Investigating Occlusion and Discretization Problems in Image Space Blurring Techniques

Brian A. Barsky [a,b], Michael J. Tobias [a], Daniel R. Horn [a], Derrick P. Chu [a,c]

[a] *Computer Science Division, University of California, Berkeley, California, 94720-1776, USA*

[b] *School of Optometry, University of California, Berkeley, California, 94720-2020, USA*

[c] *Computer Science Department, University of California, Los Angeles, California, 90095-1596, USA*

Abstract

Traditional computer graphics methods render images that appear sharp at all depths. Adding blur can add realism to a scene, provide a sense of scale, and draw a viewer's attention to a particular region of a scene. Our image based blur algorithm needs to distinguish whether a portion of an image is either from a single object or is part of more than one object. This motivates two approaches to identify objects after an image has been rendered. We illustrate how these techniques can be used in conjunction with our image space method to add blur to a scene.

Key words: Image forming and processing, Visual perception
PACS: 42.30.Va, 42.66.Si

1 Introduction

Images rendered with traditional computer graphics techniques appear completely in focus. However, images formed by optical systems have depth of field; that is, some regions of the image are in focus while other are blurred. This is useful to draw a viewer's attention to a specific region of the image and can help to indicate the scale of a particular scene. The rendering of a 3D

Email address: barksy@cs.berkeley.edu (Brian A. Barsky).
URL: http://www.cs.berkeley.edu/optical (Brian A. Barsky).

scene requires transforming from *object space* to *image space*. This transformation is parameterized by a camera angle and location. There are algorithms that can be used to apply blur in object space[2] where the entire geometry of the scene is present. Yet, after the transformation to image space, there are a wider variety of methods for applying depth of field.[3]

2 Object Space Methods for Depth of Field

Since scene geometry information is readily available in object space, this is the most straightforward and ideal space in which to compute blur. Distributed raytracing, first proposed by Cook et al. in 1984[6], sends multiple rays from the aperture of the lens, through each pixel on the image plane, and into the scene. Kolb et al.[11] proposed a more realistic algorithm for computing object-space blur. For each pixel in the image, their approach emits multiple rays from the film plane, through a system of lens elements, and computes where these rays intersect the scene. Using this technique, the final color of each pixel is the average of the colors at all the nearest ray-object intersections for the rays emerging from this pixel.

Inherent in the distributed technique are questions of the density and spatial distribution of samples across the lens; these issues have been addressed by Cook[5], Dippe and Wold[7], Lee et al.[12], and Kajiya[10]. Further details are provided in our survey paper on object space techniques[2]. Additionally, simulating depth of field using these distributed ray tracing approaches is more computationally expensive than standard ray tracing. The increase in computation cost over standard ray tracing is proportional to the number of rays per pixel.

3 An Image Space Method for Depth of Field

Image space techniques have been developed to avoid this increase in computation time. Potmesil and Chakravarty[13] were the first to propose an image space technique that smears colors over a region (the *circle of confusion*) for each pixel. The efficiency of the technique was improved by Rokita[14]. In 1994, Shinya[15] solved occlusion issues present in Potmesil's solution by using a sub-pixel buffer for each pixel to track all rays entering that pixel from a lens.

There has also been work regarding depth of field with respect to light fields, which results in realtime[8] and non-realtime[9] techniques. For a more detailed discussion of light fields and blur, the reader is referred to our survey

Fig. 1. Original Tin Toy image.

Fig. 2. Depth information, scaled from near (shown in white) to far (in black), for the Tin Toy image.

Fig. 3. The image is separated by depth into discrete sub-images.

paper on image space techniques[3].

In 2002, we proposed an alternate approach[1] which is also applied in image space. The technique takes as input a given image with a depth map (Figures 1 and 2) and then separates the image according to depth into discrete sub-

Fig. 4. Final blurred Tin Toy image focused on the Tin Toy.

Fig. 5. Final blurred Tin Toy image focused on the baby.

images (Figure 3). Each sub-image is blurred independently and then they are combined to produce the resulting image which has depth of field; this is shown in Figures 4 and 5.

4 Problems with Image Space Blurring Techniques

Using image space techniques raises two concerns: the *occlusion problem* and the *discretization problem*.

