Stochastic Sampling in Computer Graphics

ROBERT L. COOK
Pixar

Ray tracing, ray casting, and other forms of point sampling are important techniques in computer graphics, but their usefulness has been undermined by aliasing artifacts. In this paper it is shown that these artifacts are not an inherent part of point sampling, but a consequence of using regularly spaced samples. If the samples occur at appropriate nonuniformly spaced locations, frequencies above the Nyquist limit do not alias, but instead appear as noise of the correct average intensity. This noise is much less objectionable to our visual system than aliasing. In ray tracing, the rays can be stochastically distributed to perform a Monte Carlo evaluation of integrals in the rendering equation. This is called *distributed ray tracing* and can be used to simulate motion blur, depth of field, penumbrae, gloss, and translucency.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

General Terms: Algorithms

Additional Key Words and Phrases: Antialiasing, filtering, image synthesis, Monte Carlo integration, motion blur, raster graphics, ray tracing, sampling, stochastic sampling

1. INTRODUCTION

Because pixels are discrete, computer graphics is inherently a sampling process. The pixel size determines an upper limit to the frequencies that can be displayed. This limit, one cycle every two pixels, is called the *Nyquist limit*. An attempt to display frequencies greater than the Nyquist limit can produce aliasing artifacts, such as "jaggies" on the edges of objects [6], jagged highlights [26], strobing and other forms of temporal aliasing [19], and Moiré patterns in textures [6]. These artifacts are tolerated in some real-time applications in which speed is more vital than beauty, but they are unacceptable in realistic image synthesis.

Rendering algorithms can be classified as *analytic* or *discrete* according to how they approach the aliasing problem. Analytic algorithms can filter out the high frequencies that cause aliasing before sampling the pixel values. This filtering tends to be complicated and time consuming, but it can eliminate certain types of aliasing very effectively [3, 6, 8, 9, 15]. Discrete algorithms, such as ray tracing,

This research was done when Pixar was the computer division of Lucasfilm Ltd. An earlier version of this paper was prepared as an unpublished Lucasfilm Technical Memo #94, "Antialiased Point Sampling," 1983.

Author's address: Pixar, P.O. Box 13719, San Rafael, CA 94913-3719.

only consider the image at regularly spaced sample points. Since they ignore everything not at these points, they appear by their nature to preclude filtering the image. Thus they are plagued by seemingly inherent aliasing artifacts. This is unfortunate, for these algorithms are much simpler, more elegant, and more amenable to hardware implementation than the analytic methods. They are also capable of many features that are difficult to do analytically, such as shadows, reflection, refraction [13, 24], constructive solid geometry [21], motion blur, and depth of field [5].

There are two existing discrete approaches to alleviating the aliasing problem: *supersampling* and *adaptive sampling*. Supersampling involves using more than one regularly spaced sample per pixel. It reduces aliasing by raising the Nyquist limit, but it does not eliminate aliasing. No matter how many samples are used, there are still frequencies that will alias. In adaptive sampling, additional rays are traced near edges [24]; the additional rays are traced midway between previously traced rays. Unlike supersampling, this approach can antialias edges reliably, but it may require a large number of rays, and it complicates an otherwise simple algorithm.

In this paper a new discrete approach to antialiasing called *stochastic sampling* is presented. Stochastic sampling is a Monte Carlo technique [11] in which the image is sampled at appropriate nonuniformly spaced locations rather than at regularly spaced locations. This approach is inherently different from either supersampling or adaptive sampling, though it can be combined with either of them. Stochastic sampling can eliminate all forms of aliasing, including unruly forms such as highlight aliasing.

With stochastic sampling, aliasing is replaced by noise of the correct average intensity. Frequencies above the Nyquist limit are still inadequately sampled, and they still appear as artifacts in the image. But a highly objectionable artifact (aliasing) is replaced with an artifact that our visual systems tolerate very well (noise).

In addition to providing a solution to the aliasing problem, stochastic sampling also provides new capabilities for discrete algorithms such as ray tracing. The physical equations simulated in the rendering process involve integrals over time, lens area, specular reflection angle, etc. Image-synthesis algorithms have usually avoided performing these integrals by resorting to crude approximations that assume instantaneous shutters, pinhole cameras, mirror or diffuse reflections, etc. But these integrals can be easily evaluated by stochastically sampling them, a process called Monte Carlo integration. In a ray-tracing algorithm, this involves stochastically distributing the rays in time, lens area, reflection angle, etc. This is called *probabilistic* or *distributed ray tracing* [5]. Distributed ray tracing allows the simulation of fuzzy phenomena, such as motion blur, depth of field, penumbrae, gloss, and translucency.

2. UNIFORM POINT SAMPLING

Before discussing stochastic sampling, we first review uniform sampling and the source of aliasing. In a point-sampled picture, frequencies greater than the Nyquist limit are inadequately sampled. If the samples are uniformly spaced,

these frequencies can appear as aliases, that is, they can appear falsely as low frequencies [4, 17, 20].

To see how this happens, consider for the moment one-dimensional sampling; we refer to the dimension as time. Let a signal $f(t)$ be sampled at regular intervals of time, that is, at times nT for integer n, where T is the time period between samples, so that $1/T$ is the sampling frequency. The Nyquist limit is half the sampling frequency, or $0.5/T$. This sampling is equivalent to multiplication by the *shah* function $\mathrm{III}(t/T)$, where

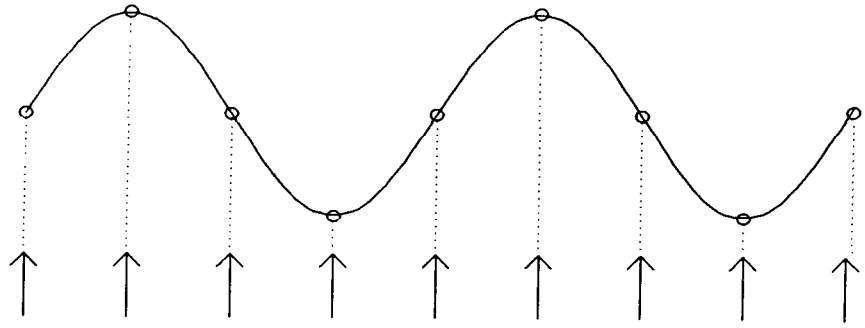
$$\mathrm{III}(x) = \sum_{n=-\infty}^{\infty} \delta(x - n),$$

where δ is the Kronecker delta function. After sampling, information about the original signal $f(t)$ is preserved only at the sample points. The sampling theorem states that, if $f(t)$ contains no frequencies above the Nyquist limit, then sampling followed by an ideal reconstruction filter reproduces the original signal $f(t)$ exactly.

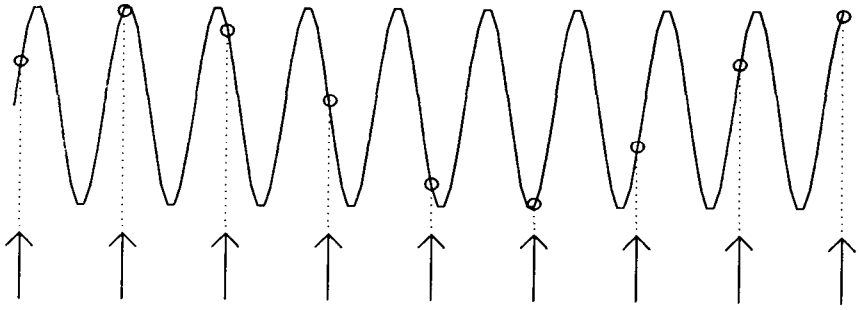
This situation is shown in Figure 1 for a sine wave. In Figure 1a, the frequency of the sine wave is below the Nyquist limit of the samples, and the sampled values accurately represent the function. But, in Figure 1b, the frequency of the sine wave is above the Nyquist limit of the samples. The sampled values do not accurately represent the sampled sine wave; instead they look as if they came from a low-frequency sine wave. The high-frequency sine wave appears incorrectly under the alias of this low-frequency sine wave.

