Spatial Anti-aliasing for Animation Sequences with Spatio-temporal Filtering

Mikio Shinya
NTT Human Interface Laboratories
3-9-11 Midori-cho, Musashino-shi
Tokyo 180, Japan
email: shinya@nttarm.ntt.jp
tel: +81 422 59 2648

Abstract

Anti-aliasing is generally an expensive process because it requires super-sampling or sophisticated rendering. This paper presents a new type of anti-aliasing filter for animation sequences, the *pixel-tracing filter*, that does not require any additional sample nor additional calculation in the rendering phase. The filter uses animation information to calculate correlation among the images, and sub-pixel information is extracted from the sequence based on the correlation. Theoretical studies prove that the filter becomes an ideal anti-aliasing filter when the filter size is infinite.

The algorithm is simple image processing implemented as post-filtering. The computational cost is independent of the complexity of the scene. Experiments demonstrate the efficiency of the filter. Almost complete anti-aliasing was achieved at the rate of about 30 seconds per frame for very complex scenes at a resolution of 256×256 pixels. The pixel tracing filter provides effective anti-aliasing for animation sequences at a very modest computational cost.

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

Additional Keywords and Phrases: Anti-aliasing, Spatio-temporal filtering, Computer Animation

1 Introduction

Aliasing artifacts have been troublesome in the field of graphics for a long time. These problems are particularly bad in animation sequences, since flickering thin objects and traveling jaggies are very noticeable.

For viewers in general, these spatio-temporal artifacts are more noticeable than the purely spatial ones in still images. To detect spatial aliasing, the *true* images (e.g., continuous lines or checker board patterns) should be inferred from the sampled image by intelligent, high-level visual processing. On the other hand, spatio-temporal aliasing can be detected by low-level vision processes (e.g., flicker detection and optical flow segmentation) without deep knowledge. This may seem rather negative, but it also implies a positive aspect: there may be easier ways to detect and remove aliasing in animation sequences.

Usually, there is strong correlation among the successive frames of motion pictures. This correlation allows efficient image compression in video codecs (coder/decoder) [NETRA]. This motivates us to extract sub-pixel information from image sequences, which could reduce aliasing artifacts.

This paper mathematically analyzes spatio-temporal characteristics of motion image sequences, and clarifies the useful features of their spectrum. Based on the analysis, a new type of anti-aliasing algorithm is proposed. In the algorithm, the image sequences are filtered with a linear shift-variant spatio-temporal filter called the *pixel-tracing filter*. Through the image sequence, the filtering process traces the pixels corresponding to the same object point, and the weighted sum of their colors is calculated. Theoretical studies prove that the filter acts as an ideal anti-aliasing filter when the filter size is infinite.

Unlike most anti-aliasing algorithms, this algorithm is achieved by post-filtering. The advantages are:

- fast execution independent of the scene complexity (e.g., number of polygons),
- simplicity of implementation,
- no dependence on the rendering process.

Experiments showed that the algorithm was efficient in terms of computational cost and provided effective image improvement.

 ACM-0-89791-601-8/93/008/0289 $01.50

2 Related Work

There are too many studies of anti-aliasing to review exhaustively, so only spatio-temporal approaches are briefly mentioned here. There are two major methods of spatio-temporal anti-aliasing: super-sampling and analytic calculation. In the super-sampling scheme, distributed ray tracing [COOK84] and alpha-blending [HAEBERLI] with stochastic sampling [DIPPE,COOK86] are the most successful and commonly used. Their advantages are simplicity and generality, but the disadvantage is a computational cost that is proportional to the rate of super-sampling. Although adaptive sampling [LEE] and optimal sampling patterns [MITCHELL] have been investigated, image improvement by super-sampling is generally computationally expensive. The analytic approach, on the other hand, is attractive because an exact solution can be calculated in relatively modest computation time. However, algorithms usually involve rather complicated processes, such as three-dimensional scan-conversion [GRANT] and analytic filtering of polygons [CATMULL84], and are only applicable to particular object primitives (typically polygons). In short, both approaches directly calculate sub-pixel or sub-frame information and then apply local filters.

Our approach differs from the above methods in three ways. First, our approach does not require any additional sample or additional calculation in the rendering phase. Second, it evaluates sub-pixel information from the image sequences themselves, taking the advantage of global spatio-temporal correlation. Third, our method uses a temporally global filter to removes spatial aliasing while other methods attempt to produce motion-blur by local temporal filtering.

3 Fourier Analysis

Temporal variation in animation sequences is usually due to the motion of the camera and objects. In this section, we mathematically analyze the spatio-temporal spectra of image sequences of moving objects. The velocity on the image plane is first assumed to be constant in time and space; analyses with spatial and temporal variation follow. The analyses provide an ideal anti-aliasing filter with infinite integral under certain conditions. Throughout this section, a one-dimensional space (image) is assumed for simplicity, but extension to two-dimensional images is mathematically straightforward.

