Available online at www.sciencedirect.com

SCIENCE DIRECT®

Graphical Models 67 (2005) 584–599

Graphical Models

www.elsevier.com/locate/gmod

Elimination of artifacts due to occlusion and discretization problems in image space blurring techniques ☆

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

[a] *Computer Science Division, University of California, Berkeley, CA 94720-1776, USA*
[b] *School of Optometry, University of California, Berkeley, CA 94720-2020, USA*
[c] *Computer Science Department, University of California, Los Angeles, CA 90095-1596, USA*

Received 30 September 2003; received in revised form 6 June 2004; accepted 10 January 2005
Available online 21 July 2005

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.
© 2005 Elsevier Inc. All rights reserved.

PACS: 42.30.Va; 42.66.Si

Keywords: Image forming and processing; Visual perception

☆ All requests for permission to use the figures in this article should be addressed to B.A. Barsky.
* Corresponding author.
E-mail address: barsky@cs.berkeley.edu (B.A. Barsky).
URL: http://www.cs.berkeley.edu/~barsky/ (B.A. Barsky).

1524-0703/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2005.01.009

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 others 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 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 [3] 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 [4].

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 ray tracing, first proposed by Cook et al. [7], sends multiple rays from the aperture of the lens, through each pixel on the image plane, and into the scene. Kolb et al. [12] 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 [6], Dippe and Wold [8], Lee et al. [13], and Kajiya [11]. Further details are provided in our survey paper on object space techniques [3]. 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 [14] 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 [15]. In 1994, Shinya [16] 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 [9] and non-realtime [10] techniques. For a more detailed discussion of light fields and blur, the reader is referred to our survey paper on image space techniques [4].

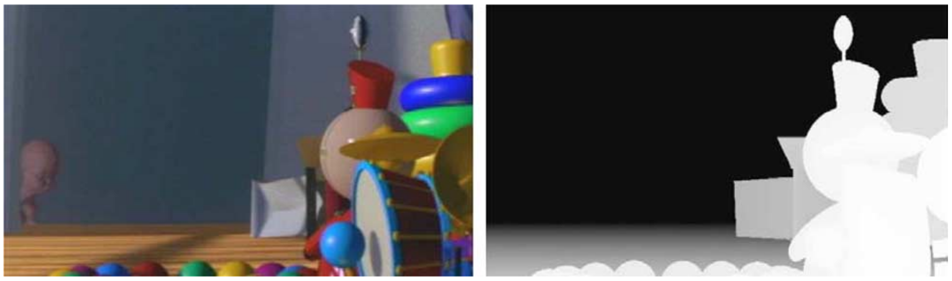


Fig. 1. Original Tin Toy image (left) and its associated depth information (right), which is scaled from near (shown in white) to far (in black). © 2005 by B.A. Barsky.

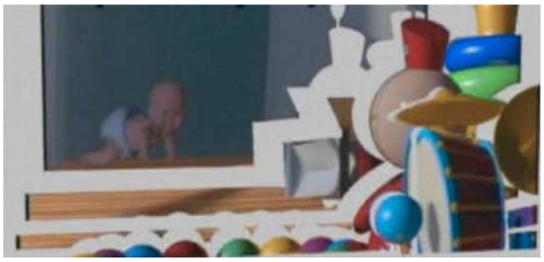


Fig. 2. The image is separated by depth into discrete sub-images. © 2005 by B.A. Barsky.

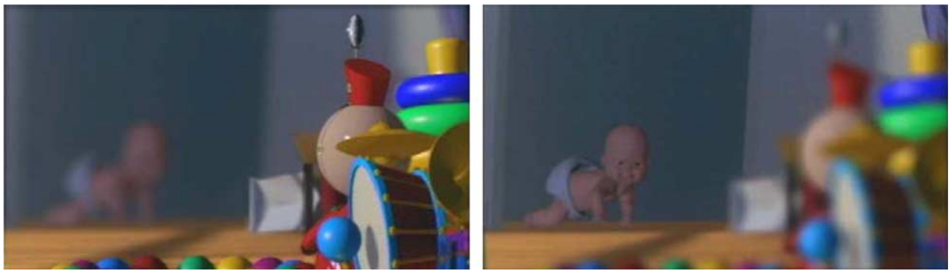


Fig. 3. Final blurred Tin Toy image focused on the Tin Toy (left) and focused on the baby (right). © 2005 by B.A. Barsky.

In 2002, we proposed an alternate approach [2], updated in 2004 [1], which is also applied in image space. The technique takes as input a given image with a depth map (Fig. 1) and then separates the image according to depth into discrete sub-images (Fig. 2). 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 Fig. 3.