Figure 2 shows this situation in the frequency domain. The Fourier transform of f is denoted by F; the Fourier transform of the shah function $\mathrm{III}(t/T)$ is another shah function $(1/T)\mathrm{III}(tT)$. Figure 2a shows the Fourier transform of the signal in Figure 1a, a single sine wave whose frequency is below the Nyquist limit. Sampling involves convolving the signal with the sampling grid of Figure 2b to produce the spectrum shown in Figure 2c. An ideal reconstruction filter, shown in Figure 2d, would extract the original signal, as in Figure 2e. In Figures 2f–2j, the same process is repeated for the signal in Figure 1b, a single sine wave whose frequency is above the Nyquist limit. In this case, the sampling process can fold the high-frequency sine wave into low frequencies, as shown in Figure 2h. These false frequencies, or aliases, cannot be separated from frequencies that are a part of the original signal. The part of the spectrum extracted by the reconstruction filter contains these aliases, as shown in Figure 2j.

Sampling theory thus predicts that, with a regular sampling grid, frequencies greater than the Nyquist limit can alias. The inability to reproduce those frequencies is inherent in the sampling process, but their appearance as aliases is a consequence of the regularity of the sampling grid. If the sample points are not regularly spaced, the energy in those frequencies can appear as noise, an artifact that is much less objectionable than aliasing. In the case of uniform sampling, aliasing is precisely defined; in the case of nonuniform sampling, we use the term aliasing to mean artifacts with distinct frequency components, as opposed to noise.



(a) Point sampling within the Nyquist limit



(b) Point sampling beyond the Nyquist limit

Fig. 1. Point sampling shown in the spatial domain. The arrows indicate the sample locations, and the circles indicate the sampled values. In (a), the sine wave frequency is within the Nyquist limit, so the sampled values accurately represent the signal. In (b), the sine wave frequency is above the Nyquist limit, and the sampled values incorrectly represent a low-frequency sine wave that is not present in the signal.

3. POISSON DISK SAMPLING

An excellent example of a nonuniform distribution of sample locations is found in the human eye. The eye has a limited number of photoreceptors, and, like any other sampling process, it has a Nyquist limit. Yet our eyes are not normally prone to aliasing [25]. In the fovea, the cells are tightly packed in a hexagonal pattern, and aliasing is avoided because the lens acts as a low-pass filter. Outside of the fovea, however, the cells are further apart and thus the sampling rate is lower, so we might expect to see aliasing artifacts. In this region, aliasing is avoided by a nonuniform distribution of the cells.

The distribution of cones in the eye has been studied by Yellott [27]. Figure 3a is a picture of the distribution of cones in an extrafoveal region of the eye of a rhesus monkey, which has a photoreceptor distribution similar to that in the human eye. Yellott took the optical Fourier transform of this distribution, with the result shown in Figure 3b. This distribution is called a *Poisson disk distribution*, and it is shown schematically in the frequency domain in Figure 4b. There is a spike at the origin (the dc component) and a sea of noise beyond the Nyquist

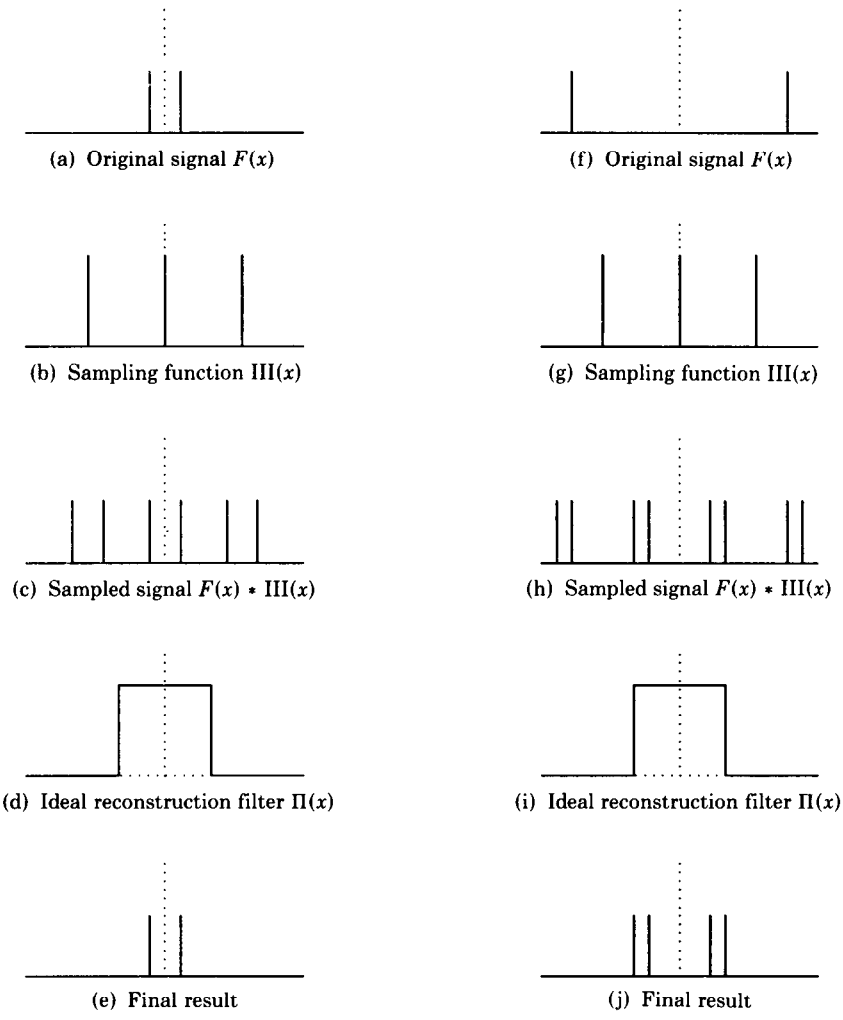


Fig. 2. Point sampling shown in the frequency domain. The original signal $F(x)$ is convolved with the sampling grid $\mathrm{III}(x)$, and the result is multiplied by an ideal reconstruction filter $\Pi(x)$. The process is shown for a sine wave with a frequency below the Nyquist limit in (a)–(e) and above the Nyquist limit in (f)–(j).

limit. In effect, the samples are randomly placed with the restriction that no two samples are closer together than a certain distance.

Now let us analyze point sampling using a Poisson disk sampling distribution instead of a regular grid. Figure 4a shows a signal that is a single sine wave whose frequency is below the Nyquist limit. Convolution with the Poisson sampling grid of Figure 4b produces the spectrum in Figure 4c. The ideal reconstruction filter of Figure 4d would extract the original signal, Figure 4e. Figure 4f shows a sine wave whose frequency is above the Nyquist limit. Convolution with the Poisson sampling grid produces the spectrum in Figure 4h. An ideal reconstruc-

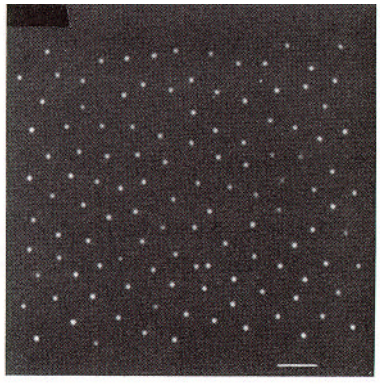


Fig. 3a. Monkey eye photoreceptor distribution.

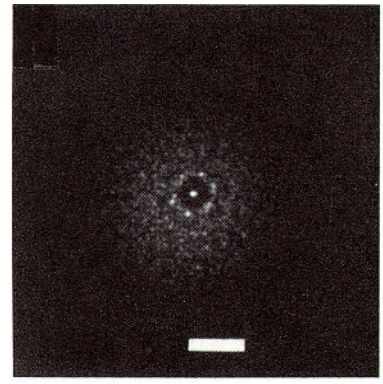
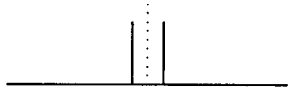
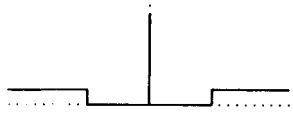


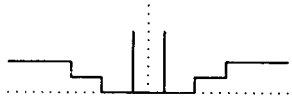
Fig. 3b. Optical transform of monkey eye.



