Improvements on the Pixel-tracing Filter: Reflection/Refraction, Shadows, and Jittering

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

Abstract

Pixel-trace filtering is an efficient anti-aliasing technique for sequence animation because its computation cost is constant regardless of scene complexity. Although it works well in simple cases, it has shortcomings with steady objects, moving shadows, reflective/refractive objects, and deforming objects.

This paper significantly improves pixel-trace filtering and eliminates all these restrictions. Jittered sampling is introduced to eliminate the aliasing experienced with steady objects. Beam/pencil tracing techniques are applied to reflective/refractive objects. The G-buffer scheme is adopted to deal with moving shadows and deforming objects. Several experiments demonstrate that our improvements successfully eliminate the restrictions. It is also shown that the temporal-filtering feature of pixel-trace filtering is advantageous for flicker reduction, and an appropriate filter size is suggested based on human flicker perception.

With these improvements, pixel-trace filtering becomes a powerful anti-aliasing method, and we believe that it can be a standard anti-aliasing technique for computer animation.

Keywords: Anti-aliasing, Spatio-temporal filtering, Computer Animation, Flicker Perception

1 Introduction

Because sampling techniques are used in computer image synthesis, aliasing artifacts are unavoidable whenever the sampling rate is insufficient. A simple, but successful anti-aliasing technique is stochastic super-sampling [DIPPE, COOK86], where several stochastically selected points are sampled in each pixel area.

A major problem of this technique is its high computational cost, which is proportional to scene complexity (e.g., the number of polygons) and the number of samples. Ironically, complex scenes usually contain high frequency information, which requires a high sampling rate. This makes the technique extremely expensive for complex scenes.

The author proposed a constant-time anti-aliasing algorithm with respect to scene complexity for animation sequences [SHINYA93]. The algorithm traces image points through the image sequence using animation information, and collects sub-pixel information for filtering. It works well in simple cases, such as walk-through scenes without reflection/refraction, but involves several restrictions, as pointed out in [SHINYA93].

Correlated motion When object motion is correlated with the sampling pattern, aliasing artifacts cannot be removed. A typical example is an object that remains at the same screen position.

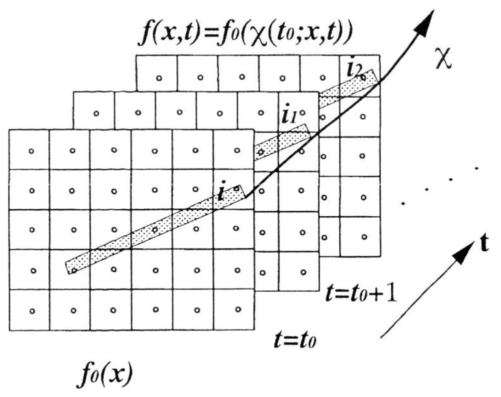
Refraction/reflection In the case of refraction/reflection, the basic algorithm cannot trace the image points because conventional ray traces do not provide transformation between the screen and the object space.

Temporal change of object brightness Since the algorithm assumes constant object brightness over time, brightness changes cause some artifacts. For example, shadows of moving objects become blurred by the filtering.

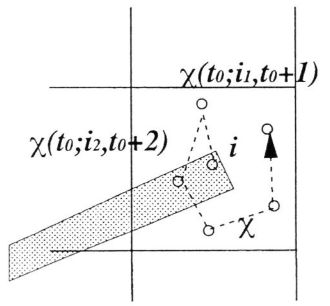
Object Deformation Since the basic algorithm relies on a simple linear motion, deforming objects cannot be managed.

These limit the practical application of the pixel-tracing filter.

This paper provides practical solutions to the above problems. First, the jittered sampling technique [DIPPE, COOK86] is applied to the algorithm to eliminate the correlation between motion and sampling. Second, the beam and pencil tracing approach [HECKBERT, SHINYA87] is adopted to trace refracted/reflected image points. Third, the G-buffer scheme [SAITO] is introduced to handle temporal brightness changes. Theoretical studies and experiments confirm the efficiency of the algorithm. Finally, we will discuss human flicker perception and flicker reduction by the pixel-tracing filter.



(a) Animation sequence.



(b) Collected samples.

Figure 1: Basic idea of the pixel-tracing filter.

2 Review of the pixel tracing filter

This section briefly reviews the basic idea and theories of the pixel tracing filter. Details can be found in [SHINYA93].

2.1 Basic idea

Animation sequences usually have strong correlation between successive frames. The most typical situation is that image points in a frame move along a velocity field, called an *optical flow*. Through such sequences, the same images are repeatedly sampled at slightly different points in different frames. This allows us to reduce aliasing artifacts by spatio-temporal filtering.

Consider the situation shown in Figure 1, where the initial image f_0 is moving through the sequence. In the figure, the grids show pixel areas and the circles indicate sample points. The image motion can be described by flow function χ. When the image point x_0 at t_0 moves to x_1 at t_1, flow χ is defined as

$$\chi(t_1; x_0, t_0) = x_1.$$

Using χ, the image sequence can be represented by

$$f(x,t) = f_0(\chi(t_0; x, t)). \qquad (1)$$

Eq. 1 also means that sample (x,t) is identical to sample $(\chi(t_0;x,t),t_0)$. Thus, sampling at i_j at $t = t_0+j$ is equivalent to sampling at $\chi(t_0; i_j, t_0+j)$ at $t = t_0$. In the example, all $x_j = \chi(t_0; i_j, t_0+j)$ are located at the same pixel area, and their average

$$\sum_j^n f(i_j, t_0 + j)/n = \sum_j f(x_j; t_0)/n$$

reduces aliasing just as super-sampling with box filtering does.

A big advantage in computer generated animation is that we can calculate exact image motion from animation data, which allows efficient anti-aliasing, as well as image compression [WALLACH].

