Post-filtering for Depth of Field Simulation with Ray Distribution Buffer

Mikio Shinya
NTT Human Interface Laboratories
3-9-11 Midori-cho, Musashino-shi, Tokyo 180, Japan
email: shinya@nttarm.ntt.jp

Abstract

In real photography, focus control plays an important role in emphasizing the subject in the photo. In computer graphics, however, focus simulation, or depth of field simulation, is used only in a limited way because it is too expensive if super-sampling is used.

This paper proposes an efficient depth of field simulation method that can be realized as post-filtering. To deal with the partial occlusion of out-of-focus objects, the ray distribution buffer (RDB) is introduced. Z-buffering with the RDBs performs hidden surface removal for each distributed ray, as in distributed ray tracing. Experiments demonstrate that depth of field simulation is not an expensive process with the RDB method.

Keywords: Realistic Image Synthesis, Filtering, Depth of Field.

1 Introduction

The rendering process involves lighting simulation and imaging simulation. Although lighting models have been greatly improved, most systems still use the simplest camera model, the pin-hole camera model. With the pin-hole model, images are in focus everywhere on the screen. However, focus control is an important technique in real photography, especially in portraits to emphasize the subject.

Potmesil et al. first introduced a lens and aperture model into computer graphics to simulate focus effects, or depth of field [POTMESIL]. In his method, the intensity distribution, or the point spread function (the PSF), is calculated from the diffraction theory, and pin-hole camera images are filtered with the calculated PSFs. The advantage of the method is that the computational cost is independent of scene complexity (e.g., the number of polygons), since it can be implemented as post-filtering. The shortcoming is, however, that it cannot successfully simulate partial occlusion of out-of-focus objects because of its linear filtering feature. This problem limits its applications.

To solve the partial occlusion problem, Cook, et al., applied distributed ray tracing to the depth of field simulation [COOK84]. An almost equivalent process can be also performed by using accumulation buffers [HAEBERLI] and taking advantage of modern graphics hardware. This super-sampling approach realizes partial occlusion effects, but is computationally expensive with a cost proportional to the scene complexity and the number of samples.

In this paper, we propose a post-filtering method which approximates super-sampling methods at much lower cost. The method calculates the PSFs for each pixel in the original pin-hole image, like Potmesil's method. In addition, however, the proposed method also calculates the direction of out-of-focus rays. According to the direction, the color and depth of the rays are stored at each pixel in the form of a sub-pixel buffer, called the ray distribution buffer (RDB). Elements of RDB correspond to the sample rays in distributed ray tracing, and z-buffering of the RDBs can solve the occlusion problem as does the super-sampling approach. The computation cost of this method is still independent of scene complexity. Experiments demonstrate that the depth of field simulation is now feasible at reasonable computation cost.

2 Camera models

This section reviews three camera models, the pin-hole camera model, the diffraction model and the

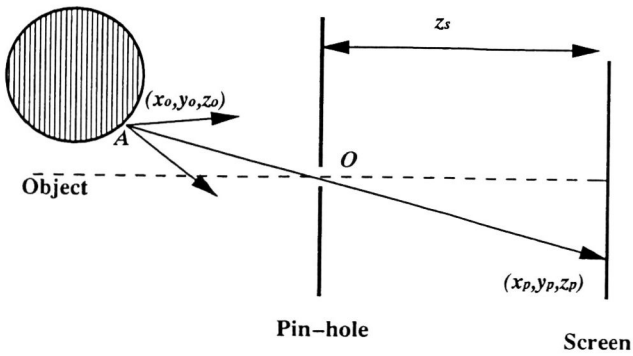


Figure 1: Pin-hole camera model.

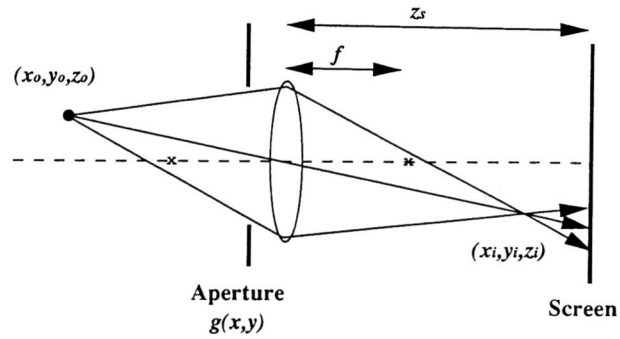


Figure 2: Thin lens imaging system.

geometric model. For the reason described below, we adopt the geometric model in our method.