3.1 Preparation

Let x be the image coordinate in pixels and t be the time in frames. Let a real function $f_0(x)$ be the image at $t = t_0$, and $f(x;t)$ be the image sequence. The spatial Fourier transform of f is defined by

$$F_0(\xi) = \int f_0(x)\exp(\imath\xi x)dx,$$

$$F(\xi;t) = \int f(x;t)\exp(\imath\xi x)dx,$$

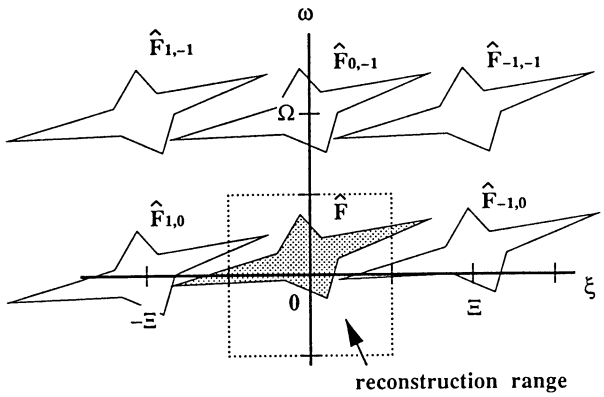


Figure 1: Aliasing in the Fourier Domain.

where ξ denotes the spatial angular frequency (rad/pixel), and $\imath$ is the imaginary unit, $\imath^2 = -1$. Similarly, the temporal Fourier transform is defined by

$$\hat{F}(\xi,\omega) = \int F(\xi;t)\exp(\imath\omega t)dt,$$

where ω is the temporal angular frequency in rad/frame.

The sampled image sequence $f_s(x,t)$ is represented by

$$f_s(x;t) = f(x;t)\sum_{k,l}\delta(x - 2\pi k/\Xi)\delta(t - 2\pi l/\Omega),$$

where Ξ and Ω are the sampling frequencies in space and time. When one point per pixel per frame is sampled, $\Xi = 2\pi$, and $\Omega = 2\pi$. The Fourier transform of f_s is

$$\hat{F}_s(\xi,\omega) = \sum_{n,m}\hat{F}_{n,m}(\xi,\omega), \qquad (1)$$

where

$$\hat{F}_{n,m}(\xi,\omega) = \hat{F}(\xi + n\Xi, \omega + m\Omega).$$

Equation 1 indicates that replicas of $\hat{F}$ appear, centered at the grid points $(-n\Xi, -m\Omega)$, as illustrated in Figure 1.

When $\hat{F}(\xi,\omega) \neq 0$ outside the Nyquist frequencies $(\pm\Xi/2, \pm\Omega/2)$, some replicas intrude on the reconstruction range, causing aliasing artifacts. In other words, anti-aliasing can be achieved if replicas $\hat{F}_{n,m}$ can be filtered out. Therefore, anti-aliasing can be regarded as a process which calculates filtered images from the sampled images, and consequently, our objective is to find some mapping

$$f_s \longmapsto \int f_0(x)w(x_0 - x)dx$$

for any x_0. Here, $w(x)$ denotes some desirable spatial anti-aliasing filter.

The notation defined here is listed in Table 1. An introduction to sampling theory and aliasing can be found in [FOLEY].

Table 1: Symbols and notation

x	position on the image (pixel)
t	time (frame)
ξ	spatial angular frequency (rad/pixel)
ω	temporal angular frequency (rad/frame)
$f_0(x)$	image at $t = t_0$
$f(x;t)$	image at t
$f_s(x;t)$	sampled image sequence
$F_0(\xi)$	the spatial spectrum of f_0
$\hat{F}(\xi,\omega)$	the spatio-temporal spectrum of f.
$\hat{F}_s(\xi,\omega)$	the spatio-temporal spectrum of f_s.
Ξ	spatial sampling frequency
Ω	temporal sampling frequency
$\hat{F}_{n,m}$	the replica of $\hat{F}$ centered at $(-n\Xi, -m\Omega)$
$w(x)$	spatial anti-aliasing filter
$g(x,t)$	shift variant spatio-temporal filter
$\hat{G}(\xi,\omega)$	the spatio-temporal spectrum of w

3.2 Constant Velocity Motion

First, let us consider the simplest motion, constant velocity motion. In this case, the image at t can be represented by

$$f(x;t) = f_0(x + v_0(t_0 - t)), \tag{2}$$

where v_0 is the velocity of the pattern. Its spatio-temporal spectrum is

$$\begin{aligned}\hat{F}(\xi,\omega) &= \int \exp(\imath\omega t)dt \int f_0(x + v_0(t_0 - t))\exp(\imath\xi x)dx \\ &= \int F_0(\xi)\exp(\imath\xi v_0(t - t_0))\exp(\imath\omega t))dt \\ &= 2\pi F_0(\xi)\exp(-\imath\xi v_0 t_0)\delta(v_0\xi + \omega), \end{aligned} \tag{3}$$

where δ is Dirac's delta function and we used the equality

$$\int \exp(\imath uv)dv = 2\pi\delta(u).$$

Equation 3 clarifies a very important fact: *the spectrum $\hat{F}$ can be separated from the replicas even though the spatial spectrum $F_0(\xi)$ ranges beyond the Nyquist frequency.* Figure 2 illustrates this situation. The replicas can be filtered out as shown in the figure if velocity v_0 is known. Fortunately, the velocity can be easily calculated from animation data in graphics applications. Thus, an ideal anti-aliasing filter in this case looks like[1]

$$\hat{G}_v(\xi,\omega) = 2\pi\delta(v_0\xi + \omega). \tag{4}$$

The linear filtering in the Fourier domain $\hat{G}_v\hat{F}_s$ is equivalent to convolution in real time-space, that is,

$$\int\int f_s(x,t)\delta((x_0 - x) - (t_0 - t)v_0)dxdt. \tag{5}$$

This motivates us to study more general cases.