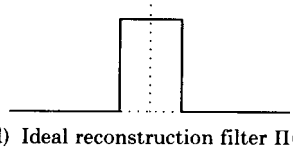
(a) Original signal $F(x)$



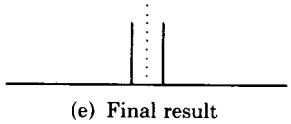
(b) Sampling function $P(x)$



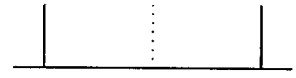
(c) Sampled signal $F(x) * P(x)$



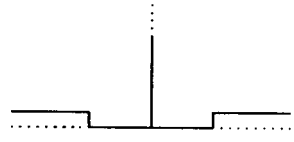
(d) Ideal reconstruction filter $\Pi(x)$



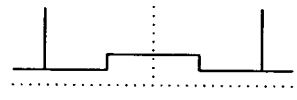
(e) Final result



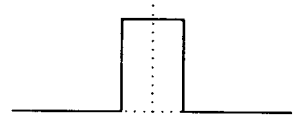
(f) Original signal $F(x)$



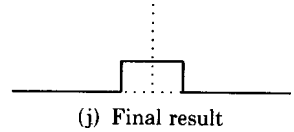
(g) Sampling function $P(x)$



(h) Sampled signal $F(x) * P(x)$



(i) Ideal reconstruction filter $\Pi(x)$



(j) Final result

Fig. 4. Poisson disk sampling shown in the frequency domain.

tion filter would extract noise, as shown in Figure 4j. This noise replaces the aliasing of Figure 2j.

The minimum distance restriction decreases the magnitude of the noise. For example, film grain appears to have a random distribution [23], but without the minimum distance restriction of a Poisson disk distribution. With a purely random distribution, the samples tend to bunch up in some places and leave large gaps in other places. Film does not alias, but it is more prone to noise than the eye.

One possible implementation of Poisson disk sampling to image rendering is straightforward, though expensive. A lookup table is created by generating random sample locations and discarding any locations that are closer than a certain distance to any of the locations already chosen. Locations are generated until the sampling region is full. Filter values that describe how each sample affects the neighboring pixels are calculated, and these filter values must be normalized. The locations and filter values are stored in a table. This method would produce good pictures, but it would also require a large lookup table. An alternative method, jittering a regular grid, is discussed in the next section.

4. JITTERING A REGULAR GRID

4.1 Theory

Jittering, or adding noise to sample locations, is a form of stochastic sampling that can be used to approximate a Poisson disk distribution. There are many types of jitter; among these is additive random jitter, which can eliminate aliasing completely [22]. But the discussion in this paper is limited to one particular type of jitter: the jittering of a regular grid. This type of jitter produces good results and is particularly well suited to image-rendering algorithms.

The Fourier transform of a jittered grid (shown later in Figure 11b) is similar to the Fourier transform of a Poisson disk distribution (shown in Figure 4b). An analysis like that in Figures 2 and 4 shows that the results are not quite so good as those obtained with Poisson disk sampling. The images are somewhat noisier and some very small amount of aliasing can remain. We now look at this noise and aliasing quantitatively.

Jitter was analyzed in one dimension (time) by Balakrishnan [2], who calculated the effect of *time jitter*, in which the nth sample is jittered by an amount ζ_n so that it occurs at time $nT + \zeta_n$, where T is the sampling period (see Figure 5a). If the ζ_n are uncorrelated, Balakrishnan reports that jittering has the following effects:

—High frequencies are attenuated.

—The energy lost to the attenuation appears as uniform noise. The intensity of the noise equals the intensity of the attenuated part of the signal.

—The basic composition of the spectrum otherwise does not change.

Sampling by itself cannot be regarded as a filter, because sampling is not a linearly shift-invariant process. Balakrishnan showed, however, that the combination of jittered sampling plus an ideal reconstruction filter is a linearly shift-invariant process, even though the sampling by itself is not [2], so it is in this context that we can talk about frequency attenuation.

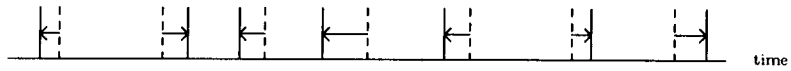


Fig. 5a. Time jitter. Regularly spaced sample times are shown as dashed lines, and the corresponding jittered times are shown as solid lines. Each sample time is jittered by an amount ζ so that the nth sample occurs at time $nT + \zeta_n$ instead of at time nT, where T is the sample period.

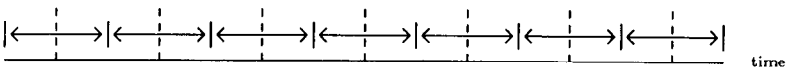


Fig. 5b. White noise jitter for $\gamma = 0.5$. Regularly spaced samples, shown as dashed lines, are jittered so that every time has an equal chance of being sampled.

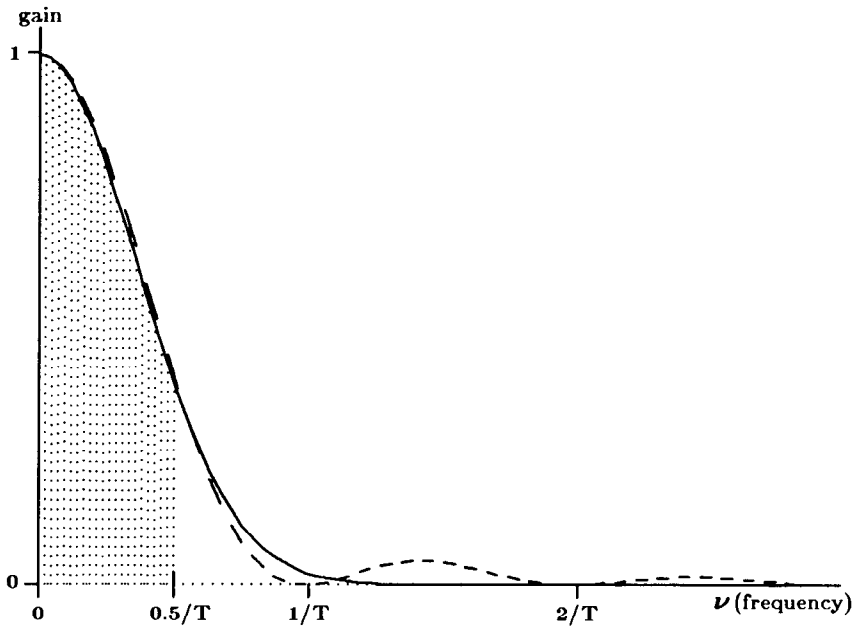


Fig. 6. Attenuation due to jitter. The broken line shows the filter for white noise jitter, the solid line for Gaussian jitter. The shaded area is inside the Nyquist limit.

Uncorrelated jitter is jitter in which any two jitter amounts ζ_n and ζ_m are uncorrelated. Balakrishnan analyzed two types of uncorrelated jitter: *Gaussian jitter* and *white noise jitter.* For Gaussian jitter, the values of ζ are chosen according to a Gaussian distribution with a variance of σ^2. The gain as a function of frequency ν is then

$$e^{-(2\pi\nu\sigma)^2}. \tag{1}$$

This function is plotted with a solid line in Figure 6 for $\sigma = T/6.5$. With white noise jitter, the values of ζ are uniformly distributed between $-\gamma T$ and γT (see

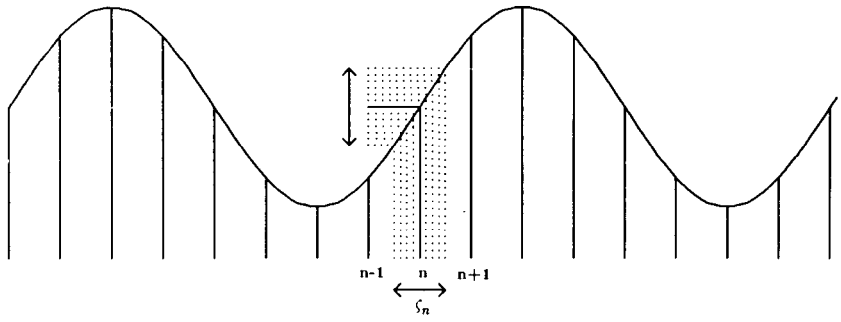


Fig. 7a. The effect of white noise jitter on a sine wave with a frequency below the Nyquist limit. Sample n occurs at a random location in the dotted region. The jitter indicated by the horizontal arrow results in a sampled value that can vary by the amount indicated by the vertical arrow.

