Study of Supersampling Methods for Computer Graphics Hardware Antialiasing

Michael E. Goss and Kevin Wu

michael_e_goss@hp.com

Visual Computing Department
Hewlett-Packard Laboratories
Palo Alto, California

December 5, 2000

Abstract

A study of computer graphics antialiasing methods was done with the goal of determining which methods could be used in future computer graphics hardware accelerators to provide improved image quality at acceptable cost and with acceptable performance. The study focused on supersampling techniques, looking in detail at various sampling patterns and resampling filters.

This report presents a detailed discussion of theoretical issues involved in aliasing in computer graphics images, and also presents the results of two sets of experiments designed to show the results of techniques that may be candidates for inclusion in future computer graphics hardware.

Table of Contents

1. Introduction 1
1.1 Graphics Hardware and Aliasing 1
1.2 Digital Images and Sampling 1
1.3 This Report 2
2. Antialiasing for Raster Graphics via Filtering 3
2.1 Signals and the Fourier Transform 3
2.2 Systems 5
2.3 Practical Resampling Filters 8
2.4 Application of Resampling Filters 10
2.5 Results 13
3. Description of Supersampling Pattern Experiments 17
3.1 Software Implementation 18
3.2 Supersample patterns 18
3.2.1 Regular Grid 18
3.2.2 Irregular Grid 18
3.2.2.1 N-Queens Supersampling 18
3.2.2.2 Edge-Optimized Irregular Grid 19
3.2.3 Jittered Grid 20
3.3 Resampling filters 22
3.3.1 Box Filter 23
3.3.2 Tent Filter 23
3.3.3 Gaussian ½ Filter 24
3.4 Results 24
3.4.1 Box Filter 24
3.4.1.1 Conventional Aliased Rendering 25
3.4.1.2 Regular Grid, Box Filter 26
3.4.1.3 N-Queens Irregular Grid, Box Filter 26
3.4.1.4 Edge-Optimized Irregular Grid, Box Filter 27
3.4.1.5 Jittered Grid, Box Filter 28
3.4.2 Tent Filter 29
3.4.3 Gaussian ½ Filter 31
3.4.4 Filter Comparisons 33
3.4.5 Limits to Image Quality 34
3.4.6 Animation Results 35
4. Conclusions 36

Table of Figures

Figure 1: Rasterization systems. (a) Conventional point sampling. (b) Ideal antialiasing with filtering. (c) Practical antialiasing with supersampling and filtering. 6
Figure 2: Resampling filter impulse responses and normalized transfer functions. 9
Figure 3: Supersample locations (blue crosses) relative to final sample locations (red filled diamonds). Supersample rate $N = 2$ in left diagram; $N = 3$ in right diagram. 11
Figure 4: Configuration of 10 unique weights in an 8×8 even resampling filter. Each unique weight has a unique color and letter code. 13
Figure 5: Pointed-sampled checkerboard pattern. 14
Figure 6: Antialiased checkerboard pattern. When viewing the page after a 90º clockwise rotation, the supersample rates from top to bottom are 2×2, 4×4, and 8×8, and the resampling-filter functions from left to right are rectangle, triangle, and cubic. 15
Figure 7: Subjective antialiasing quality for the supersample rates and filters in Figure 6. 16
Figure 8 – Frequency spectra for Poisson Disc and Uniform Jitter sample patterns 21
Figure 9: Regular Grid, Box Filter 25
Figure 10: N-Queens Irregular Grid, Box Filter 27
Figure 11: Edge-Optimized Irregular Grid, Box Filter 28
Figure 12: Jittered Grid, Box Filter 29
Figure 13: Regular Grid, Tent Filter 30
Figure 14: N-Queens Irregular Grid, Tent Filter 30
Figure 15: Jittered Grid, Tent Filter 31
Figure 16: Regular Grid, Gaussian ½ Filter 31
Figure 17: N-Queens Irregular Grid, Gaussian ½ Filter 32
Figure 18: Jittered Grid, Gaussian ½ Filter 32
Figure 19: 4 sample per pixel filter comparison 33
Figure 20: 16 sample per pixel filter comparison 34
Figure 21: 256 samples per pixel, Jittered Grid, ... 35

1. Introduction

1.1 Graphics Hardware and Aliasing

Graphics hardware accelerators have, almost without exception, used incremental raster conversion of lines and polygons as the primary method of rendering images. An undesirable side effect of this technique is the generation of "jaggies," ragged stair-step patterns apparent on edges that are not oriented at exact multiples of 45 degrees. Aggregations of very small polygons also generate somewhat unpredictable results when the polygon size comes close to or below the size of one pixel. These effects are typically worsened in animation sequences, where false apparent motion of these artifacts proves very distracting to the viewer.

These artifacts are two of the most common effects of *aliasing*. Aliasing is defined to occur when a continuous signal (in this case the image projection of the geometric model of a scene) is sampled (converted to discrete pixels) using sample locations (pixels) that are too widely spaced relative to the smallest details of the scene. Because mathematically defined lines and edges contain infinitely sharp edges (and thus infinitely small detail), it is not possible to exactly represent a computer graphics image using pixels. Antialiasing techniques attempt to generate approximations that more closely resemble the ideal scene by minimizing the effects of aliasing.

1.2 Digital Images and Sampling

A digital image is a collection of point samples that can be used to reconstruct a continuous[*] image to be viewed. Pixels are sometimes characterized as area samples, but since each pixel is represented by a single value (per color channel), the area sample characterization is not a very accurate formalization [1].

The typical placement of point samples for image sampling is on a square grid. If the sampling frequency in each direction is greater than twice the maximum frequency present in the continuous image that is sampled, then it is possible to exactly reconstruct that original continuous image from the samples. An image meeting this criterion relative to a particular sampling frequency is said to be *band-limited* relative to that sampling frequency.

In practice, it is rare that the image to be sampled is, in fact, band-limited. In the most common case for graphics hardware, the image to be sampled is made up of lines and polygons projected on to the view plane. Any sharp edge, such as a line or polygon edge, contains non-zero amplitude at any arbitrarily high frequency. This implies that it is not possible to use samples to

[*] An image is considered to be a two-dimensional function of image coordinates. A *continuous* image has a valid value at any coordinates within the image space, a *digital* image has a valid value only at integer coordinate (pixel) locations. For gray scale images, the function is single-valued. For color images, we generally consider each color channel to be a separate image function.

exactly represent any image that has sharp edges. The best that we can do in this situation is to generate sample values that represent a reasonable approximation to the ideal image.

In theory, the correct method for avoidance of aliasing when sampling a signal is to pass the signal through a low-pass filter *before* any sampling is done. The low-pass filter removes any frequencies greater than or equal to half the sampling frequency. The samples are then able to exactly represent the filtered signal without any aliasing. Correct sampling requires computation of the 2D projection of all 3D geometric primitives in a scene, determination of the visible parts of each primitive, and then formation of a band-limited representation of the collection of visible partial primitives. Then a low-pass filter must be applied to this representation before sampling. Even if a practical method were available to implement this procedure, it would undoubtedly be complex and time-consuming to generate an image using this approach.

In practice, correct sampling is generally not feasible, so some approximation must be made. For most signals, an approximation method can take advantage of the fact that signal content generally declines as the frequency increases. The most common technique, *supersampling*, takes advantage of this by first sampling the signal at a frequency higher than the desired sample rate. This supersampled signal still has artifacts due to aliasing, but the artifacts are less prominent than if the signal was sampled at the target rate. A filter *reconstructs* a new continuous signal from the supersamples and *attenuates* the frequency content above a threshold so that the signal resampled at the target rate exhibits fewer aliasing artifacts.

Note that *super*sampling [2] is paradoxically also referred to as *sub*sampling [3]. These terms generally refer to the same process; we will use the term “supersampling” in this report.

1.3 This Report

This report describes two studies performed in support of the *Bonanza* graphics hardware design effort to determine the effectiveness of antialiasing techniques which have been described in the literature and appear feasible for hardware implementation. Section 2 describes a study of filters used in conjunction with supersampling, using supersample patterns on a regular grid (see Figure 3). The cost and effectiveness of various filters are compared, and subjective image quality is examined. Section 3 describes a study of sampling patterns and sample counts, examining the effectiveness of irregular and jittered grids for supersampling, in conjunction with supersampling filters.

2. Antialiasing for Raster Graphics via Filtering

Computer graphics on a raster display is the process of rendering a synthesized continuous-domain image at pixels in a regular grid. A *pixel* is a *point sample*; a pixel is not a little square [1]. A polygon renderer samples polygons at *points* on scan lines: this operation validates this definition of a *pixel*. Sampling of a continuous-domain signal leads to aliasing [1,3,4,5,6,7]. In raster graphics, aliasing is most noticeable as jagged silhouette edges, particularly in animations when the staircase patterns crawl along these edges. Aliasing also appears as changing moiré patterns during animation of fine periodic textures or geometric structures. These artifacts are distracting to viewers. The process of reducing the aliasing artifacts is known as *antialiasing*.

A full analysis of aliasing arising from sampling requires application of the Fourier transform, which yields a signal's frequency spectrum. Sampling of a continuous-domain signal replicates the signal's spectrum at multiples of the sampling frequency. Aliasing results when the sampling frequency is too low for the signal, resulting in overlap of the replicated and offset spectral lobes. An increase in the sampling frequency (or equivalently, a reduction in the sampling interval) can help to reduce aliasing by increasing the separation of spectral replicas in the frequency domain. However, many signals such as silhouette edges have significant high-frequency components so that a very high sampling frequency is needed to reduce aliasing. In raster graphics, the final resolution of a given screen has an upper bound so decreasing the sampling interval of the display is impractical.

For a given sampling frequency, the most effective means of reducing aliasing is to apply a low-pass filter to the signal. This operation reduces the high-frequency components, and when it precedes sampling, aliasing artifacts are less perceptible. The sampling theorem states that a function whose Fourier transform is zero for absolute frequencies $|f| > f_c$ is completely specified by samples taken at a uniform interval not greater than $1/(2f_c)$ except for any harmonic term with zeros at the sampling points [4]. A corollary of the sampling theorem is that the low-pass filter's ideal cutoff frequency is half of the sampling frequency. In all discrete signal processing applications, a tradeoff exists between quality and computational expense when performing antialiasing. The characteristics of the filter are among the considerations in the design of an antialiasing solution.

This section reviews three topics: the source of aliasing when sampling continuous-domain signals, filtering, and supersampling. Then it compares various filters: their shapes in the spatial and frequency domains, cost of implementation, and quality.

2.1 Signals and the Fourier Transform

Bracewell [4] gives a comprehensive presentation of the Fourier transform, both its theory and applications. This subsection adopts the function definitions and properties from that book.

In the context of signal processing, certain functions occur frequently in sampling and filtering applications. Some of these are defined in Table 1. The sampling function is an infinite train of uniformly spaced impulses, where the notation $\delta(x)$ denotes the Dirac impulse. The impulse can sample a function with unit weight at a point with infinitesimal extent. The impulse satisfies the following properties: $\delta(x) = 0$ for $x \neq 0$ and $\int_{-\infty}^{\infty} \delta(x)\,dx = 1$.

Table 1: Function Definitions

Function Name	Definition
Rectangle	$\Pi(x) = \begin{cases} 1 & \lvert x\rvert < 1/2 \\ 1/2 & \lvert x\rvert = 1/2 \\ 0 & \lvert x\rvert > 1/2 \end{cases}$
Triangle	$\Lambda(x) = \begin{cases} 1-\lvert x\rvert & \lvert x\rvert \le 1 \\ 0 & \lvert x\rvert > 1 \end{cases}$
Sinc	$\operatorname{sinc}(x) = \dfrac{\sin(\pi x)}{\pi x}$
Sampling	$\mathrm{III}(x) = \sum_{i=-\infty}^{\infty} \delta(x-i)$

The Fourier transform yields the frequency spectrum of a signal. Some important Fourier transform pairs and properties are in Table 2. The convolution operation is denoted by the asterisk symbol *. Some properties of the Fourier transform can be described as follows. Under the convolution property, convolution in one domain becomes multiplication in the other domain. The similarity property expresses that compressing a signal's domain expands the domain of the signal's Fourier transform pair and attenuates the magnitude. A property of sampling a continuous-domain signal is that the operation replicates the frequency spectrum at multiples of the sampling frequency.

Table 2: Fourier transform pairs.

Spatial Domain	Frequency Domain
$\Pi(x)$	$\operatorname{sinc}(f)$
$\operatorname{sinc}(x)$	$\Pi(f)$
$\Lambda(x)$	$\operatorname{sinc}^2(f)$
$\exp(-\pi x)$	$\exp(-\pi f)$
$\mathrm{III}(x)$	$\mathrm{III}(f)$
$g(x) * h(x)$	$G(f)H(f)$

$g(x)h(x)$	$G(f) * H(f)$
$g(ax)$	$\lvert a\rvert^{-1} G(f/a)$
$g(x)\,\mathrm{III}(Nx) = \frac{1}{N}\sum_{i=-\infty}^{\infty} g\left(\frac{i}{N}\right)\delta\left(x - \frac{i}{N}\right)$	$G(f) * \frac{1}{N}\mathrm{III}\left(\frac{f}{N}\right) = \sum_{k=-\infty}^{\infty} G(f - Nk)$

The antialiasing methods for images apply to two-dimensional (2D) signals and filters. For simplicity, the systems analyzed in the next subsection are one-dimensional (1D): their extension to 2D is straightforward. In addition, the filters are separable, meaning that a 2D function is the product of two 1D functions: $h(x, y) = a(x)b(y)$. For 2D filters, the individual 1D functions usually are identical: $h(x, y) = a(x)a(y)$.

2.2 Systems

Analysis of aliasing in the rasterization pipeline requires a model in the form of a systems diagram as shown in Figure 1(a). The continuous-domain input signal $g(x)$ represents the scene. The renderer samples the scene at uniform one-pixel intervals to produce the sampled signal $g^*(x)$, which is stored in the frame buffer. The reconstruction function $r(x)$ represents the video display circuitry and the physics of the display unit, which may be a cathode-ray tube, liquid-crystal display, or some other device. This function should be a good low-pass filter, which interpolates the discrete samples, but often the actual hardware is only a modest approximation to the ideal. Upon convolving the sampled signal with the reconstruction function, the final output signal $q(x)$ represents the displayed image seen by viewers.