[1]Strictly speaking, the filter $\hat{G}_v$ involves a convergence problem because infinite animation sequences are assumed here. This will be solved in the next section.

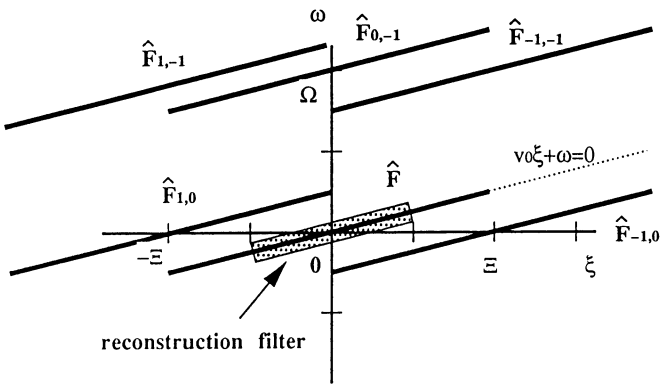


Figure 2: Spatio-temporal spectrum of constant velocity motion.

3.3 General Motion

Let us consider general motion. When the image point x_0 at t_0 moves to x_1 at t_1, we denote the motion by

$$x_1 = \chi(t_1; x_0, t_0). \tag{6}$$

For example, the flow χ_v for constant motion is:

$$\chi_v(t; x_0, t_0) = x_0 + v(t - t_0).$$

Note that the reciprocity generally holds from the definition

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

To avoid convergence problems, a finite animation sequence should be considered. With the flow χ, the sequence can be described as:

$$f(x,t) = \begin{cases} f_0(\chi(t_0;x,t)) & \text{if } t \in [-T/2, T/2] \\ 0 & \text{otherwise,} \end{cases}$$

where T is the length of the animation. The sampled image sequence are represented by

$$\begin{aligned} f_s(x,t) &= f(x,t)\sum_{k,l}\delta(x - 2\pi k/\Xi)\delta(t - 2\pi l/\Omega), \\ \hat{F}_s(\xi,\omega) &= \sum_{n,m}\int_{-T/2}^{T/2} dt \int_{-\infty}^{\infty} f_0(\chi(t_0;x,t)) \\ &\quad \exp(\imath(\xi + n\Xi)x + \imath(\omega + m\Omega)t)dx \\ &= \sum_{n,m}\hat{F}_{n,m}. \end{aligned}$$

Next, let us consider the anti-aliasing filter g. Filtering for constant motion, Eq. 5, can be rewritten as

$$\int\int f_s(x,t)\delta(x_0 - \chi_v(t_0;x,t))dxdt.$$

By analogy, we set our filter kernel g as

$$\begin{aligned} g(x,t) &= (1/T)w(x_0 - \chi(t_0;x,t))(\partial\chi/\partial x)_{t_0,t} \\ &= (1/T)w(x_0 - \chi(t_0;x,t))D_\chi(t_0;x,t) \end{aligned} \tag{7}$$

for space-variant filtering at (x_0, t_0):

$$h(x_0, t_0) = \int\int f_s(x,t)g(x,t)dxdt. \quad (8)$$

Here, $w(x)$ represents some appropriate anti-aliasing filter, such as a sinc-function, Gauss function, box function, and so on. The factor $1/T$ is the normalization constant, and $D_\chi = (\partial\chi/\partial x)$ compensates for image magnification variation due to spatially non-uniform motion.

Now, we prove that the filtering defined by Eq. 8 becomes an ideal anti-aliasing filter in the limit that $T \to \infty$. From the Parseval Identity, Eq. 8 can be rewritten as

$$\begin{aligned} h(x_0,t_0) &= (1/2\pi)^2 \int\int \hat{F}_s(\xi,\omega)\hat{G}^*(\xi,\omega)d\xi d\omega \\ &= (1/2\pi)^2 \sum_{n,m} \int\int \hat{F}_{n,m}(\xi,\omega)\hat{G}^*(\xi,\omega)d\xi d\omega \\ &= \sum_{n,m} h_{n,m}, \end{aligned}$$

where $\hat{G}^*$ denotes the complex conjugate of $\hat{G}$. The function $\hat{G}$ is the spatio-temporal spectrum of g, calculated by

