in both a change in the perspective projection and a change in depth of field. The perspective projection can be tilted, for example, to make the top of a tall building not appear smaller than the bottom, when photographed from the ground, even though the top is much farther away than the bottom. More relevant to us is that a view camera enables the plane of sharp focus to be oriented generally such that it does not have to be parallel to the film/image plane. Thus, the plane that is in sharp focus might be an entire kitchen floor, even though the floor spans a great range of depths.

Barsky and Pasztor developed methods for simulating the view camera camera model for computer generated images [Barsky and Pasztor 2004]. One approach used distributed ray tracing and the other used the accumulation buffer.

For a more thorough survey of existing depth of field techniques, we refer the reader to a pair of surveys by Barsky et al. where the techniques have been separated into object space [2003a] and image space [2003b] techniques.

Su, Durand, and Agrawala presented a method for subtly directing the viewers' attention within an image [Su et al. 2005]. Their input is a region that is to be de-emphasized. This region then has its texture deemphasized via a novel image processing operation.

Chastine and Brooks have a 3D visualization of molecules, and wish to emphasize some atoms over others, to clarify a cluttered image [Chastine and Brooks 2005]. Their technique uses what they call a blur shader, which modulates transparency such that de-emphasized atoms remain relatively opaque in the center while fading to complete transparency towards the edges. Interestingly, although there is no actual convolution or blurring of any kind, the appearance of the atoms does qualitatively resemble being blurred. Their goal of drawing attention to specific parts of a scene, and their approach of making some parts of the scene appear blurred are reminiscent of some of our discussions in the present paper.

Interactive digital photomontage uses graph cut segmentation and gradient domain fusing to seamlessly cut and paste bits and pieces of various photographs to form new images [Agarwala et al. 2004]. The input photographs are generally of the same scene, but with some variable changing. One variable that can be changed is focus. Agarwala et al. use this to combine photographs of a single object taken with various focus settings into a single image where everything is in focus.

None of these depth of field algorithms provides a technique that can be modified to achieve our goal of arbitrary blur fields. Physically oriented methods based on raytracing or accumulating samples are too closely tied to the laws of optics to be coerced into producing the kind of effects we want. Although distributed ray tracing can be modified to allow blur to vary arbitrarily as a function of depth, this requires curved rays, i.e., nonlinear raytracing (mention some nonlinear raytracing references here). Nonlinear raytracing is extremely computationally demanding, and this exigency is only compounded by the need for many rays per pixel. Furthermore, nonlinear raytracing does not handle blur fields that vary along the x or y directions. Simply changing the ray profile per pixel does indeed change the amount of blur from pixel to pixel, but it violates the blur semantics we require. Blur semantics are discussed later in this paper, in Section 5.

The postprocessing methods inherently blur some regions more than others, in accordance with depth. Therefore, it would seem promising to modify a postprocessing algorithm to simply blur according to the blur field, rather than strictly adhering to the circle of confusion size as dictated by optics. Postprocessing methods are flawed by the fact that the blurring of foreground objects causes parts of background objects that should be occluded to become partially visible. For small amounts of blur, this is not too important; however, the entire object can become transparent for large amounts of blur. This could occur when a small object is close to the camera and the aperture is large. In the case of large amounts of blur, the typical postprocessing method lacks sufficient information to avoid erroneously rendering parts of occluded objects as partially visible. Furthermore, no existing postprocessing method can ensure the blur semantics that we require. Although our method is indeed a postprocessing method, we differ from prior work both in the nature of the intermediate representation that is postprocessed as well as in the 2D blur algorithm that operates on the intermediate representation.

Krivanek's point splatting algorithm is only applicable to point cloud models. Furthermore, it cannot be modified to support arbitrary blur fields because quickly varying blur values can lead to cracks opening up at the seam. Our method, being a postprocessing method, can potentially be used in conjunction with any initial representation (point cloud, polygon mesh) and with any rendering method.

5 Blur Field Semantics

The selection of a 2D blur operation to apply to the layers is based on the following considerations. Our algorithm must smoothly generalize the blur produced by realistic depth of field.

The relevant property of depth of field relates to partial occlusion along the edges of an out-of-focus object. Since a camera lens has a finite aperture, the edges of a blurred object become semi-transparent. We require this effect. However, we must decide how partial occlusion should behave in the presence of arbitrary blur fields. We cannot fall back on any physically-realizable camera or camera model to guide us here.