Pin-hole camera model Figure 1 illustrates the pin-hole camera model. Among the rays reflected off the object point A, only ray OA goes through pin-hole O because the pin-hole is infinitely small. This is why the image is in focus everywhere. With the geometry shown in the figure, the image point (x_p, y_p, z_p) is described as

$$\begin{aligned} x_p &= x_o z_s / z_o, \\ y_p &= y_o z_s / z_o. \\ z_p &= z_s. \end{aligned} \tag{1}$$

When finite lenses are used in imaging systems, the depth of field effect is generally observed. There are two theories for the analysis, the diffraction theory and the geometric theory.

Geometric model Figure 2 shows a single ideal thin lens imaging system. The rays emitted (or reflected) from the object point (x_o, y_o, z_o) are focused on the image point (x_i, y_i, z_i) and then spread out onto the screen. From the imaging formula, we have

$$\begin{aligned} 1/z_i &= 1/f - 1/z_o \\ x_i &= x_o z_i / z_o, \\ y_i &= y_o z_i / z_o, \end{aligned} \tag{2}$$

where f is the focal length of the lens.

Since the imaging rays form a pyramidal cone with the aperture as its base, the imaging area on the screen is the intersection of the cone with the screen plane. The intensity distribution on the screen, or the PSF, is

$$\begin{aligned} h(x_s, y_s) &= g(Mx_s - (M-1)x_i, My_s - (M-1)y_i), \\ M &= z_i/(z_s - z_i), \end{aligned} \tag{3}$$

where $g(x, y)$ represents the aperture transmittance. For example,

$$g(x,y) = \begin{cases} 1 & \text{if } x^2 + y^2 \le R^2, \\ 0 & \text{otherwise,} \end{cases}$$

for a circular aperture with radius R. In this case, the boundary of the support of the function h is a circle, the *circle of confusion*.

Diffraction model Figure 3 shows diffraction in a lens imaging system. The spherical wave emitted from object point (x_o, y_o, z_o) is diffracted by the aperture. With the Fresnel approximation[1], the intensity at (x_s, y_s, z_s) is represented as

$$\begin{aligned} h(x_s, y_s) \propto & \Big| \int\int \exp\{2\pi\imath/\lambda \cdot \\ & (\alpha(x^2 + y^2)/2 + (xx_s + yy_s)/z_s)\} \\ & g(x,y)dxdy \Big|^2, \end{aligned} \tag{4}$$

where $g(x, y)$ is the aperture and

$$\alpha = 1/z_o + 1/z_s - 1/f.$$

The major difference from the geometric PSF (Eq. 3) is the ringing nature of diffraction. As seen in Eq. 4, the diffraction pattern strongly depends on wavelength λ. Thus, the diffraction effect

[1]The Fresnel approximation is a second order approximation for far fields and neglects the third order terms such as $O((x/z_s)^3)$ and so on. By the way, the Fraunhofer approximation is a first order approximation.

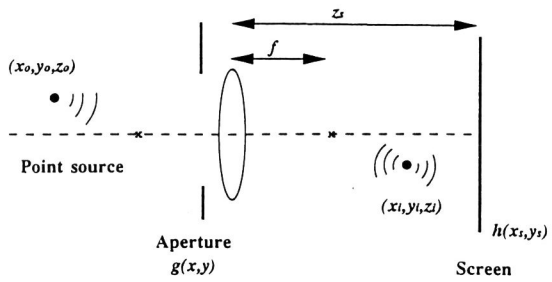


Figure 3: Diffraction model.

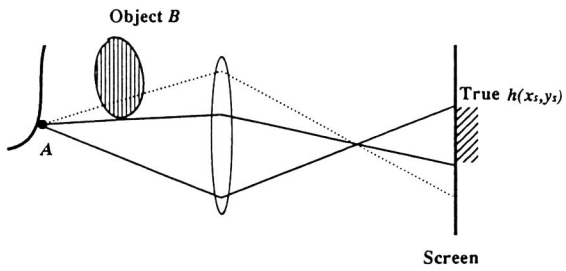


Figure 4: Partial occlusion.

