Anti-aliasing of Images Using Line Sampling

Robert M. Kotredes

under the direction of
Mr. Ray Jones
Mitsubishi Electric Research Labs

Research Science Institute
August 7, 1999

Abstract

Sampled images in computer graphics suffer from a problem known as aliasing. Aliasing causes artifacts such as jagged edges on fonts and polygons. There are several ways to remove these artifacts from images, known as anti-aliasing. The two most common anti-aliasing methods are exact area convolution and supersampling. This paper introduces a new anti-aliasing method, line sampling, which is significantly faster than exact area convolution and generates images of a higher quality than supersampling.

1 Introduction

1.1 What is aliasing

Aliasing is defined as "A high frequency signal masquerading as a low frequency signal"[5]. Aliasing in computer images is most commonly seen as "The low frequency ripples, stair-steps, or jaggies [...] caused by the high-frequency components of the polygon edges" [5]. Aliasing occurs because computers store images as discrete samples of data and not the continuous image they represent. The sampling theorem states that "[an] input signal can be reconstructed exactly from its samples only if it is band-limited to some maximum frequency which is less than one-half the sampling rate"[5]. This rate is known as the Nyquist limit. A computer can not correctly reconstruct an input signal if the signal has a frequency greater than one cycle per two pixels. Unexpected patterns occur when this is the case, as shown in Appendix A.

1.2 What can be done to stop aliasing

Anti-aliasing methods usually apply some sort of filter to the continuous image within a certain area around a pixel. The purpose of the filter function is to bandlimit the frequencies in the continuous image to below the Nyquist limit. If $w(x, y)$ is the filter function, and $s(x, y)$ is the continuous image, then the value of an anti-aliased image A at the point (X, Y) is given by the anti-aliasing integral:

$$A(X, Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(x - X, y - Y) s(x, y) dy dx \tag{1}$$

There is no perfect filter function w The sinc function $\frac{sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$ is mathematically the ideal bandlimiting filter, but causes ringing in images (see Appendix B). A commonly used filter function is the Gaussian, $\frac{1}{2\pi R^2} e^{\frac{-(x^2+y^2)}{2R}}$, where R is the radius of the filter. The Gaussian has infinite extent in both the x and y dimensions, but it is of negligible height at $\sqrt{x^2 + y^2} > R$,

so it is common to clamp the function to 0 outside of this "footprint". A Gaussian of radius R equal to the distance between two adjacent pixels is used in this paper. Exact area convolution, supersampling, and line sampling are three methods of evaluating this integral.

1.2.1 Exact area convolution

Exact area convolution [3, 4] evaluates the integral exactly within the footprint of the filter. This requires knowing the continuous image around the pixel. Computing visibility is easy for simple 2D images, but not for polygons in 3D that might occlude and intersect. However, once visibility is resolved, the images generated are as accurate as possible.

1.2.2 Regular supersampling

Regular supersampling is a method of approximating the anti-aliasing integral by evaluating the integrand at several regularly spaced sample points and combining the samples. It is generally very fast to evaluate the integrand at any given point, but it takes many samples for this method to converge. In general, it takes N samples to get an approximation within $\frac{1}{\sqrt{N}}$[8]. Aliases still occur, and aliases caused by such regularly spaced sampling patterns produce patterned artifacts, which are easily detectable by the human eye[2].

1.2.3 Stochastic supersampling

Stochastic supersampling[2] is a variant of regular supersampling, with the sample points placed at quasi-random locations. This quasi-random sampling replaces patterned aliases with noise. The human eye is more forgiving of noise than patterns, so stochastic supersampling produces more acceptable images[2]. Stochastic supersampling converges faster than regular supersampling, with N samples required to get an approximation within $\frac{1}{N}$[8].

1.2.4 Line Sampling

Line sampling is a new method of anti-aliasing. It is similar to both exact area convolution and point sampling. It is more accurate than point sampling because of its continuous nature. It reduces the double integral of equation 1 to a single integral which can be evaluated much more efficiently. It is also easier to solve the occlusion problem for a one-dimensional sample than for multiple dimensions as in exact area convolution.