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 Fig. 4. Thus, without additional input, the colors from parts of the scene that are 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. These data comprise 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 Fig. 5. Otherwise, when the adjacent pixels in

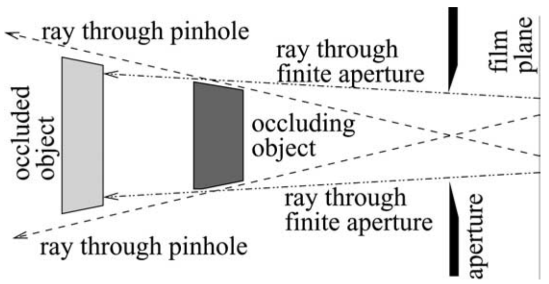


Fig. 4. An object that is visible through a finite aperture can be occluded when viewed through a pinhole. © 2005 by B.A. Barsky.

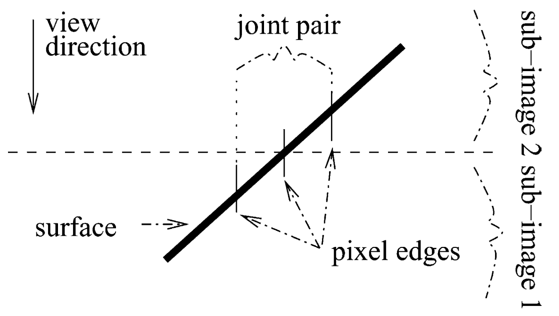


Fig. 5. A joint-pair is a pair of adjacent pixels belonging to the same object but residing in different sub-images. © 2005 by B.A. Barsky.

different sub-images correspond to different objects, we call them a *disjoint-pair*, as illustrated in Fig. 6.

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 Fig. 7. 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 Fig. 8.

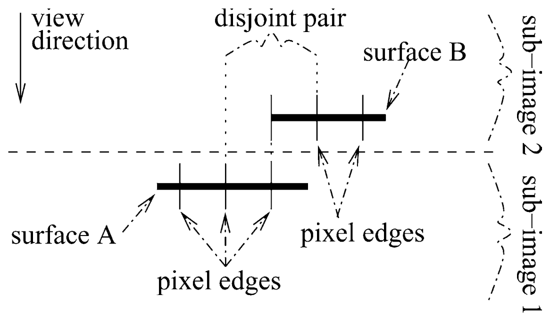


Fig. 6. A disjoint-pair is a pair of adjacent pixels belonging to different objects and residing in different sub-images. © 2005 by B.A. Barsky.

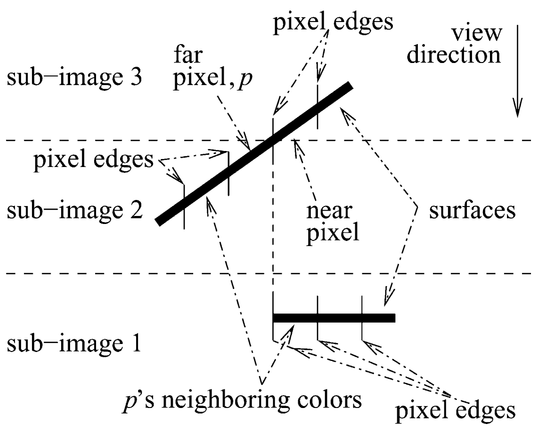


Fig. 7. The neighboring colors of a pixel p are the colors of those pixels in the nearest sub-image at or behind the sub-image containing the pixel p. © 2005 by B.A. Barsky.

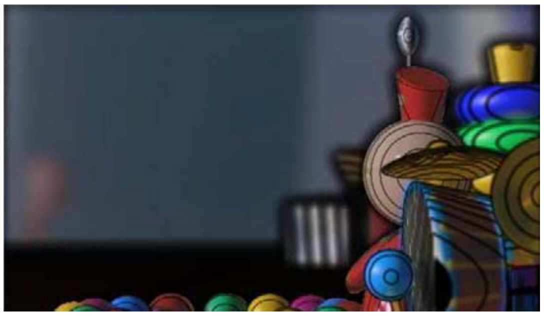


Fig. 8. Black band artifacts appear at the locations where the sub-images are separated, unless the algorithm properly distinguishes between joint-pairs and disjoint-pairs. © 2005 by B.A. Barsky.

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. In our first approach, the Canny edge detection [5] algorithm is applied to draw borders between objects and hence identify them. Our second approach uses the depth difference of adjacent pixels to identify objects.