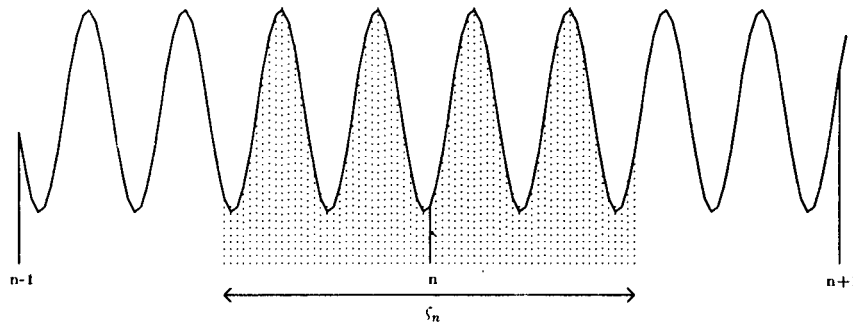


Fig. 7b. The effect of white noise jitter on a sine wave with a frequency above the Nyquist limit. The jitter indicated by the horizontal arrow results in a sampled value that is almost pure noise.

Figure 5b). The gain in this case is

$$\left[\frac{\sin(2\pi\gamma\nu T)}{2\pi\gamma\nu T}\right]^2, \tag{2}$$

as shown with a dashed line in Figure 6 for $\gamma = \frac{1}{2}$.

From this we can see that jittering a regular grid does not eliminate aliasing completely, but it does reduce it substantially. The Nyquist limit of $0.5/T$ is indicated in the figure by the shaded area. Notice that the width of the filter can be scaled by adjusting γ or σ. This gives control of the trade-off between decreased aliasing and increased noise.

For an intuitive explanation of these equations, consider the sine wave shown in Figure 7a, with samples at regularly spaced intervals λ as shown. These samples are inside the Nyquist limit and therefore sample the sine wave properly. Jittering the location of each sample n by some ζ_n in the range $-\lambda/2 < \zeta_n < \lambda/2$ is similar to adding some noise to the amplitude; note that the basic sine wave

frequency is not lost. This noise is less for sine waves with a lower frequency relative to the sampling frequency.

Now consider the sine wave shown in Figure 7b. Here the sampling rate is not sufficient for the frequency of the sine wave, so regularly spaced samples can alias. The jittered sample, however, can occur at any amplitude. If there are exactly a whole number of cycles in the range $-\lambda/2 < \zeta_n < \lambda/2$, then the amplitude that we sample is random, since there is an equal probability of sampling each part of the sine wave. In this case none of the energy from the sine wave produces aliasing; it all becomes noise. This corresponds to the zero points of the dashed line in Figure 6. If the sine wave frequency is not an exact multiple of λ, then some parts of the wave will be more likely to be sampled than others. In this case there is some attenuated aliasing and some noise because there is some chance of hitting each part of the wave. This attenuation is greater for higher frequencies because with more cycles of the wave there is less preference for one part of the wave over another. Note also that the average signal level of the noise (the dc component or gray level) is equal to the average signal level of the sine wave. The gray level of the signal is preserved.

4.2 Implementation

The extension of jittering to two dimensions is straightforward. Consider a pixel as a regular grid of one or more rectangular *subpixels*, each with one sample point. Each sample point is placed in the middle of a subpixel, and then noise is added to the x and y locations independently so that each sample point occurs at some random location within its subpixel.

Once the visibility at the sample points is known, the sample values are filtered with a reconstruction filter and resampled on a regular grid of pixel locations to obtain the pixel values. How to do this reconstruction properly is an open problem. The easiest reconstruction filter to compute is a box filter. Each pixel value is obtained by simply averaging the sample values in that pixel. Weighted reconstruction filters with wider filter kernels give better variance reduction. In this case the filter values are a function of the position of each sample point relative to the surrounding pixels. The value of each pixel is the sum of the values of the nearby sample points multiplied by their respective filter values; this total is normalized by dividing by the total of the filter values.

If the random components of the sample locations are small compared with the width of the filter, the effect of the random components on the filter values can usually be ignored. The filter values can then be calculated in advance for the regularly spaced grid locations. These filter values can be prenormalized and stored in a lookup table. Changing filters is simply a matter of changing the lookup table.

5. DISTRIBUTED RAY TRACING

In the previous section, we applied stochastic sampling to the two-dimensional distribution of the sample points used for determining visibility in a z buffer or ray-casting algorithm. But the intensity of a pixel on the screen is an analytic function that may involve several nested integrals: integrals over time, over the pixel region, and over the lens area, as well as an integral of reflectance times

illumination over the reflected hemisphere and an integral of transmittance times illumination over the transmitted hemisphere. These integrals can be tremendously complicated.

Image-rendering algorithms have made certain simplifying assumptions in order to avoid the evaluation of these integrals. But the evaluation of these integrals is essential for rendering a whole range of fuzzy phenomena, such as penumbrae, blurry reflections, translucency, depth of field, and motion blur. Thus image rendering has usually been limited to sharp shadows, sharp reflections, sharp refractions, pinhole cameras, and instantaneous shutters. Recent exceptions to this are the radiosity method [10] and cone tracing [1].

The rendering integrals can be evaluated with stochastic sampling. If we regard the variables of integration as additional dimensions, we can perform a Monte Carlo evaluation of the integrals by stochastically distributing the sample points (rays) in those additional dimensions. This is called probabilistic or distributed ray tracing.

—Distributing reflected rays according to the specular distribution function produces gloss (blurry reflection).
—Distributing transmitted rays produces translucency (blurry transparency).
—Distributing shadow rays through the solid angle of each light source produces penumbrae.
—Distributing ray origins over the camera lens area produces depth of field.
—Distributing rays in time produces motion blur.

Distributed ray tracing is discussed in detail in a previous paper [5], and others have extended the results found there [7, 12, 14] (also personal communications from D. Mitchell and from T. Whitted). This section summarizes the distributed ray-tracing algorithm from the viewpoint of stochastic sampling.

5.1 Nonspatial Jittering

One way to distribute the rays in the additional dimensions is with uncorrelated random values. For example, one could pick a random time for each ray or a random point on a light source for each shadow ray. This approach produces pictures that are exceedingly noisy, owing to the bunching up of samples (as illustrated later in Figure 11d). We can reduce the noise level by using a Poisson disk distribution, ensuring that the samples do not bunch up or leave large gaps that are unsampled. As before, we use jittering to approximate a Poisson disk distribution.

To jitter in a nonspatial dimension, we use randomly created prototype patterns in screen space to associate the sample points with a range of that dimension to sample, then jitter to pick the exact location within each range. In the case of sampling in time to produce motion blur, we divide the frame time into slices and randomly assign a slice of time to each sample point. The exact time within each slice is then determined by jittering.

For example, to assign times in a pixel with a 4-by-4 grid of sample points, one could use a random distribution of the numbers 1–16, such as the one shown in

Fig. 8. Example of a prototype time pattern.

7	11	3	14
4	15	13	9
16	1	8	12
6	10	5	2

Figure 8. The sample in the xth column and the yth row would have a prototype time

$$t_{xy} = \frac{P_{xy} - 0.5}{16},$$

where P_{xy} is the value shown in the xth column and the yth row of the prototype pattern in Figure 8. A random jitter of $\pm\frac{1}{32}$ is then added to this prototype time to obtain the actual time for a sample. For example, the sample in the upper left subpixel would have a time $\frac{6}{16} \leq t \leq \frac{7}{16}$.