$$\begin{aligned} \hat{G}(\xi,\omega) &= (1/T)\int\int w(x_0 - \chi(t_0;x,t))(\partial\chi/\partial x) \\ &\quad \exp(\imath(\xi x + \omega t))dtdx \\ &= (1/T)\int\int w(x_0 - u) \\ &\quad \exp(\imath\chi(t;u,t_0)\xi)\exp(\imath\omega t)dudt, \end{aligned}$$

where $u = \chi(t_0; x, t)$. Then, the integral $h_{n,m}$ can be evaluated as

$$\begin{aligned} h_{n,m} &= 1/(2\pi)^2(1/T)\int_{-T/2}^{T/2} \exp(\imath(\omega + m\Omega)t_1)dt_1 \\ &\quad \int f_0(\chi(t_0;x_1,t_1))\exp(\imath(\xi + n\Xi)x_1)dx_1 \\ &\quad \int\int w(x_0 - u)\exp(-\imath\chi(t_2;u,t_0)\xi) \\ &\quad \exp(-\imath\omega t_2)dudt_2 \\ &\quad \int d\xi \int d\omega \\ &= (1/T)\int_{-T/2}^{T/2} \exp(\imath m\Omega t_1)dt_1 \\ &\quad \int f_0(\chi(t_0;x_1,t_1))\exp(\imath n\Xi x_1)dx_1 \\ &\quad \int\int w(x_0 - u)\delta(t_1 - t_2)\delta(x_1 - \chi(t_2;u,t_0))dudt_2 \\ &= \int w(x_0 - u)f_0(u)du \\ &\quad \int_{-T/2}^{T/2} \exp(\imath n\Xi\chi(t_1;u,t_0)\exp(\imath m\Omega t_1)dt_1/T, \end{aligned}$$

where we used the reciprocity $\chi(t_0;\chi(t_1;u,t_0),t_1) = u$. Consequently,

$$\lim_{T\to\infty} h_{n,m} = \int w(x_0 - u)f_0(u)du(\lim_{T\to\infty} K_n(m\Omega;u)/T),$$

where $K_n(\omega;u)$ is the Fourier transform of the function k_n,

$$k_n(t;u) = \exp(\imath n\Xi\chi(t;u,t_0)).$$

Obviously,

$$h_{0,0} = \int w(x_0 - u)f_0(u)du.$$

On the other hand, when K_n is not singular at $\omega = m\Omega$, the aliasing pattern tends to 0, as

$$\lim_{T\to\infty} h_{n,m} = 0.$$

This completes the proof.

Note that $K_n(m\Omega, u)$ can be singular when, for example, motion is periodic with a frequency of $(m\Omega/n)$, or constant motion with a velocity of $(m\Omega/n\Xi)$.

3.4 Discrete Filtering

The filtering Eq. 8 can also be represented in a discrete form. By setting $\Xi = 2\pi$ and $\Omega = 2\pi$ (1 sample/pixel/frame sampling), we have

$$\begin{aligned} h(x_0,t_0) &= \int\int f_s(x;t)g(x,t)dxdt \\ &= (1/T)\int_{-T/2}^{T/2} dt \int_{-X/2}^{X/2} f(x;t)g(x,t) \\ &\quad \sum_{k,l} \delta(x-k)\delta(t-l)dtdx \\ &= (1/T)\sum_{k=-X/2}^{X/2}\sum_{l=-T/2}^{T/2} f(k;l)w(x_0 - \chi(t_0;k,l)) \\ &\quad D_\chi(t_0;k,l) \end{aligned} \quad (9)$$

for T-frame image sequences at the X pixel image resolution. Since Eq. 9 is a finite weighted sum of the sampled images, it can be directly computed.

The magnification factor $D_\chi = (\partial\chi/\partial x)$ compensates for image distortion due to non-uniformity of motion flow. For spatially uniform motion (more generally, incompressible flow), $D_\chi \equiv 1$. Furthermore, since $D_\chi(t_0;t,x) \to 1$ as $t \to t_0$, we can assume

$$D_\chi \simeq 1,$$

when the filter size T is small. If non-uniformity is not negligible, we have to evaluate D_χ point by point. Analytic formulae for D_χ are given in the Appendix.

For practical implementation, we slightly modify the filtering equation Eq. 9. By assuming local uniformity of motion flows, we have

$$h(x_0,t_0) = (1/T)\sum_{k,l} f(k;l)w(\chi(l;x_0,t_0) - k), \quad (10)$$

where we used the uniformity $x - y = \chi(t;x,t') - \chi(t;y,t')$. The advantage of Eq. 10 over Eq. 9 is that only one flow $\chi(l;x_0,t_0)$ should be traced for (x_0,t_0) rather than all flows $\chi(t_0;l,k)$.

The normalization factor $(1/T)$ relies on

$$\lim_{T\to\infty}(1/T)\sum_{l=-T/2}^{T/2}\sum_{k} w(\chi(l;x_0,t_0)-k)=1,$$

and would cause a normalization problem for finite T. Thus, it is better to adopt explicit normalization such as

$$h(x_0,t_0)=\sum_{k,l} f(k;l)w(\chi(l;x_0,t_0)-k)/\sum_{k,l} w(\chi(l;x_0,t_0)-k). \tag{11}$$

4 Algorithm

This section shows a simple algorithm for applying the anti-aliasing filter. When only one velocity field occupies the image, the only problem is to calculate the flow $\chi(t;x_0,t_0)$. However, when more than two velocity fields overlap, the filter should be separately applied because the theories rely on the uniqueness of the field. This happens when the projections of differently moving objects overlap (Figure 3).