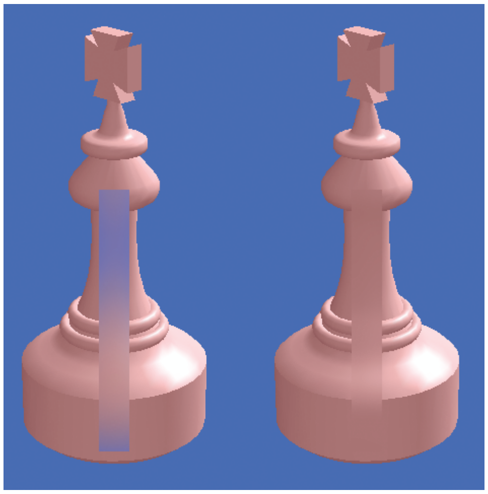

Figure 2: Left: Naive convolution. Notice how the blurred region is transparent. Right: Anisotropic diffusion. Notice how the blurred region remains opaque, as it should.

Consider, for example, a simple scene consisting of two layers: a foreground object and a background. We would like to blur the front layer according to a blur field, while maintaining plausible partial occlusion. First, we explored a naive spatially variant convolution (on each layer). Certainly this results in images that are more or less blurred in accordance with the blur field, and indeed we obtain partial transparency along the edges. However, transparency also occurred in isolated blurred regions interior to the foreground object. This does not correspond to anything present in realistic depth of field, and this is a highly objectionable artifact. Figure 2, left, shows an example of such a hole. Holes should not appear in the middle of an object, simply because the object is blurred in the middle.

We now examine the cause of, and solution to, this artifact. Transparency is represented as values of alpha less than 1. The initial unblurred image has some region where the object lies. Alpha is always 1 inside this region (in the case of opaque objects). Outside the object, the pixels have alpha of 0, indicating that nothing is there. During convolution, blurred pixels near the edge are averages of both the interior and the exterior of the object, leading to alpha values between 0 and 1. This results in a reasonable approximation to the partial occlusion found in depth of field. However, when the blur value inside an object is large, whereas the blur value towards the edges of the object are small, then the interior of the object can be averaged with pixels outside the object, despite the fact that the edge pixels themselves may remain perfectly sharp. We would avoid this artifact if only those low-blur pixels on the edge acted as barriers through which colors cannot leak. This notion leads us to the key property that our 2D blur algorithm must possess: the averaging process must be aware of the blur values along the *entire path* that a color must traverse in order to reach a destination.

This property is inherently upheld by anisotropic diffusion. Consider heat transfer on a sheet of metal. Making the analogy between heat and brightness in an image, imagine the idea of “painting a grayscale image” on the metal by heating some areas more than others. Over time, the heat diffuses, blurring the image. If the metal is more conductive in some places and less conductive in others, then at any given time, some portions of the image will be more blurred than others. Most importantly, if there is a barrier of low conductivity between two regions of high conductivity, then no heat will flow from one region of high conductivity to the other. Therefore we take inspiration from anisotropic diffusion in building our 2D blur operation. Figure 2, right, shows how anisotropic diffusion prevents a hole from appearing inside an object.

6 Algorithm Overview

There are several steps in this algorithm.

First, the scene is rendered as a collection of layers.

Associated with each layer is a position map. This position map is required because the pixels within any given layer need not all occupy the same depth. Thus, our layers are more like 3D objects and less like the flat images that the term “layer” may suggest. A position map is similar to a depth map, except that it stores the full three dimensional world coordinates of the object seen under each pixel. The position map is implemented as a high dynamic range image, where red, green, and blue correspond to x, y, and z coordinates. Figure 3 shows a typical position map.

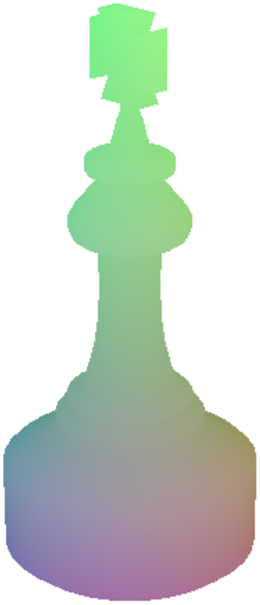

Figure 3: A typical position map. This is a linearly scaled and clamped high dynamic range image. The red, green, and blue color channels represent x, y, and z world coordinates, respectively.