6. Approaches for object identification

6.1. Edge detection technique

Our first method for identifying objects begins by using a variant of the Canny edge detection algorithm [5]. 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 detection algorithm. Fig. 9 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 sub-image, 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 Fig. 10.

Using the teapot scene (Fig. 12), an image (Fig. 13) is created which contains many artifacts when no object identification technique is applied. Fig. 14 demonstrates the result of the Edge Detection method to eliminate these artifacts.

The details of this method are illustrated in Fig. 11, which shows how it operates at a given depth. The pixels corresponding to the current depth plane are shown as dark gray. To determine the correct blurred color for a particular pixel in the sub-image at this given depth, we need to know the colors of the surrounding pixels, some of which may lay outside the current sub-image. This requirement motivates

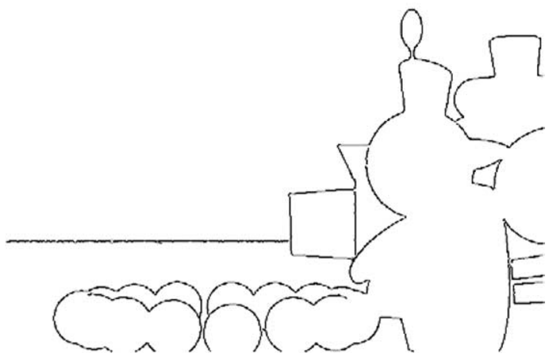


Fig. 9. Using depth map information as input, the edge detection algorithm identifies where object edges lie in the image. © 2005 by B.A. Barsky.

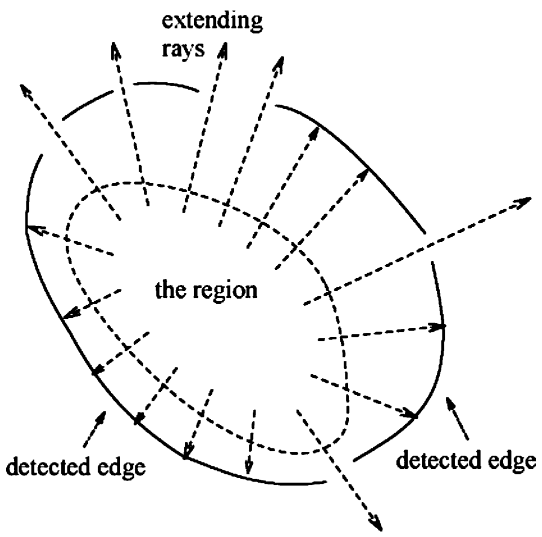


Fig. 10. Extending the region involves taking the union of the line segments that begin within the original region and do not intersect the detected edges. © 2005 by B.A. Barsky.

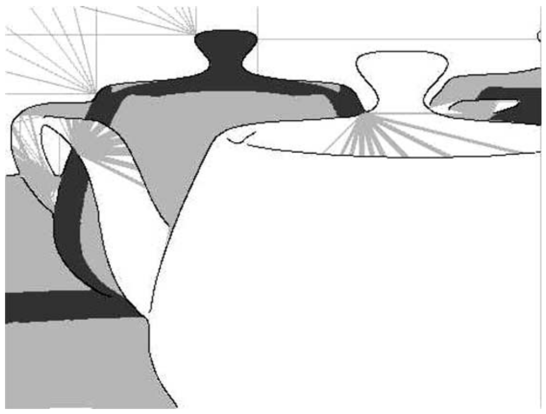


Fig. 11. Internal processing of the Edge Detection method. © 2005 by B.A. Barsky.

this process of extension, where pixels are added to the current sub-image with respect to edge curves (which form the black outline of the teapot objects in Fig. 11). These added pixels are shown in light gray in Fig. 11.

Note that before extending the sub-images, each pixel in the original unblurred image belonged to a unique sub-image. However, during sub-image extension, each of these pixels might be added to other sub-images.