Note that correlation between the spatial locations and the locations in other dimensions can cause aliasing. For example, if the samples on the left side of the pixel are consistently at an earlier time than those on the right side of the pixel, an object moving from right to left might be missed by every sample, whereas an object moving from left to right might be hit by every sample.

5.2 Weighted Distributions

Sometimes we need to weight the samples. For example, we may want to weight the reflected samples according to the specular reflection function, or we may want to use a weighted temporal filter. One approach would be to distribute the samples evenly and then later weight each ray according to the filter. A better approach is *importance sampling* [11], in which the sample points are distributed so that the chance of a location being sampled is proportional to the value of the filter at that location. This avoids the multiplications necessary for the weighting and also puts the samples where they will do the most good.

In order to use jitter to do importance sampling, we divide the filter into regions of equal area, as shown in Figure 9. Each region is sampled by one sample point, with the samples spaced further apart for smaller filter values and closer together for larger filter values. Each sample point is positioned at the center of its region and then jittered to a random location in the region. Note that the size of the jitter varies from sample to sample. If the filter shape is known ahead of time, a list of the centers and jitter magnitudes for each region can be precomputed and stored in a lookup table.

For example, for the reflection ray, we create a lookup table based on the specular reflection function. Given the angle between the surface normal and the incident ray, this lookup table gives a range of reflection angles plus a jitter magnitude for determining an exact reflection angle within that range. For any given reflection ray, the index into this table is determined using its ancestral

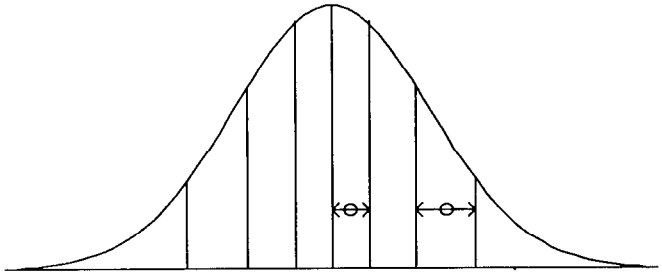


Fig. 9. Importance sampling. The samples are distributed so that they sample regions of equal area under the weighting function. The prototype sample location and jitter range is shown for two of the sampling regions.

primary ray in screen space to associate it with a randomly generated prototype pattern of table indices.

5.3 Summary of Distributed Ray Tracing

The distributed ray-tracing algorithm is illustrated in Figure 10. For each primary ray:

—Determine the spatial location of the ray by jittering.

—Determine the time for the ray from jittered prototype patterns.

—Move the camera and the objects to their location at that time.

—Determine the focal point by constructing a ray from the eye point (center of the lens) through the screen location of the ray. The focal point is located on this ray so that its distance from the eye point is equal to the focal distance.

—Determine the lens location for the ray by jittering a location selected from a prototype pattern of lens locations.

—The primary ray starts at the lens location and goes through the focal point. Determine the visible point for this ray using standard ray-casting or ray-tracing techniques.

—Trace a reflection ray. The direction of the reflection ray is determined by jittering a set of directions that are distributed according to the specular reflection function. This is done with a lookup table; the lookup table index is based on a screen space prototype pattern that assigns indices to primary rays and their descendants. The reflection direction is obtained from the lookup table and then jittered. The range of the jitter is also stored in the table.

—Trace a transparency ray if the visible object is transparent. The direction of the transparency ray is determined by jittering a set of directions that are distributed according to the specular transmission function.

—Trace the shadow rays. For each light source, determine the location on the light for the shadow ray, and trace a ray from the visible point to that location on the light. The chance of tracing the ray to a location on the light should be proportional to the intensity and projected area of that location as seen from the visible point on the surface.

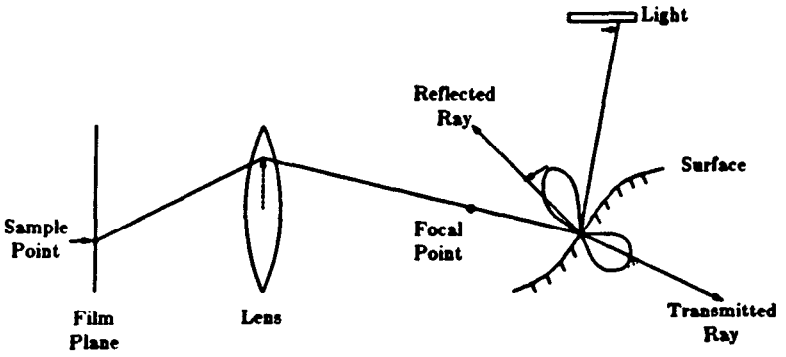


Fig. 10. Distributed ray tracing.

6. EXAMPLES

The jitter used in these examples is white noise jitter with $\gamma = 0.5$. An example of this distribution is shown in Figure 11a, and the Fourier transform of Figure 11a is shown in Figure 11b. Notice how Figure 11b resembles the Fourier transform of a Poisson disk distribution (shown in Figure 4b). By contrast, a pure Poisson distribution of samples with no minimum distance restriction is shown in Figure 11d, and the Fourier transform of Figure 11d is shown in Figure 11e. The C code in Figure 11c was used to generate Figure 11a, and the C code in Figure 11f was used to generate Figure 11d.

In Figures 12 and 13, a box filter was used for a reconstruction filter to accentuate the noise problems. In all of the other examples, the following Gaussian filter was used:

$$e^{-d^2} - e^{-w^2},$$

where d is the distance from the center of the sampling region to the center of the pixel, and $w = 1.5$ is the filter width distance, beyond which the filter was set to zero. The effect of jitter on the filter values was ignored.

Consider the comb of triangular slivers illustrated in Figure 12a. Each triangle is 1.01 pixels wide at the base and 50 pixels high. The triangles are placed in a horizontal row 1.01 pixels apart. If the comb is sampled with a regular grid, aliasing can result as depicted in Figure 12b. A comb containing 200 such triangular slivers is rendered in Figures 12c–f.

In Figure 12c the comb is rendered with a single sample at the center of each pixel. Figure 12d also has one sample per pixel, but the sample location is jittered by $\zeta = \pm\frac{1}{2}$ pixel in x and y. Figure 12c is grossly aliased: there are just a few large triangles spaced 100 pixels apart. This aliasing is replaced by noise in Figure 12d. Because there is only one sample per pixel, each pixel can only be white or black, but in any given region, the percentage of white pixels equals the percentage of that region that is covered by the triangles. Note that the white pixels are denser at the bottom, where the triangles are wider.

In Figure 12e the same comb is rendered with a regular 4-by-4 grid of samples. In Figure 12f the regular 4-by-4 grid is jittered by $\zeta = \pm\frac{1}{8}$ pixel in x and y. Again the regularly spaced samples alias; this time there are a few large overlapping triangles spaced $\frac{100}{4} = 25$ pixels apart. This aliasing is replaced by noise in the

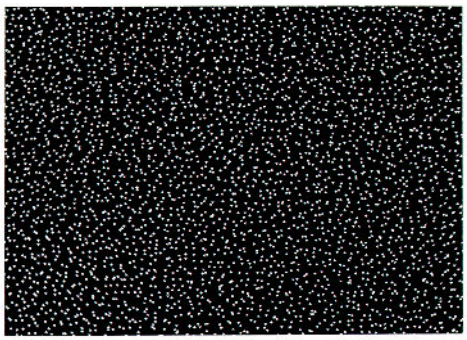


Fig. 11a. Distribution pattern of jittered samples.