We will be applying a spatially varying blur operator to each layer, which is inherently a two-dimensional process. The blur field, however, exists in three dimensions. We need a two-dimensional blur field for the layer blurring step. These two-dimensional blur fields are nonplanar slices of the three-dimensional blur field. The mapping between the two-dimensional blur field slice and the full three-dimensional blur field is performed by the aforementioned position map.

As will be explained in the next section, blur values are actually needed for some pixels outside the object itself. These pixels are those that lie within a certain distance of the object. We find these pixels by growing the region representing the object within the layer. The position map inherently cannot contain any information for pixels outside the object; thus, these blur values cannot be read from the three-dimensional blur field. Therefore reasonable blur values must be extrapolated from the known data.

Next, our carefully chosen, two-dimensional spatially varying blur operation is applied to each layer.

Finally, the layers are composited from back to front, with alpha blending.

7 Algorithm Details

We now describe the region growing, blur field extrapolation, and repeated convolution steps in full detail.

7.1 Region Growing

Often, only a small portion of a layer is covered. Thus, it would be a waste of computational resources to applying expensive blurring operations to the entire image when most of it is empty. On the other hand, a blurred object expands in size; thus, it is not sufficient only to blur the pixels lying inside the unblurred object. We therefore designate a superset of the object as the region to be blurred. This region is initialized to the object itself. The region is then grown one ring at a time by finding all the pixels lying outside the current region, but in contact with it. Rings are added until the region is sufficiently large to encompass any potential growth caused by the blurring operation. Figure 4 shows an object and the corresponding grown region. It is quite often the case that this expanded region is significantly smaller than the total size of the layer, resulting in a dramatic speedup compared to blurring the entire layer.

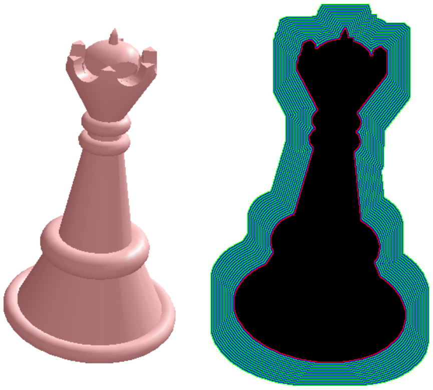

Figure 4: A chess piece and the surrounding region within which blur will be applied.

7.2 Blur Field Extrapolation

We now have blur values for each pixel within the object within the layer. However, our two-dimensional blur operation requires blur values outside the object, because a blurred image is larger than the original image. The region of the blurred image outside the original object is very important, since this is where the fuzzy edges characteristic of out-of-focus objects are. We cannot look up these blur values directly from the three-dimensional blur field, since there are no appropriate world coordinates present in the position map. Therefore, we must fabricate exterior blur values. This blur extrapolation step must ensure a smooth blur field that seamlessly merges with the interior of the object. Exterior blur values are calculated as a weighted average of the edge blur values, where the weights decrease with increasing distance from the edge point of interest. Note that the weights are normalized. That is, points farther from the edge are not less blurred; these points are merely influenced by the various edge points to a greater or lesser extent. Figure 5 illustrates what a typical extrapolated blur field looks like.

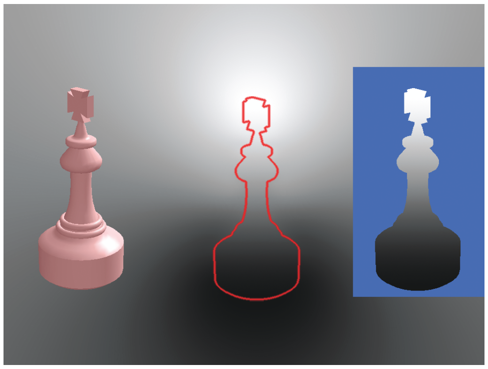

Figure 5: Left: A layer. Right: The blur field slice for the layer. Middle: The blur values outside the object have been extrapolated.