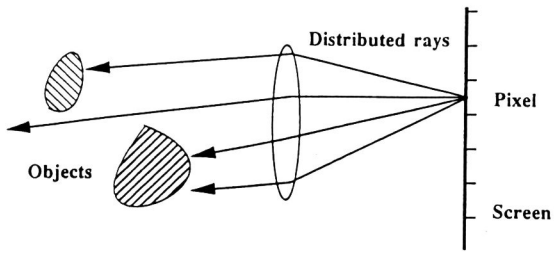


Figure 5: Distributed ray tracing.

is more significant for monochromatic light, typically, laser light. Fortunately, however, since natural light usually has broad spectral distribution, the diffraction fringes are hardly noticeable in reality. For this reason, we adopted the geometric model to simulate the depth of field.

3 Previous Methods

Linear filtering When neglecting occlusion effects, the imaging process is a linear optical process, and thus, can be simulated by linear filtering operation, as proposed by Potmesil [POTMESIL].

This method is a two-pass method. The first pass is the usual rendering process, such as ray tracing and z-buffering, producing (pin-hole camera) image $I_0(x_p, y_p)$ and its depth image $z_o(x_p, y_p)$. In the second pass, the diffraction pattern h is calculated from z-values according to the geometric model or diffraction model. Object point (x_o, y_o, z_o) can be calculated from Eq. 1 for $(x_p.y_p)$ and the PSF h can be calculated by Eq. 3 or Eq. 4. The linear filtering

$$I(x_s, y_s) = \int h(x_s, y_s) I_0(x, y) dx dy \qquad (5)$$

results in images that appear to have a depth of field.

The advantage of this method is that the computational cost is independent of scene complexity. Its shortcoming is its neglect of partial occlusion. As shown in Figure 4, some rays emitted from Point A are occluded by Object B, which affects the intensity distribution pattern. This causes serious artifacts especially with objects in focus. Plate 1 shows an example. In the figure, a thin black rectangle is located in front of a textured polygon. The black rectangle, which is in focus, looks partially 'transparent' in Plate 1-b, which is indeed unacceptable.

Distributed ray tracing To solve the partial occlusion problem, distributed ray tracing was applied to simulate depth of field [COOK84]. Similar computation can be also achieved by the accumulation buffer with z-buffering [HAEBERLI], which can take advantage of advanced graphics hardware.

In this approach, several sample rays passing through the lens are traced from each pixel (Figure 5), and the pixel value is determined by the average of the intensity of the distributed rays. Since hidden surface removal is achieved in every ray tracing, the partial occlusion effects are taken into account. Plate 1-c shows the result of the accumulation buffer method. The partial occlusion artifact is completely solved here.

To reduce aliasing artifacts, the stochastic sampling technique [DIPPE,COOK86] can be applied. The disadvantage of this method is, however, its high computational cost, which is proportional to scene complexity and the number of samples.

4 RDB Method

Basic idea The key to dealing with occlusion is to classify imaging rays according to their direction and to apply hidden surface removal, as in distributed ray tracing. For this purpose, we introduce a sub-pixel structure wherein each sub-pixel element represents a sample distribution ray, as shown in Figure 6-a. Let us call this sub-pixel buffer the *ray distribution buffer* (RDB). Each RDB element stores an rgb value and a z-value.

Consider the situation shown in Figure 6-b. The rays emitted from object point (x_o, y_o, z_o) are focused on (x_i, y_i, z_i) and then spread out over the screen. At (x_s, y_s), the incoming ray direction (s_x, s_y, s_z) can be calculated by

$$\begin{aligned} s_x &= (x_s - x_i)/d, \\ s_y &= (y_s - y_i)/d, \\ s_z &= \sqrt{1 - s_x^2 - s_y^2}, \end{aligned} \tag{6}$$

where

$$d = \sqrt{(x_s - x_i)^2 + (y_s - y_i)^2 + (z_s - z_i)^2}.$$

With the RDB, z-buffering is applied to these incoming rays to solve the partial occlusion problem. According to the ray direction (Eq. 6), the corresponding RDB elements are calculated. If z_o is smaller than the stored z-value, the z- and rgb-values are substituted, just as in conventional z-buffering.

The RDB elements for the imaging rays can be determined as in Figure 6-c. The ray direction at the four corners of pixel $\mathbf{s}_i$ $(i = 0, 1, 2, 3)$ are calculated by Eq. 6 (Figure 6-c). The RDB elements whose directions lie in the quadrateral $\mathbf{s}_0 - \mathbf{s}_1 - \mathbf{s}_2 - \mathbf{s}_3$ are determined as the corresponding elements (Figure 6-d).