The system in Figure 1(a) performs conventional point sampling. The sampled signal and its Fourier transform are as follows:

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

$$G^*(f) = G(f) * \mathrm{III}(f) = \sum_{n=-\infty}^{\infty} G(f - n)$$

The sampled signal's spectrum $G^*(f)$ is the sum of replicas of the original spectrum $G(f)$ at integral multiples of the sampling frequency of 1 cycle/pixel. If the original signal has spectral energy beyond 0.5 cycles/pixel, then the replicas overlap, resulting in aliasing. In computer graphics, this condition is true for silhouette edges so the jagged appearance is noticeable, particularly during animation.

Since the sampling frequency is fixed at 1 cycle/pixel, reduction of aliasing requires a low-pass filter with a sharp cutoff at 0.5 cycles/pixel. The ideal antialiasing system in Figure 1(b)

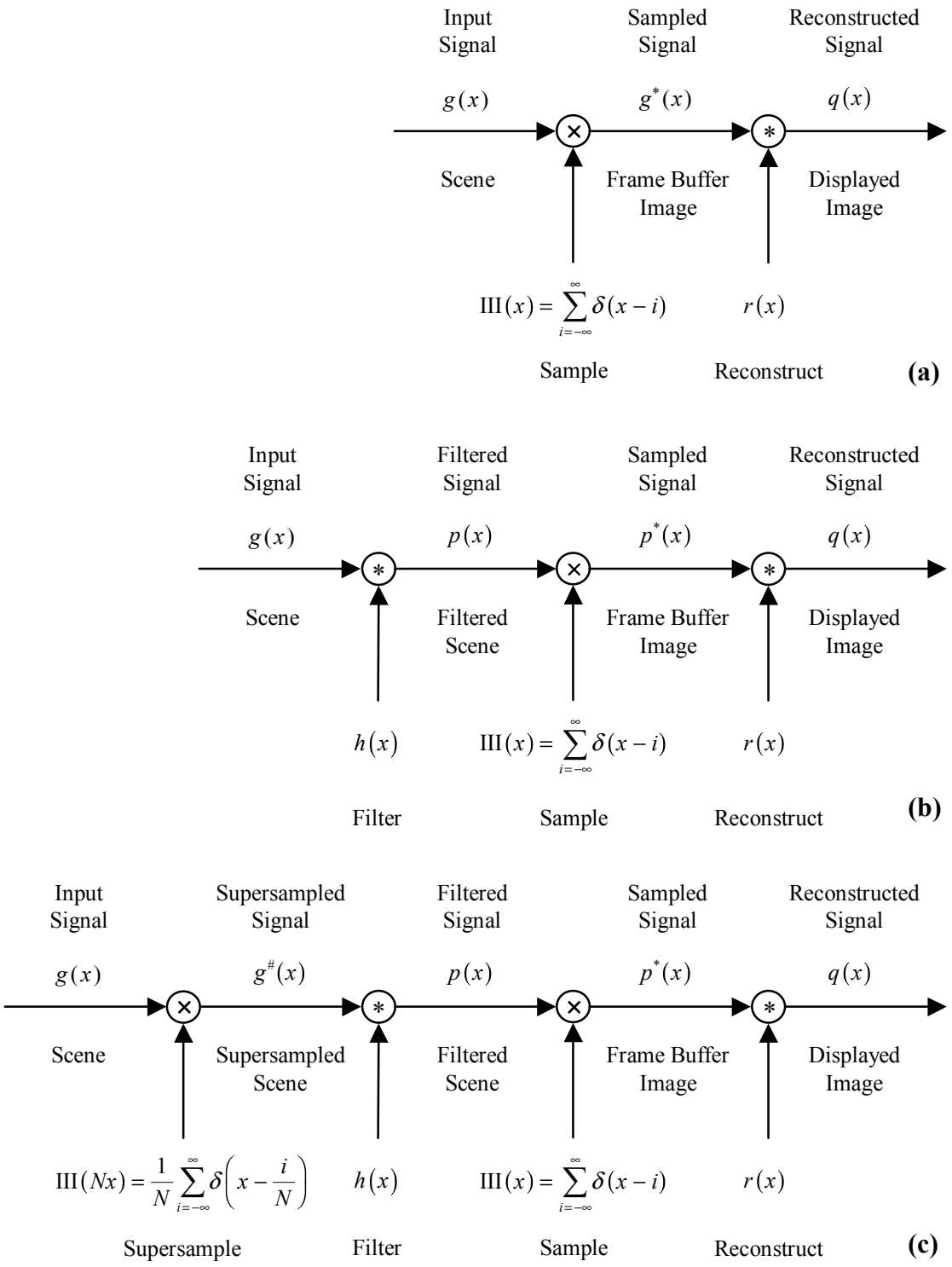


Figure 1: Rasterization systems. (a) Conventional point sampling. (b) Ideal antialiasing with filtering. (c) Practical antialiasing with supersampling and filtering.

convolves the input signal $g(x)$ with the filter impulse-response $h(x)$ to yield a filtered signal $p(x)$. This stage precedes sampling, which produces the sampled signal $p^*(x)$. If the filter is the ideal low-pass filter with transfer function $H(f) = \Pi(f)$, or equivalently, impulse response $h(x) = \text{sinc}(x)$, then this system eliminates the high frequency content that causes aliasing when sampling. In addition, if the reconstruction function is ideal, meaning that $r(x) = \text{sinc}(x)$, then the filtered signal $p(x)$ can be recovered exactly. However, implementation of the ideal antialiasing system is impractical because it requires convolution of two continuous-domain signals.

In the practical antialiasing system of Figure 1(c), a supersampling stage precedes the filtering stage of the ideal antialiasing system. Supersampling at a rate of *N* cycles/pixel yields the supersampled signal $g^{\#}(x)$, which is the new input to the filter $h(x)$, which we now call the *resampling filter* because it resides between two sampling stages. This system is simpler to implement because the convolution of a discrete signal $g^{\#}(x)$ with the continuous resampling-filter's impulse response $h(x)$ is a sum rather than an integral.

$$\begin{aligned} p(x) &= g^{\#}(x) * h(x) \\ &= \frac{1}{N} \sum_{i=-\infty}^{\infty} g\left(\frac{i}{N}\right) \delta\left(x - \frac{i}{N}\right) * h(x) \\ &= \frac{1}{N} \sum_{i=-\infty}^{\infty} g\left(\frac{i}{N}\right) h\left(x - \frac{i}{N}\right) \end{aligned}$$

Even better, the filtered signal $p(x)$ is sampled (at the final rate of 1 cycle/pixel) so it is required only at discrete locations:

$$p^*(x) = \sum_{j=-\infty}^{\infty} p(j)\, \delta(x - j)$$

$$p(j) = p(x)\big|_{x=j} = \frac{1}{N} \sum_{i=-\infty}^{\infty} g\left(\frac{i}{N}\right) h\left(j - \frac{i}{N}\right).$$

The discrete signal $p(j)$ is the output of a discrete convolution and the implementation is straightforward. In a graphics system, this is the signal stored in the frame buffer.

The practical antialiasing system is susceptible to aliasing. The supersample rate is *N* cycles/pixel so any signal energy above the frequency *N*/2 cycles/pixel results in aliasing. Fortunately, this threshold is *N* times higher than in the conventional point sampling system, and

aliasing artifacts are less noticeable in the antialiasing system because the envelope of the frequency response of most practical signals tends to decrease as the frequency increases.

The objective in designing the antialiasing system is to minimize the implementation's effort while maximizing the visual quality. The design parameters are the supersample rate *N* and the resampling filter $h(x)$.

2.3 Practical Resampling Filters

The resampling filter has a major role in an antialiasing system. The resampling filter is characterized by its shape and extent, which is finite in order to make the discrete convolution realizable and practical. The ideal resampling filter has the impulse response $h(x) = \text{sinc}(x)$ with Fourier transform $H(f) = \Pi(f)$, which has a flat pass-band, a sharp cutoff at $\pm 1/2$, and total attenuation in the stop-band. It is impractical to implement because its spatial domain extends to infinity and its envelope decays slowly at $O(1/x)$. Practical resampling filters seek to emulate the ideal filter. The two general categories are polynomial approximation or interpolation functions and windowed functions. The polynomial functions have a kernel consisting of a small sum of powers of the spatial coordinate. The windowed functions are the product of a function with infinite extent, typically $\text{sinc}(x)$, and a window function with finite extent. Some common window functions are the Hann/Hamming, Blackman, Kaiser, and Lanczos [6]. Table 3 lists some examples of polynomial and windowed functions. Wolberg [6] and Blinn [3] describe these functions in detail.

Table 3: Some Practical Resampling Filters

Filter Name	Function
Rectangle	$h(x) = \Pi(x)$
Triangle	$h(x) = \Lambda(x)$
Cubic	$h(x) = \begin{cases} (a+2)\lvert x\rvert^3 - (a+3)\lvert x\rvert^2 + 1 & 0 \le \lvert x\rvert < 1 \\ a\lvert x\rvert^3 - 5a\lvert x\rvert^2 + 8a\lvert x\rvert - 4a & 1 \le \lvert x\rvert < 2 \\ 0 & 2 \le \lvert x\rvert \end{cases}$
Lanczos-*M* Sinc	$h(x) = \text{sinc}\left(\frac{x}{M}\right)\Pi\left(\frac{x}{2M}\right)\text{sinc}(x)$
Blinn's Nice Function [3]	$h(x) = \text{sinc}\left(\frac{x}{4}\right)\Pi\left(\frac{x}{8}\right)\text{sinc}\left(\frac{3x}{4}\right)$
Lanczos-*M* Gaussian	$h(x) = \text{sinc}\left(\frac{x}{M}\right)\Pi\left(\frac{x}{2M}\right)\exp\left(-\frac{\pi x^2}{\sqrt{2}}\right)$

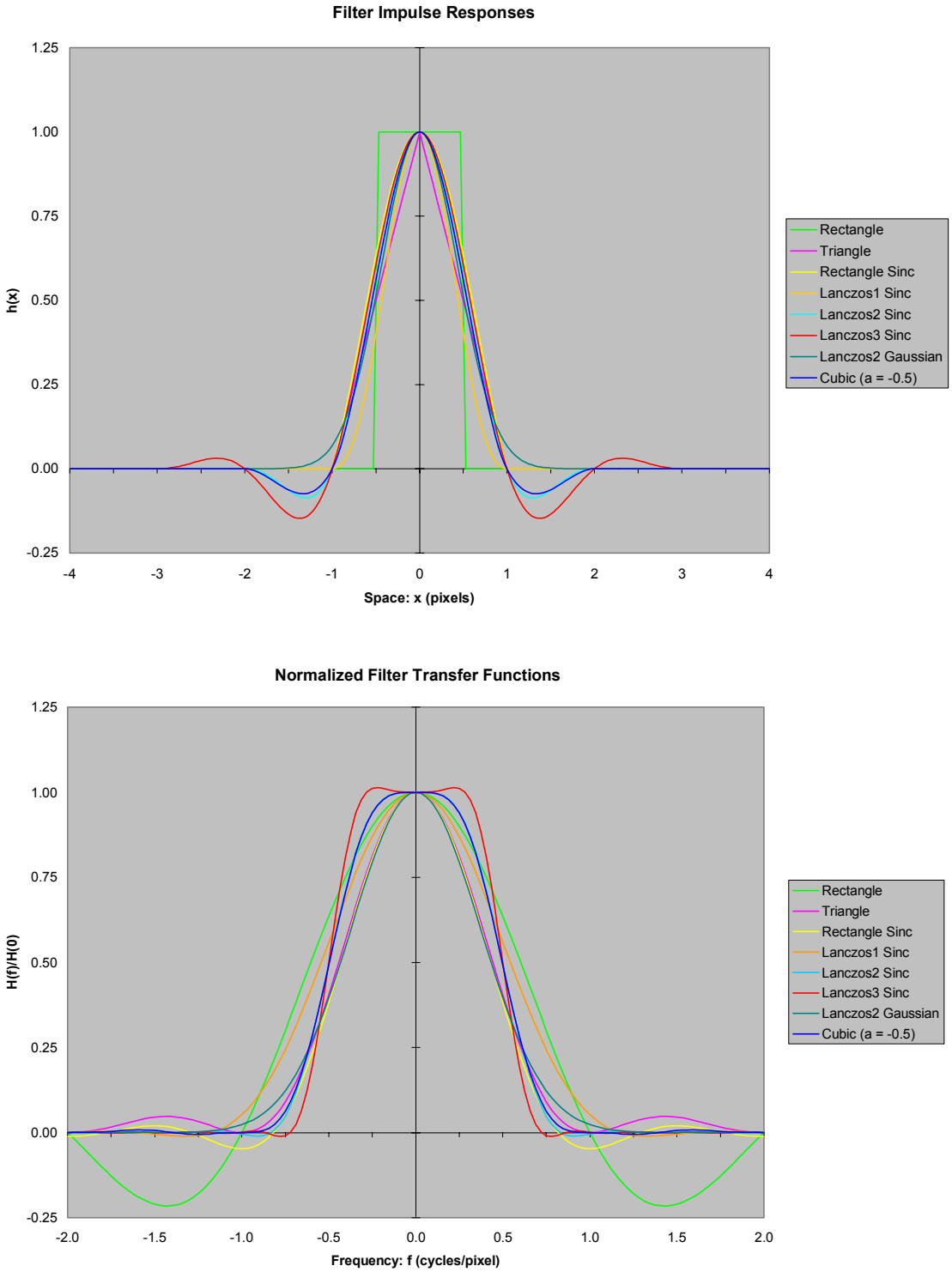


Figure 2: Resampling filter impulse responses and normalized transfer functions.

Figure 2 shows plots of the impulse responses and the Fourier transforms of some of these resampling filters. An impulse response with narrow extent has a low-cost implementation with discrete convolution, but it is a poor approximation to the ideal filter and the result is a poor transfer function. For example, the rectangle function has the narrowest extent $[-0.5, 0.5]$, and its transfer function $\text{sinc}(f)$ has a slow transition from pass-band to stop-band and large oscillations in the stop-band. Another example, the Lanczos3 Sinc filter, has a wide extent $[-3, 3]$. Its frequency response exhibits characteristics of a better low-pass filter: flatter pass-band, faster transition from pass-band to stop-band, and higher attenuation in the stop-band. Higher quality requires a resampling filter with a wide spatial extent.