To determine the correct blurred color for a given sub-image pixel, we need to add pixels outside the current sub-image. However, this must be done in such a way that the added pixels will only affect the blurred colors in the current sub-image;

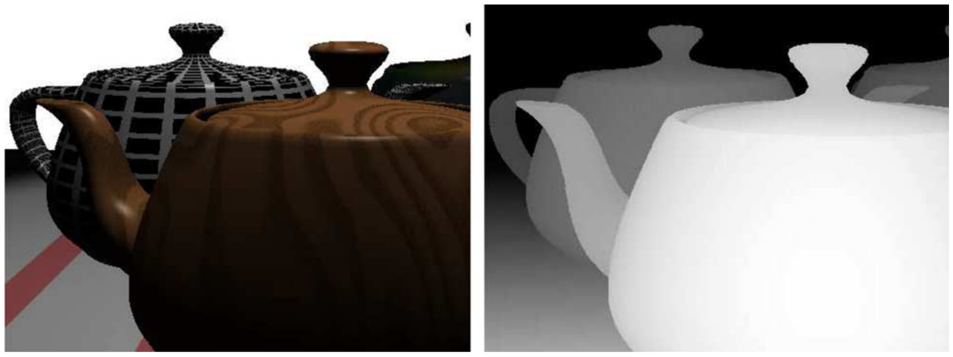


Fig. 12. Original teapot scene (left) and its associated depth information (right). © 2005 by B.A. Barsky.

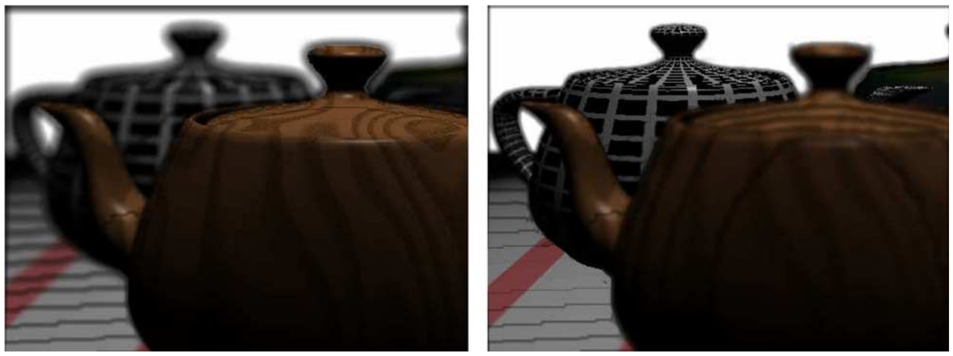


Fig. 13. Blurred teapot scene without object identification focused on the foreground (left) and background (right). Black band artifacts appear at the locations where the sub-images are separated. © 2005 by B.A. Barsky.

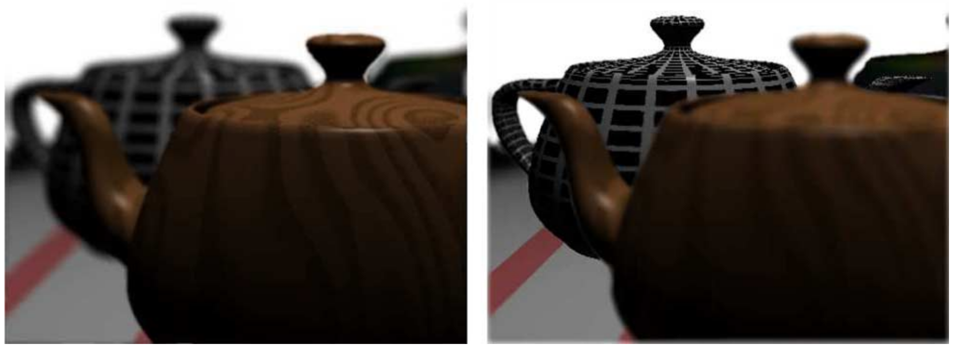


Fig. 14. Artifacts eliminated by the Edge Detection method for the teapot scene focused on the foreground (left) and background (right). © 2005 by B.A. Barsky.

consequently, the composition of all the blurred sub-images will produce a correct final result.

To this end, we set the alpha values of the added pixels to zero. Doing so confines the effects of the added pixels to the area corresponding to the current sub-image in the final composite image. Note that the pixels that are added in one sub-image have no effect on the blur calculations for other sub-images.

Thus, extending the sub-images yields a complete and correct set of colors from the original image from which we can convolve with the blur filter to determine each pixel's new blurred color. In particular, this method corrects the black-band problem, in which pixels straddling the edge of a sub-image are blurred with unknown black pixels, by blurring instead with the correct neighboring pixels that have now been added to the sub-image.

6.2. Adjacent pixel difference technique

The second 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. Fig. 15 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.

In Section 6.1, Fig. 14 demonstrated the results of the Edge Detection method when applied to the original teapot scene (Fig. 12). This eliminated the artifacts illustrated in Fig. 13 and generated a correctly blurred image. We will now use the adjacent pixel difference technique to generate a similar artifact-free blurred image, which is shown in Fig. 17.