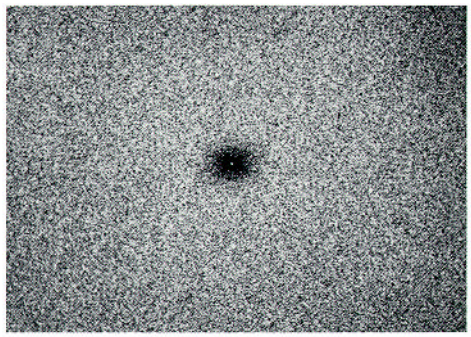


Fig. 11b. Fourier transform of the pattern in Figure 11a.

```
/* Draw a jittered sample pattern in a 512x512 frame buffer.  There is one  */
 * sample in each sample region of 8x8 pixels, for a total of 4096 samples. */
DrawJitterPattern() {
      double Random();                    /* returns a random number in the range 0–1 */
      int x,y;                            /* (x,y) is the corner of the sample region */
      int jx,jy;                          /* (jx,jy) is the jitter */
      for (y=0; y<512; y+=8) {
            for (x=0; x<512; x+=8) {
                  jx = 8*Random();
                  jy = 8*Random();
                  SetPixelToWhite(x+jx,y+jy);
            }
      }
}
```

Fig. 11c. C program that generated the pattern in Fig. 11a.

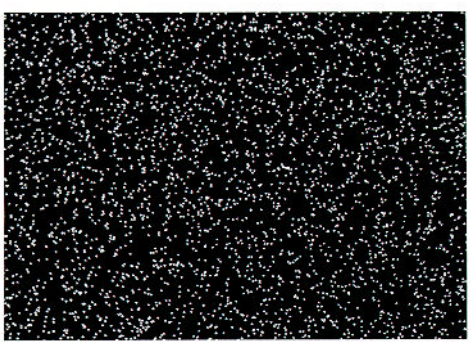


Fig. 11d. Distribution pattern of randomly placed samples.

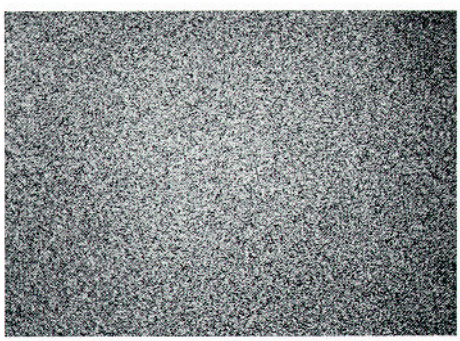


Fig. 11e. Fourier transform of the pattern in Fig. 11d.

```
/* Draw a random sample pattern with 4096 samples. */
DrawPoissonPattern() {
      double Random();                    /* returns a random number in the range 0–1 */
      int n, sx,sy;                       /* (sx,sy) is the sample location */
      for (n=0; n<4096; n++) {
            sx = 512*Random();
            sy = 512*Random();
            SetPixelToWhite(sx,sy);
      }
}
```

Fig. 11f. C program that generated the pattern in Fig. 11d.

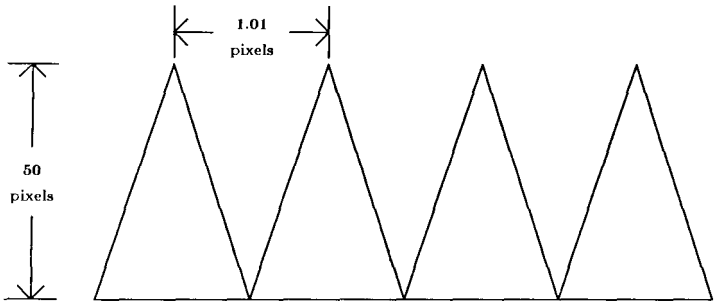


Fig. 12a. Schematic diagram of the comb of triangles example. The triangles are 50 pixels high and 1.01 pixels apart.

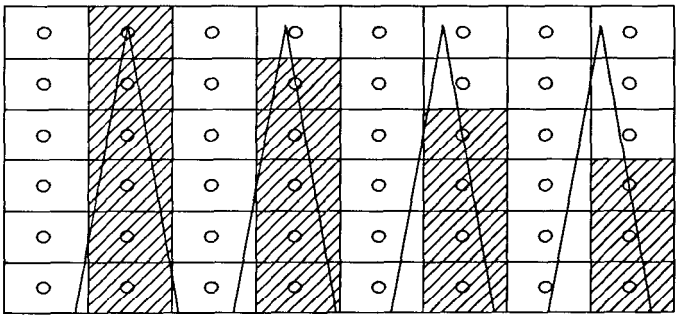


Fig. 12b. The comb of triangles aliases when rendered with a regular grid of sample points in the manner shown here. Samples are shown as circles, and pixels are shown as rectangles. Pixels with samples inside a triangle are shaded.

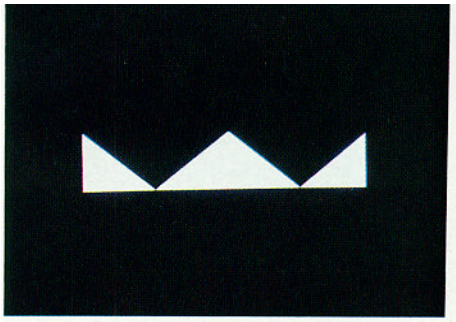


Fig. 12c. Comb rendered with a regular grid, one sample per pixel.

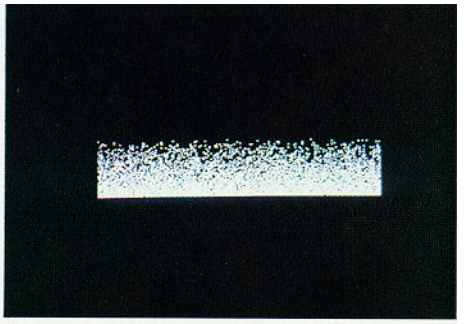


Fig. 12d. Comb rendered with a jittered grid, one sample per pixel.

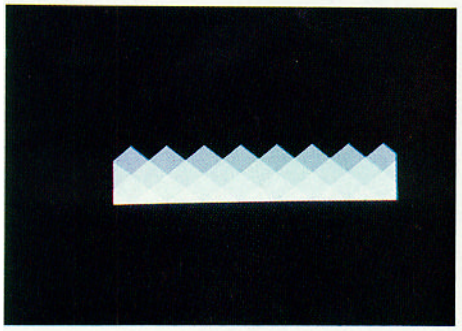


Fig. 12e. Comb rendered with a regular grid, 16 samples per pixel.

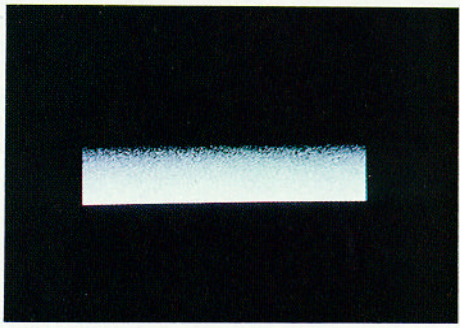
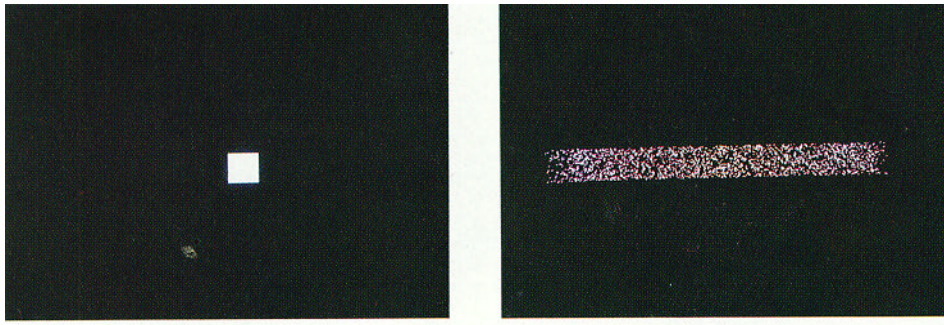
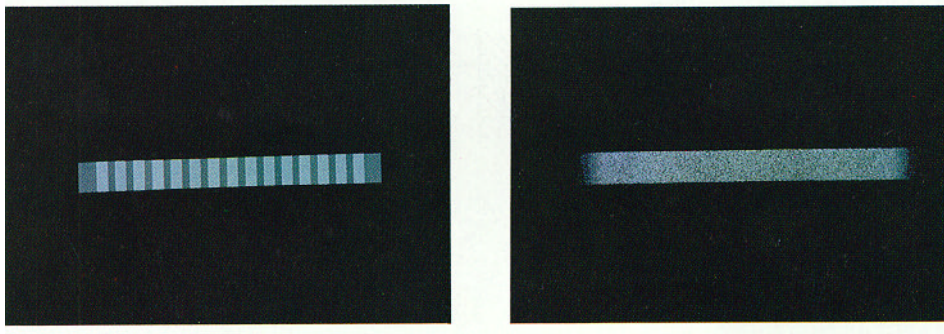


Fig. 12f. Comb rendered with a jittered grid, 16 samples per pixel.