Some antialiasing methods in computer graphics employ subpixel masks including the A-buffer [8] and its derivatives or improvements [9, 10, 11]. These methods attempt to determine the overlap between an area primitive at a silhouette edge and the little square enclosing a pixel, which is a point sample. The little square centered on a pixel at (i, j) has an edge length of 1 pixel and covers the area $\left\{(x, y) \,\middle|\, i - \frac{1}{2} \le x < i + \frac{1}{2},\ j - \frac{1}{2} \le y < j + \frac{1}{2}\right\}$ [1]. The filter for these methods is the rectangle function $h(x) = \Pi(x)$, which is the worst of the resampling filters in Figure 2. Better resampling filters extend outside of the little square. The extent for a resampling filter should be at least $[-1,1] \times [-1,1]$ as this is the area covered by the primary lobe of the ideal 2D resampling filter $h(x, y) = \text{sinc}(x)\text{sinc}(y)$.

2.4 Application of Resampling Filters

In the previous subsection, we have examined practical resampling filters independent of the sampling stages in the antialiasing system. Now consider the application of filters for sampling in a bounded region. Let the images generated in a computer graphics system span a space of $W \times H$ pixels. The pixels are located at $\left\{(x, y) \,\middle|\, x = 0, 1, \ldots, W-1; \quad y = 0, 1, \ldots, H-1\right\}$. In the little square model, the image occupies the space $\left\{(x, y) \,\middle|\, -\frac{1}{2} \le x < W - \frac{1}{2}, -\frac{1}{2} \le y < H - \frac{1}{2}\right\}$. The supersample rate in both the *x* and *y* directions is *N*. The image spanned by the supersampled pixels needs to be identical to that for the final pixels. This constraint establishes the locations of the supersampled pixels:

$$x = \frac{2i + 1 - N}{2N} \qquad i = 0, 1, \ldots, NW - 1$$
$$y = \frac{2j + 1 - N}{2N} \qquad j = 0, 1, \ldots, NH - 1$$

The supersample rate N does not need to be an integer. However, the implementation is simplified when N is a positive integer. The x and y offset of the nearest supersample relative to any final pixel is $1/(2N)$ when N is even or 0 when N is odd. Figure 3 shows examples of both cases.

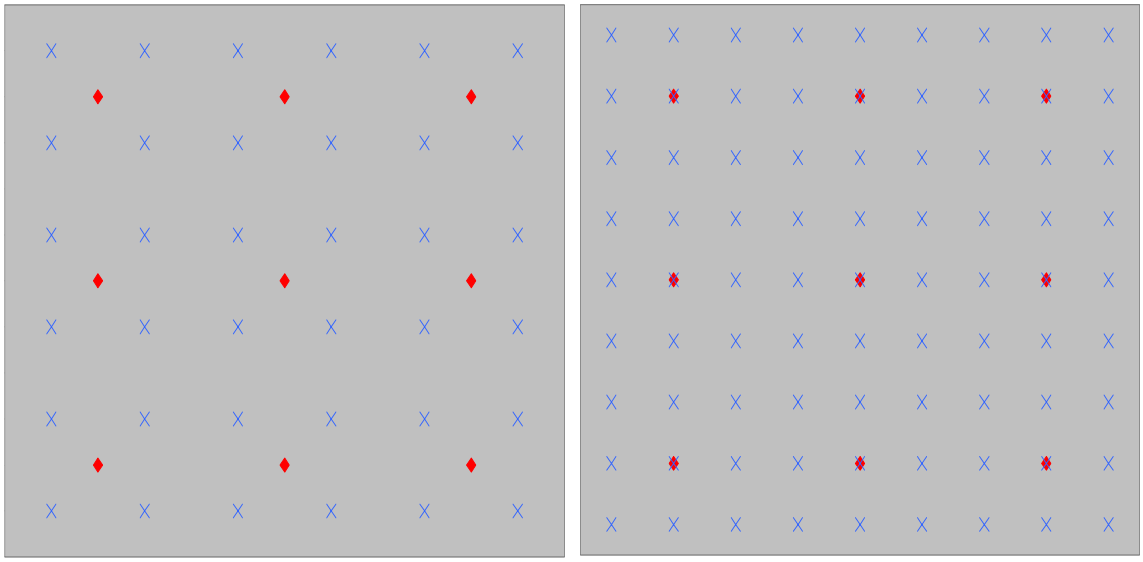


Figure 3: Supersample locations (blue crosses) relative to final sample locations (red filled diamonds). Supersample rate $N = 2$ in left diagram; $N = 3$ in right diagram.

Table 4 lists the kernel sizes of some examples of practical resampling filters with finite extent. Let the filter consist of $S \times S$ discrete values. The discrete convolution for application of a 2D filter with integer supersample rate N is as follows:

$$p(k,l) = \sum_{i=\frac{-S+N}{2}}^{\frac{S+N}{2}-1} \sum_{j=\frac{-S+N}{2}}^{\frac{S+N}{2}-1} g\left(k - \frac{2i+1-N}{2N}, l - \frac{2j+1-N}{2N}\right) \hat{h}\left(\frac{2i+1-N}{2N}, \frac{2j+1-N}{2N}\right)$$

$$k = 0, 1, \ldots, W-1$$

$$l = 0, 1, \ldots, H-1$$

where the normalized filter impulse response is

$$\hat{h}\left(\frac{2i+1-N}{2N}, \frac{2j+1-N}{2N}\right) = \frac{h\left(\frac{2i+1-N}{2N}, \frac{2j+1-N}{2N}\right)}{\sum_{m=\frac{-S+N}{2}}^{\frac{S+N}{2}-1} \sum_{n=\frac{-S+N}{2}}^{\frac{S+N}{2}-1} h\left(\frac{2m+1-N}{2N}, \frac{2n+1-N}{2N}\right)}.$$

Table 4: Resampling filter kernel sizes.

Filter	N is even	N is odd
Rectangle	$N \times N$	$N \times N$
Triangle	$2N \times 2N$	$(2N-1) \times (2N-1)$
Cubic	$4N \times 4N$	$(4N-1) \times (4N-1)$
Lanczos-M Sinc or Gaussian	$2MN \times 2MN$	$(2MN-1) \times (2MN-1)$
Blinn's Nice Filter [3]	$8N \times 8N$	$(8N-1) \times (8N-1)$

The reason for normalizing the impulse response is to make the gain of the filter unity in regions where the color does not vary. The normalized impulse response can be computed in advance when the supersample rate N is an integer because it is spatially invariant in convolution. Note that the 2D filters under consideration are separable: $h(x, y) = h(x)h(y)$.

All the previous examples of practical resampling filters are even functions: $h(x) = h(-x)$. This symmetry can help reduce the workload of performing the convolution. Let the resampling filter be real and even with a total of $S \times S$ discrete values. Then the maximum number of unique values is $U = \frac{1}{2}\left(\left\lceil \frac{S}{2} \right\rceil\right)\left(\left\lceil \frac{S}{2} \right\rceil + 1\right)$. Table 5 shows some examples of various filter kernel sizes. The number of unique values is much less than the total number of values in a filter kernel. By grouping the supersamples with a common filter weight, an implementation can reduce the number of multiplications per pixel from $S \times S$ to U. Figure 4 shows the configuration of 10 unique weights in an even 8×8 filter kernel.

Table 5: Number of weights in 2D even resampling filters.

Filter size S	Maximum unique values U	Total values $S \times S$
2	1	4
3	3	9
4	3	16
5	6	25
6	6	36
7	10	49
8	10	64
9	15	81
10	15	100

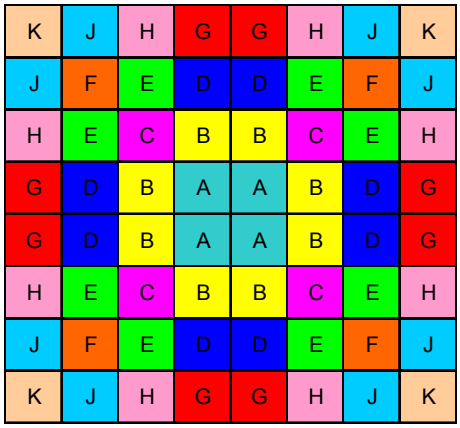


Figure 4: Configuration of 10 unique weights in an 8×8 even resampling filter. Each unique weight has a unique color and letter code.

2.5 Results

In a practical antialiasing system, the design parameters are the supersample rate and the resampling filter's extent and shape. Increasing the rate or extent increases the subjective quality at the expense of higher workload. A common test for antialiasing systems is a checkerboard pattern viewed in perspective at an oblique angle. The foreground has abrupt signal transitions at longer polygon edges and the background has periodic high frequency content. A desirable property for the antialiasing feature of a computer graphics system is high quality in both cases, meaning reduction of jagged edges in the foreground and moiré patterns in the background.

Raster images showing antialiasing examples should be viewed at 100% scale on a computer monitor. This is the standard configuration of a real-time computer graphics system, and it matches the practical antialiasing system in Figure 1(c). Any rescaling and resampling introduces additional stages to the system. Printing these images also introduces dithering artifacts. With these provisos in mind, Figure 5 shows a point-sampled checkerboard pattern in perspective rendered at 280×403 pixels. (When viewing the page after a 90º clockwise rotation, the foreground is at the bottom of the image.) The corresponding system is shown in Figure 1(a), and it lacks the supersampling and filtering stages. When viewed on a computer monitor at full size, the image exhibits jagged edges in the foreground and moiré patterns in the background.

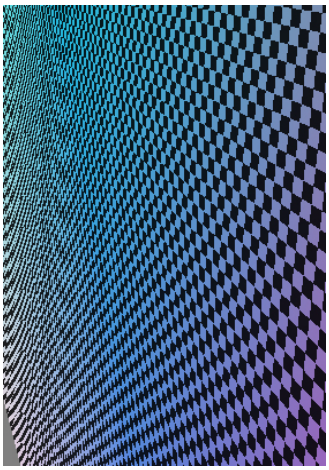


Figure 5: Pointed-sampled checkerboard pattern.

Figure 6 shows the same checkerboard pattern antialiased with various supersample rates and resampling filters. When viewing the page after a 90º clockwise rotation, the supersample rates from top to bottom are 2×2, 4×4, and 8×8, and the resampling-filter functions from left to right are rectangle, triangle, and cubic. Some general observations are apparent upon viewing the composite image on a computer monitor. All examples exhibit improved quality over the point-sampled image in Figure 5. At 2×2 supersampling, the foreground edges still appear jagged for all filters, but less so than for the point-sampled case. Increasing the supersample rate to 4×4 helps to eliminate most of the jagged appearance in the foreground edges, especially for the triangle and cubic filters. At 8×8 supersampling, the foreground edges have no jagged appearance for all filters. Moiré patterns in the background are visible in all cases. Increasing the supersample rate and/or filter extent tends to reduce this artifact. The cubic filter produces a sharper image for a given supersample rate because its negative secondary lobes make it a better approximation to the ideal filter.

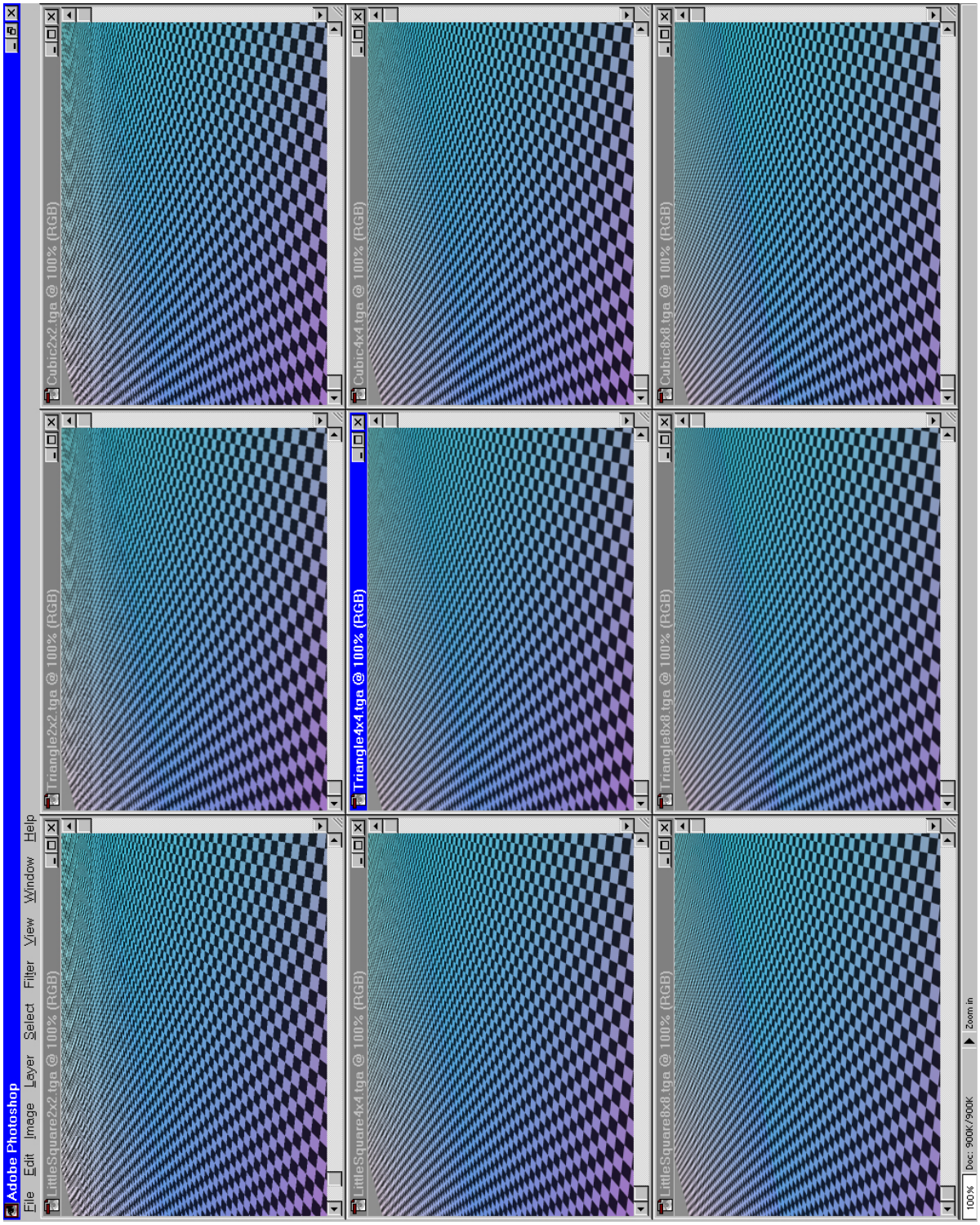


Figure 6: Antialiased checkerboard pattern. When viewing the page after a 90° clockwise rotation, the supersample rates from top to bottom are 2×2, 4×4, and 8×8, and the resampling-filter functions from left to right are rectangle, triangle, and cubic.