Thus, the keys to the implementation are how to evaluate the motion flow χ and how to separate fields of different velocity. To deal with multiple flows, we adopt the filtering equation Eq. 11, assuming local uniformity of flows.

Data From animation models, we receive animation data for the sequence, such as transformation of objects and camera parameters. From the rendering process, RGB values, z-values, and object-id values are provided for each pixel at each frame, for example, in the form of G-buffers [SAITO]. Here, the object-id's are only used to identify object motion, and can be omitted for walk-through scenes. Let us denote these values for the pixel (k_x,k_y) at the frame l by rgb$[k_x][k_y][l]$,z$[k_x][k_y][l]$, and id$[k_x][k_y][l]$, respectively.

As the work space to capture multiple flows, we have a list structure of rgb, z, and α for each pixel, denoted by rgb_{flow}[ix][iy], $z_{flow}[ix][iy]$, and $\alpha_{flow}[ix][iy]$.

Pixel Tracing We now treat two-dimensional images. Let us denote two-dimensional vectors and their x,y-components by using the arrow and suffix notation, such as $\vec{k}=(k_x,k_y)$.

The motion flow, $\vec{\chi}(l;\vec{k}_0,l_0)=(\chi_x,\chi_y)$, corresponding to the sample point $\vec{k}_0=(k_{0x},k_{0y})$ at $t=l_0$, can be easily calculated from the animation information, the object-id, A, and the z-value (Figure 3). Let the transformation from the object coordinate of Object A to the screen space at t be $T_A(t)$. Then, the corresponding object point $p_A=(x_A,y_A,z_A,w_A)$ is given by

$$p_A=(k_{0x},k_{0y},z[k_{0x}][k_{0y}][l_0],1)T_A^{-1}(l_0).$$

At $t=l$, the object point p_A is projected by $T_A(l)$, and thus, the flow $\vec{\chi}$ and the corresponding depth $\zeta(l;\vec{k}_0,l_0)$ can be calculated as

$$\begin{aligned}\vec{\chi}(l;\vec{k}_0,l_0) &= (x/w,y/w)\\ \zeta(l;\vec{k}_0,l_0) &= z/w.\\ (x,y,z,w) &= p_AT_A(l_0)\\ &= (k_{0x},k_{0y},z[k_{0x}][k_{0y}][l_0],1)\\ &\quad T_A^{-1}(l_0)T_A(l)\end{aligned} \tag{12}$$

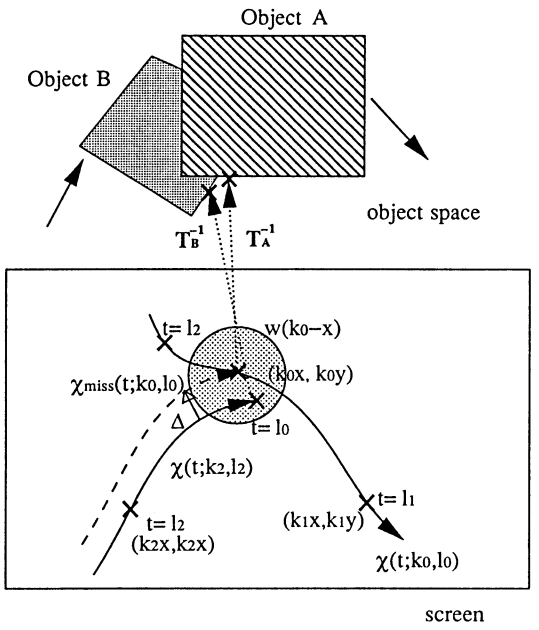


Figure 3: Pixel-tracing.

When the sample point misses an object, the filtering can be applied in the following way. In the example in the Figure 3, Object B fails to hit the sample point at $\vec{k}_0,t=l_0$, but pixel tracing from $\vec{k}_2=(k_{2x},k_{2y}),t=l_2$, reveals that Object B should exist in the reconstruction area of $\vec{k}_0,t=l_0$ because the traced point $\chi(l_0;\vec{k}_2,l_2)$ lies in the area. Therefore, we trace the flow for $\vec{k}_0$ by

$$\vec{\chi}_{miss}(l;\vec{k}_0,l_0)=\vec{\chi}(l;\vec{k}_2,l_2)+\Delta, \tag{13}$$

where

$$\Delta=\vec{k}_0-\vec{\chi}(l_0;\vec{k}_2,l_2).$$

Separation and Summation To separate different velocity fields, we adopt a simple rule, that is, *when the difference between two flows is smaller than some threshold, we regard them as the same flow.*

This can be described as follows. If both of the inequalities,

$$\|\vec{k}_0-\vec{\chi}(l_0;\vec{k}_1,l_1)\|<d_{th}, \tag{14}$$

and

$$\|\vec{k}_1-\vec{\chi}(l_1;\vec{k}_0,l_0)\|<d_{th}, \tag{15}$$

hold, the two sampled data are judged to belong to the same flow. Here, $\|\cdot\|$ denotes some norm on the image plane. The threshold d_{th} can be determined, for example, according to the diameter of the support of the filter kernel $w(\vec{x})$.