2 Theory behind line sampling

2.1 How line samples work

Line samples have only one dimension, simplifying the problems that plague exact area convolution. For instance, it is easier to determine where a 1 dimensional line sample intersects a polygon than to determine the intersection of a two dimensional circle and the same polygon (Figure 1). From a line sample, we can approximate the anti-aliasing integral at

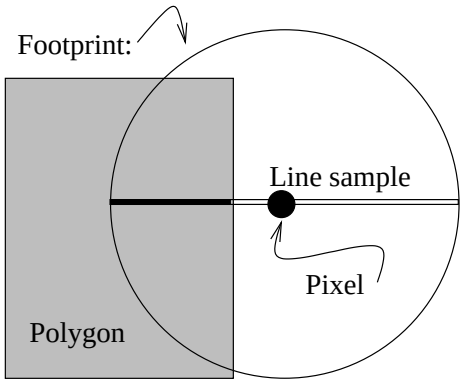

Figure 1: The intersection of a line and a polygon

the corresponding pixel. This approximation assumes that all edges intersected by the line sample are perpendicular to the line sample (Figure 2).

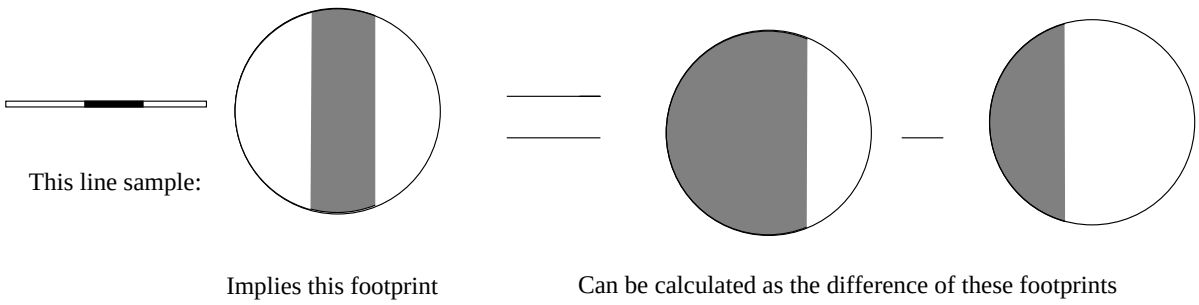

Figure 2: Calculation of a pixel from it's implied footprint

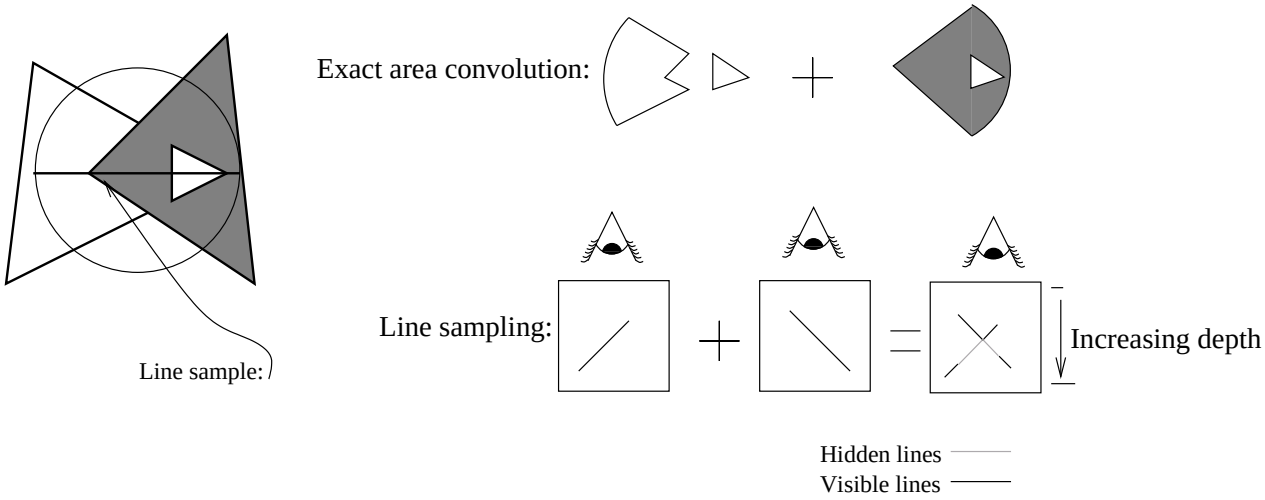

Figure 3: Occlusion is simpler with line samples

2.2 The Occlusion problem

The biggest problem with the exact area convolution method is resolving occlusion. It is difficult to calculate the 2D visible portions of polygons that intersect or occlude in 3D, but it is easier to do so along a 1D line sample. The top of Figure 3 shows the complicated shapes that can occur when using exact area convolution. The bottom of Figure 3 shows the equivalent scene for a line sample, where resolving the occlusion is simpler.