Figure 7 shows the subjective antialiasing quality (on a scale of 100) based on one author's ratings. Increasing the supersample rate from 2×2 to 4×4 gives larger relative gain than increasing it from 4×4 to 8×8. Upgrading the filter from rectangle to triangle gives a slightly larger relative gain than upgrading from triangle to cubic.

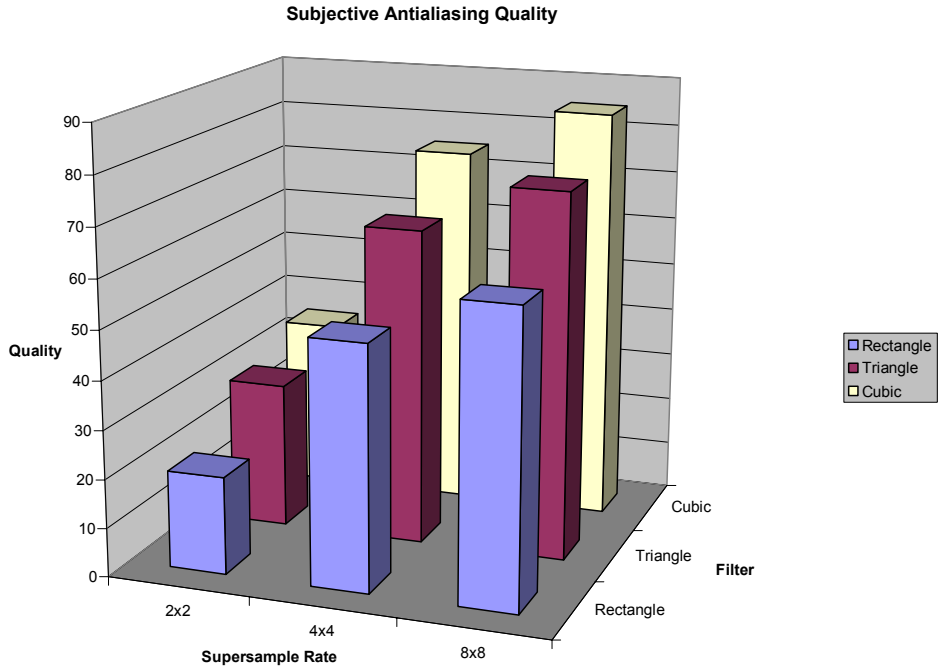


Figure 7: Subjective antialiasing quality for the supersample rates and filters in Figure 6.

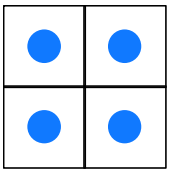
3. Description of Supersampling Pattern Experiments

The previous section described the effects of various types of resampling filters used in the supersampling process. This section describes a separate set of experiments performed to determine the effect of using sampling patterns other than a regular grid for supersampling.

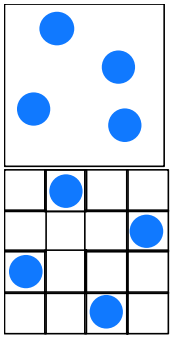
The final sample geometry of an image in a graphics system is constrained to a rectangular regular grid due to the construction of the video refresh system, which paints a regular series of scan lines on a video monitor or other display. It is possible, however, to use a different pattern for the supersamples, since resampling is necessary in any case to get the final pixel values.

Three broad categories of supersampling patterns have been described in the signal processing and computer graphics and imaging literature:

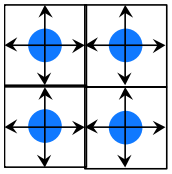
1. **Regular Grid:** the conventional method in which samples are taken on a rectangular grid at evenly spaced intervals. The example at right shows 4x supersampling on a regular grid (the actual sample points are located at the center of the circles). Regular hexagonal grids have also been described in the literature[12], but were not included in this study.



2. **Irregular Grid:** a set of irregularly positioned supersamples is chosen. The same set of supersample positions is used for every pixel. Supersample positions may be selected from anywhere within the pixel coverage area (top) or at positions on a higher resolution supersample grid (bottom).



3. **Jittered:** supersample positions are randomly displaced from regular grid positions. A different random pattern is used for every pixel. A given pixel retains the same pattern over every frame of an animation.



Some antialiasing implementations have used line and/or polygon edge related techniques to mask aliasing effects. For this study, we considered only full scene approaches to antialiasing.

Fragment techniques which use a bit map or a quantized coverage map to represent coverage [8,9,10,11] are in effect a form of supersampling, and are subject to similar aliasing effects. The sampling results of this study would be applicable to such techniques with minor adjustments. For the most part, these techniques use a box filter as a resampling filter, which reduces image quality, although some software implementations have implemented better filtering [13,14].

3.1 Software Implementation

The goal of this study was to examine the results of three supersampling methods (regular grid, irregular grid, and jittered) in combination with three main resampling filters (box, tent, and Gaussian). The sample scene chosen was similar to that used in Section 2, an obliquely viewed checkerboard. However, in this case flat shading without lighting effects was used to eliminate any image variation caused by light sources, and an infinite extent checkerboard was used.

In order to evaluate jittered sampling, it was necessary to build a renderer that allowed a different supersampling pattern for each pixel in the scene. Rather than build an all-new jittered incremental rasterizer, a much simpler ray-casting method was used in the rasterization software. This also allowed a simple data representation in which an infinite extent checkerboard could be easily represented as a single primitive. Since development time was more important than execution time, the renderer was built completely using the MathWorks *MATLAB* numerical computing system [15], version 5.2. MATLAB provides extensive vector computation and image processing capabilities that greatly aided development of the renderer.

While allowing the use of irregular and jittered grids, the ray casting method used in the renderer can also reproduce the results calculated by an incremental renderer on a regular grid (except for approximations introduced by quantization errors). This allows comparison of images generated using the various grid types.

3.2 Supersample patterns

3.2.1 Regular Grid

The regular grid supersample pattern was used as the base method for comparison.

3.2.2 Irregular Grid

An important consideration for generation of an irregular grid is the number of bits required for representation of the supersample positions. Use of a larger number of bits for the supersample positions requires additional data paths in hardware, and therefore increased expense. Two approaches were examined, one using a limited number of bits but with limited sample placement precision, and the other using many more bits to allow effectively unlimited sample placement precision.

3.2.2.1 N-Queens Supersampling

The *N-Queens* method uses a sparsely populated regular grid. For N samples, an $N \times N$ supersample grid is used within the pixel. Each supersample is placed such that no other supersample occupies the same row, column, or diagonal of the grid (this is an extension of the

N-Rooks sampling method [16]). The patterns that were used in this study are shown in Table 6 (MATLAB code was developed to search for the patterns).

Table 6: N-Queens Irregular Grid Sample Patterns

N=4

```
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```

N=5

```
1 0 0 0 0
0 0 0 1 0
0 1 0 0 0
0 0 0 0 1
0 0 1 0 0
```

N=8

```
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

N=9

```
1 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0
```

N=16

```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
```

3.2.2.2 Edge-Optimized Irregular Grid

It is possible to optimize the pattern of supersamples according to some specified criteria, in the hope that this will result in improvement in rendering of a wider class of images. Cross [17] has developed a genetic algorithm search process that tries to optimize sample distribution for improved rendering of edges. Patterns for several supersample densities are included in the article. This method has the disadvantage for hardware implementation that considerable precision may be required to represent the supersample positions, adding expense to the hardware. It appears that if the precision is reduced, the positions may converge to a pattern

Table 7: Edge Optimized Irregular Grid Sample Locations

foursamples	
x	*y*
0.274942	0.884325
0.797099	0.207128
0.765063	0.715779
0.122774	0.282758

sixteensamples	
x	*y*
0.755279	0.0497319
0.384479	0.688268
0.666094	0.868388
0.317172	0.0331764
0.729309	0.43103
0.0867931	0.368519
0.322668	1.0
0.442302	0.572752
0.889074	0.606985
0.0343768	0.191404
0.910321	0.872547
0.92479	0.345332
0.289126	0.389783
0.896551	0.141167
0.23357	0.678942
0.11281	0.526939

similar to the N-Queens solution. As will be seen in the results, the subjective difference in image quality between the N-Queens and Edge-Optimized methods is slight.

The code to generate these sampling patterns is not available at the *Graphics Gems* FTP site, and we were not successful in contacting the author, so full information on how to generate these patterns is not available. Since the quality of the results is similar to the N-Queens method, which is easier to implement, this is not a real problem.

Sampling patterns used in this study can be found in Cross [17] on p. 362, and are also shown in Table 7. The first pattern shown in "`foursamples`" and the first pattern shown in "`sixteensamples`" were used for four and sixteen sample supersampling, respectively. In the coordinate system used, the pixel center is located at (x,y) location (0.5, 0.5) and the origin is at the lower left.

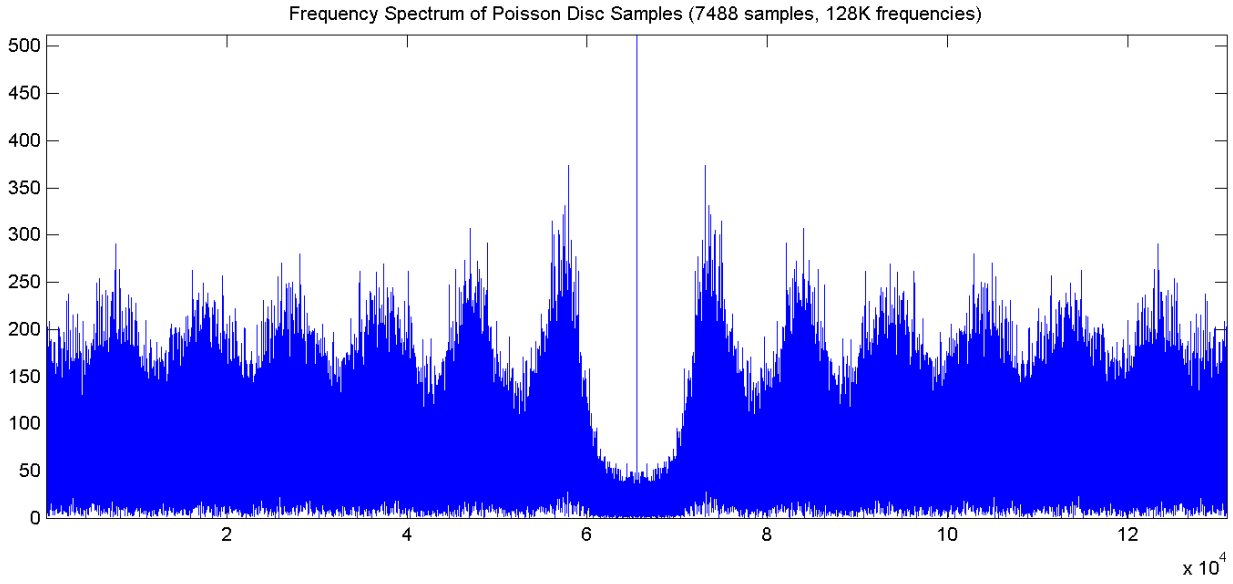
3.2.3 Jittered Grid

One of the most objectionable features of aliasing is the presence of spurious regular patterns which the human visual system is very good at recognizing. It is possible to add noise to the sampling process in such a way that aliasing results in noisy, irregular patterns which can be less objectionable to the visual system. As will be seen in the results, however, achieving good results requires more samples than is desirable.

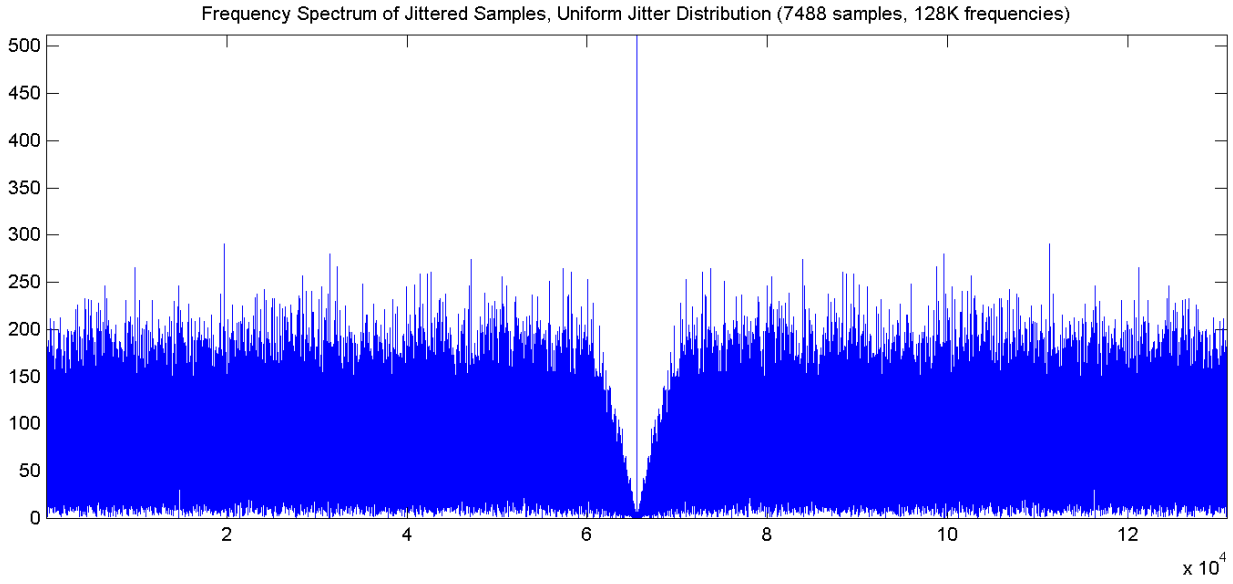
Noise can be added to the sampling process by randomly perturbing the locations of the supersample points. Cook [18] describes Poisson Disk sampling as a method to accomplish this. A minimum spacing distance λ between supersample locations is specified, and then supersample locations are randomly placed over the entire image area with the constraint that none are spaced more closely than λ. The frequency spectrum of a Poisson Disk sampling pattern shows a central area with a very small amount of noise, surrounded by higher frequencies dominated by noise. In theory, sampling with such a pattern will convert energy in these higher (aliased) frequencies to noise rather than to coherent lower frequency signals (see Figure 4 in [18]).