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. 15. An example of a first degree difference map (right) resulting from applying a horizontal difference to the first 16 digits of π (left). © 2005 by B.A. Barsky.

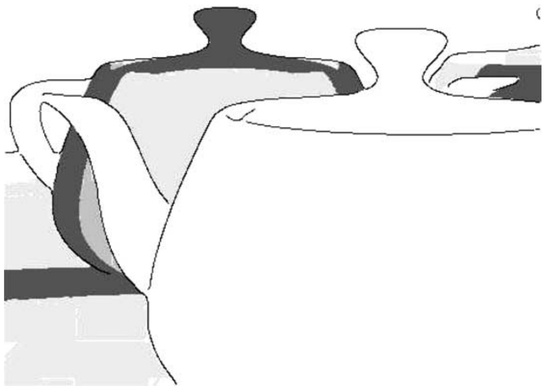


Fig. 16. Internal processing of the Adjacent Pixel Difference method. © 2005 by B.A. Barsky.

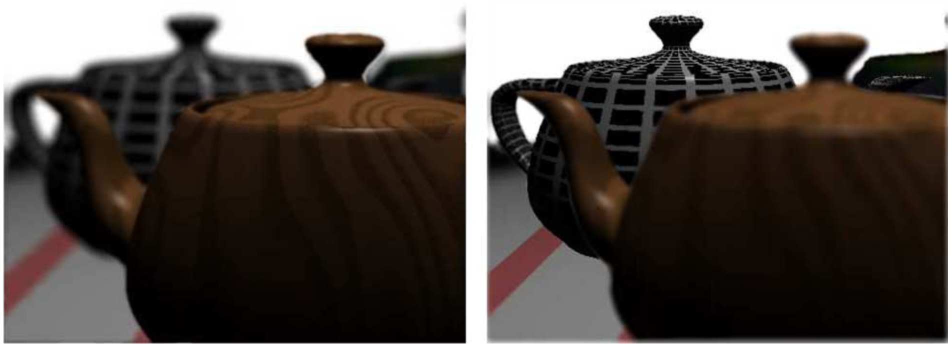
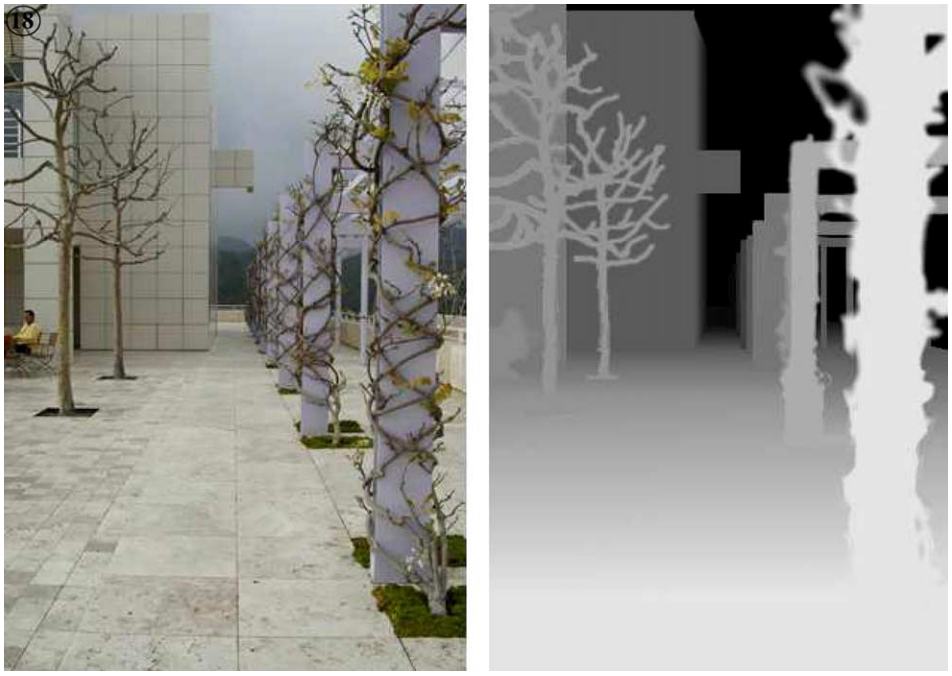