(a) One sample per pixel, no jitter

(b) One sample per pixel, with jitter



(c) Sixteen samples per pixel, no jitter

(d) Sixteen samples per pixel, with jitter

Fig. 13. Fast-moving polygon.

jittered version, Figure 12f. Notice, though, that the noise is greatly reduced compared with Figure 12d.

Figure 13 shows a small white square moving across the screen. Figure 13a was rendered with no jitter and one sample per pixel, so the image is still. Figure 13b was rendered with jitter and one sample per pixel; the image is now blurred but is extremely noisy because, with only one sample, each pixel can be only one of two colors-the color of the square or the color of the background. Notice, though, that in any given region the number of pixels that are white is proportional to the amount of time the square covered that region; thus the percentage of white pixels is constant in the middle and ramps off at the ends. Figure 13c was rendered with no jitter and 16 samples per pixel, and Figure 13d with jitter and 16 samples per pixel. Notice the reduction in the noise level with the additional samples.

Figure 14a is the ray-traced picture *1984,* with a closeup of the 4-ball shown in Figure 14b. The 4-ball remains stationary for most of the time the shutter is open and moves quickly to the upper right just before the shutter closes. The blur is quite extreme, and yet the image looks noisy instead of aliased. This picture was made with 16 samples per pixel.

Figures 15a and 15b are two frames from the short film *The Adventures of André & Wally B.* [18]. These extreme examples of motion blur were rendered

Fig. 14a. *1984*, by Thomas Porter.

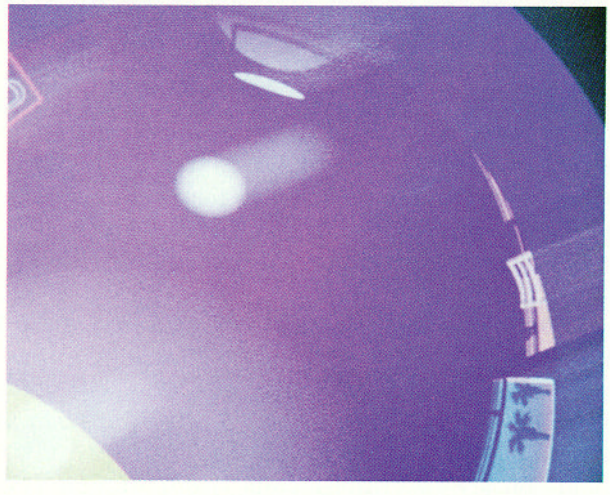


Fig. 14b. Close-up of *1984*.

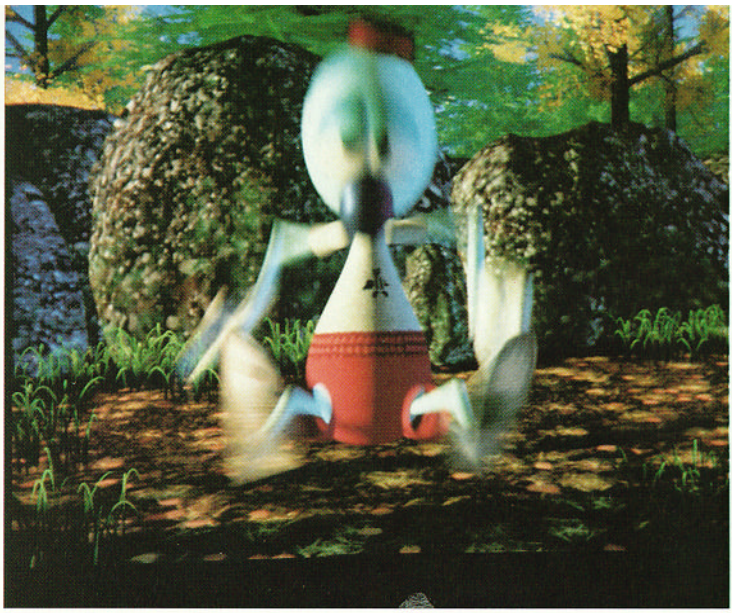


Fig. 15a. Example of motion blur from *The Adventures of André & Wally B.*

Fig. 15b. Example of motion blur from *The Adventures of André & Wally B.*

with a scan-line algorithm that uses point sampling and a z buffer to determine visibility. In these frames a very simple adaptive method automatically used 16 samples per pixel for most pixels and 64 samples per pixel for pixels that contain objects that move more than 8 pixels in x or y within the frame time.

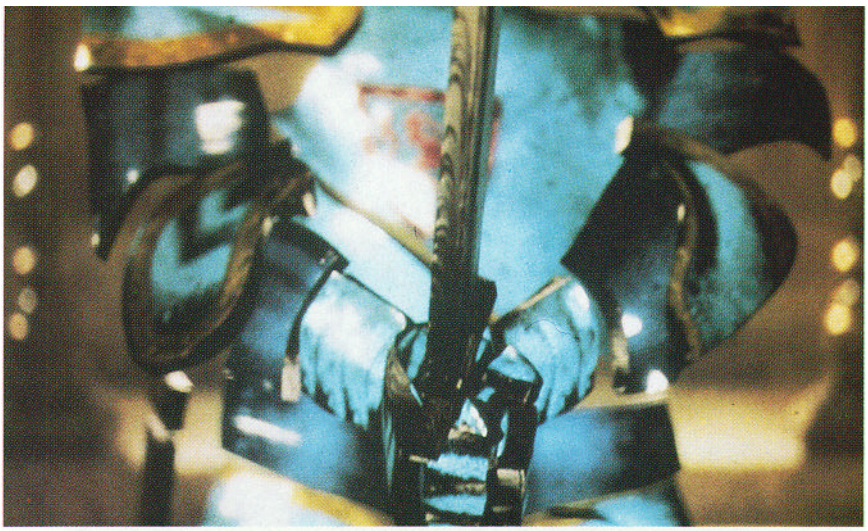


Fig. 16. Example of depth of field from *Young Sherlock Holmes* (Copyright 1985, Paramount Pictures Corp.).

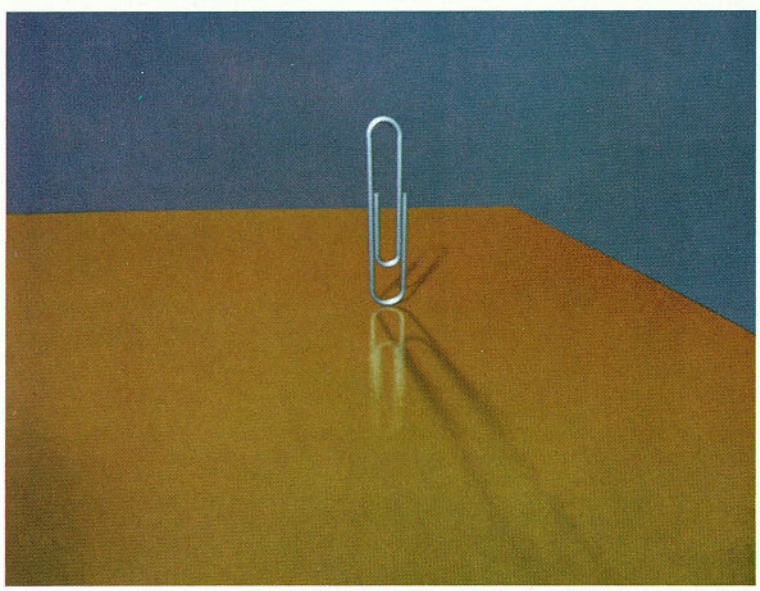


Fig. 17. Example of penumbrae and blurry reflection.