2.3 How convolution can be computed efficiently

Because we have reduced the sample to a line (horizontal, without loss of generality), the function s in the integral is only dependent upon x. After substituting $S(x)$ in for $s(x, y)$ and the equation for the Gaussian for w, the anti-aliasing equation (1) can be simplified as follows:

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{\frac{-((x-X)^2+(y-Y)^2)}{2}} S(x)dydx = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{\frac{-(x-X)^2}{2}} S(x)dx \qquad (2)$$

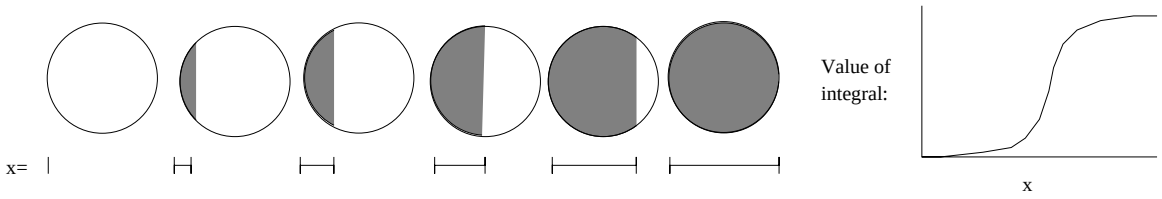

Figure 4: Value of the anti-aliasing integral as a polygon covers the footprint

This integral, which is exact for edges perpendicular to a line sample, can be calculated for continuously changing edge positions (Figure 4). As illustrated in Figure 2, these integral values can be used to compute the value for a continuous image with multiple edges.

2.4 Multiple line samples

One problem does arise when taking a single line sample for each pixel. The line sample is most accurate when edges intersect the line sample at right angles. Consider an edge in a pixel's footprint, parallel to a line sample, but not intersecting it (Figure 5). The approximated footprint will be empty, even though the real footprint is almost half full. This is avoided by having multiple line samples evenly spread around the footprint, all centered at the pixel, such that an edge is not able to cover much of the footprint without intersecting at least one of the line samples. The results of this research have shown that two perpendicular line samples are sufficient to accurately compute the anti-aliasing integral

in almost all cases.

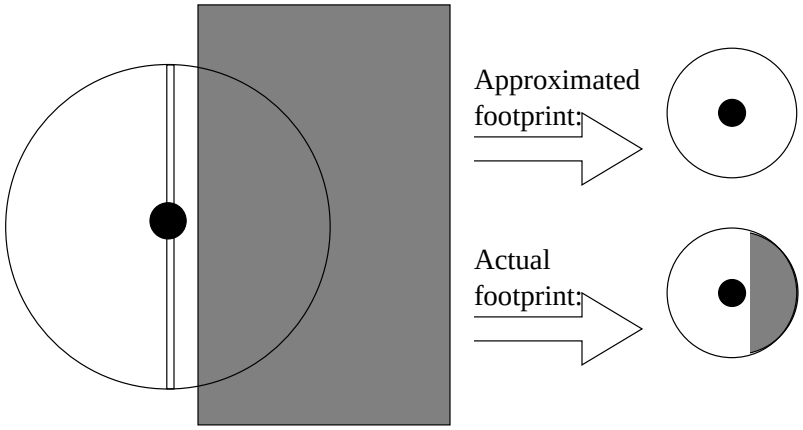

Figure 5: A Parallel edge leads to an erroneous approximation

2.5 Combining line samples

In order to use multiple line samples, there needs to be a method of combining their information into the value for a single pixel. An edge that is parallel to one line sample will not be anti-aliased correctly by that line sample, but such an edge would be better anti-aliased by a line sample with a different orientation. Unfortunately, the simple averaging of such an aliased line sample with an anti-aliased line sample gives a result which is still visibly aliased.