In practice, it turns out that Poisson Disk sampling generates some noticeable artifacts not mentioned by Cook. Figure 3b in [18] shows an optical frequency transform of the photoreceptor distribution in a monkey's eye, which according to Cook has a Poisson Disk distribution. A careful look at this figure shows faint alternating concentric light and dark bands around the center of the two-dimensional transform.

In order to examine this banding phenomenon, code was written to generate one-dimensional Poisson Disk sampling patterns, and the Fourier transforms of the patterns. Figure 8(a) below shows one such transform. There is a very regular fluctuation in the frequencies outside the central area of the transform, similar to the alternating bands of the two-dimensional transform in [18]. This can result in noticeable artifacts in images generated using such a sampling pattern. Other authors [19,20] have taken issue with a number of other points made by Cook.



(a)



(b)

Figure 8 – Frequency spectra for Poisson Disc and Uniform Jitter sample patterns

Dippé and Wold [21] compare Poisson Disc ("minimum distance Poisson") sampling to uniform jitter, and find similar peaks in the frequency distribution (see Figure 4 in [21]). They find that a uniform jitter produces better results than minimum distance Poisson (see Figures 5 and 6 in [21]). Figure 8(b) above shows the frequency spectrum of a set of uniform samples equal in number to the Poisson Disc samples used to generate (a). Notice that the noise in the outer part of the spectrum in (b) does not contain the objectionable low frequency elements seen in (a). This will result in fewer artifacts in a sampled image.

The jittered grid images for this study were generated by using this type of uniform random jitter relative to a regular grid. Designate the spacing between regular grid supersamples in the x and y directions as ε. Two random values were generated for each supersample of each pixel, uniformly distributed in the range $(-\varepsilon/2,+\varepsilon/2)$. One random value was added to the x coordinate of the supersample, the other to the y coordinate. A separate random value was used for every supersample of every pixel in the image. For an animation, the same supersample locations were used for every frame of the animation.

3.3 Resampling filters

Section 2 gave a thorough discussion of the importance of the resampling filter in reducing the effects of aliasing. Three different filters (described below) were used for this part of the sampling study: the conventional Box filter, the Tent filter, and the Gaussian ½ filter.

Because this study looked at filtering samples which may be on an irregular grid, we will first define some new terminology to deal with this extension to conventional filtering. At a given pixel, the reconstructed gray scale value G_p of the pixel p is calculated as a normalized weighted average of S_p, the set of supersamples within the support of the resampling filter. For this study, three different resampling filters were used. Designate x_{ip} and y_{ip} as the distances of supersample i from the center of p along the x and y axes, respectively. G_i designates the gray-scale intensity of supersample i. The weight applied to each supersample is w_{ip}. The pixel gray scale value is computed by summing the weighted supersample gray values. It is then normalized by dividing this sum by the sum of all the weights:

$$G_p = \frac{\sum_{i \in S_p} w_{ip} G_{ip}}{\sum_{i \in S_p} w_{ip}}$$

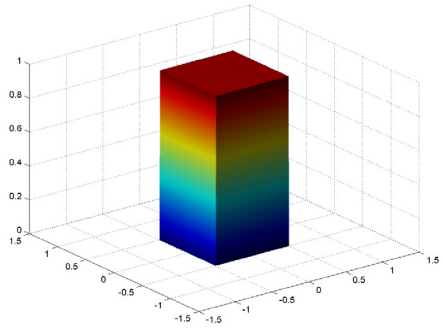
The particular filter chosen determines the w_{ip} values and the members of S_p.

3.3.1 Box Filter

The box filter is computed simply as the average of all supersamples in a unit square surrounding the pixel. Unlike the other filters described, the box filter has non-overlapping support (each supersample affects only a single pixel value), and so each pixel value can be computed independently. The box filter also has the smallest support and therefore uses the smallest number of supersamples to compute its value.

$$S_p = \left\{ i \middle| x_{ip} \in \left[-\tfrac{1}{2}, \tfrac{1}{2}\right), y_{ip} \in \left[-\tfrac{1}{2}, \tfrac{1}{2}\right) \right\}$$

$$w_{ip} = 1$$

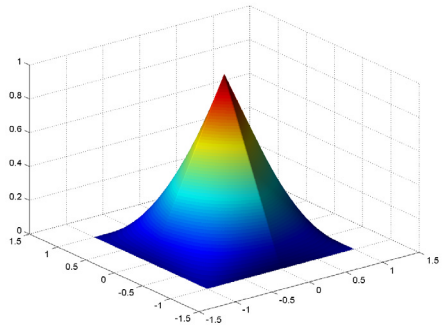


3.3.2 Tent Filter

Supersamples are weighted as a product of the linear distance in x and y from the pixel. The support of the filter extends a distance of 1 in the x and the y directions from the pixel center.

$$S_p = \left\{ i \middle| x_{ip} \in [-1,1), y_{ip} \in [-1,1) \right\}$$

$$w_{ip} = \left(1 - \left|x_{ip}\right|\right)\left(1 - \left|y_{ip}\right|\right)$$

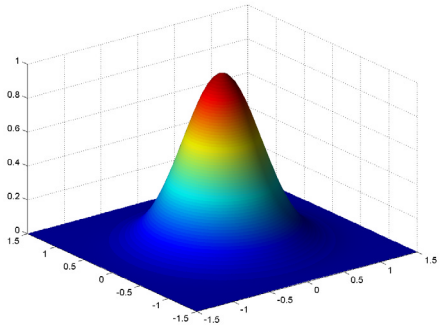


3.3.3 Gaussian ½ Filter

The weighting function used is a product of Gaussian functions in the x and y direction which have a height of ½ at a distance of ½ from the center [22]. This has an effective support radius of about 1.5, since the magnitude at this distance is $\frac{1}{512}$, which is half of the typical (eight bit) pixel gray scale quantization value. In practice the function is truncated at this distance in x and y.

$$S_p = \left\{ i \middle| x_{ip} \in [-1.5,1.5), y_{ip} \in [-1.5,1.5) \right\}$$

$$w_{ip} = 2^{-4\left(x_{ip}^2 + y_{ip}^2\right)}$$



3.4 Results

The following results include images illustrating antialiasing results. Because these images will not be rendered faithfully on most printers, any critical examination of these images should be performed using the originals. Information about the images (and animations) described below can be obtained on the web at

http://www.hpl.hp.com/research/cp/cmsl/research/3d/antialiasing

No quantitative image comparisons are presented in this section for two reasons:

1. For purposes of hardware rendering, subjective image quality (quality perceived by the viewer) is the object of antialiasing procedures. The goal of this study was to determine the most cost effective way to increase subjective image quality.
2. No studies have been done to develop useful quantitative metrics for evaluation of subjective image (the signal-to-noise ratio often used in image processing is not strongly correlated with subjective image quality).

Instead of quantitative studies, we instead present side-by-side comparisons of equivalent images generated by different techniques for subjective evaluation by the reader.

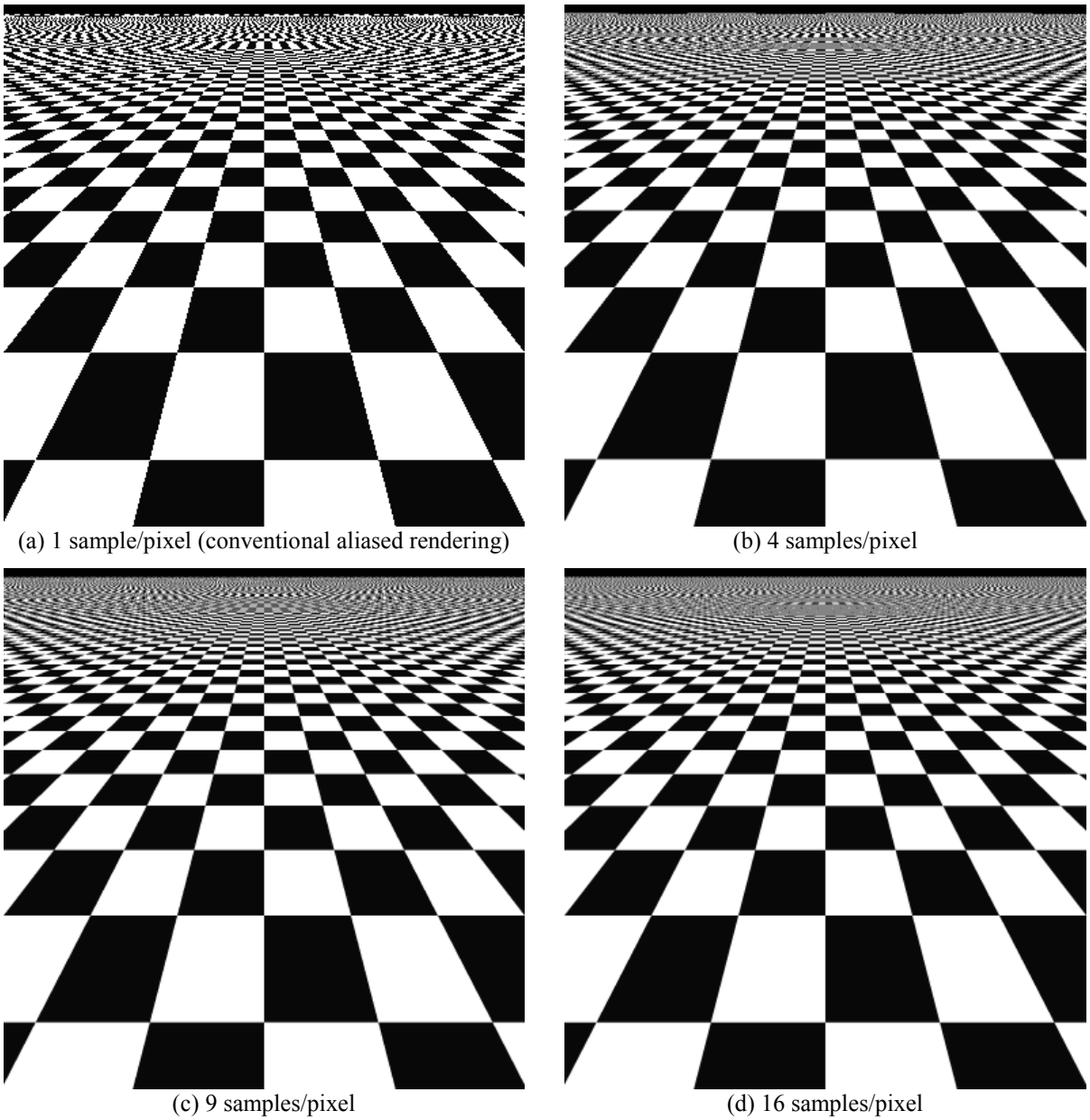
3.4.1 Box Filter

The first results shown were generated using a box filter (see 3.3.1 above). This is the least expensive method, since pixels have non-overlapping support. As will be seen in later results, the

higher performance resulting from this simple filter comes at the cost of noticeable reduction in image quality.

3.4.1.1 Conventional Aliased Rendering

For comparison purposes with later figures, Figure 9 (a) shows the checkerboard rendered with conventional point sampling. A single sample is taken in the center of each pixel.



(a) 1 sample/pixel (conventional aliased rendering)

(b) 4 samples/pixel

(c) 9 samples/pixel

(d) 16 samples/pixel

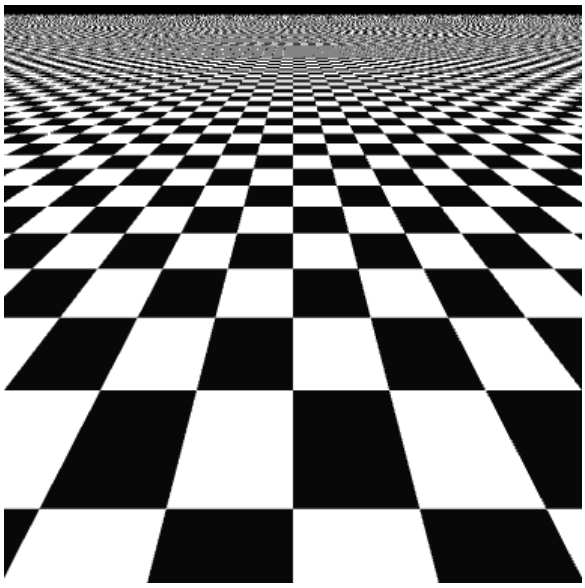
Figure 9: Regular Grid, Box Filter

3.4.1.2 Regular Grid, Box Filter

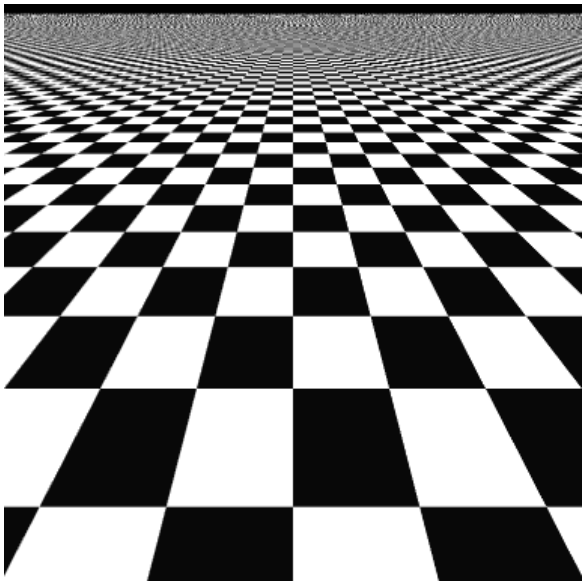
Figure 9 (b) through (d) shows the effects of what is currently the most common hardware antialiasing technique. These images were rendered with multiple supersamples per pixel on a regular grid. Rendering of the edges in the foreground and the areas near the horizon improve noticeably as the number of samples increases. However, moiré patterns in the middle distance are still very pronounced, even with 16 samples.