Consider the extrapolated blur value $B2D(x,y)$ at a pixel (x,y) lying outside the object (but inside the expanded region). Let n be the number of pixels on the edge of the object, and g be a weighting function causing the relative contribution of each edge pixel to decrease with distance. Then $B2D(x,y)$ can be computed as:

$$B2D(x,y) = \frac{\sum_{i=1}^{n} B2D(x_i,y_i) * g(x_i,y_i,x,y)}{\sum_{i=1}^{n} g(x_i,y_i,x,y)}. \tag{3}$$

where $g = \frac{1}{d(x_i,y_i,x,y)^2}$, and $d(x_i,y_i,x,y)$ is the Euclidean distance between pixels (x_i,y_i) and (x,y); that is, $d = \sqrt{(x_i-x)^2+(y_i-y)^2}$.

7.3 Repeated Convolution In the Spirit of Anisotropic Diffusion

Two-dimensional spatially varying blur is performed via repeated convolution in a process inspired by anisotropic diffusion. Whereas we might have started with the partial differential equation describing heat flow [Carslaw and Jaeger 1959], discretized it in space and numerically integrated it through time, our intent is merely to blur images, not accurately simulate the behavior of heat. Therefore, we move directly to a simple implementation consisting of repeated convolution, without making any claims as to the accuracy of our method as a means for simulating diffusion.

A number of iterations are carried out. For each iteration, each pixel is averaged with its four neighbors, where the blur value for each pixel determines the weights in the average. Importantly, these weights are not normalized. Thus the weights are absolute, not merely relative to one another. If the neighboring pixels have low blur values, then it is appropriate to use a low weight for those neighbor pixels. The center pixel's weight is calculated based on the weights of the neighbors, such that the weights will always sum to unity, taking the place of normalization in ensuring that no unwanted lightening or darkening occurs in regions where the blur value is changing. This local convolution is repeated so that the object can be effectively blurred by kernels of any size. Repeated convolution rather than a single convolution with a wider kernel is necessary to prevent colors from leaking across regions of low blur.

Our blur operator works as follows: To calculate the blurred value of pixel (x,y) after one iteration, we average pixel (x,y) with its four neighbors, which we refer to as north, south, east, and west, denoted by C_n, C_s, C_e and C_w, respectively. The pixel at location (x,y) is referred to as "center", and is denoted by C_c. C_i denotes the image after i iterations. We calculate C_{i+1}, the color of pixel (x,y) after the $i+1$'th iteration, using the following formula.

$$C_{i+1} = W_c * C_c + \frac{1}{4} * W_n * C_n + \frac{1}{4} * W_s * C_s + \frac{1}{4} * W_e * C_e + \frac{1}{4} * W_w * C_w$$

Where $C_c = C_i(x,y)$, $C_n = C_i(x,y-i)$, $C_s = C_i(x,y+1)$, $C_e = C_i(x+1,y)$, and $C_w = C_i(x-1,y)$. W_c is the weight for the center pixel. It is chosen such that all the weights will sum to unity, ensuring no unwanted brightening or dimming even when adjacent blur values are vastly different. That is, $W_c = 1.0 - \frac{1}{4} * W_n - \frac{1}{4} * W_s - \frac{1}{4} * W_e - \frac{1}{4} * W_w$. This formulation has the disadvantage of ignoring the B2D value for the center pixel, but the advantage of having the required property that low B2D values for the surrounding pixels lead to low amounts of blur, while high B2D values for the surrounding pixels lead to the four surrounding pixels being the dominant contribution to the blurred center pixel, leading to a high amount of blur.

We considered the anisotropic diffusion formulation proposed by Perona and Malik, but found it unsuitable for our needs because their blur values (conductance coefficients, in the terminology of heat flow) are defined as links between adjacent pixels, whereas our blur field semantics demand control over how blurred individual pixels are, not how much flow there is between pairs of adjacent pixels.