Jittered sampling can be also achieved by assigning a jittered offset δs[ix][iy] to sample ray directions at each pixel in the pre-processing phase. When determining the corresponding RDB elements, $\mathbf{s}_i + \delta$s[ix][iy] is used instead of $\mathbf{s}_i$ (Figure 6-e).

Procedure The procedure can be summarized in the following way.

1) Render the pin-hole camera image rgb[ix][iy] and the z-image z[ix][iy] by a conventional method.

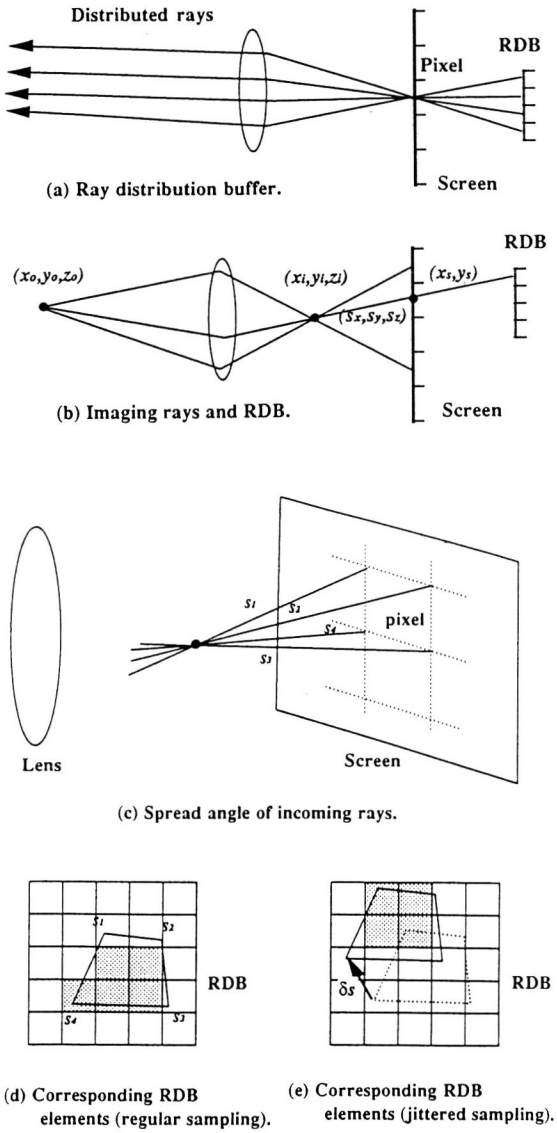


Figure 6: Ray distribution buffer.

Table 1: CPU time comparison.

z_f	3 (meter)	5	17
RDB method	212.7 (sec)	117.9	155.0
ACC method	1321 (sec)	1321	1321

2) Initialize all RDBs.

3) When jittered sampling is applied, set jittered offset δs[ix][iy].

4) For all pixels (ix, iy):

   i) Calculate the object point (x_o, y_o, z_o) and the circle of confusion according to Eqs. 1, 2 and 3.

   ii) For all pixels in the circle of confusion:

      a) Calculate the ray direction according to Eq. 6 and determine the corresponding RDB elements.

      b) For all the corresponding elements, if z[ix][iy] is smaller than the stored z-value, replace the rgb and z-values with z[ix][iy] and rgb[i][iy].

5) For each pixel, calculate the average of the rgb-values in its RDB to yield the final image.

5 Experiment and discussion

Partial occlusion Figure Plate 1-d shows the simulation result of the RDB method using the same parameters as employed in Plate 1-b. The partial occlusion artifacts observed in Figure Plate 1-b are completely suppressed as in Plate 1-c.

Comparison with the accumulation buffer method Depth of field was simulated by the RDB method and the accumulation buffer method to allow a comparisons to be made. The parameter z_f is the z-value of objects in focus. The F-number (f/R) was fixed at 1.0. The image resolution is 640×480 and the RDB resolution (super-sampling rate) is 9×9. As seen in Plate 2, the results of the two methods are almost identical, as theoretically expected. Table 1 shows the required computation time on an IRIS Crimson R4400 (150 MHz) RealityEngine.