4.1 Occlusion Problem

Although processing in image space allows an increase in speed, it has the problem that the scene geometry is missing, which we refer to as the occlusion problem. A lens with a finite aperture allows more of the scene to be visible than would be seen through an infinitesimal pinhole, as illustrated in Figure 6. Thus, without additional input, the colors from parts of the scene that are

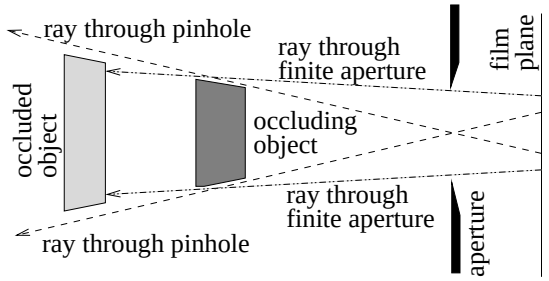

Fig. 6. An object that is visible through a finite aperture can be occluded when viewed through a pinhole.

behind objects would have to be approximately reconstructed using the border colors of visible objects.

To address the need for occluded scene information, the algorithm can also take as input scene data that are behind the foreground objects. This data comprises both color and depth information. Thus, in addition to the data at the first intersection, this approach can use all the subsequent intersection data located along the infinite ray drawn from the viewpoint through any pixel in the original scene. This allows the algorithm to obtain colors from occluded objects, instead of approximating the reconstruction of occluded color data.

However, if it is not possible to determine additional intersections from the geometry itself, then we must approximate the occluded pixels. If we assume that the hidden pixels have similar colors to those of the neighboring pixels that are visible, then we can approximate the hidden pixels by a weighted sum of the visible pixels. This is equivalent to convolving the visible sub-images with a well-chosen Gaussian kernel and then using the results as approximations for the occluded areas. Note that since these approximated pixels are used only for blurring, accuracy is not critical.

4.2 Discretization Problem

The second problem, which we call the discretization problem, results from separating the image by depth. At adjacent pixels in different sub-images, the calculation of depth of field is complicated. This arises because these adjacent pixels may or may not correspond to the same object.

If an object spans more than one sub-image, there will be adjacent pixels that correspond to the same object but reside in different sub-images. We refer to a pair of such pixels as a *joint-pair*, as shown in Figure 7. Otherwise, when the adjacent pixels in different sub-images correspond to different objects, we call them a *disjoint-pair*, as illustrated in Figure 8.

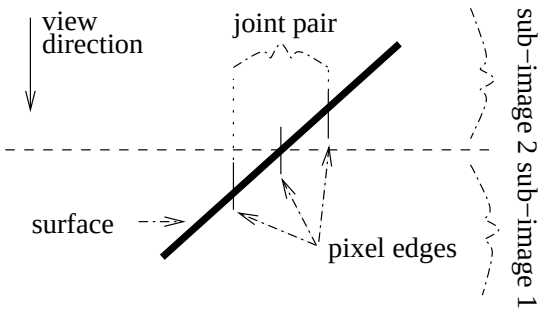

Fig. 7. A joint-pair is a pair of adjacent pixels belonging to the same object but residing in different sub-images.

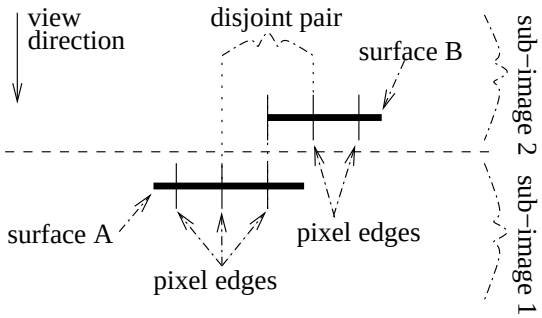

Fig. 8. A disjoint-pair is a pair of adjacent pixels belonging to different objects and residing in different sub-images.

When two pixels reside in distinct sub-images, it is difficult to determine whether they form a joint-pair or disjoint-pair.

Artifacts are introduced into the image when a single object straddles two sub-images and the sub-images are blurred. Consider a pixel p; its neighboring colors are the colors of those pixels in the nearest sub-image at or behind the sub-image containing the pixel p, as depicted in Figure 9. As presented earlier in this section, these neighboring colors are either determined from the first intersection, or determined from latter intersections, or approximated. In a joint-pair, we refer to the pixels in the near and far sub-images as the *near pixels* and *far pixels*, respectively. The artifact arises when the far pixel is averaged with neighboring colors behind the near pixel that do not match the far pixel's color. The neighboring colors are often black, which is the default background color. Consequently, a black blurred band occurs at the intersection of the object with the separation of the sub-images that it spans, as can be seen in Figure 10.