8 Results

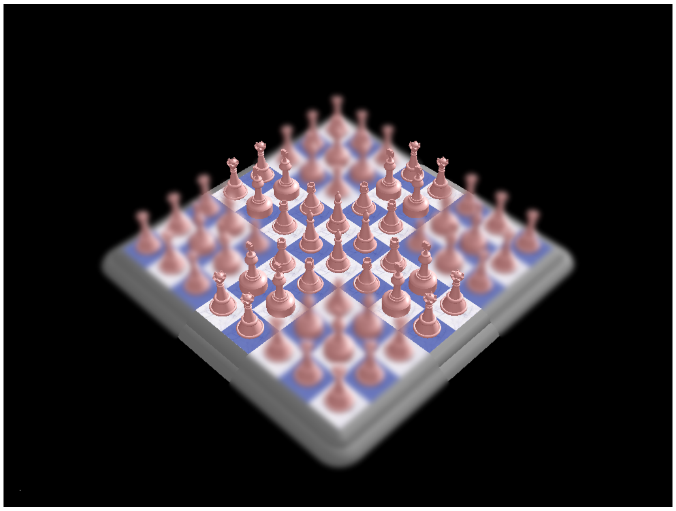

Figure 6: The volume that is in focus is shaped like a plus sign.

We have applied generalized depth of field to a chessboard scene. A fully in-focus version of this scene was rendered using a pinhole camera model. The scene was rendered into layers and corresponding position maps. The board itself is one layer, and each chess piece forms another layer. Figure 1 shows, from left to right, the original scene, a blur field, and the blurred scene corresponding to that blur field. This particular blur field is a vertical cylinder of focus centered on one chess piece. Figure 6 shows a highly populated chessboard with a plus-shaped volume in focus. In Figure 7, we see a visualization of the chess scene intersecting with a blur field. This blur field has a volume of focus oriented along a column of the chessboard. Figure 8 shows the blurred scene corresponding to the aforementioned blur field.

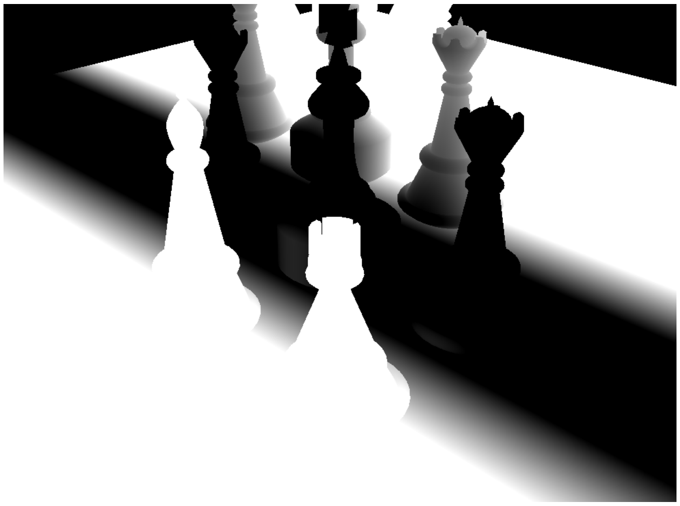

Figure 7: This blur field has a swath of acceptable focus which is planar, vertical, and aligned exactly with one row of the chessboard.

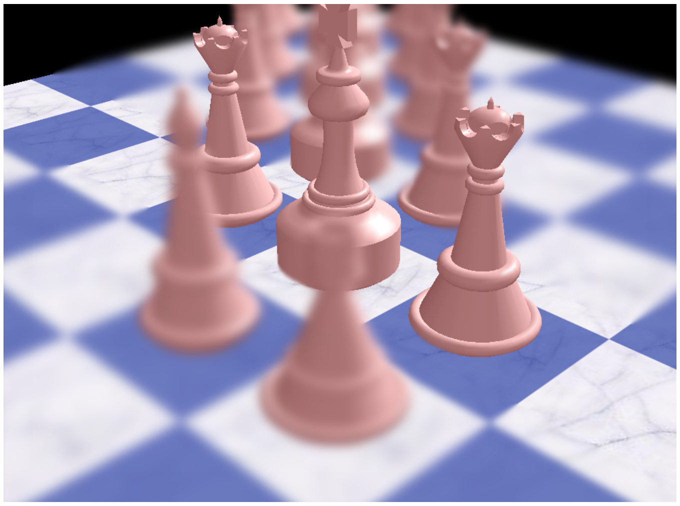

Figure 8: The chess scene blurred with the swath of acceptable focus from Figure 7.

By varying the blur field over time, generalized depth of field can be used to create interesting variations on the traditional focus pull. For example, the cylindrical focal volume can smoothly glide from chess piece to chess piece. Alternatively, the scene can gradually come into focus as a sphere of focus emerges from the center of the scene and gradually expands in size. Please see the accompanying video for examples of generalized depth of field in motion.