Computation time versus the RDB resolution The computation time versus the RDB resolution is plotted for $z_f = 3$m and $z_f = 1$m using the same scene as in Figure 7. The image resolution is 256×256. As shown in the figure, the increase of the CPU time with the RDB resolution is very slow, particularly for $z_f = 1$. The computational cost is proportional to the total number of the processed RDB elements. In the worst case, the filter kernel size is very large, and thus, the number of RDB elements per imaging ray is nearly one. In this case, the computational cost become $O(n \times m \times \bar{\delta_r})$, where $n \times m$ is the image resolution and $\bar{\delta_r}$ is the average filter kernel size. Thus, the worst case cost is independent of scene complexity and the RDB resolution. This is why the computation time is almost constant for $z_f = 1$ in Figure 7. For reference, the computation time and the super-sampling rate is also shown for the accumulation buffer method.

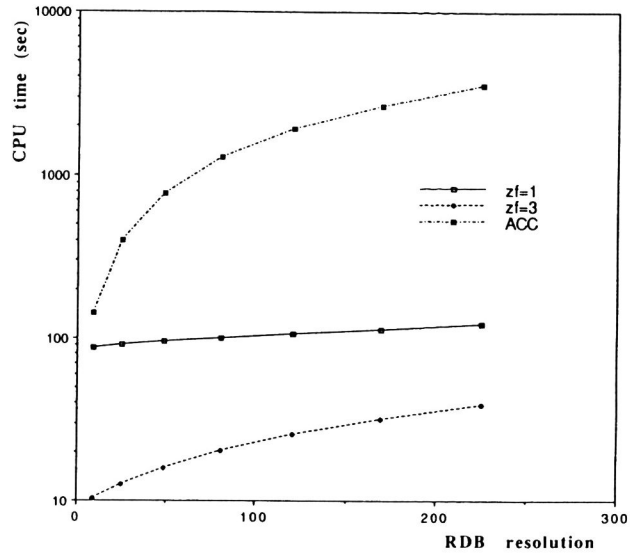


Figure 7: RDB resolution versus CPU time. In the figure, ACC indicates the accumulation buffer methods.

6 Conclusion

A new method for depth of field simulation has been developed. This method is a post-filtering process as is Potmesil's method, but the partial occlusion artifacts are reduced by introducing the ray distribution buffer (RDB). Its efficiency was demonstrated by experiments, and the worst case analysis shows that the computation cost is independent of scene complexity and the RDB resolution. With this method, depth of field simulation

can be achieved at a reasonable cost.

Acknowledgments

The author would like to thank the GI reviewers for their valuable comments. He also wishes to thank Takashi Sakai and Kazuyoshi Tateishi for their administrative supports, Atsushi Kajiyama for his help in preparing the paper, Toki Takahashi, Takafumi Saito, and Toshi Tanaka for a helpful discussion.

References

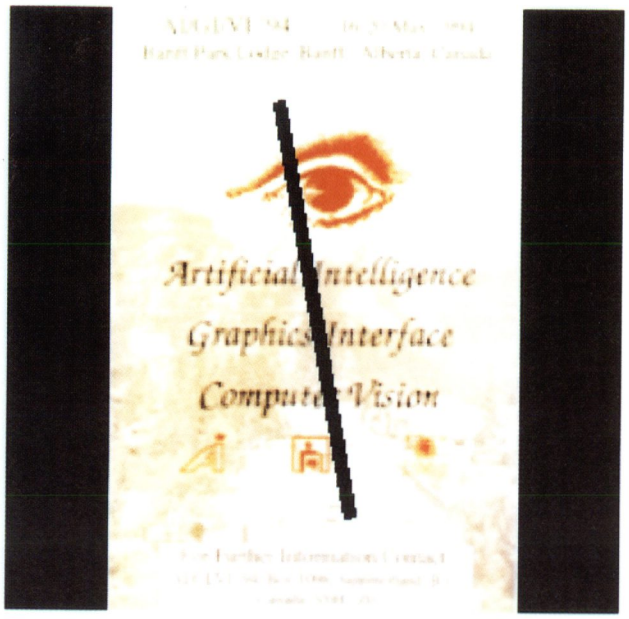
[COOK84] R. L. Cook, T. Porter, L. Carpenter, 'Distributed Ray Tracing', Computer Graphics **18**, No.3, pp.137-145, 1984.

[COOK86] R. L. Cook, 'Stochastic Sampling in Computer Graphics', ACM Trans. Graphics, **5**, No.1, pp.51-57, 1986.