2.2 Theories

The basic idea can be formulated in the following way. We assume that the image sequence is a deformed pattern of the initial image, as defined by Eq. 1. Pixel-tracing filtering can be then represented as

$$h(x_0, t_0) = \int_{-T/2}^{T/2} \int f_s(x,t) g(x,t) dx dt, \qquad (2)$$

where $f_s(x,t)$ is the sampled image sequence of $f(x,t)$, and $g(x,t)$ is the spatio-temporal filter kernel, described by

$$g(x,t) = (1/T) w(x_0 - \chi(t_0; x, t))(\partial\chi/\partial x),$$

where $w(x)$ is a desirable anti-aliasing filter. The filtering result can be calculated as

$$\begin{aligned} h(x_0,t_0) &= \sum_{n,m} h_{n,m} \\ h_{n,m} &= \int w(x_0 - x') f_0(x') dx' \\ &\quad \times \int_{-T/2}^{T/2} \exp(\imath n \Xi \chi(t'; x', t_0)) \\ &\quad \times \exp(\imath m \Omega t') dt'/T, \end{aligned}$$

where Ξ and Ω are the spatial and temporal sampling frequency, respectively.

Obviously,

$$h_{0,0} = \int w(x_0 - x') f_0(x') dx'.$$

When the filter size T tends to infinity,

$$\lim_{T\to\infty} h_{n,m} = \int w(x_0 - x')f_0(x')dx' \qquad (3)$$

$$\times(\lim(1/T)K_n(m\Omega; x')), \quad (4)$$

where K_n is the Fourier transform of the function k_n,

$$k_n(t;x') = \exp(\imath n \Xi \chi(t;x',t_0)).$$

When $K_n(m\Omega;x')$ is not singular for all m, $n(\neq 0)$ integer, i.e.,

$$|K_n(m\Omega;x)| < \infty, \qquad (5)$$

the motion flow χ is not correlated with sampling, and the aliasing pattern tends to zero,

$$h(x_0,t_0) \to \int w(x_0 - x')f(x')dx'.$$

This theory itself is solid, but involves some practical problems with the assumptions and computability:

Image sequence model (Eq. 1) Most three-dimensional image sequences do not satisfy Eq. 1 because of occlusion and temporal brightness changes.

Correlated motion (Eq. 5) When object motion is correlated with the sampling pattern, Eq. 5 does not hold, and aliasing artifacts are not removed. This happens when, for example, objects remain at the same screen position, in which case $\chi \equiv 0$.

Computation of motion flow Rigid body motion can be simply represented by a 4×4 matrix for each object. With deforming objects, however, transformation is generally a complicated non-linear mapping. Reflection and refraction also makes the calculation of motion flow χ difficult.

The occlusion problem can be solved by separating multiple (occluding) flows based on flow vector comparison and by alpha-blending the filtering results [SHINYA93]. In this paper, we focus on the remaining problems: correlated motion, refraction/reflection, deforming objects, and temporal brightness change.

3 Improvements

3.1 Breaking motion correlation

Stochastic sampling is a very powerful tool for anti-aliasing, and we introduce it to the pixel-tracing technique to break the correlation between motion and sampling. If we jitter the sample position in each frame, the pixel-tracing filter acts as a purely spatial jittered sampling filter for steady objects, as shown in Figure 2. More generally, we can prove that random jittering eliminates the correlation between object motion and sampling.

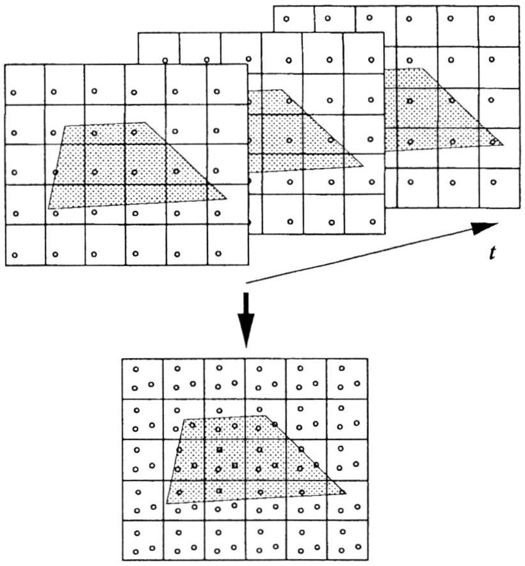


Figure 2: Temporal jittered sampling and steady objects.

We denote the displacement of sample points by $\chi_j(t)$, and the flow due to object motion by $\chi_o(t;x_0,t_0)$. The total motion flow χ is represented by their sum,

$$\chi(t;x_0,t_0) = \chi_o(t;x_0,t_0) + \chi_j(t).$$

Assuming that the jitter is a stationary noise and independent of object motion, Eq. 4 becomes

$$\begin{aligned}
&\lim_{T\to\infty} h_{n,m} \\
&= \int w(x_0 - x')f_0(x')dx' \\
&\quad \times[\lim_{T\to\infty} \int_{-T/2}^{T/2} \exp(\imath n\Xi\chi(t';x',t_0)) \\
&\quad \times \exp(\imath m\Omega t')dt'/T], \\
&= \int w(x_0 - x')f_0(x')dx' \int \exp(\imath n\chi_j\Xi)dP(\chi_j) \\
&\quad \times \lim_{T\to\infty} \int_{-T/2}^{T/2} (1/T)\exp(\imath n\Xi\chi_0(t';x',t_0)) \\
&\quad \times \exp(\imath m\Omega t')dt',
\end{aligned}$$

where $dP(\chi_j)$ represents the probability density function of χ_j. The third integral can be evaluated as

$$|\int_{-T/2}^{T/2} \exp(\imath(n\Xi\chi(t';x',t_0)) \times \exp(\imath m\Omega t')dt'| \leq \int_{-T/2}^{T/2} dt = T.$$

When χ_j is a uniform noise on $[-\pi/\Xi, \pi\Xi]$,

$$\int \exp(\imath n\chi_j\Xi)dPr(\chi_j) = 0,$$

for all $n \neq 0$. Thus,

$$h(x_0, t_0) = \sum h_{n,m} \rightarrow \int w(x - x')f_0(x')dx'.$$

This completes the proof.

3.2 More general image sequence model

When object brightness changes, the fundamental assumption of Eq. 1 does not hold any more. The most typical example is that shadows of moving objects are blurred by filtering[1], as shown in Figure 10. In some sense, shadow blurring may not sound terrible. However, the blurring size depends on the velocity of the motion, and, for example, when a moving object suddenly stops, its shadows start to shrink, which looks pretty strange.