9 Future Work

Currently, the scene must be split into a series of layers in order for our algorithm to achieve partial occlusion. Although any correct handling of partial occlusion would require some form of layering, our present approach does not work well for scenes that cannot easily be split into layers. Our layers are nonplanar, alleviating an important restriction in previous post-processing approaches to depth of field. The chessboard, for example, is one layer, while in previous methods it would have had to be split into many layers, because it spans a deep range of depths.

However, our method cannot presently handle a complex layer that occludes itself. Depth complexity is handled solely through different layers; hence, there cannot be any depth complexity within a single layer. A reasonable way to solve this would be to operate on a layered depth image (LDI) [Shade et al. 1998]. An LDI records the colors and depths of every object that is hit along each ray emanating from the center of projection. LDIs do not explicitly group the scene into objects or layers, but they do retain the colors of objects that are occluded. Thus, they have the benefits of layers without the disadvantages, and hence would make an excellent representation on which a future version of our algorithm could operate.

Generalized depth of field would be a useful tool in photography. Unfortunately, no real lens can possibly achieve this effect, since it would need to violate the laws of optics. We believe that by processing and combining a series of exposures taken with a camera over a variety of focus settings, and with the help of computer vision techniques to estimate depth, generalized depth of field should be achievable for real world scenes.

10 Conclusion

In a real camera, the control over depth of field is limited by the laws of physics and by physical constraints. The only degrees of freedom are: the distance at which the camera lens is focused, the f/stop of the lens, and the focal length of the lens. The depth of field effect has previously been simulated in computer graphics, but with the same limited control as found in real camera lenses. In this paper, we have generalized depth of field in computer graphics by allowing the user to independently specify the amount of blur at each point in three-dimensional space. Generalized depth of field provides a novel tool to emphasize an area of interest within a 3D scene, to pick objects out of a crowd, and to render a busy, complex picture more understandable by focusing only on relevant details that may be scattered throughout the scene.

Our algorithm first renders the scene into a set of nonplanar layers and associated position maps. Then, each layer is blurred according to a 3D blur field provided by the user. Choosing a suitable blur algorithm for the layers is critical. Using straightforward convolution would result in artifacts. We observe that such artifacts occur because straightforward convolution would not respect our blur semantics, which specify the intended meaning of the blur field to resemble depth of field, and not just any arbitrary blur. Thus, we develop appropriate blur semantics such that the blur algorithm will properly generalize depth of field.

It is important to note that our method not only enables new effects, but it can also simulate realistic depth of field. This can be done merely by constructing a blur field based on the circle of confusion of a conventional lens. A realistic blur field corresponding to a conventional camera would vary only along the depth axis, remaining constant within planes parallel to the image plane. The variation along the depth axis is related to the circle of confusion. Specifically, the blur value at a given depth is specified by equation 2 in section 3.2. Alternatively, a blur field corresponding to a view camera with tilted plane of sharp focus could be simulated by letting the circle of confusion vary along a tilted axis, rather than the depth axis.

Our approach has advantages over existing approaches for simulating realistic depth of field. Although our approach is a postprocessing technique, unlike other postprocessing methods, it handles occlusion correctly, like the multisampling approaches do. Unlike the other postprocessing methods, our approach is not based on the assumption that objects can be approximated as planar layers that are aligned parallel to the image plane, nor does our method suffer from discretization issues. Conversely, multisampling methods handle occlusion properly and do not suffer from discretization issues, but their performance degrades in proportion to the time it takes to render the scene. Thus, a scene that comprises many small polygons with complex shaders, which would already be slow to render due to its complexity, would need to be rendered many times in a multisampling method. On the other hand, postprocessing methods (such as ours) operate on an image based representation that is output by the renderer. Thus, our method requires no more time to process a scene of a million polygons than for a hundred polygons.

Generalized depth of field can simulate camera models for which no corresponding real-world camera can exist. In that sense, this is a kind of non-photorealistic rendering. Another way of considering this work is as representing a real camera, but in a universe whose laws of optics behave differently from how they behave in the real world. Thus, we might categorize generalized depth of field as non-physically-realistic rendering. Thinking along these lines engenders the notion that perhaps entirely new classes of heretofore unimagined images might be created by carefully breaking other laws of the image formation process.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. In *Proceedings of SIGGRAPH 2004*, ACM.