3.4.1.3 N-Queens Irregular Grid, Box Filter

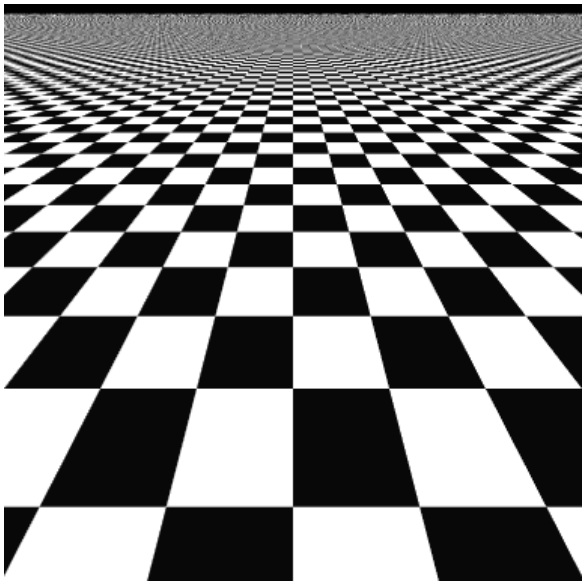
Figure 10(a-c) shows the results of using N-Queens sampling (see 3.2.2.1 above) with the same number of samples per pixel as in Figure 9 (b-d). The irregular grid of the N-Queens pattern



(a) 4 samples/pixel



(b) 9 samples/pixel



(c) 16 samples/pixel

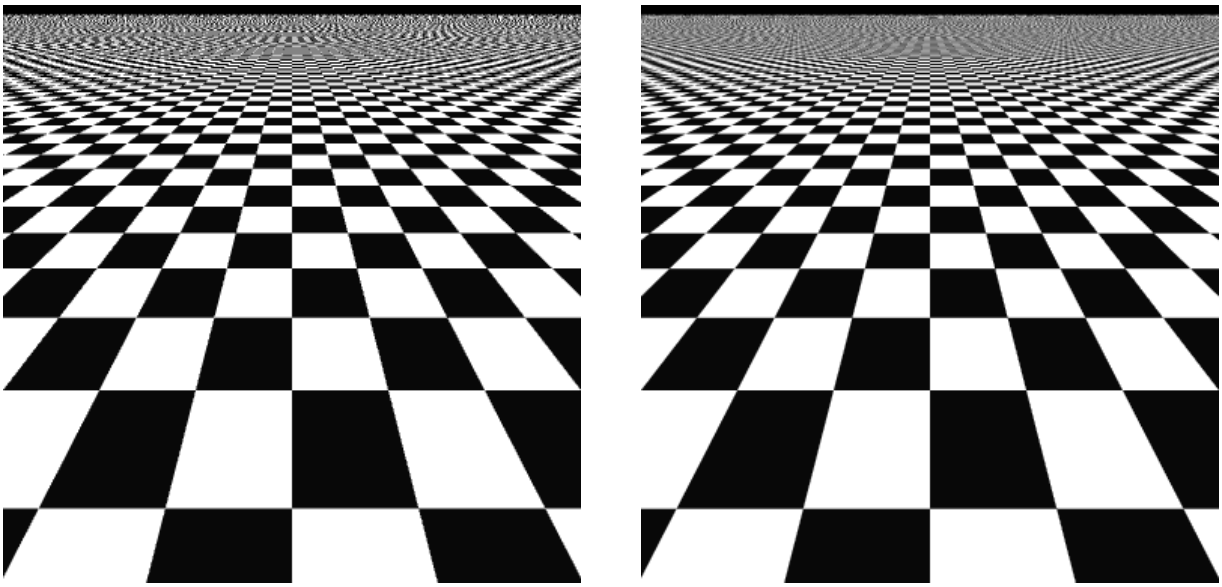
Figure 10: N-Queens Irregular Grid, Box Filter

greatly reduces the presence of moiré patterns in the middle distance when compared with the regular grid, giving noticeably better image quality.

3.4.1.4 Edge-Optimized Irregular Grid, Box Filter

Figure 11 shows the images resulting from use of the edge-optimized irregular grid method (see 3.2.2.2 above). Comparison with Figure 10 (a) and (c) above shows very slight improvement in image quality in comparison to the N-Queens method. The extra cost of the edge-optimized grid

relative to the N-Queens grid indicates that for hardware implementation the N-Queens is probably the more cost-effective option.

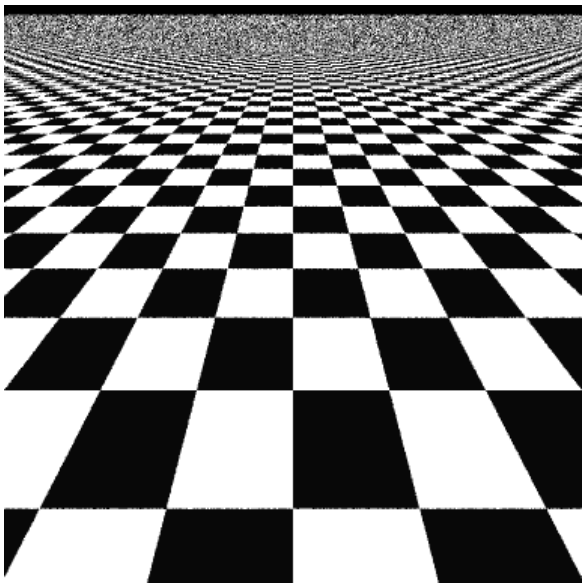


(a) 4 samples/pixel (b) 16 samples/pixel

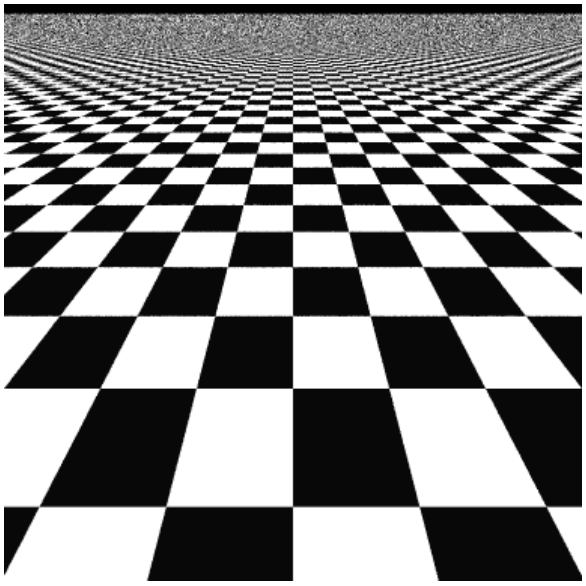
Figure 11: Edge-Optimized Irregular Grid, Box Filter

3.4.1.5 Jittered Grid, Box Filter

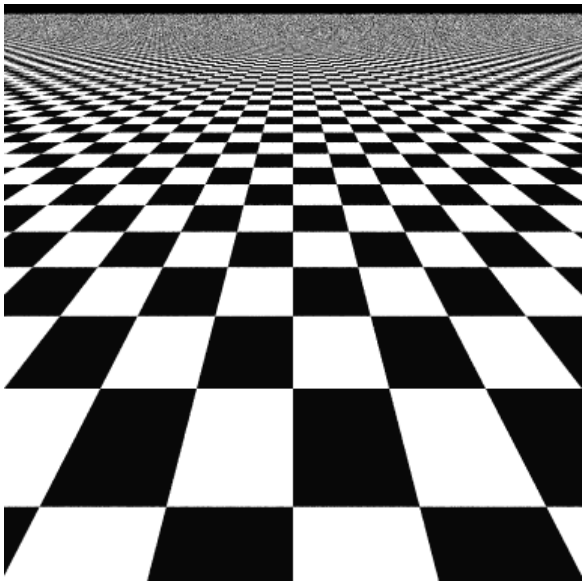
Figure 12 shows images generated using a jittered grid (see 3.2.3 above). As expected, the moiré patterns in the mid-distance and far distance are effectively converted to noise, even at 4 samples/pixel. However, at fewer than 16 samples per pixel an unacceptable level of noise is generated along edges such as those of the squares in the foreground. Animations with fewer than 16 samples per pixel also show noticeable moving noise patterns on edges and in the distance. With 16 or more supersamples per pixel, the jittered grid generates significantly better image quality than the other methods at the same supersample rate, but at a greatly increased cost. As with edge-optimized sampling, the arbitrary placement of supersamples also involves increased hardware cost, especially since supersample locations differ for every pixel in the image.



(a) 4 samples/pixel



(b) 9 samples/pixel



(c) 16 samples/pixel

Figure 12: Jittered Grid, Box Filter

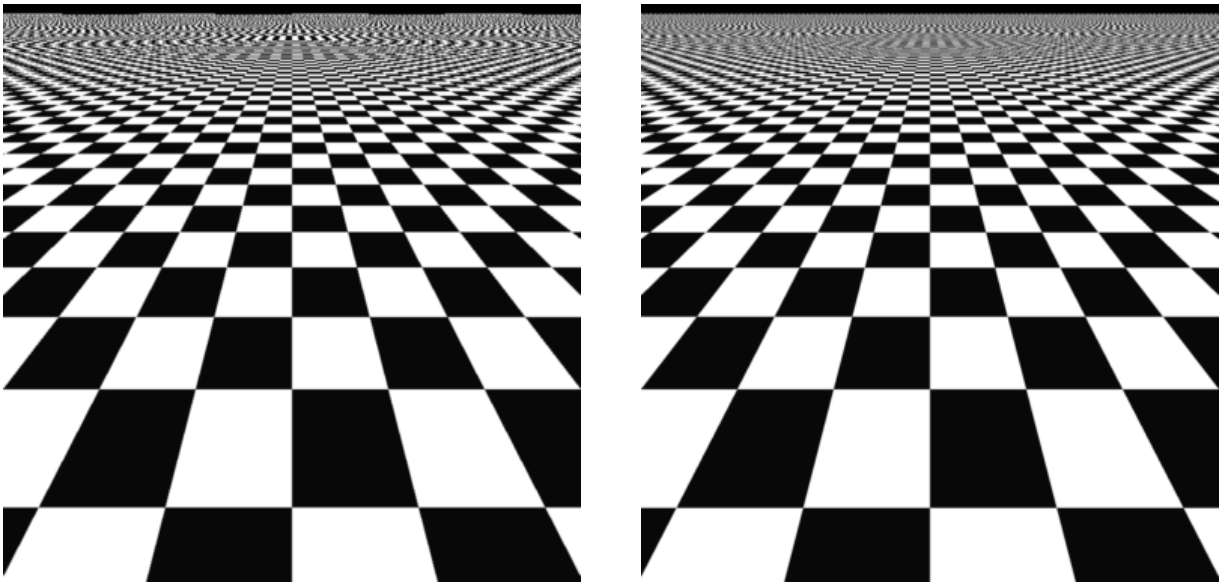
3.4.2 Tent Filter

Although the box filter is the least expensive resampling filter to compute, substantial image quality benefits may be realized by using a better resampling filter. The cost of a filter is strongly related to the size of the filter support. To improve upon the results from the box filter, the support of the filter must be enlarged and will overlap with neighboring pixels. The tent filter (see 3.3.2 above) is a reasonable compromise between computation cost and image quality. At 4

samples/pixel, the support of the tent filter covers 16 supersamples, and at 16 samples/pixel, the support covers 64 supersamples.

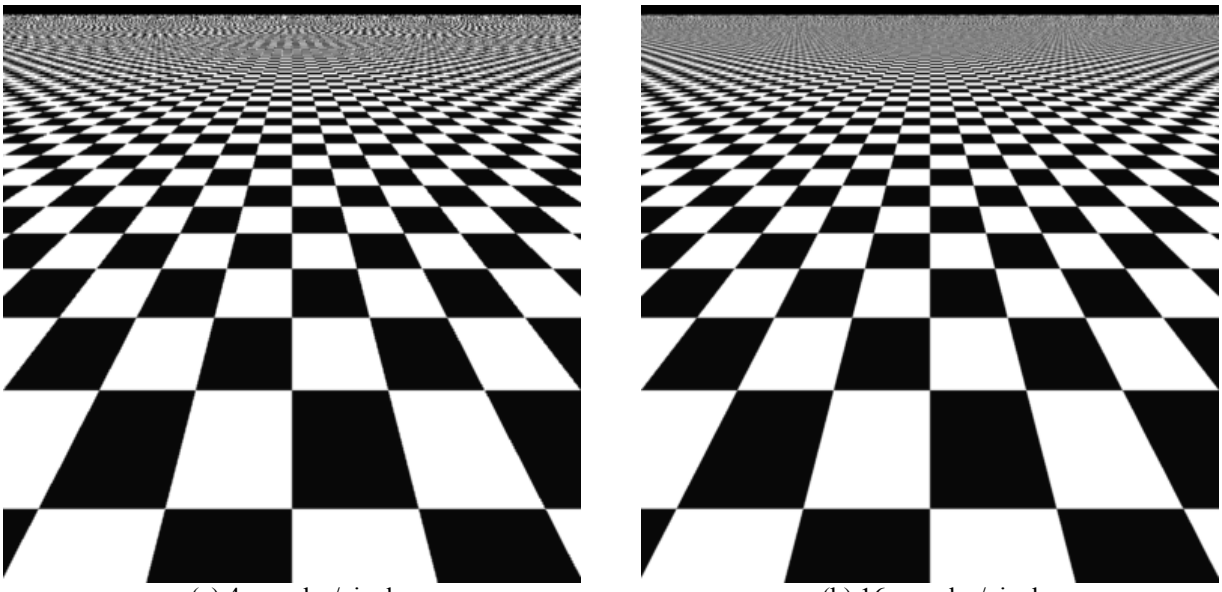
The results for the tent (and later the Gaussian ½) filter are shown for three sampling patterns (regular grid, N-Queens, and jittered) at two sampling densities (4 and 16 samples per pixel). The edge-optimized grid and 9 sample per pixel images may be viewed at the web site