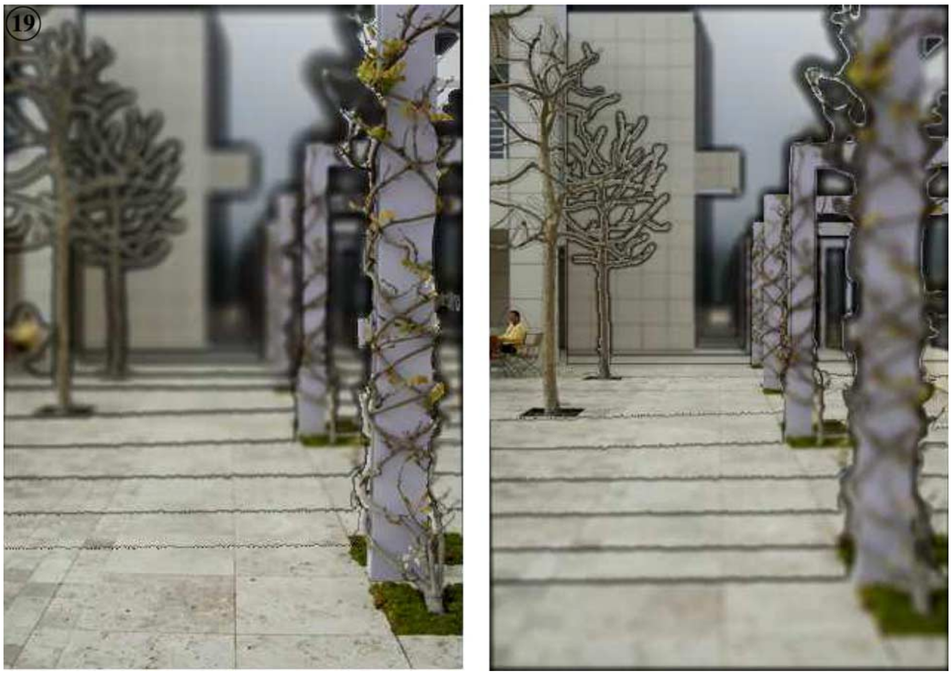
Fig. 17. Artifacts eliminated by the Adjacent Pixel Difference method for the teapot scene focused on the foreground (left) and background (right). © 2005 by B.A. Barsky.

Fig. 16 shows the operation of the Adjacent Pixel Difference method at a given depth, using similar coloring conventions as in Fig. 11 in Section 6.1, which illustrated the Edge Detection method. Dark gray corresponds to the current depth plane, and light gray represents the pixels added during the extension of the sub-image. Unlike the Edge Detection method which extends the current sub-image with respect to the edge curves, the Adjacent Pixel Difference method does so instead with respect to the groupings of object identifiers. The edge curves of the Edge Detection method are not used in the Adjacent Pixel Method. However, we display the black outline of the

Fig. 18. Original photograph of a plaza at the Getty museum (left) and its associated depth map (right). © 2005 by B.A. Barsky.

Fig. 19. Blurred Getty photograph without object identification focused on the foreground (left) and background (right). Black band artifacts appear at the locations where the sub-images are separated. © 2005 by B.A. Barsky.


18


19

teapots formed by the edge curves solely to facilitate a visual comparison of the two methods for extending the sub-image.

7. Analysis of results for object identification

Figs. 11 and 16 demonstrate that both methods can be subject to errors in their intermediate steps when extending a sub-image. The set of edges generated by the Edge Detection method as well as the groupings of object identifiers in the Adjacent Pixel Difference method do not always precisely correspond to the image geometry.

For example, errors arise in the Edge Detection method when the edge curves do not completely outline the object in question, leaving small gaps, as shown in Fig. 11. In this case, narrow streaks of pixels outside the object, corresponding to the extending rays that pass though the gaps between edge curves, are erroneously added to the current sub-image. A noisy depth map is sometimes the underlying cause of these types of errors. However, Canny edge detection specifically addresses noise concerns by using a Gaussian in the convolution with its input. This reduces the errors from noise sufficiently for our purposes.

Related problems arise when a noisy depth map is provided as input to the Adjacent Pixel Difference method. Here, pixels can be misidentified as belonging to one object when in fact the scene geometry would place them in a different object.

In either method, significant errors in object identification lead to artifacts that are similar to those occurring throughout Fig. 13, except that they will be local to the poorly identified object.