A line sample intersected by a perpendicular edge should have more weight than a line sample which is almost parallel to that edge. More weight is therefore given to line samples that intersect an edge perpendicularly than those which intersect closer to parallel. The total weight of a line sample is the sum of the weights given by all of the edges it intersects.

A blending function, such as a linear or cubic function, can use the weight of each line sample and its value to get a combined value for that pixel.

2.5.1 Non-visible edges

Non-visible edges occur when two triangles of the same color are adjacent to each other. In Figure 6, there are several near-horizontal non-visible edges that intersect the vertical line sample. These would skew the weight of the vertical line sample. The horizontal line sample, which would have a lower weight, is in fact the better anti-aliasing line sample. In order to correctly anti-alias this pixel, the weighting system needs to recognize those edges that are non-visible, and not weight the line sample based on the orientations of those edges.

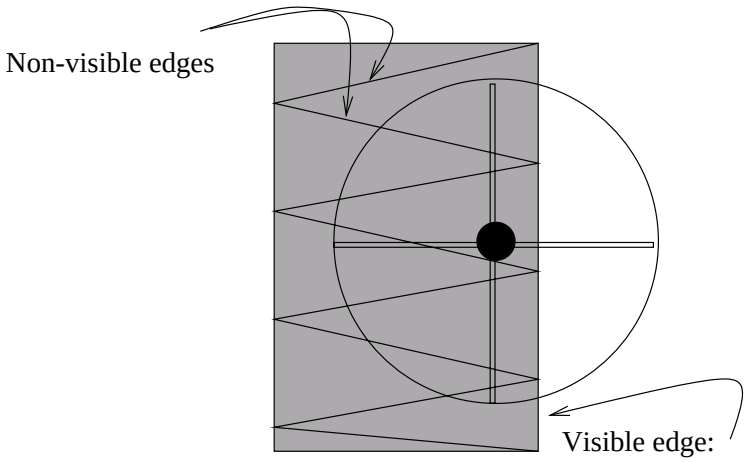

Figure 6: Weighting based on the non-visible edges would cause the visible edge to be poorly anti-aliased

2.5.2 Virtual visible edges

When two polygons intersect in three dimensions, they form a line of points on both polygons. This edge needs to be anti-aliased, even though it is not explicitly defined. The weighting system needs a method of correctly weighting line samples for such virtual edges. The direction of the edge can be computed from the normals of the polygons, and the weight can be computed from that direction using the same method as for explicit edges.

2.6 Failure cases

When trying to anti-alias an image consisting of a very large number of polygons, as is sometimes the case, line sampling would require a proportionally large amount of memory, possibly exceeding the amount available. Also, if there are polygons that are smaller than a pixel in all directions, they might not intersect any of the line samples, and therefore not affect the image. *Stochastic line sampling*, in which the orientations of the line samples are randomized, might partially alleviate this problem.

3 Method

3.1 Choosing line samples

In our experiments we used two line samples per pixel, oriented horizontally and vertically, centered at each pixel. Two line samples were found to be sufficient for all the images we tested. We investigated two possible ways of implementing these line samples.

3.1.1 The memory intensive, but fast method

One way of implementing line sampling is to keep a record of horizontal and vertical scanlines; a scanline is a line sample which spans the entire continuous image. After determining the intersections and occlusions for the polygons along each scanline, we can consider a small section of each of the relevant line samples for each pixel (Figure 7).

3.1.2 The much slower, but less memory intensive method

Alternatively, we investigated having two small line segments chosen for each pixel. After finishing the first pixel, the two line segments can be deleted and the memory reused for the next pixel. This method is significantly slower than the scanline method. However, the scanline method only works for line samples oriented along scanlines, while this method