BAKER, A. A. 1985. *Applied Depth of Field*. Focal Press.

BARSKY, B. A., AND PASZTOR, E. 2004. Rendering skewed plane of sharp focus and associated depth of field. In *SIGGRAPH 2004 Tech Sketch*, ACM.

BARSKY, B. A., BARGTEIL, A. W., GARCIA, D. D., AND KLEIN, S. A. 2002. Introducing vision-realistic rendering. In *Eurographics Rendering Workshop, posters*, 26–28.

BARSKY, B. A., HORN, D. R., KLEIN, S. A., PANG, J. A., AND YU, M. 2003. 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).*, 246–255.

BARSKY, B. A., HORN, D. R., KLEIN, S. A., PANG, J. A., AND YU, M. 2003. Camera models and optical systems used in computer graphics: Part ii, image based techniques. In *Proceedings of the 2003 International Conference on Computational Science and its Applications (ICCSA'03).*, 256–265.

BARSKY, B. A., TOBIAS, M. J., AND CHU, D. R. H. D. P. 2003. Investigating occlusion and discretization problems in image space blurring techniques. In *First International Conference on Vision, Video, and Graphics*, 97–102.

BARSKY, B. A., TOBIAS, M. J., CHU, D. P., AND HORN, D. R. 2005. Elimination of artifacts due to occlusion and discretization problems in image space blurring techniques. *Graphical Models 67*, 6, 584–599.

BARSKY, B. A. 2004. Vision-realistic rendering: simulation of the scanned foveal image from wavefront data of human subjects. In *Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, ACM, 73–81.

BERTALMIO, M., FORT, P., AND SANCHEZ-CRESPO, D. 2004. Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards. In *IEEE Second International Symposium on 3DPVT*, IEEE.

BUDIN, C. 2004. zdof: A fast, physically accurate algorithm for simulating depth-of-field effects in synthetic images using z-buffers. In *SIGGRAPH 2004 Tech Sketch*, ACM.

CARSLAW, H. S., AND JAEGER, J. C. 1959. *Conduction of Heat in Solids, 2nd edition*. Oxford University Press.

CHASTINE, J., AND BROOKS, J. 2005. Emphasizing the area of interest using real-time shaders. In *ACM SIGGRAPH 2005 Poster Session*, ACM.

COOK, R. L., PORTER, T., AND CARPENTER, L. 1984. Distributed ray tracing. In *ACM SIGGRAPH 1984 Conference Proceedings*, ACM, 137–145.

DEMERS, J. 2004. Depth of field: A survey of techniques. In *GPU Gems*, 375–390.

DIPPE, M. A. Z., AND WOLD, E. H. 1985. Antialiasing through stochastic sampling. In *ACM SIGGRAPH 1985 Conference Proceedings*, ACM, 69–78.

ERICKSON, B., AND ROMANO, F. 1999. *Professional Digital Photogrpahy*. Prentice Hall.

FEARING, P. 1996. Importance ordering for real-time depth of field. In *Proceedings of the Third International Conference on Computer Science*, 372–380.

GOLDBERG, N. 1992. *Camera Technology: The Dark Side of the Lens*. Academic Press.

GREENLEAF, A. R. 1950. *Photographic Optics*. The Macmillan Company.

GRÖLLER, M. E. 1995. Nonlinear raytracing - visualizing strange worlds. *Visual Computer 11*, 5, 263–274.

HAEBERLI, P., AND AKELEY, K. 1990. The accumulation buffer: hardware support for high-quality rendering. In *ACM SIGGRAPH 1990 Conference Proceedings*, ACM, 309–318.

HEIDRICH, W., SLUSALLEK, P., AND SEIDEL, H.-P. 1997. An image-based model for realistic lens systems in interactive computer graphics. In *Proceedings of Graphics Interface 1997*, Canadian Human Computer Communication Society, 68–75.

ISAKSEN, A., MCMILLAN, L., AND GORTLER, S. J. 2000. Dynamically reparameterized light fields. In *ACM SIGGRAPH 2000 Conference Proceedings*, ACM.

JENKINS, F. A., AND WHITE, H. E. 1976. *Fundamentals of Optics*. McGraw-Hill, Inc.

KINGSLAKE, R. 1992. *Optics in Photography*. SPIE – The International Society for Optical Engineering.