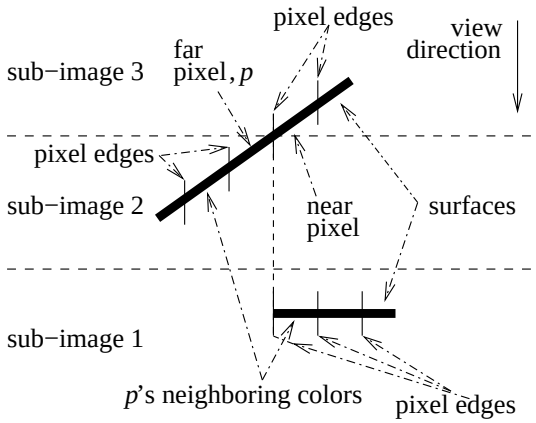

Fig. 9. The neighboring colors of a pixel p are the colors of those pixels in the nearest sub-image at or behind the sub-image containing the pixel p.

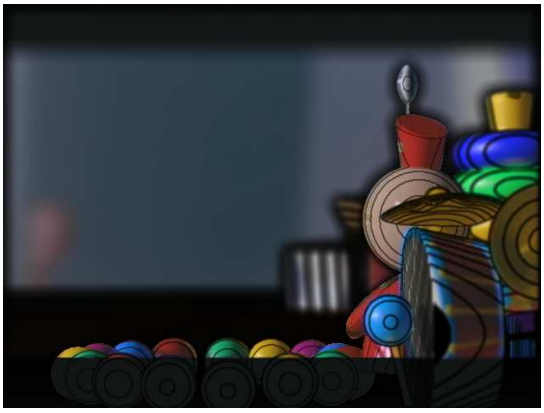

Fig. 10. Black bands appear at the locations where the sub-images are separated, unless the algorithm properly distinguishes between joint-pairs and disjoint-pairs.

5 Object Identification as a Solution for Image Space Artifacts

To eliminate these band artifacts that arise when an object is separated into multiple discrete sub-images, the algorithm attempts to identify entire objects within the image. This eliminates the artifact by avoiding the separation of objects across sub-images. Instead, when a large object straddles several sub-images, each sub-image will include the entire object instead of only a part of that object. Consequently, the object will have minimal artifacts due to blurring.

We will now consider two approaches for object identification to properly blur the scene. Our first approach uses the depth difference of adjacent pixels to identify objects. In our second approach, the Canny edge detection[4] algorithm is applied to draw borders between objects and hence identify them.

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3

2	−3	3
−4	7	−4
2	−2	−3
2	−2	6

Fig. 11. An example of a first degree difference map (right) resulting from applying a horizontal difference to the first 16 digits of π (left).

6 Approaches for Object Identification

6.1 Adjacent Pixel Difference Technique

The first technique for including points from objects that span several sub-images assumes a surface with a given order of continuity. As input to the algorithm, we select the order of continuity, denoted C^n, of the surface. In addition, the bound on the nth derivative of depth with respect to the image plane coordinates is selected such that adjacent pixels within the bound correspond to the same object. Since image space is a discrete representation of continuous geometry, we use the difference as the discretized counterpart of the derivative. Figure 11 illustrates a first degree difference map for an arbitrary image.

The algorithm assigns an object identifier to each pixel and then groups together those pixels that share an object identifier. Once all objects are located, it is straightforward to determine whether the neighboring colors should be obtained from objects in front of, at, or behind, the current sub-image.

6.2 Edge Detection Technique

Our second method for identifying objects begins by using a variant of the Canny edge detection algorithm[4]. The Canny algorithm takes as input an intensity map for the image, and it convolves the intensity map with the first derivative of a Gaussian function. The algorithm then marks pixels in the resulting array whose magnitude exceeds a specified upper threshold. These marked pixels are grouped into edge curves based on the assumption that neighboring marked pixels that have consistent orientations belong to the same edge.