To deal with temporal brightness changes, we generalize the filtering operation given by Eq. 2. First, we introduce the *luminance function* $\mathcal{L}$, which calculates pixel colors from geometric and lighting parameters, such as the object position, normal vector, shading properties of object points, and so on. By setting these parameters as u, image f_0 can be written as

$$f_0(x_0) = \mathcal{L}_{t0}(u(x_0, t_0)).$$

For rigid bodies, these parameters do not change over time, so parameters $u(x,t)$ can be described by

$$u(x,t) = u(\chi(t_0; x, t), t_0),$$

and image sequence $f(x,t)$ by

$$f(x,t) = \mathcal{L}_t(u(\chi(t_0; x,t), t_0)). \tag{6}$$

The image sequence described by Eq. 6 is a fairly general model, and allows any temporal brightness changes. When the luminance function

[1]This is different from the motion blur induced by finite (sub-frame) exposure time since the filter interval T is considerably larger than a frame (typically, more than 16 frames).

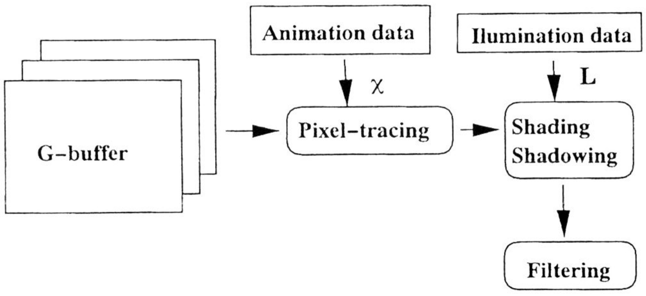


Figure 3: Pixel-tracing with G-buffer.

$\mathcal{L}_t$ is constant over time, Eq. 6 is identical with Eq. 1.

Instead of filtering equation Eq. 2, we generalize the filtering as

$$h(x_0, y_0) = \int\int g(x,t)\mathcal{L}_{t0}(u(\chi(t_0; x,t), t_0))dxdt, \tag{7}$$

It is obvious that this filter acts as an ideal anti-aliasing filter for image sequences Eq. 6.

The actual calculation can be illustrated as shown in Figure 3. The parameters u necessary for the luminance function are stored in a G-buffer. During filtering, pixel colors are evaluated by the current luminance function $\mathcal{L}_{t0}$.

The luminance function $\mathcal{L}_t$ consists of shading and shadowing calculation, which can be very expensive. However, when shadow information is stored, for example, in the form of shadow buffers [REEVES] or radiosities, process $\mathcal{L}$ is inexpensive and is still independent of scene complexity.

3.3 Computation of motion flows

3.3.1 Deformation

For rigid body motion, tracing the object position $x_o(t)$ from the screen space is straightforward. It can be calculated using

$$x_o(t) = X(t)X^{-1}(t_0)S^{-1}(t_0)x_i(t_0),$$

where $X(t)$ is the transformation matrix from the world coordinates to the object local coordinates, S is the transformation from the world to the screen coordinates, and x_i is the image point.

However, the transformation varies non-linearly over space for deforming objects, such as free form deformation [SEDERBERG], hierarchical deformation [FORSEY] and physically-based deformation [TERZOPOULOS]. Fortunately, these transformations generally assume parameterization of surfaces (objects), and object points are easily evaluated from the parameters. Thus, we store the

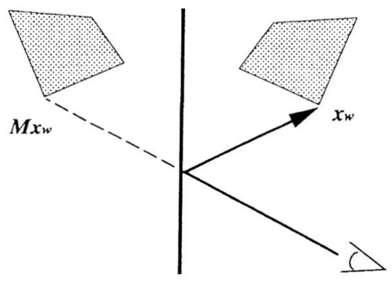


Figure 4: Beam tracing.

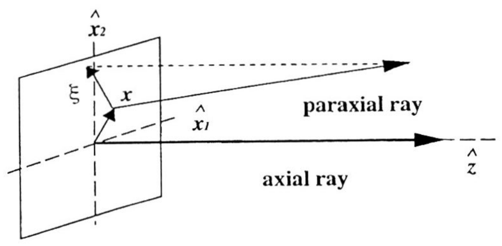


Figure 5: Ray coordinate.

surface parameter $s(x_i, t)$ of each pixel for deforming objects, and trace motion by

$$x_o(t) = X(t)\sigma(s(x_i, t_0), t), \tag{8}$$

where σ represents the parameterized deforming surface.

Another advantage of storing parameter s is that most parameters required by $\mathcal{L}$ (e.g., the surface normal, position) can be also calculated from s. This allows us to apply the generalized filter Eq. 7 to deforming objects.

3.3.2 Beam/pencil tracing

Although conventional ray tracers cannot provide transformation between the screen space and object spaces for reflected/refracted rays, beam tracing [HECKBERT] and pencil tracing [SHINYA87] explicitly performs this transformation. For planar reflection, beam tracing is more convenient, while pencil tracing is applied to general cases.

Beam tracing Images reflected from a planar surface are simple mirror symmetric images of the original scenes, as shown in Figure 4. Consequently, reflection can be represented by the mirror symmetric transformation M. For the plane $n_1x + n_2y + n_3z = d$, the transformation M is represented by

$$M = \begin{pmatrix} 1+2n_1^2 & 2n_1n_2 & 2n_1n_3 & -2dn_1 \\ 2n_1n_2 & 1+2n_2^2 & 2n_2n_3 & -2dn_2 \\ 2n_1n_3 & 2n_2n_3 & 1+2n_3^2 & -2dn_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

When the transformation from the world coordinate to the screen coordinate is T_p, the transformation for the reflected image is

$$x_s = T_pMx_w, \tag{9}$$

where x_s and x_w are the screen and world coordinates, respectively.

Pencil tracing General reflection and refraction are not as simple as planar reflection. However, the paraxial theory provides the first order approximation of the transformation.

To specify paraxial rays,[2] we define a ray coordinate, shown in Figure 5. In the coordinate system, the $\hat{z}$-axis is the direction of the axial ray (the center ray), and the axial ray passes through the origin. Since a ray can be defined by its direction and a point through which it passes, paraxial rays are defined by their intersection with the $\hat{x}_1$-$\hat{x}_2$ plane (x) and the projection of their direction onto the plane (ξ).