This cuts down considerably on the noise level and helps avoid needless computation. Others have since found ways to add more samples adaptively based on an estimate of the variance of the image in each pixel [12, 14].

Figure 16 shows a frame of a computer-synthesized stained-glass man from *Young Sherlock Holmes* [16]. The camera is focused on the sword, with the body

out of focus. This was also rendered with a scan-line algorithm, but in this case, no adaptive method was used to change the number of samples per pixel; instead, there were always 16 samples per pixel. The sequence is also motion blurred.

The paper clip in Figure 17 shows penumbrae and blurry reflection, rendered with 16 samples per pixel. Other examples of distributed ray tracing have appeared in a previous paper [5]. In all cases, areas of extreme blur become noisy instead of aliasing.

7. DISCUSSION AND CONCLUSIONS

With correctly chosen nonuniform sample locations, high frequencies appear as noise instead of aliasing. The magnitude of this noise is determined by the sampling frequency. We have found that using 16 samples per pixel produces an acceptable noise level in most situations, with more needed only for high-frequency situations, such as frames that are extremely motion blurred or out of focus. Stochastic sampling should also work well when integrated with adaptive sampling. This has been the subject of some recent research [12, 14].

The human eye uses a Poisson disk distribution of photoreceptors. A simple and effective approximation to a Poisson disk distribution can be obtained by jittering a regular grid. When this technique is extended to distributed ray tracing, the locations in the nonspatial dimensions can be chosen by jittering randomly generated prototype patterns. Weighted functions can be evaluated using importance sampling.

Stochastic sampling involves some additional computation. Because the samples are not regularly spaced, forward differencing cannot be used to exploit pixel-to-pixel coherence. Compared with standard ray tracing, distributed ray tracing requires additional calculations to move objects to their correct location for each ray. Moving and out-of-focus objects also require a more sophisticated bounding calculation, and these objects must often be intersected with a larger number of rays.

Aliasing has been a major problem for ray-tracing and ray-casting algorithms, and this problem is solved by stochastic sampling. The shading calculations, which have traditionally been point sampled, are automatically antialiased with stochastic sampling, eliminating problems such as highlight aliasing. Another potential application is texture map sampling. Extended to distributed ray tracing, stochastic sampling also provides a solution to motion blur, depth of field, penumbrae, blurry reflections, and translucency.

ACKNOWLEDGMENTS

I would especially like to thank Tom Porter, who made the *1984* picture, suggested the extension of the two-dimensional technique to motion blur, and helped test many of the ideas. Alvy Ray Smith found the article on the distribution of cells in the eye. Andy Moorer and Jim Kajiya helped with the theory, and a number of discussions with Loren Carpenter were invaluable. The idea of dithering sample locations originally came from Rodney Stock, who provided inspiration and motivation for this work. Jack Yellott provided the pictures in Figure 3. Thanks also to the many people at Lucasfilm who made *The Adventures of André & Wally B.* and the stained-glass man sequence from *Young Sherlock Holmes.*

REFERENCES

1. AMANATIDES, J. Ray tracing with cones. *Comput. Graph. 18*, 3 (July 1984), 129–145.
2. BALAKRISHNAN, A. V. On the problem of time jitter in sampling. *IRE Trans. Inf. Theory* (Apr. 1962), 226–236.
3. BLINN, J. F. Computer display of curved surfaces. Ph.D. dissertation, Computer Science Dept., Univ. of Utah, Salt Lake City, 1978.
4. BRACEWELL, R. N. *The Fourier Transform and Its Applications.* McGraw-Hill, New York, 1978.
5. COOK, R. L., PORTER, T., AND CARPENTER, L. Distributed ray tracing. *Comput. Graph. 18*, 3 (July 1984), 137–145.
6. CROW, F. The use of greyscale for improved raster display of vectors and characters. *Comput. Graph. 12*, 3 (Aug. 1978), 1–5.
7. DIPPE, M. A. Z., AND WOLD, E. H. Antialiasing through stochastic sampling. *Comput. Graph. 19*, 3 (July 1985), 69–78.
8. FEIBUSH, E., LEVOY, M., AND COOK, R. L. Synthetic texturing using digital filtering. *Comput. Graph. 14*, 3 (July 1980), 294–301.
9. GARDNER, G. Y. Simulation of natural scenes using textured quadric surfaces. *Comput. Graph. 18*, 3 (July 1984), 11–20.
10. GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., AND BATTAILE, B. Modeling the interaction of light between diffuse surfaces. *Comput. Graph. 18*, 3 (July 1984), 213–222.
11. HALTON, J. H. A retrospective and prospective survey of the Monte Carlo method. *SIAM Rev. 12*, 1 (Jan. 1970), 1–63.
12. KAJIYA, J. T. The rendering equation. *Comput. Graph. 20*, 4 (Aug. 1986), 143–150.
13. KAY, D. S., AND GREENBERG, D. P. Transparency for computer synthesized images. *Comput. Graph. 13*, 2 (Aug. 1979), 158–164.
14. LEE, M. E., REDNER, R. A., AND USELTON, S. P. Statistically optimized sampling for distributed ray tracing. *Comput. Graph. 19*, 3 (July 1985), 61–67.
15. NORTON, A., ROCKWOOD, A. P., AND SKOLMOSKI, P. T. Clamping: A method of antialiasing textured surfaces by bandwidth limiting in object space. *Comput. Graph. 16*, 3 (July 1982), 1–8.
16. PARAMOUNT PICTURES CORP. *Young Sherlock Holmes.* Stained glass man sequence by D. Carson, E. Christiansen, D. Conway, R. Cook, D. DiFrancesco, J. Ellis, L. Ellis, C. Good, J. Lasseter, S. Leffler, D. Muren, T. Noggle, E. Ostby, W. Reeves, D. Salesin, and K. Smith. Pixar and Lucasfilm Ltd., 1985.
17. PEARSON, D. E. *Transmission and Display of Pictorial Information.* Pentech Press, London, 1975.
18. PIXAR. *The Adventures of André & Wally B.* By L. Carpenter, E. Catmull, R. Cook, T. Duff, C. Good, J. Lasseter, S. Leffler, E. Ostby, T. Porter, W. Reeves, D. Salesin, and A. Smith. July 1984.
19. POTMESIL, M., AND CHAKRAVARTY, I. Modeling motion blur in computer-generated images. *Comput. Graph. 17*, 3 (July 1983), 389–399.
20. PRATT, W. K. *Digital Image Processing.* Wiley, New York, 1978.
21. ROTH, S. D. Ray casting for modeling solids. *Comput. Graph. Image Process. 18* (1982), 109–144.
22. SHAPIRO, H. S., AND SILVERMAN, R. A. Alias-free sampling of random noise. *SIAM J. 8*, 2 (June 1960), 225–248.
23. SOCIETY OF PHOTOGRAPHIC SCIENTISTS AND ENGINEERS. *SPSE Handbook of Photographic Science and Engineering.* Wiley, New York, 1973.
24. WHITTED, T. An improved illumination model for shaded display. *Commun. ACM 23*, 6 (June 1980), 343–349.
25. WILLIAMS, D. R., AND COLLIER, R. Consequences of spatial sampling by a human photoreceptor mosaic. *Science 221* (July 22, 1983), 385–387.
26. WILLIAMS, L. Pyramidal parametrics. *Comput. Graph. 17*, 3 (July 1983), 1–11.
27. YELLOTT, J. I., JR. Spectral consequences of photoreceptor sampling in the rhesus retina. *Science 221* (July 22, 1983), 382–385.

Received March 1985; accepted June 1986