In the example in Figure 3, Inequality 15 is not true, so the two samples are processed as different flows. Note that

the projection points of the same object do not necessarily belong to the same flow because of perspective.

With this criteria function, same_flow(), which returns 1 when two samples are judged as being in the same flow and 0 otherwise, the actual filtering for each velocity flow becomes

$$\begin{aligned} \mathrm{rgb}_{flow} &= \sum_{l}\sum_{\vec{k}} \mathrm{same\_flow}(\vec{k}, l; \vec{k}_0, l_0) w(\vec{\chi}(l; \vec{k}_0, l_0) - \vec{k}) \\ &\quad \mathrm{rgb}[k_x][k_y][l] / \sum \mathrm{same\_flow}() w() \\ \alpha_{flow} &= \sum \mathrm{same\_flow}() w() / \sum w(), \\ z_{flow} &= \min(z_{flow}, \zeta(l; \vec{k}_0, l_0)) \end{aligned} \tag{16}$$

Here, α represents the 'coverage' of this flow. Note that the summation with $\vec{k}$ can be calculated only in a neighborhood of the flow $\vec{\chi}$ when the filter $w(\vec{x})$ is compactly supported.

At each pixel, we store the calculated RGB values, α-values, and z-values for all flows as a list, like the A-buffer structure [CARPENTER]. After the filtering, we sort the list with respect to the z-values at each pixel, and the final RGB values are determined by simple α-blending in the order of the sorted list, from near to far,

$$\mathrm{rgb}_{final} = \alpha_{flow1}\mathrm{rgb}_{flow1} + (1-\alpha_{flow1})\alpha_{flow2}\mathrm{rgb}_{flow2} + \cdots. \tag{17}$$

Procedure The procedure for filtering the frame l_0 can be summarized in the following way.

1) For each sample, $\vec{k}_0$, at frame $l = l_0$, do the following for all frames l within the filter.
   i) Calculate $\vec{\chi}(l; \vec{k}_0, l_0)$ according to Eq. 12.
   ii) Calculate the weighted sum according to Eq. 16.
2) For $l \neq l_0$ within the filter, do the following for all the samples, $\vec{k}$.
   i) Calculate the flow $\vec{\chi}(l_0; \vec{k}, l)$ according to Eq. 12.
   ii) For all samples $\vec{k}_0'$ at l_0 such that $w(\vec{k}_0' - \vec{\chi}(l_0; \vec{k}, l)) \neq 0$, do the following.
      a) If $\vec{k}, l$ belongs to any flow listed for $\vec{k}_0', l_0$, skip b) and c).
      b) Calculate the flow $\vec{\chi}_{miss}(l'; \vec{k}_0', l_0)$ according to Eq. 13, and calculate the weighted sum according to Eq. 16.
      c) Append the result to the list of $\vec{k}_0', l_0$.
3) For each pixel at l_0, sort the resulting list with respect to the z-values and apply alpha-blending according to Eq. 17.

As this filter involves a pixel tracing process, we call it the *pixel-tracing filter*.

5 Experiments and Discussion

The first experiment shows the influence of filter size, T. The test pattern is an almost horizontal (1 degree from the horizon) thin rectangle with the width of 1/8 pixel, constantly moving in the vertical direction at a speed of $0.22 = 11/50$ pixel/frame. Note that $\lim_{T\to\infty}(h_{50,11}/T) \neq 0$.

Figures 4-a, -b, and -c show the original image, and the results of applying filters of various sizes. As the filter kernel $w(x)$, we used the box function with one pixel area. Figures 4-d and -e show the analytic solution and the root-mean-square error of the filtered results with respect to the filter size. As shown in the figure, effective anti-aliasing was achieved with a filter size 128. The remaining error comes from the replica $\hat{F}_{50,11}$.

The second experiment shows an example of non-uniformly accelerated motion. The test pattern is rotating radial thin rectangles. As shown in Figure 5, aliasing was mostly removed with the 32-frame filter.

The final experiment shows application to a more practical image sequence taken from a walk-through scene in 'Také Tera.' In the sequence, only the camera moves and everything else is fixed in space. The original images were synthesized by using the GL library on the IRIS workstations at a resolution of 256 × 256. The scene consists of about 4M polygons. Figure 6 demonstrates the efficiency of the algorithm, where the severe aliasing artifacts seen in the original image were largely removed. The filter size was 16 frames.