When rays are refracted or reflected, their direction and position changes in a complicated way. However, as a first order approximation, the change can be represented by matrix T, called the system matrix, such that,

$$\begin{pmatrix} x' \\ \xi' \end{pmatrix} = T \begin{pmatrix} x \\ \xi \end{pmatrix}.$$

For convenience, we can also denote a system matrix by four 2×2 sub-matrices, as

$$T = \begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

The initial rays emitted from the eye can be represented by only their directions as $(0\ \xi_e)^t$. After reflection/refraction, the ray vector becomes

$$\begin{aligned} x_{ref} &= B\xi_e, \\ \xi_{ref} &= D\xi_e. \end{aligned}$$

When the ray hits the object point (x_o, z_o) after reflection/refraction,

$$x_o = x_{ref} + z\xi_{ref} = (B + zD)\xi_e,$$

or

$$x_e = (B + zD)^{-1}x_o = Rx_o. \tag{10}$$

[2]The rays that are near to a given axial ray are called paraxial rays and are said to form a pencil.

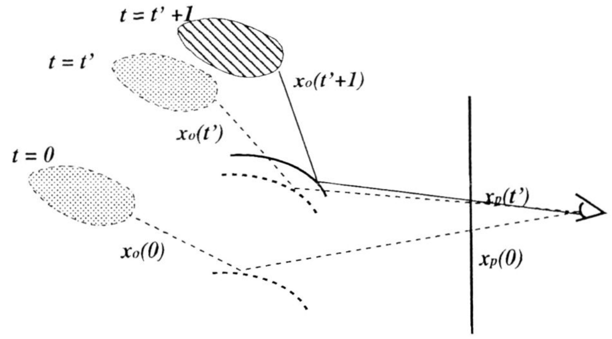


Figure 6: Tracing reflected image point.

Using Eq. 10, we can trace image points after refraction/reflection.

The system matrices can be calculated either in an analytical way ([SHINYA87]) or from the sampled rays. When the reflected/refracted ray for pixel (i, j) is known to be $(x_{i,j}, \xi_{i,j})$, matrices B and D can be calculated from these samples as

$$\begin{aligned} B &= \left(\; (x_{i-1,j} - x_{i+1,j})/2 \quad (x_{i,j-1} - x_{i,j-1})/2 \; \right), \\ D &= \left(\; (\xi_{i-1,j} - \xi_{i+1,j})/2 \quad (\xi_{i,j-1} - \xi_{i,j-1})/2 \; \right). \end{aligned}$$

Tracing image points We store the beam matrix M or system matrix R at each pixel containing a refracted/reflected ray. To efficiently trace image points through the sequence, we take advantage of temporal coherence.

At $t = 0$, the object point $x_o(0)$ is projected to pixel $x_p(0)$ (Figure 6). Assume that we know $x_p(t')$ for $x_o(t')$. The projection for $t' + 1$ (or -1) is calculated by the matrix stored for pixel $x_p(t')$ as

$$x_p^{(0)}(t'+1) = R(x_p(t'), t'+1) x_o(t'+1), \qquad (11)$$

in the case of pencil tracing.

Since the system matrix is an approximation, we should make a confirmation. We calculate

$$x_p^{(1)}(t'+1) = R(x_p^{(0)}(t'+1), t'+1) x_o(t'+1),$$

and check

$$||x_p^{(1)}(t'+1) - x_p^{(0)}(t'+1)|| < th$$

for a certain threshold th (e.g., 0.5 pixels). If not satisfied, this process is repeated. When we fail to find a solution after several iterations, we conclude that there is no corresponding projection point at $t' + 1$, and the filter treats such cases as if the sample is clipped from the screen. Because of temporal image coherence, this searching process is very efficient. For beam tracing, on the other hand, the transformation is exact, and iteration is not necessary.

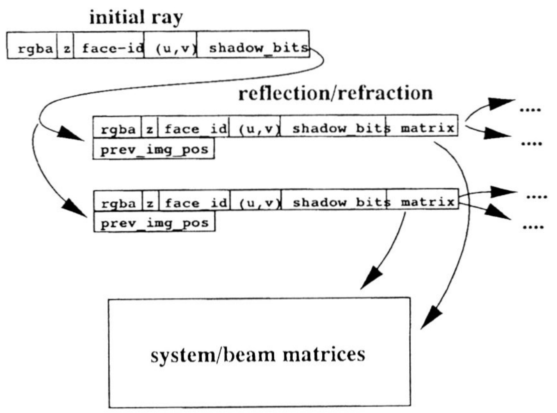

initial ray
rgba | z | face-id | (u,v) | shadow_bits
reflection/refraction
rgba | z | face_id | (u,v) | shadow_bits | matrix
prev_img_pos
rgba | z | face_id | (u,v) | shadow_bits | matrix
prev_img_pos
....
....
....
....
system/beam matrices

Figure 7: Data structure.

3.4 Data

For each ray, we store the rgba values, the z-value, the face-id, the surface parameter (u, v). For efficiency, we also store shadow-bits, where each bit represents whether the sample point is lit or not from the corresponding light.

For refracted/reflected rays, the pointer to the system matrix or the beam matrix is also stored. To trace each image point as described in Section 3.2, a data field is also prepared to record the latest image point `(ix, iy)`. To represent a ray-tree structure, each data set has pointers for reflection/refraction rays. The data structure is illustrated in Figure 7.

3.5 Procedure

We implemented the algorithm described in Section 3 using two ideas. First, we avoid random access among frames, which can be fatal if not all the necessary data can be loaded into the main memory. Second, shading and shadowing were computed only if the shadow-bit patterns were different from that of the corresponding flow.

The procedure for frame `t0` can be summarized as follows:

1) Initialize all data for `t0`.

2) For all `t` in the filter kernel, do the following.

   2.1) (If the data for `t` is not loaded, remove unnecessary data, and load the data for `t`).

   2.2) For the ray-tree at each pixel `(x,y)`, do the following.

      2.2.1) Calculate the image point `(x0,y0)` at `t0`, according to Eqs. 9, 10 and 8.