http://www.hpl.hp.com/research/cp/cmsl/research/3d/antialiasing



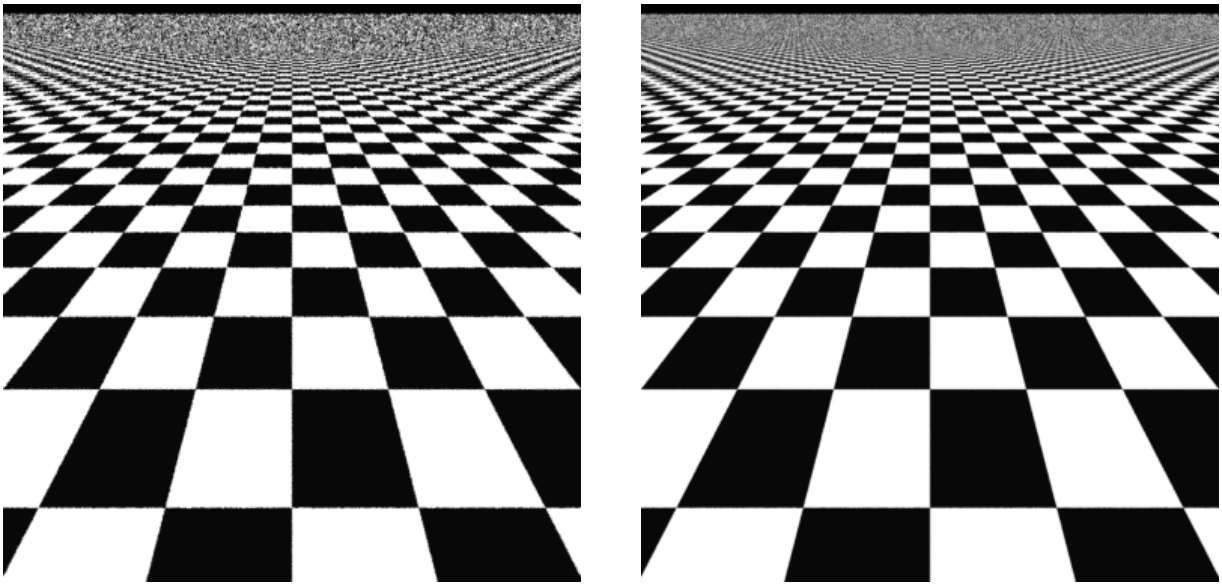
(a) 4 samples/pixel (b) 16 samples/pixel

Figure 13: Regular Grid, Tent Filter



(a) 4 samples/pixel (b) 16 samples/pixel

Figure 14: N-Queens Irregular Grid, Tent Filter

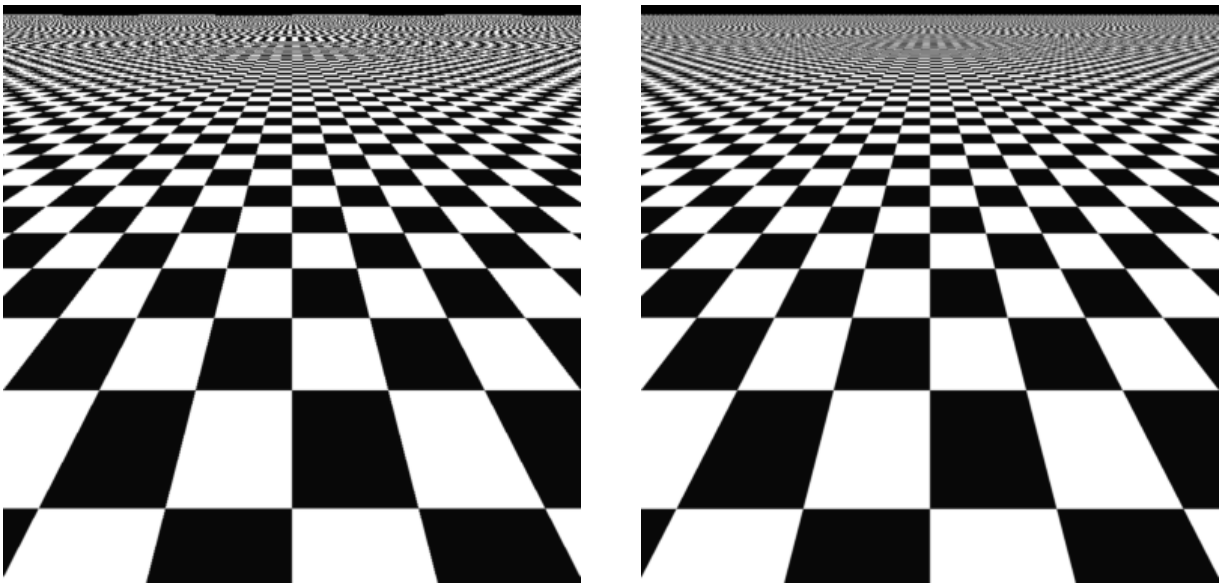


(a) 4 samples/pixel (b) 16 samples/pixel

Figure 15: Jittered Grid, Tent Filter

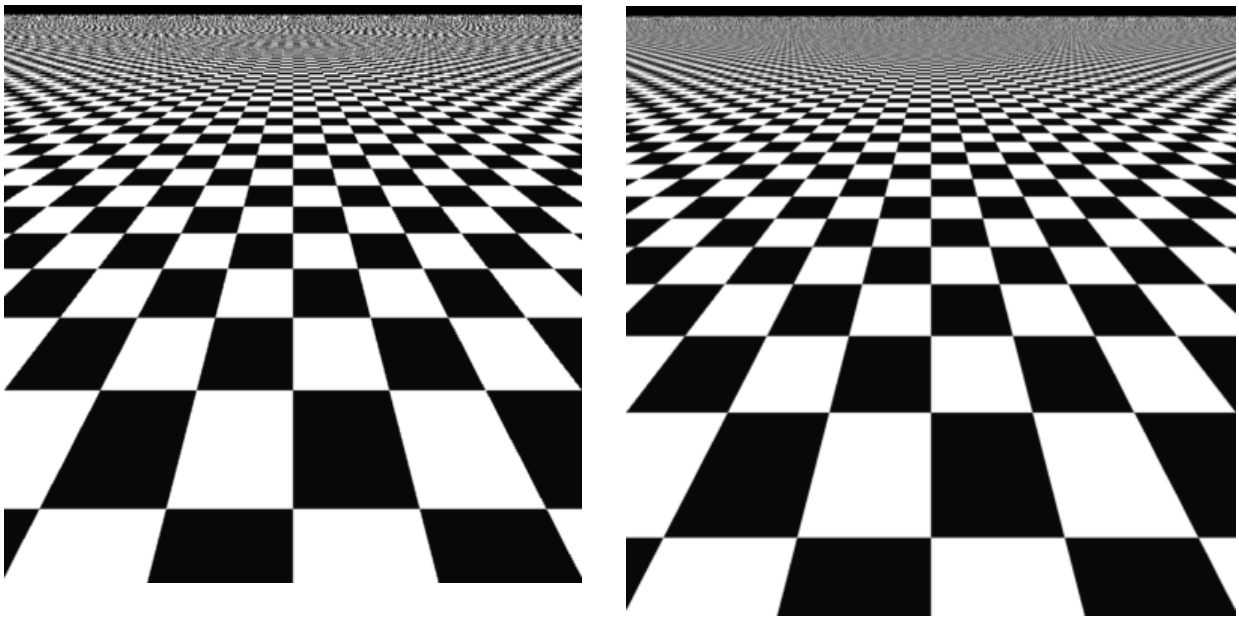
3.4.3 Gaussian ½ Filter

The Gaussian ½ Filter (see 3.3.3 above) has a larger support than the tent filter, and as the sample images show the resulting image quality is correspondingly better. At 4 samples/pixel, the support of the Gaussian ½ filter covers 36 supersamples, and at 16 samples/pixel, the support covers 144 supersamples.



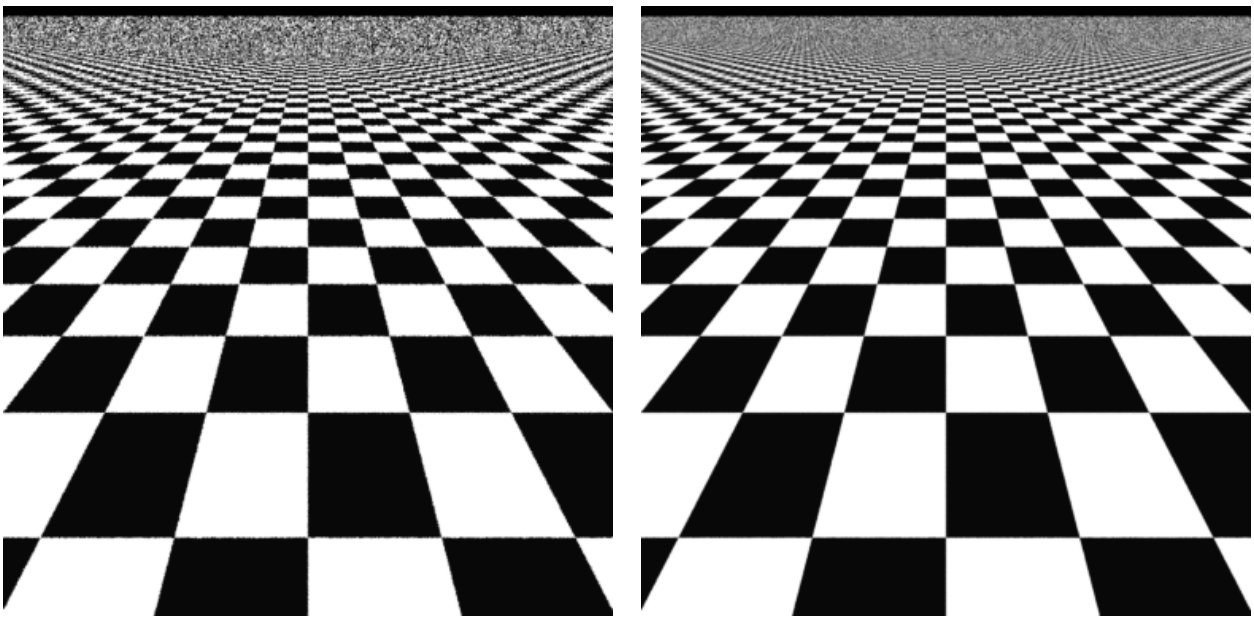
(a) 4 samples/pixel (b) 16 samples/pixel

Figure 16: Regular Grid, Gaussian ½ Filter



(a) 4 samples/pixel (b) 16 samples/pixel

Figure 17: N-Queens Irregular Grid, Gaussian ½ Filter

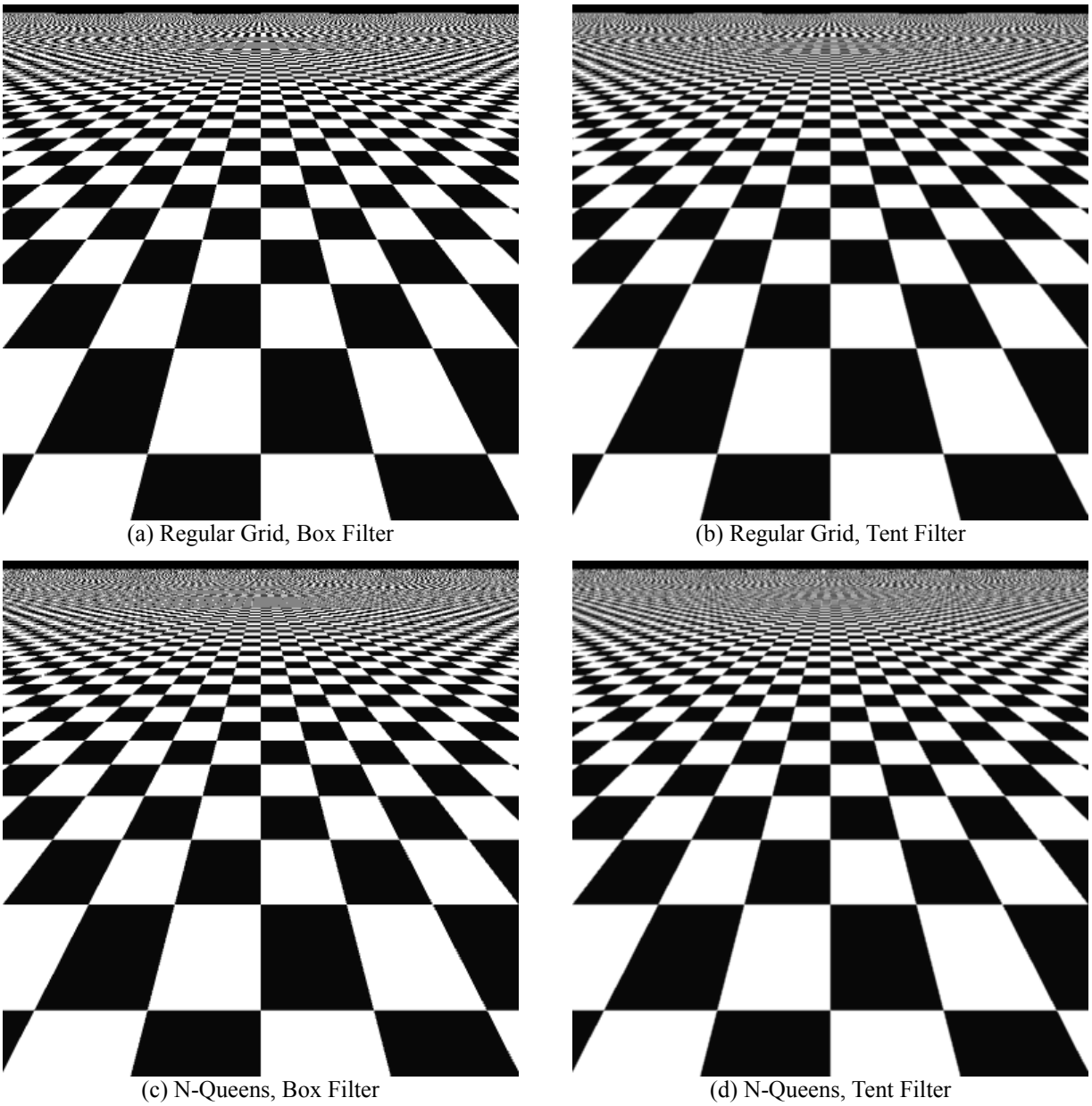


(a) 4 samples/pixel (b) 16 samples/pixel

Figure 18: Jittered Grid, Gaussian ½ Filter

3.4.4 Filter Comparisons

The most likely candidate sampling patterns for implementation in hardware in the near term will be determined by hardware cost and speed limitations. Regular grid and N-Queens sampling are the lowest cost sampling methods. The box and tent filters are the lowest cost filter methods. Figure 19 shows a side-by-side comparison of images generated using 4 samples per pixel with all four combinations of these sampling and filter methods. Figure 20 shows the same comparison for images generated using 16 samples per pixel.