The execution time is about 30 CPU seconds per frame on an IRIS Crimson R4000-50 at 256 × 256 resolution. The computation cost is directly proportional to $n_x \times n_y \times n_t$, where $n_x \times n_y$ is the image resolution, and n_t is the filter size. Considering that frame-by-frame recording onto a VCR takes about thirty seconds per frame, this powerful anti-aliasing is almost free! Furthermore, as the filtering is simple image processing with pixel-level independence, it might be possible to design parallel hardware to execute it in real-time, which would be attractive for visual simulators and virtual reality applications.

Future work includes application to reflected/refracted images, and coupling with stochastic sampling techniques. The algorithm relies on transformation between the screen space and the object space. Although conventional ray tracers cannot provide the transformation, the beam tracing/pencil tracing approach [HECKBERT,SHINYA] can calculate it in the form of system matrices and thus may be applicable to the filtering.

Since stochastic sampling techniques are powerful tools for anti-aliasing, it is an attractive idea to combine the two approaches. If we jitter the sample point of each pixel at each frame, the pixel-tracing filter acts exactly as a purely spatial filter for objects that are steady on the image plane. This means that spatial stochastic super-sampling can be performed by the pixel-tracing filter with only one point per pixel per frame sampling. This could also reduce the problems with constant velocity motion in the case of $n\Xi v_0 = m\Omega$, which we observed in Figure 4.

6 Conclusion

A new type of efficient anti-aliasing filter, the pixel-tracing filter, was proposed for animation sequences. The filter sums sub-pixel information using the correlation among images calculated from animation information. Theoretical studies prove the ability of the filter, and experimental results demonstrate the efficiency.

The algorithm is simple image processing implemented as post-filtering. The computational complexity is of constant order with regard to the complexity of scenes (e.g., number of polygons). With the pixel-tracing filter, effective anti-aliasing can be completed for animation sequences with a very modest computational cost.

Acknowledgments

The author would like to thank the Siggraph reviewers, whose comments greatly contributed to improving the theoretical part. He also wishes to thank Takahiko Kamae, Rikuo Takano, and Kazuyoshi Tateishi for their administrative support and encouragement, Atsushi Kajiyama for his technical support, and Toki Takahashi, Taka Saito, and Toshi Tanaka for helpful discussion.

References

[CARPENTER] Loren Carpenter, 'The A-buffer, An Antialiased Hidden Surface Method,' Computer Graphics **18**, No.3, pp.103-108, 1984.

[CATMULL84] Edwin Catmull, 'An Analytic Visible Surface Algorithm for Independent Pixel Processing,' Computer Graphics **18**, No.3, pp.109-115, 1984.

[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.

[FOLEY] James D. Foley, Andies van Dam, Steven K. Feiner, John F. Hughes, 'Computer Graphics Principal and Practice,' Addison-Wesley, 1990.

[GRANT] Charles W. Grant, 'Integrated Analytic Spatial and Temporal Anti-Aliasing for Polyhedra in 4-Space,' Computer Graphics **19**, No.3, pp.79-84, 1985.

[HAEBERLI] P. Haeberli, K. Akeley, 'The Accumulation Buffer: Hardware Support for High-Quality Rendering,' Computer Graphics, **24**, No.4, pp.309-318, 1990.

[HECKBERT] P. S. Heckbert, P. Hanrahan, 'Beam Tracing Polygonal Objects,' Computer Graphics, **18**, No.3, pp.119-128, 1984.

[LEE] Mark E. Lee, Richard A. Redner, and Samuel P. Uselton, 'Statistically Optimized Sampling for Distributed Ray Tracing,' Computer Graphics **19**, No.3, pp.61-67, 1985.

[MITCHELL] D. Mitchell, 'Spectrally Optimal Sampling for Distributed Ray Tracing,' Computer Graphics **25**, No.4, pp.157-164, 1991.

[NETRA] A. N. Netravali and B. G. Haskell, 'Digital Pictures - Representation and Compression,' Prenum Press, 1988.

[SAITO] Takafumi Saito and Toki Takahashi, 'Comprehensible Rendering of 3-D Shapes,' Computer Graphics **24**, No.4, pp.197-206, 1990.

[SHINYA] M. Shinya, T. Takahashi, and S. Naito, 'Principles and Applications of Pencil Tracing,' Computer Graphics, **21**, No.4, pp. 45-54, 1987.

Appendix: Calculation of D_χ

Here, we derive D_χ in Eq. 9 for two-dimensional images. We assume motion flows $\vec{\chi}(t; x_0, y_0, t_0)$ represented by Eq. 12. In the two-dimensional case, D_χ becomes the Jacobian of $\vec{\chi}$,

$$\begin{aligned} D_\chi &= \begin{vmatrix} (\partial\chi_x/\partial x_0)_{y0} & (\partial\chi_x/\partial y_0)_{x0} \\ (\partial\chi_y/\partial x_0)_{y0} & (\partial\chi_y/\partial y_0)_{x0} \end{vmatrix} \\ &= (\partial\chi_x/\partial x_0)_{y0}(\partial\chi_y/\partial y_0)_{x0} \\ &\quad -(\partial\chi_x/\partial y_0)_{x0}(\partial\chi_y/\partial x_0)_{y0}. \end{aligned}$$