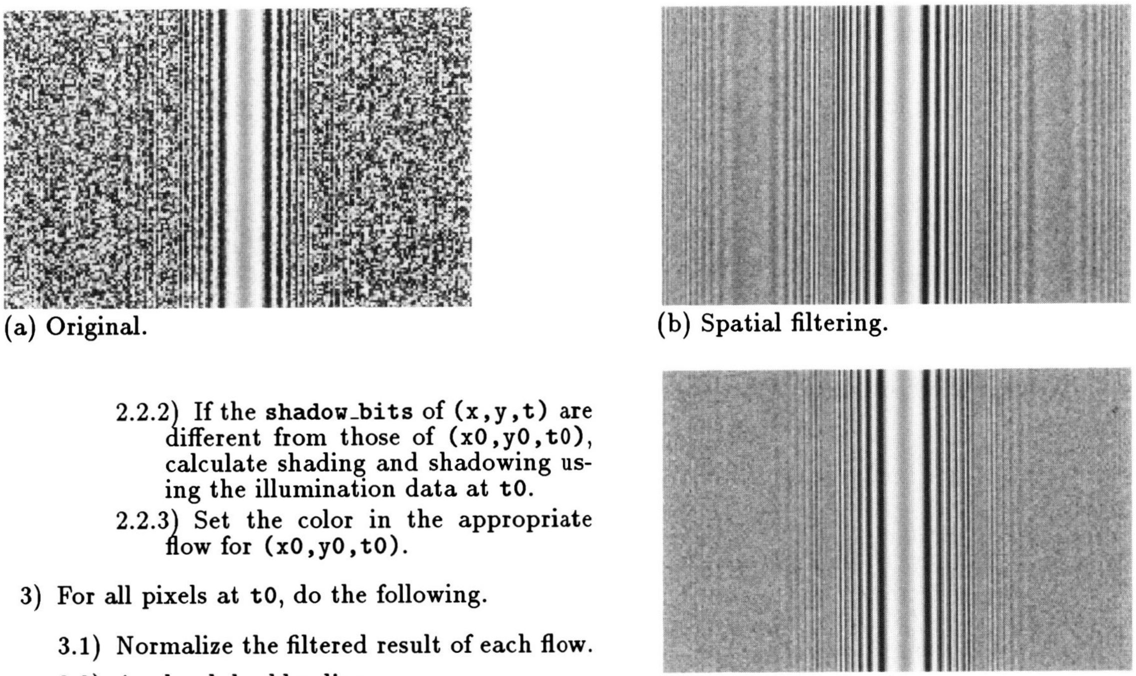

(a) Original.
(b) Spatial filtering.
(c) Pixel-trace filtering.

Figure 8: One-dimensional Fresnel pattern.

```
        2.2.2) If the shadow_bits of (x,y,t) are
               different from those of (x0,y0,t0),
               calculate shading and shadowing using
               the illumination data at t0.
        2.2.3) Set the color in the appropriate
               flow for (x0,y0,t0).
3) For all pixels at t0, do the following.
    3.1) Normalize the filtered result of each flow.
    3.2) Apply alpha-blending.
```

In this procedure, Step 2.1 is executed only when the available memory is not sufficient. The handling of flows (the flow separation, normalization, and alpha-blending) is almost the same as in the previous method. In the current implementation, we adopted the shadow-buffer technique [REEVES] to keep the shadowing calculation cost at a reasonable level. We store beam matrices for planar reflection and system matrices for other cases.

4 Another criterion: flickering

In animation sequencies, spatial aliasing appears as annoying flicker, and thus, flicker reduction is a major aspect of anti-aliasing. In this section, we consider a new anti-aliasing criterion based on *visible flicker reduction.*

One of the advantages of the pixel-tracing filter over purely spatial super-sampling is its temporal feature of low-pass filtering, which directly reduces flickering due to aliasing. To demonstrate the difference, we applied the pixel-tracing filter and a spatial super-sampling filter to a moving one-dimensional Fresnel pattern, defined by

$$f(x, y; t) = \sin(k(x - vt)^2).$$

Note that this pattern has a constant power spectrum over the entire frequency.

Figure 8-a shows the result of spatial filtering. We can clearly observe aliasing patterns even with 64 samples per pixel[3]. In an animation, these patterns become even more noticeable because of flickering. Figure 8-b shows the result from the pixel-tracing filter. Due to the temporal-filtering feature, flicking is largely eliminated, and the visual quality is much improved.

In the pixel-tracing filter, this flicker criterion can also estimate the appropriate number of samples per pixel, or the temporal filter size. In terms of the sampling theory, it is hard to determine the super-sampling rate because the frequency band of information is generally unknown in computer graphics applications. Fortunately, we do know the basic human flicker perception characteristics, and from these we can estimate the necessary filter size.

It is well-known that human flicker perception has band-pass temporal characteristics, and that we are not so sensitive to low frequency flicker. The pixel-tracing filter acts as a temporal low-pass filter and, roughly speaking, its cut-off frequency is approximately the inverse of the filter size. Thus, if the cut-off frequency is lower than the sensitive flicker frequency band, visible flicker can be successfully reduced. To demonstrate this flicker sensitivity, we conducted a simple psychophysical experiment to measure flicker thresholds. The details

[3]These patterns are caused by the finitely supported filter kernel, a one-pixel Fourier window in this case. If we adopt the ideal anti-aliasing filter (a sinc function over the whole image space), this can be eliminated. However, functions with a small support are generally used in practice, and this artifact is difficult to avoid.

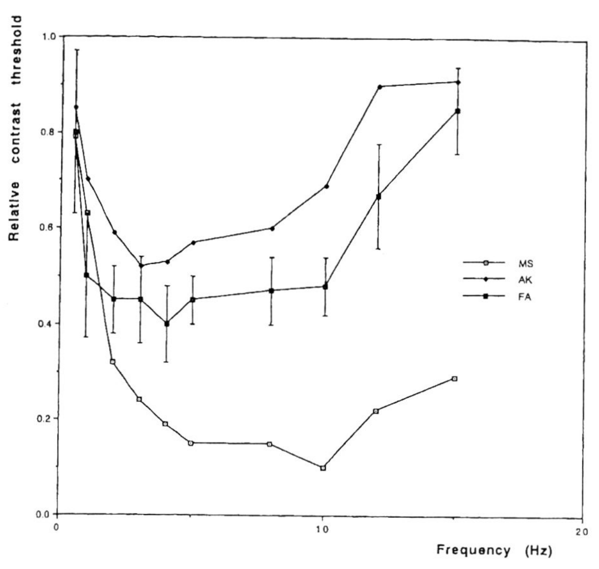


Figure 9: Flicker threshold.

of the experiment are described in the Appendix.