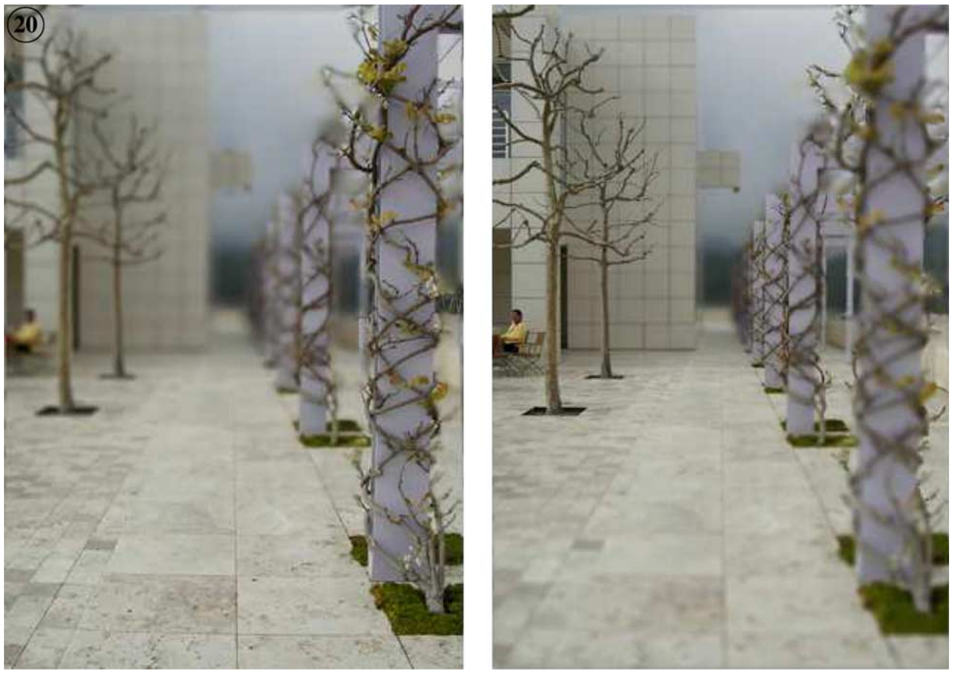
Nevertheless, our two methods prove to be sufficiently robust, since it is rare that errors in the intermediate steps of the sub-image extension process result in noticeable disturbances in the final result. Figs. 14 and 17 demonstrate such robustness for the Edge Detection method and Adjacent Pixel Difference method, respectively. Both of these two methods yield dramatically improved images in comparison to the significant artifacts that can occur without object identification (Fig. 13).

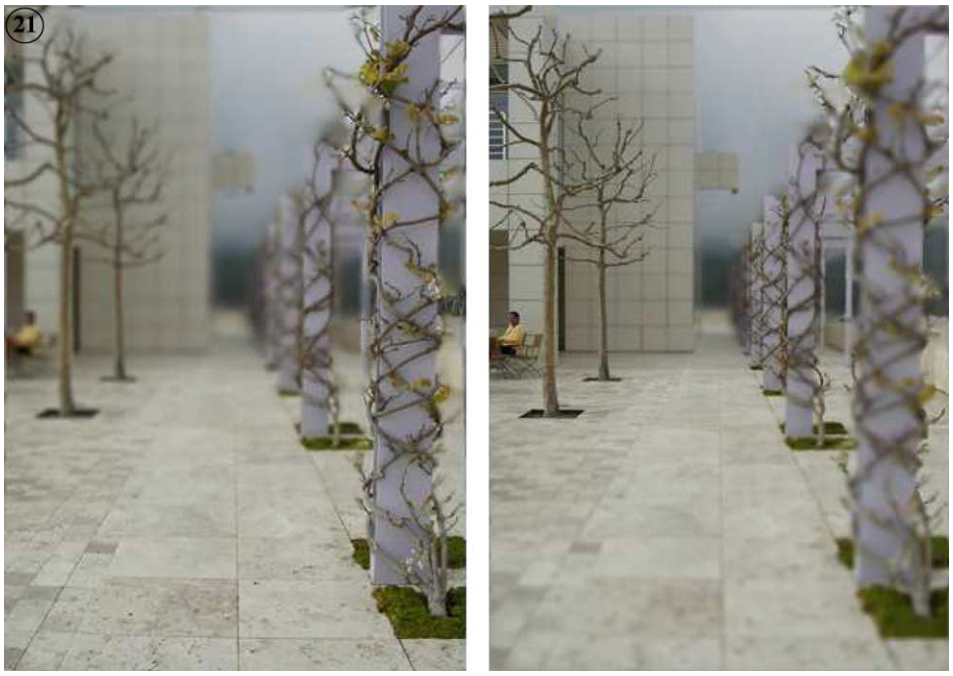
Our approaches are not limited to synthetic images. When applying our blur algorithm to photographs of real scenes (with associated depth maps), both of the object identification methods prevent artifacts. For example, in Fig. 18, the left image is a photograph of a plaza at the Getty museum in Los Angeles, with all depths in sharp focus, and the right image is its associated depth map. Fig. 19 shows the artifacts that occur without object identification. Our algorithm yields an image that is free of artifacts regardless of where the focus is set. Figs. 20 and 21 show the results of the Edge Detection method and Adjacent Pixel Difference method, respectively.