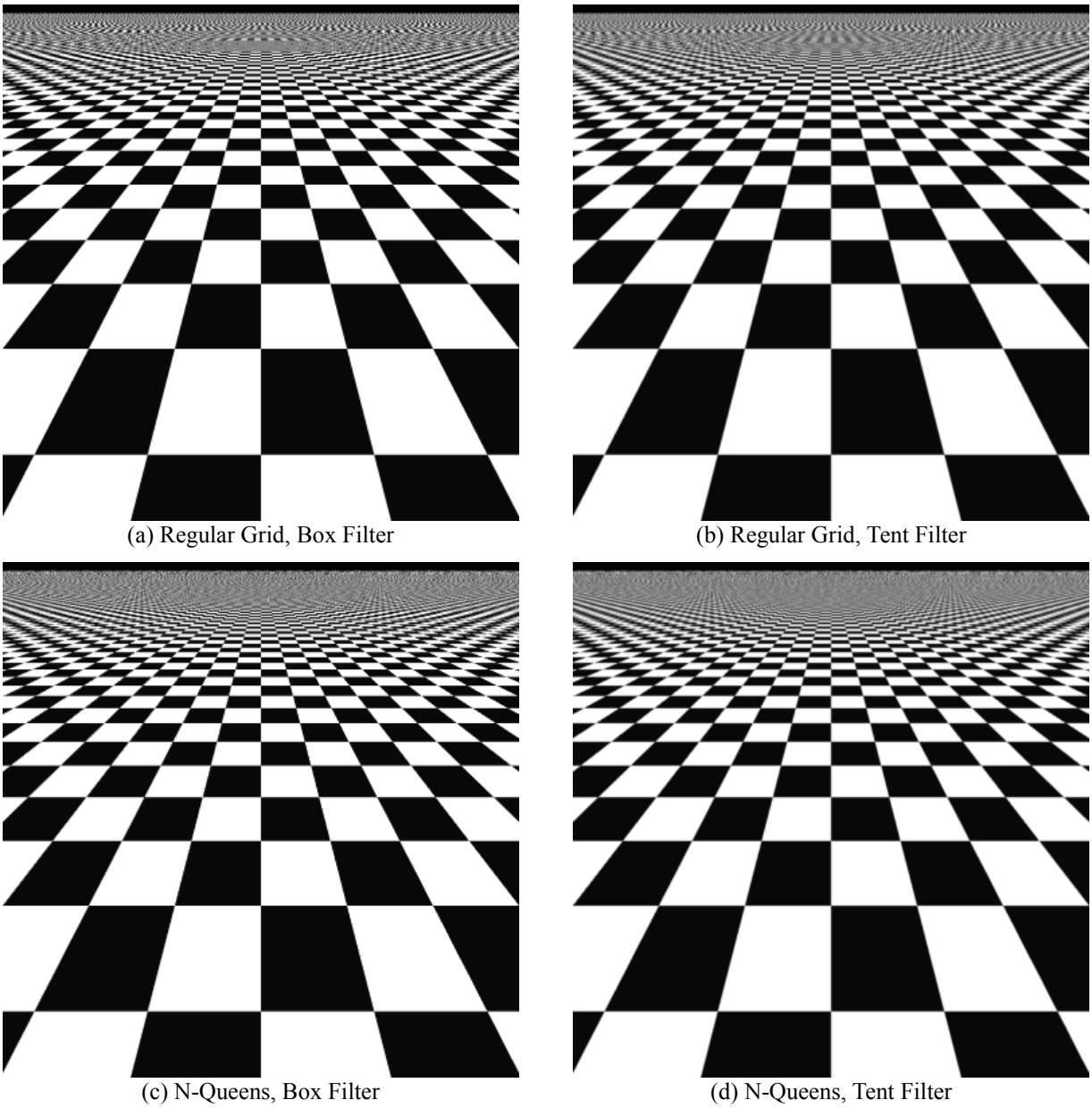
(a) Regular Grid, Box Filter

(b) Regular Grid, Tent Filter

(c) N-Queens, Box Filter

(d) N-Queens, Tent Filter

Figure 19: 4 sample per pixel filter comparison



(a) Regular Grid, Box Filter (b) Regular Grid, Tent Filter

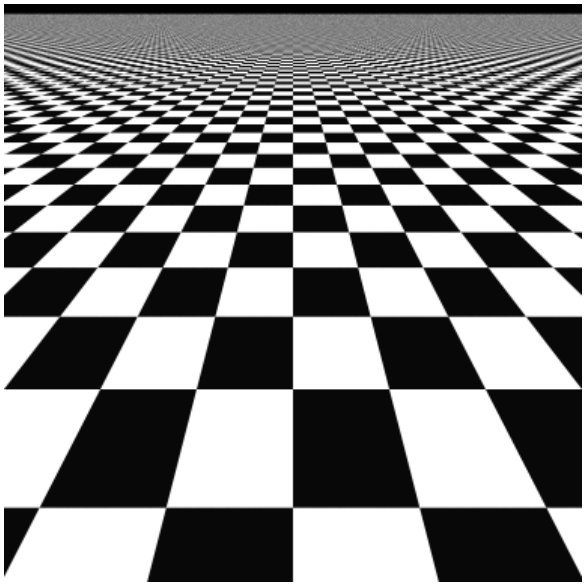
(c) N-Queens, Box Filter (d) N-Queens, Tent Filter

Figure 20: 16 sample per pixel filter comparison

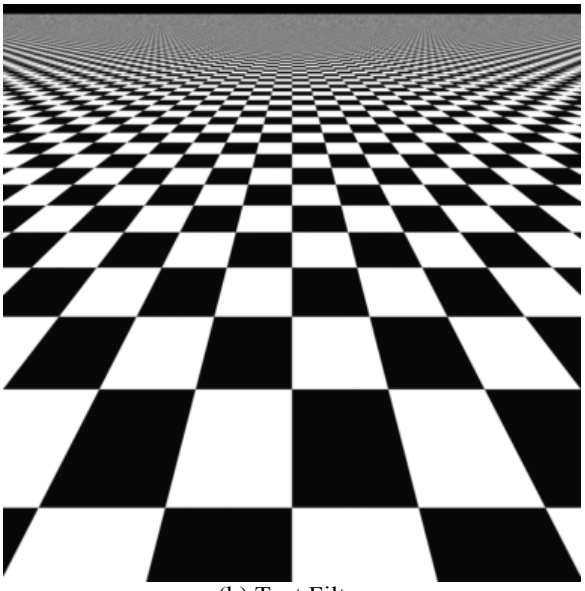
3.4.5 Limits to Image Quality

As noted above, jittered grid sampling gave best results with 16 samples or more per pixel, but at a cost that is probably impractical for hardware implementation in the near future. To give an example of the quality possible if cost is not a concern, Figure 21 shows images generated with 256 samples per pixel using the jittered grid sampling method. Images were generated using each of the three filter techniques. The differences between the Tent Filter and the Gaussian ½ filter

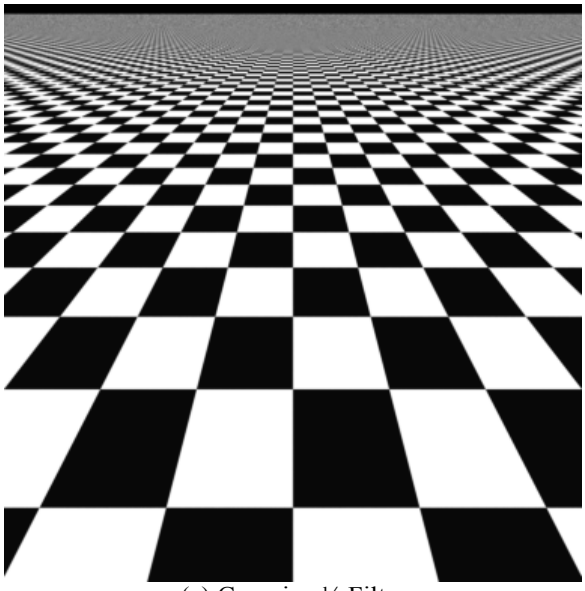
are nearly imperceptible. However, the Box Filter image still shows some noticeable increase in moiré patterns in the middle distance in spite of the high sampling rate. This demonstrates that good filtering is necessary for high image quality, no matter how high the supersample rate.



(a) Box Filter



(b) Tent Filter



(c) Gaussian ½ Filter

Figure 21: 256 samples per pixel, Jittered Grid,

3.4.6 Animation Results

In addition to the still image results shown above, a number of animation sequences were generated using linear motion and spinning motion relative to the checkerboard. The results of these animations showed that the jittered grid method is not useful with less than 16 samples per

pixel, due to excessive noise patterns along moving edges. It was found that the N-Queens patterns minimized objectionable noise, and in fact gave improved results over the regular grid, consistent with the still image results.

Several animations illustrating these results will be available at web site

http://www.hpl.hp.com/research/cp/cmsl/research/3d/antialiasing

The animations are in Microsoft AVI format.

4. Conclusions

This report has examined two important issues that will affect the design of antialiasing algorithms for graphics hardware in the near future: sampling and filtering. We have shown that the pattern used for supersamples has important effects on image quality, and use of an irregular grid pattern (such as N-Queens) may result in improvements in image quality with little increase in rendering time. We have also shown that filtering of the supersamples is critically important to image quality, and that better filtering, although requiring large support for the filter, can improve image quality in ways which cannot be achieved simply by increasing the number of supersamples per pixel.

References

[1] Alvy Ray Smith, A Pixel is Not a Little Square, A Pixel is Not a Little Square, A Pixel is Not a Little Square! (And a Voxel is Not a Little Cube), Microsoft Research, Technical Memo 6, 1995, http://www.research.microsoft.com/users/alvy/Memos/default.htm .

[2] Andrew S. Glassner, *Principles of Digital Image Synthesis*, San Francisco: Morgan Kaufmann, 1995.

[3] Jim Blinn, Return of the Jaggy, in *IEEE CG&A*, 9(2):82 –89, March 1989.

[4] Ronald N. Bracewell, *The Fourier Transform and Its Applications*, McGraw-Hill, 1978.

[5] Steven A. Tretter, *Introduction to Discrete-Time Signal Processing*, John Wiley & Sons, 1976.

[6] George Wolberg, *Digital Image Warping*, IEEE Computer Society Press, 1990.

[7] Jim Blinn, What We Need Around Here Is More Aliasing, in *IEEE CG&A*, 9(1):75–79, January 1989.

[8] Loren Carpenter, The A-buffer, an Antialiased Hidden Surface Method, *Computer Graphics Proceedings*, ACM SIGGRAPH, 1998, pp. 103-108.

[9] Andreas Schilling, A New Simple and Efficient Antialiasing with Subpixel Masks, *Computer Graphics Proceedings*, ACM SIGGRAPH, 1991, pp. 133-141.

[10] Andreas Schilling and Wolfgang Strasser, EXACT: Algorithm and Hardware Architecture for an Improved A-buffer, *Computer Graphics Proceedings*, ACM SIGGRAPH, 1993, pp. 85-92.

[11] Bill Rivard, Stephanie Winner, Mike Kelley, Brent Pease, and Alex Yen, Hardware Accelerated Rendering of Antialiasing Using a Modified A-Buffer Algorithm, *Computer Graphics Proceedings*, ACM SIGGRAPH, 1997, pp. 307-316.

[12] Dan E. Dudgeon and Russell M. Mersereau, *Multidimensional Signal Processing*, Prentice-Hall, 1984.

[13] Greg Abram, Lee Westover, and Turner Whitted, Efficient Alias-free Rendering using Bit-masks and Look-up Tables, *Computer Graphics Proceedings*, ACM SIGGRAPH, 1985, pp. 53-59.

[14] Ned Greene, Hierarchical Polygon Tiling with Coverage Masks, *Computer Graphics Proceedings*, ACM SIGGRAPH, 1996, pp. 65-74.

[15] http://www.mathworks.com/products/matlab/

[16] Andrew S. Glassner, N-Rooks Sampling, *Principles of Digital Image Synthesis*, Vol. I, Morgan Kaufmann, 1995, pp. 424-426.

[17] Robert A. Cross, Sampling Patterns Optimized for Uniform Distribution of Edges, *Graphics Gems V*, Alan W. Paeth, ed., Academic Press, 1995, pp. 359-363.

[18] Robert L. Cook, Stochastic Sampling in Computer Graphics, *ACM Transactions on Graphics*, 5(2), Jan. 1986, pp. 51-72.

[19] Theo Pavlidis, Comments on "Stochastic Sampling in Computer Graphics," *ACM Transactions on Graphics*, 9(2), April 1990, pp. 233-236.

[20] Erling Wold and Kim Pépard, Comments on "Stochastic Sampling in Computer Graphics," *ACM Transactions on Graphics*, 9(2), April 1990, pp. 237-243.

[21] Mark A. Z. Dippé and Erling Henry Wold, Antialiasing Through Stochastic Sampling, *Computer Graphics Proceedings*, ACM SIGGRAPH, 1985, pp. 69-78.

[22] Ken Turkowski, Filters for Common Resampling Tasks, *Graphics Gems*, Andrew S. Glassner, ed., Academic Press, 1990, pp. 147-165.