The obtained flicker thresholds are plotted in Figure 9. Each plot is the average of ten trials, and the standard deviation is presented by vertical lines for Subject FA, whose variance was the largest. The flicker sensitivity is high (i.e., the threshold is low) between 3 and 8 Hz, and is low for 1 to 0.5 Hz. The experiment is not precise given the limitation of the display unit, and it is difficult to extract quantitative values from just this result. However, it seems reasonable to assume that one or two seconds (30 or 60 frames) would be sufficient for the filter.

5 Experiments

Example scenes The proposed algorithm was applied to the sequence shown in Figure 10-a, to yield the result shown in Figure 10-c. As shown in the figure, the blurring artifact observed in 10-b is completely removed. The filter size is 16 frames.

In Figure 10-c, however, since the base plate remains at the same screen position, aliasing artifacts are not removed for the plate, including the shadow on it. This situation can be more clearly seen in Figure 11-b. The image of the moving objects (bamboo branches) is improved while aliasing of the shadow and the trunk remains. To reduce aliasing of steady objects, we applied jittered sampling. Figure 11-c shows the result. The aliasing of the shadow is completely removed by jittered sampling. The motion of the bamboo was calculated by a modal analysis and a stochastic wind model [SHINYA92]. The leaves are rigid but the trunk and the branches are modeled by deformable beams.

Figure 12 shows a reflection example, where a thin tube is moving above a mirror. The original image sequence is shown in Figure 12-a. The blurring problem in Figure 12-b has been solved in Figure 12-c. Since the mirror is planar, a beam matrix is used for filtering. Figure 13 shows a refraction example, wherein system matrices were used. Anti-aliasing is successfully performed in the sequence. Figure 14 shows a more complicated example. Both the reflected bamboo and the shadows are effectively anti-aliased.

Computation time The required CPU time for anti-aliasing was measured on the IRIS Crimson R4400, and the results are listed in Table 1. As shown in the table, the extra cost for shading calculation with shadow buffers is very modest (Figs. 10 and 11). Actually, shading all (200×200) pixels only took 1.6 seconds for Figure 11. Shadow-bit comparison also allows further acceleration by avoiding unnecessary shading calculation.

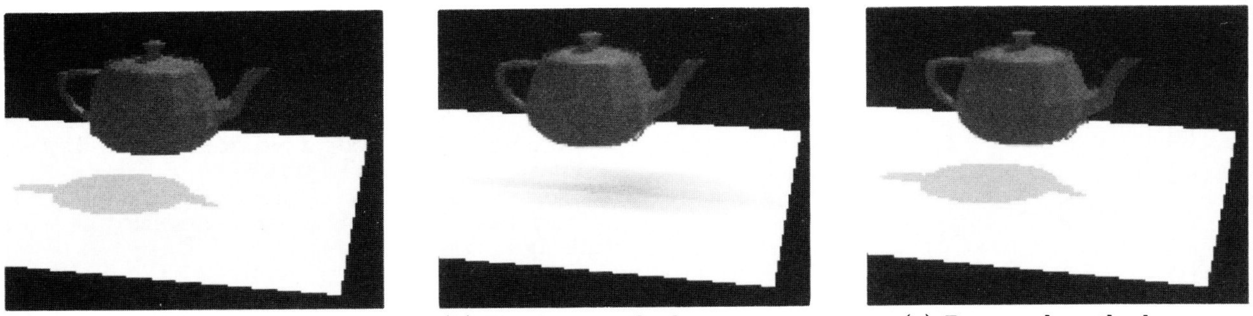
The additional computational cost for reflection/refraction calculation is also reasonable (Figs. 12, 13, and 14). Since each node of the stored ray tree is individually traced and filtered, the computational cost is proportional to the average number of ray tree nodes per pixel.

The computational cost of the pixel tracing algorithm is independent of scene complexity, hence its efficiency is significant for complex environments. For example, the scene in Figure 14 contains about 115K polygons. Our ray tracer took 6.3 minutes to create a one-sample-per-pixel image, and thus, super-sampling at 16 samples/pixel would take 100.8 minutes. The pixel tracing only took 1.6 minutes for almost equivalent anti-aliasing, thus, acceleration rates of over 50 were achieved in this example.

6 Conclusion

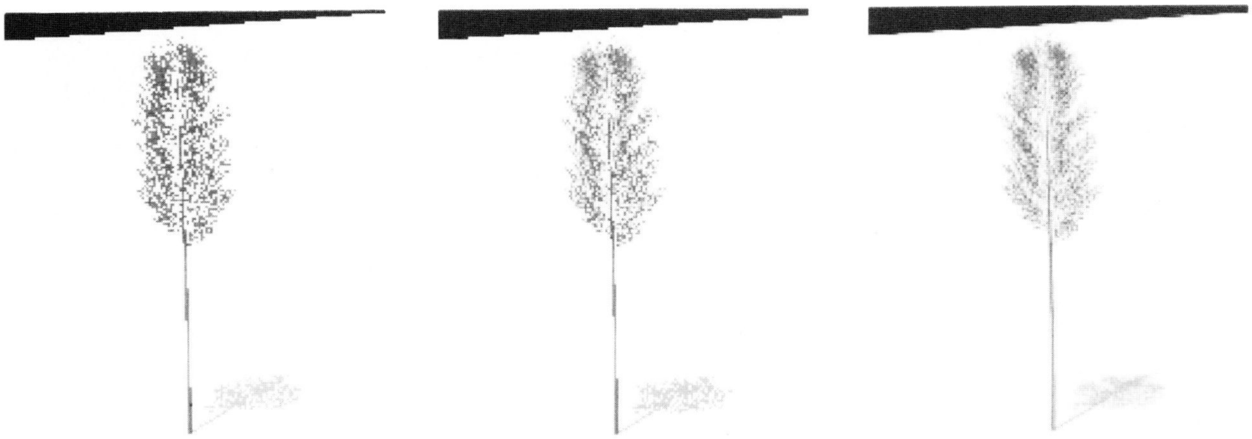
We have made several significant improvements to pixel-trace filtering, and so removed the restrictions imposed by the previous algorithm. Jittered sampling was introduced for the anti-aliasing of steady objects. Beam/pencil tracing techniques were applied to reflective/refractive objects. The G-buffer scheme was adopted to deal with moving shadows and deforming objects. Several experiments confirmed that these improvements successfully eliminate the previous restrictions. We also demonstrated the advantages of the pixel-tracing filter in terms of flicker reduction, and suggested an appropriate filter size based on human flicker perception.