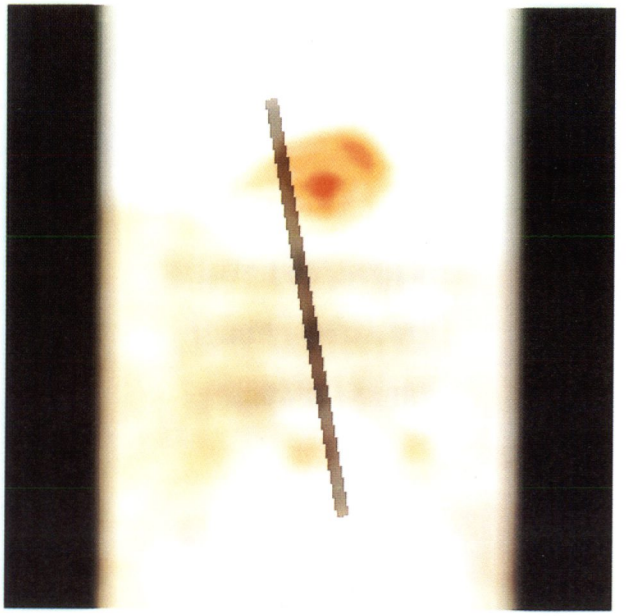
[DIPPE] M. A. Dippé, 'Anti-aliasing through Stochastic Sampling', Computer Graphics **19**, No.3, pp.69-78, 1985.

[HAEBERLI] P. Haeberli, Kurt Akeley, 'The accumulation buffer: Hardware support for high-quality rendering', Computer Graphics **24**, No. 4, pp. 309-318, 1990.

[POTMESIL] M. Potmesil, I. Chakravarty, 'A lens and aperture camera model for synthetic image generation', Computer Graphics **15**, No. 3, pp. 297-305, 1981.

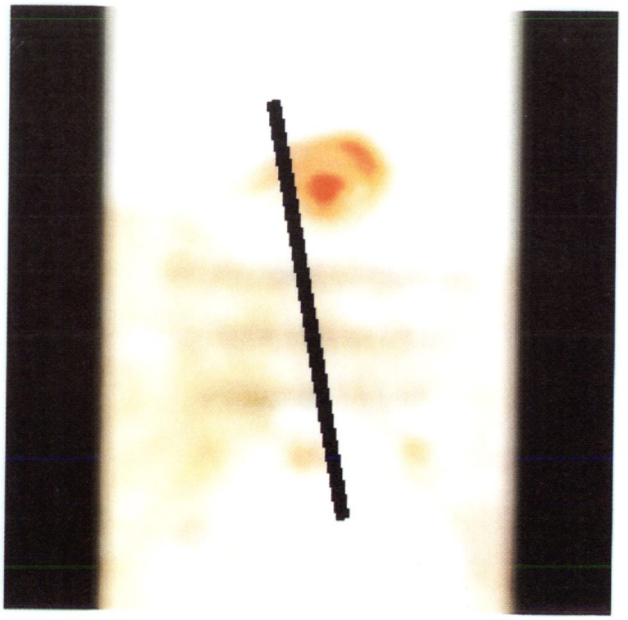


(a) Pin-hole camera image.



(b) Linear filtering method.

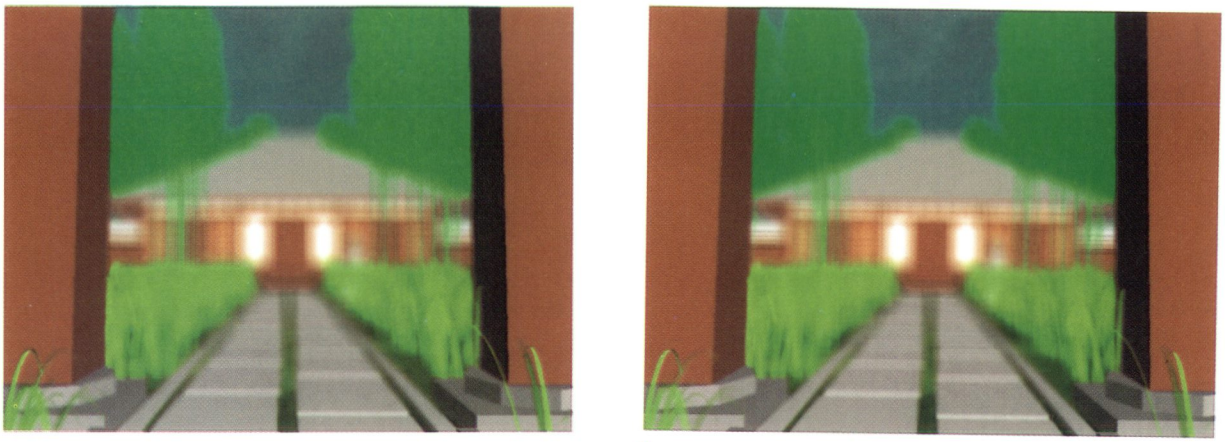
(c) Accumulation buffer method.