Our technique uses a depth map as the intensity map as input to this edge

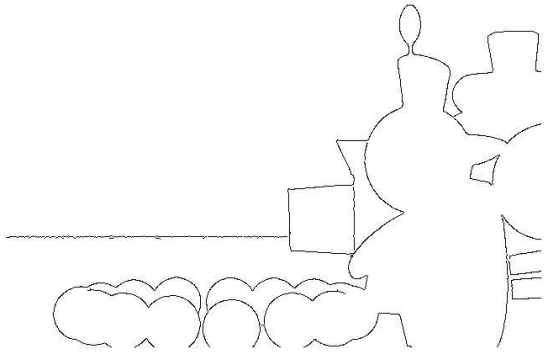

Fig. 12. Using depth map information as input, the edge detection algorithm identifies where object edges lie in the image.

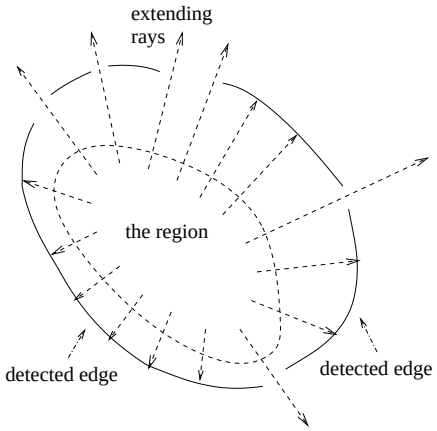

Fig. 13. Extending the region involves taking the union of the line segments that begin within the original region and do not intersect the detected edges.

detection algorithm. Figure 12 shows the result of edge detection on the example depth map. Using this variant of the Canny algorithm to segment a scene into distinct objects avoids inadequacies that are common to traditional edge detection methods. In particular, using depth information avoids erroneous detection of edges that correspond to the surface markings and shadows of objects in the scene.

Starting with the region formed by the boundary pixels in the current subimage, the algorithm extends that region until it is bounded by previously detected edges. Specifically, extending the region involves taking the union of the line segments that begin within the original region and do not intersect the detected edge segments; this is illustrated in Figure 13.

Thus, both the adjacent pixel difference technique and the Canny edge detection approach address the discretization problem that is associated with our technique for image space blurring.

7 Summary

Our two proposed approaches differ both in their inputs and in their methods to identify objects to solve the discretization problem.

The adjacent pixel difference technique can utilize every intersection between an infinite ray from the camera and the scene. It tags entire objects, since the algorithm has access to all the image space data of the scene.

The Canny variant relies only on the input image and depth map. It uses detected edges to exclude regions outside a given sub-image that are not part of an object within that sub-image.

Thus, both approaches improve our image space algorithm for depth of field by determining which regions of the image should be included in a given sub-image to create an extended sub-image. That extended sub-image may then be independently blurred and combined with other sub-images without introducing artifacts.

Acknowledgments

The authors would like to thank Adam W. Bargteil and Jeffrey A. Pang of the Computer Science Division of the University of California, Berkeley for their helpful comments. We also wish to thank Edwin E. Catmull, Christine Freeman, and Randy Nelson of Pixar Animation Studios for providing the original Tin Toy image.

References

[1] 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.

[2] 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.

[3] 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 II, Image 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.

[4] John F. Canny. A computational approach to edge detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[5] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, January 1986.

[6] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In Hank Christiansen, editor, *ACM SIGGRAPH 1984 Conference Proceedings*, pages 137–145, Minneapolis, July 23–27 1984.

[7] Mark A. Z. Dippe and Erling H. Wold. Antialiasing through stochastic sampling. In Brian A. Barsky, editor, *ACM SIGGRAPH 1985 Conference Proceedings*, pages 69–78, San Francisco, July 22–26 1985.

[8] 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.

[9] 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.

[10] James T. Kajiya. The rendering equation. In *ACM SIGGRAPH 1986 Conference Proceedings*, pages 143–150, Dallas, 1986.

[11] Craig Kolb, Don Mitchell, and Pat Hanrahan. A realistic camera model for computer graphics. In Robert L. Cook, editor, *ACM SIGGRAPH 1995 Conference Proceedings*, pages 317–324, Los Angeles, August 6–11 1995.

[12] Mark E. Lee, Richard A. Redner, and Samuel P. Uselton. Statistically optimized sampling for distributed ray tracing. In Brian A. Barsky, editor, *ACM SIGGRAPH 1985 Conference Proceedings*, pages 61–67, San Francisco, July 22–26 1985.

[13] 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).

[14] Przemyslaw Rokita. Fast generation of depth-of-field effects in computer graphics. *Computers & Graphics*, 17(5):593–595, September 1993.

[15] 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.