This paper focuses on spatial anti-aliasing, but temporal anti-aliasing (motion blur) is also important in certain applications. It is also possible to apply the method to spatio-temporal anti-aliasing by calculating sub-frame images from image flows, as discussed in [SHINYA95].



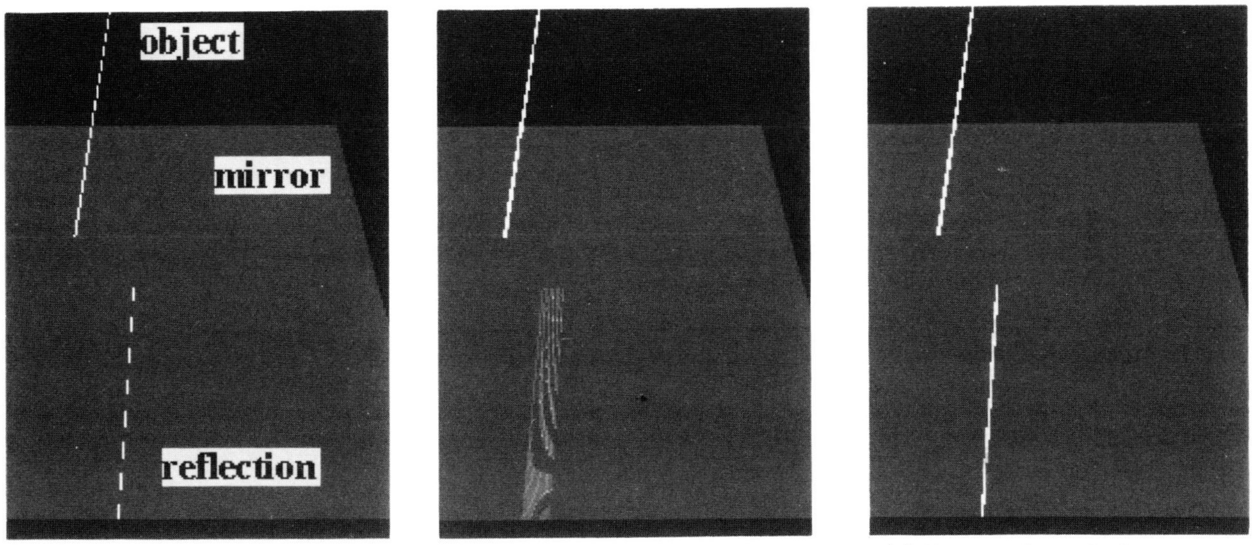
(a) Original. (b) Previous method. (c) Proposed method.

Figure 10: Blurred shadow of a moving teapot.



(a) Original. (b) Regular sampling. (c) Jittered sampling.

Figure 11: A swaying bamboo.



(a) Original. (b) Previous method. (c) Proposed method.

Figure 12: Blurring artifacts in reflection images.

Table 1: CPU time.

	image	Fig. 10	Fig. 11	Fig. 12	Fig. 13	Fig. 14
(a)	previous method	25.7 sec	29.3 sec	38.3 sec	41.7 sec	43.4 sec
(b)	+ shading	26.9 sec	30.3 sec	–	–	–
	+ reflection	–	–	49.3 sec	51.9 sec	96.7
	(b)/(a)	1.05	1.03	1.29	1.24	2.23
	resolution	256×256	200×200	256×256	256×256	256×256

Acknowledgments

The author would like to thank Tomio Kishimoto for his administrative supports, Atsushi Kajiyama for his help in making the animation sequences, and Toki Takahashi and Takafumi Saito for their helpful discussion.

References

[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', Proceedings of SIGGRAPH'85, No.3, pp.69-78, 1985.

[FORSEY] D. Forsey and R. Bartel, 'Hierarchical B-spline Refinement,' Proceeedings of SIGGRAPH'88, pp. 205-212, 1988.

[HECKBERT] P. S. Heckbert, P. Hanrahan, 'Beam Tracing Polygonal Objects,' Proceedings of SIGGRAPH'84, pp.119-128, 1984.

[REEVES] W. Reeves, D. Salesin, and R. Cook, 'Rendering Antialiased Shadows with Depth Maps,' Proceedings of SIGGRAPH'87, pp.283-291, 1987.

[SAITO] Takafumi Saito and Toki Takahashi, 'Comprehensible Rendering of 3-D Shapes', Proceedings of SIGGRAPH'90, pp.197-206, 1990.

[SHINYA87] M. Shinya, T. Takahashi, and S. Naito, 'Principles and Applications of Pencil Tracing,' Proceedings of SIGGRAPH'87, pp. 45-54, 1987.

[SHINYA92] M. Shinya and A. Fournier, 'Stochastic Motion - Motion Under the Influence of Wind, Proceedings of Eurographics'92, pp. C-119-128, 1992.

[SHINYA93] M. Shinya, 'Spatial anti-aliasing for animation sequences with spatio-temporal filtering,' Proceedings of SIGGRAPH'93, pp. 289-296, 1993.

[SHINYA95] M. Shinya, 'Spatio-temporal anti-aliasing by the pixel-tracing method,' To appear in Trans. of IEICE, D-II (In Japanese).

[SEDERBERG] T. Sederberg and P. Scott, 'Free-form deformation of Solid Geometric Models,' Proceedings of SIGGRAPG'86, pp. 151-160, 1986.