→

Fig. 20. Artifacts eliminated by the edge detection technique for the Getty image, focused on the foreground (left) and background (right). © 2005 by B.A. Barsky.

Fig. 21. Artifacts eliminated by the adjacent pixel difference technique for the Getty image, focused on the foreground (left) and background (right). © 2005 by B.A. Barsky.


20


21

8. 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 thank Adam W. Bargteil and Jeffrey A. Pang of the Computer Science Division of the University of California, Berkeley for their helpful comments, and Chintan Intwala for his work preparing images. We also thank Edwin E. Catmull, Christine Freeman, and Randy Nelson of Pixar Animation Studios for providing the original Tin Toy image.

References

[1] B.A. Barsky, Vision-realistic rendering: simulation of the scanned foveal image from wavefront data of human subjects, in: First Symposium on Applied Perception in Graphics and Visualization, co-located with ACM SIGGRAPH, Los Angeles, August 7–8, 2004, pp. 73–81.

[2] B.A. Barsky, A.W. Bargteil, D.D. Garcia, S.A. Klein, Introducing vision-realistic rendering, in: Paul Debevec, Simon Gibson (Eds.), Eurographics Rendering Workshop, Pisa, June 26–28, 2002, pp. 1–7.

[3] B.A. Barsky, D.R. Horn, S.A. Klein, J.A. Pang, M. 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), Lecture Notes in Computer Science (LNCS), Springer-Verlag, Berlin/Heidelberg.

[4] B.A. Barsky, D.R. Horn, S.A. Klein, J.A. Pang, M. 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), Lecture Notes in Computer Science (LNCS), Springer-Verlag, Berlin/Heidelberg.

[5] J.F. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (6) (1986) 679–698.

[6] R.L. Cook, Stochastic sampling in computer graphics, ACM Trans. Graph. 5 (1) (1986) 51–72.

[7] R.L. Cook, T. Porter, L. Carpenter, Distributed ray tracing, in: H. Christiansen (Ed.), ACM SIGGRAPH 1984 Conference Proceedings, Minneapolis, July 23–27, 1984, pp. 137–145.

[8] M.A.Z. Dippe, E.H. Wold, Antialiasing through stochastic sampling, in: B.A. Barsky (Ed.), ACM SIGGRAPH 1985 Conference Proceedings, San Francisco, July 22–26, 1985, pp. 69–78.

[9] W. Heidrich, P. Slusallek, H.-P. Seidel, An image-based model for realistic lens systems in interactive computer graphics, in: W.A. Davis, M. Mantei, R.V. Klassen (Eds.), Proceedings of Graphics Interface 1997, Canadian Human Computer Communication Society, May 1997, pp. 68–75.

[10] A. Isaksen, L. McMillan, S.J. Gortler, Dynamically reparameterized light fields, in K. Akeley (Ed.), Proceedings of ACM SIGGRAPH 2000, New Orleans, July 23–28, 2000, pp. 297–306.

[11] J.T. Kajiya, The rendering equation, in: ACM SIGGRAPH 1986 Conference Proceedings, Dallas, 1986, pp. 143–150.

[12] C. Kolb, D. Mitchell, P. Hanrahan, A realistic camera model for computer graphics, in: R.L. Cook (Ed.), ACM SIGGRAPH 1995 Conference Proceedings, Los Angeles, August 6–11, 1995, pp. 317–324.

[13] M.E. Lee, R.A. Redner, S.P. Uselton, Statistically optimized sampling for distributed ray tracing, in: B.A. Barsky (Ed.), ACM SIGGRAPH 1985 Conference Proceedings, San Francisco, July 22–26, 1985, pp. 61–67.

[14] M. Potmesil, I. Chakravarty, Synthetic image generation with a lens and aperture camera model, ACM Trans. Graph. 1 (2) (1982) 85–108 (Original version in ACM SIGGRAPH 1981 Conference Proceedings, August 1981, pp. 297–305).

[15] P. Rokita, Fast generation of depth-of-field effects in computer graphics, Comput. Graph. 17 (5) (1993) 593–595.

[16] M. Shinya, Post-filtering for depth of field simulation with ray distribution buffer, in: Proceedings of Graphics Interface '94, Banff, Alberta, May 1994, Canadian Information Processing Society, pp. 59–66.