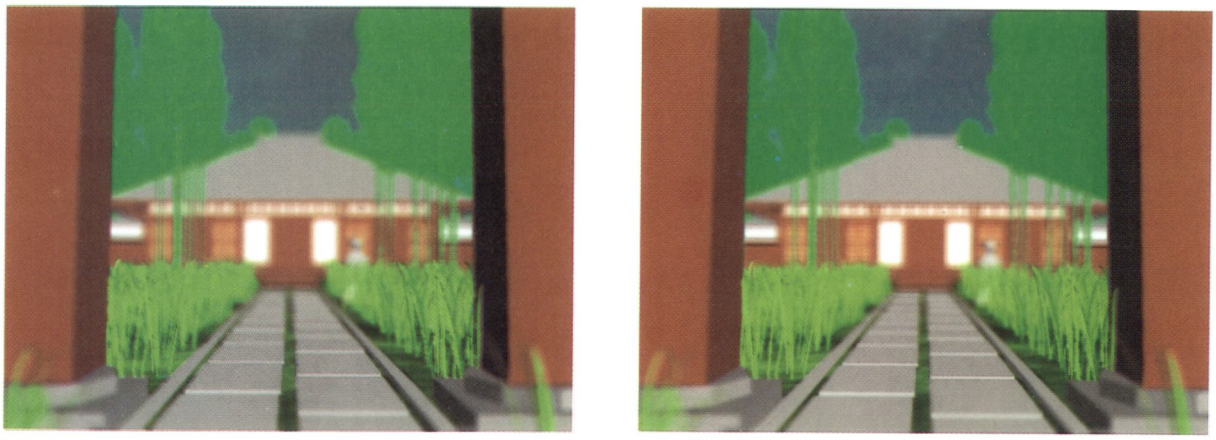
(d) RDB method.

Plate 1: Partial occlusion.

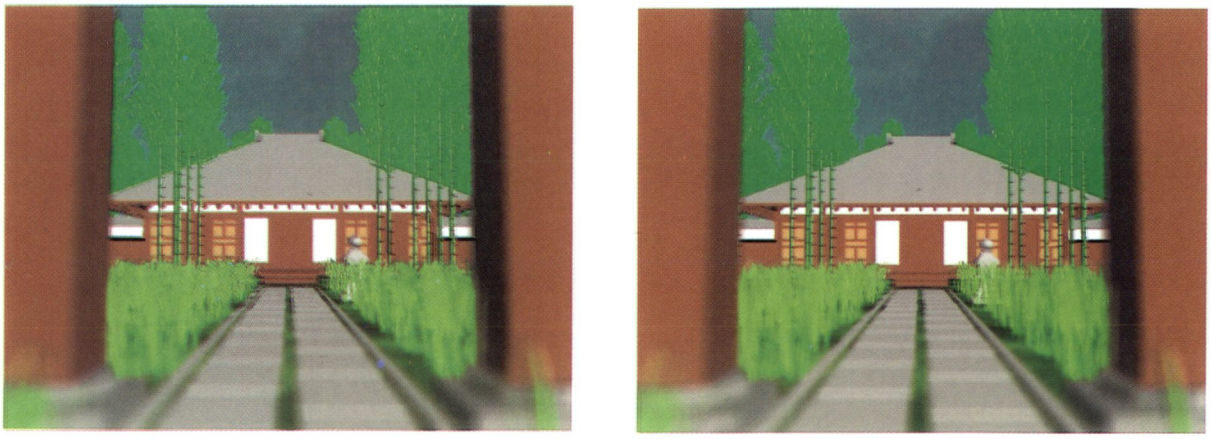
RDB method **Accumulation buffer method**



$z_f = 3m$



$z_f = 5m$



$z_f = 17m$

Plate 2: Comparison between the RDB method and the accumulation buffer method.