By setting the translation matrix

$$T_A^{-1}(t_0)T_A(t) = \{\tau_{ij}\},$$

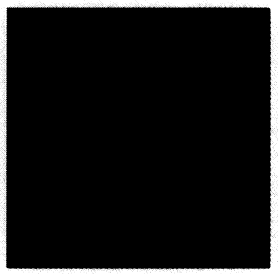
the partial deviation $(\partial\chi_x/\partial x_0)_{y0}$, etc., can be calculated as

$$\begin{aligned} (\partial\chi_x/\partial x_0)_{y0} &= (1/w)(\partial x/\partial x_0)_{y0} - (x/w^2)(\partial w/\partial x_0)_{y0} \\ &= (1/w)(\tau_{11} - (n_x/n_z)\tau_{31}) - (x/w^2) \\ &\quad (\tau_{14} - (n_x/n_z)\tau_{34}), \end{aligned}$$

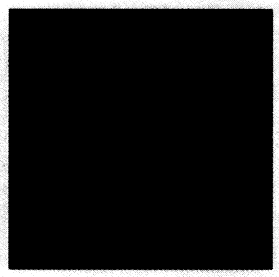
where $\vec{n} = (n_x, n_y, n_z)$ is the normal vector of the object surface at p_A, and we used

$$\begin{aligned} (\partial x/\partial x_0)_{y0} &= (\partial x/\partial x_0)_{y0,z0} + (\partial z_0/\partial x_0)_{y0}(\partial x/\partial z_0)_{x0,y0} \\ &= \tau_{11} - (n_x/n_z)\tau_{31}, \end{aligned}$$

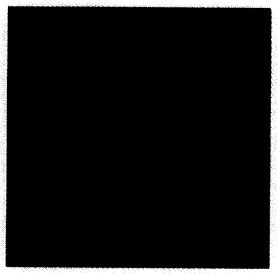
and so on.



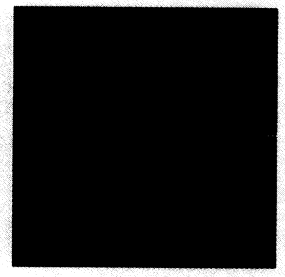
(a) Original



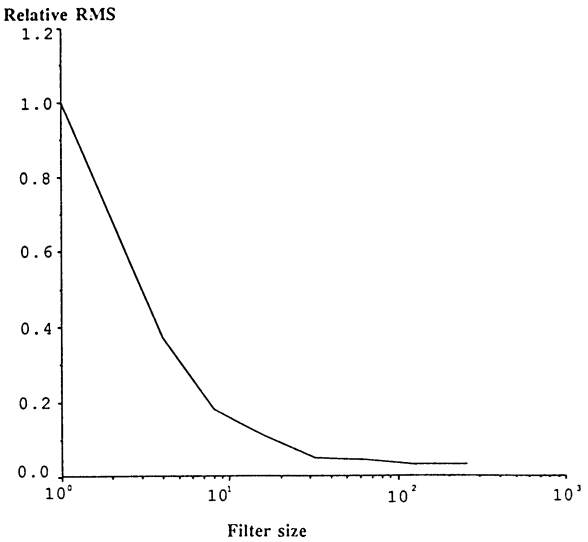
(b) Filter size = 32



(c) Filter size = 64

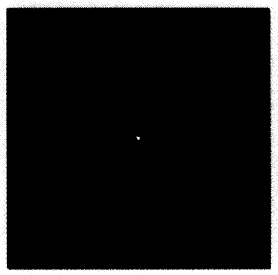


(d) Ananlytic solution

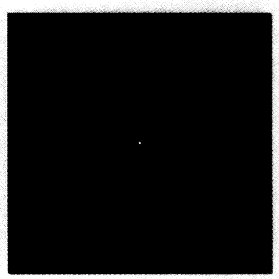


(e) Relative root-mean-square error

Figure 4: Thin rectangle.



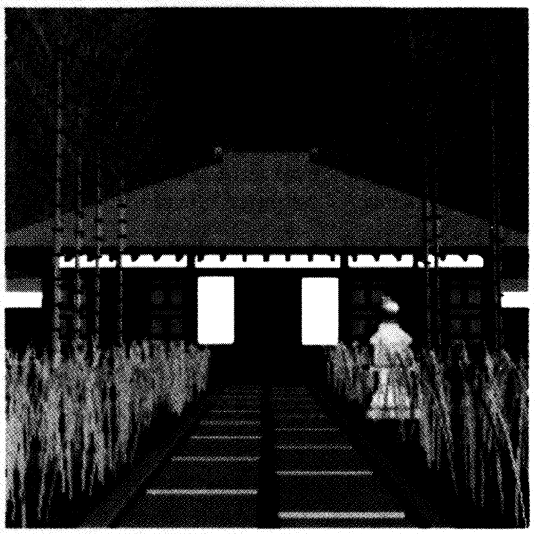
(a) Original image



(b) Filtered image

Figure 5: Rotating thin rectangles.

(a) Original image



(b) Filtered image

Figure 6: Také Tera(Bamboo Temple).