KOLB, C., MITCHELL, D., AND HANRAHAN, P. 1995. A realistic camera model for computer graphics. In *ACM SIGGRAPH 1995 Conference Proceedings*, ACM, 317–324.

KOSARA, R., MIKSCH, S., AND HAUSER, H. 2001. Semantic depth of field. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, 97.

KRIVANEK, J., ZARA, J., AND BOUATOUCH, K. 2003. Fast depth of field rendering with surface splatting. In *Computer Graphics International 2003*.

LEE, M. E., REDNER, R. A., AND USELTON, S. P. 1985. Statistically optimized sampling for distributed ray tracing. In *ACM SIGGRAPH 1985 Conference Proceedings*, 22–26.

LEVOY, M., CHEN, B., VAISH, V., HOROWITZ, M., MCDOWALL, I., AND BOLAS, M. 2004. Synthetic aperture confocal imaging. In *ACM SIGGRAPH 2004 Conference Proceedings*, ACM, 825–834.

LONDON, B., UPTON, J., KOBRE, K., AND BRILL, B. 2002. *Photography, Seventh Edition*. Prentice Hall.

MERKLINGER, H. M. 1992. *The Ins and Outs of Focus: Internet Edition*. http://www.trehnholm.org/hmmerk/download.html.

MERKLINGER, H. M. 1993. *Focusing the View Camera*. ISBN 0-9695025-2-4.

MERKLINGER, H. M. 1996. View camera focus and depth of field, parts i and ii. *View Camer magazine, July/August 1996 pp. 55-57 and September/Octor pp. 56-58*.

MUSGRAVE, F. K., AND BERGER, M. 1990. A note on ray tracing mirages. *IEEE Computer Graphics and Applications 10*, 6, 10–12.

NG, R. 2005. Fourier slice photography. In *ACM SIGGRAPH 2005 Conference Proceedings*, ACM, 735–744.

PERONA, P., AND MALIK, J. 1988. A network for multiscale image segmentation. In *IEEE Int. Symp. on Circuits and Systems*, IEEE, 2565–2568.

PERONA, P., AND MALIK, J. 1994. Scale space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell. 12*, 7, 629–639.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. In *ACM SIGGRAPH 1984 Conference Proceedings*, 253–259.

POTMESIL, M., AND CHAKRAVARTY, I. 1982. Synthetic image generation with a lens and aperture camera model. *ACM Transactions on Graphics 1*, 2, 85–108.

RAY, S. F. 2002. *Applied Photographic Optics, Third Edition*. Focal Press.

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

SCOFIELD, C. 1994. $2\frac{1}{2}$-d depth of field simulation for computer animation. In *Graphics Gems III*, Morgan Kaufmann.

SHADE, J., GORTLER, S. J., WEI HE, L., AND SZELISKI, R. 1998. Layered depth images. In *Proceedings of ACM SIGGRAPH 1998*.

SHAMAN, H. 1978. *The view camera: operations and techniques*. American Photographic Book Publishing Co.

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

SIMMONS, S. 1992. *Using the View Camera, Amphoto Revised Edition*. Watson-Guptill Publications.

STONE, J. 2004. *User's Guide to the View Camera, Third Edition*. Prentice Hall.

STROEBEL, L., COMPTON, J., AND CURRENT, I. 2000. *Basic Photographic Materials and Processes, Second Edition*. Focal Press.

STROEBEL, L. 1999. *View Camera Technique, 7th Edition*. Focal Press.

SU, S. L., DURAND, F., AND AGRAWALA, M. 2005. De-emphasis of distracting image regions using texture power maps. In *Proceedings of Texture 2005 (at ICCV)*, 119–124.

WEISKOPF, D., SCHAFHITZEL, T., AND ERTL, T. 2004. Gpu-based nonlinear ray tracing. *IEEE Computer Graphics and Applications 23*, 3, 625–633.

WILBURN, B., JOSHI, N., VAISH, V., TALVALA, E.-V., ANTUNEZ, E., BARTH, A., ADAMS, A., HOROWITZ, M., AND LEVOY, M. 2005. High performance imaging using large camera arrays. In *ACM SIGGRAPH 2005 Conference Proceedings*, ACM, 765–776.

WILLIAMS, J. B. 1990. *Image Clarity: High-Resolution Photography*. Focal Press.