[TERZOPOULOS] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, 'Elastically deformable models', Proceedings of SIGGRAPH'87, pp. 205-214, 1987.

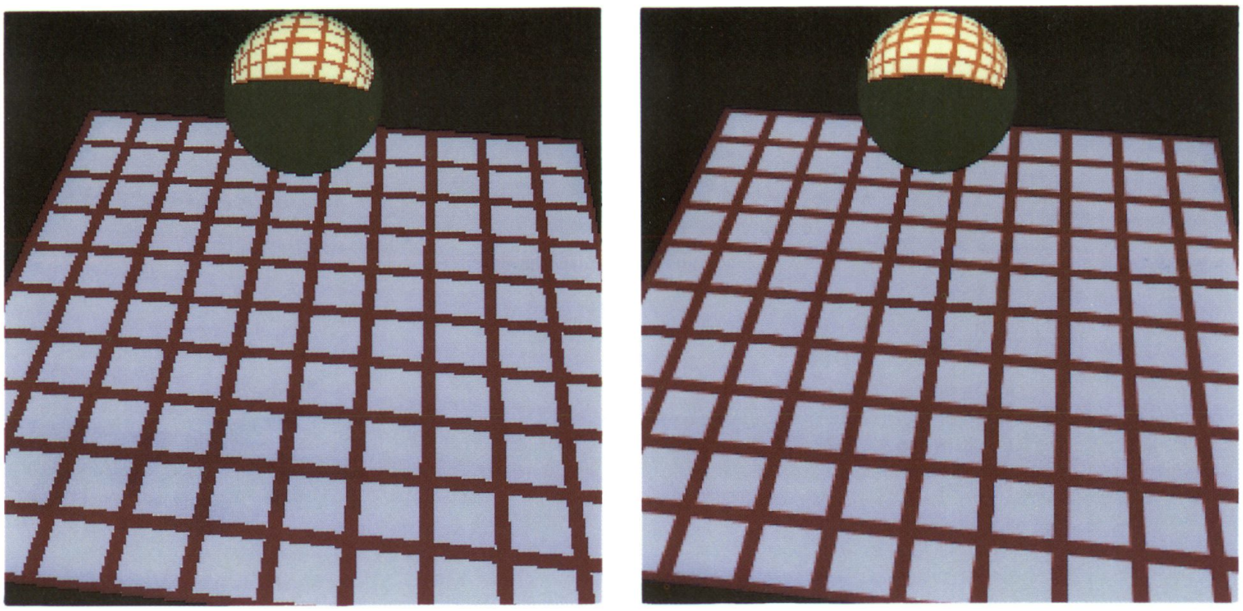
[WALLACH] D. Wallach, S. Kunapalli, and M. Cohen, 'Accelerated MPEG Compression of Dynamic Polygonal Scenes', Proceedings of SIGGRAPH'94, pp. 193-196, 1994.

Appendix: Flicker experiment

We used two white noise patterns $I_0(x,y)$ and $I_1(x,y)$ to make a sinusoidal flicker field

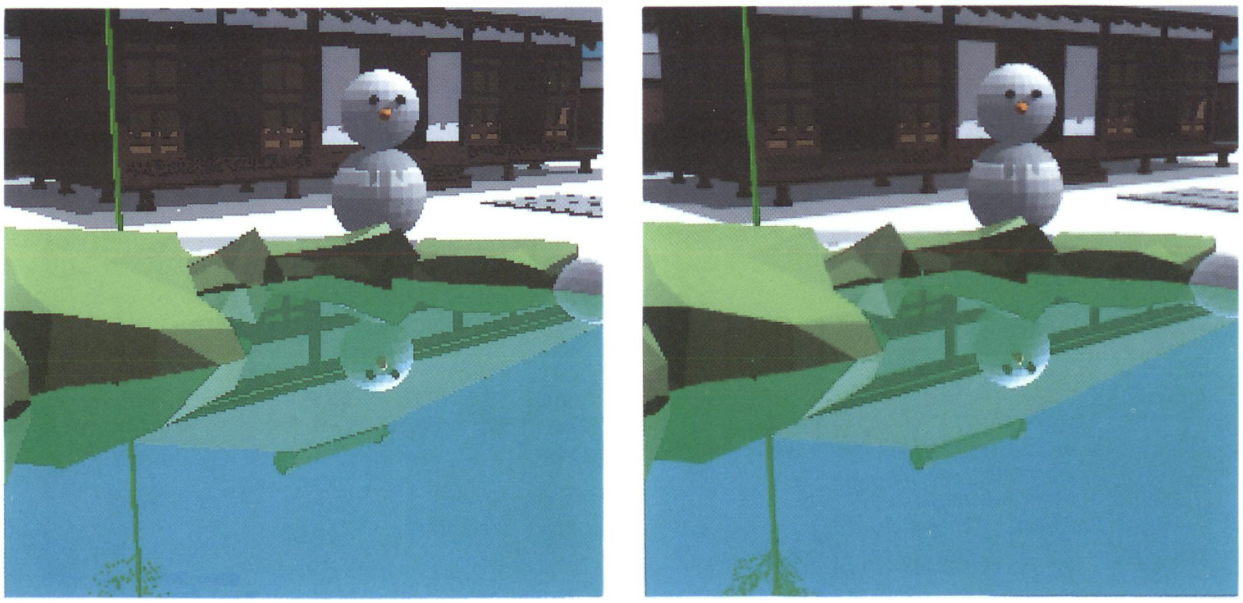
$$I(x,y;t) = I_0(x,y) + m\sin(2\pi ft)I_1(x,y),$$

where f is the frequency of the flicker. The pattern was displayed on the monitor of a graphics workstation. The flicker was realized by updating the lookup table at the frame rate (60 Hz) according to the sampled and discretized value of $m\sin(2\pi ft)$. The pattern was displayed at 256×256 pixels, about 6.5 cm, on the screen. The viewing distance from the subjects was about 1 meter. Subjects binocularly viewed the pattern and adjusted m so that the flicker was just noticeable.



(a) Original. (b) Proposed method.

Figure 13: A glass sphere.



(a) Original. (b) Proposed method.

Figure 14: A pound in Také Tera.