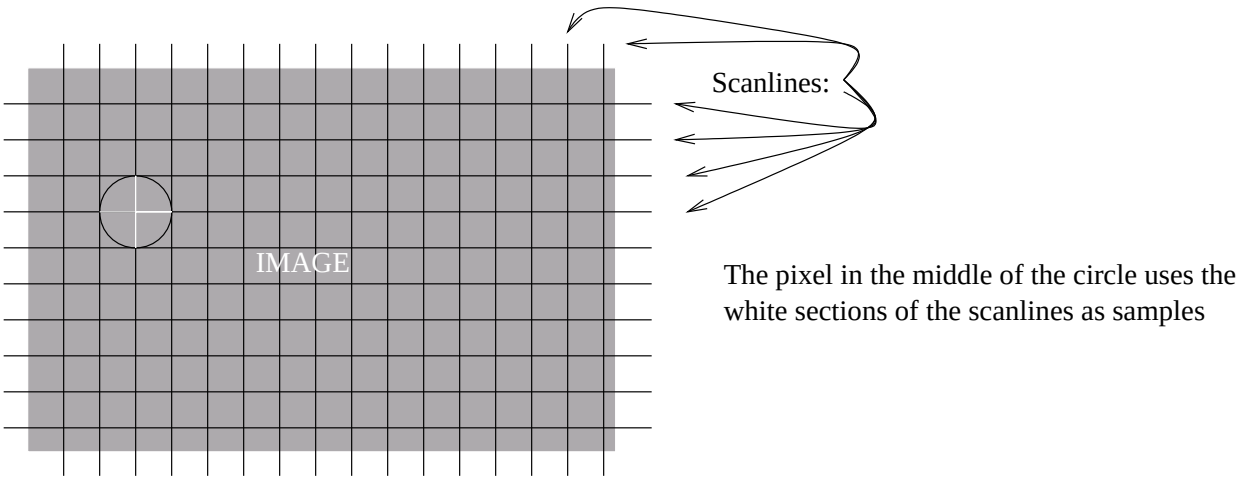

Figure 7: Scanlines covering the entire image are used as line samples

would work for independently oriented line samples.

3.2 Finding the intersection of the line samples and the triangles

To determine the intersection of a line sample and any given triangle, we used Pineda edge equations[7], a method of determining the location of a point relative to a directed edge. These edge equations have a positive value if a point is on the right side of an edge, and negative if it is on the left side. A point is internal to a triangle if the Pineda values of the point relative to all of the edges are positive when the edges are transversed clockwise. Likewise, to find where a line sample intersects a triangle, we must find where the edge equations along the line sample are positive relative to all three edges. Since the edge equations are linear in x and y, the intersection is simple to calculate.

3.3 The weighting filter

Once we have the intersection of the line sample and each polygon, we compute the visible segments along the line sample. These are then weighted by the filter, and combined with the

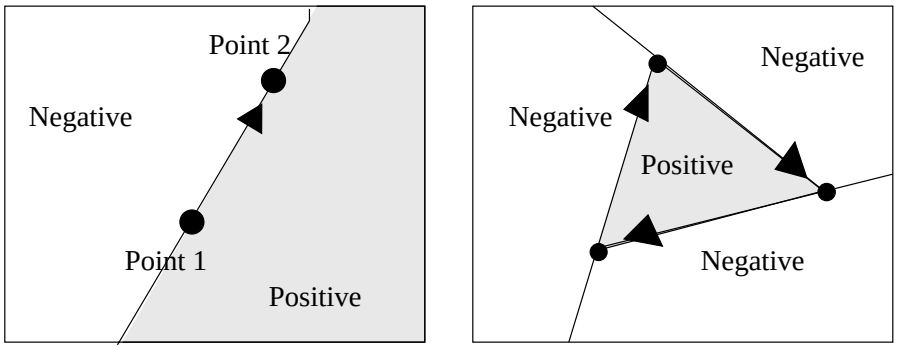

Figure 8: Pineda values can determine the location of a point relative to a directed edge

colors from the triangles to produce a pixel value. In order to use line samples, we need an efficient way to evaluate the integral from Equation 2. Because we had chosen the Gaussian for our filter, we were able to use the built-in *erf* function. The *erf* function needs to be normalized such that it returns 1 when the line sample is completely covered, and 0 when it is empty. With this normalized function, the area under the anti-aliasing filter evaluates to unity, and the filter does not darken or brighten an image incorrectly.

4 Conclusions and Future Work

This work has introduced line sampling, a new method of anti-aliasing scenes composed of two and three dimensional polygons. Line sampling simplifies the occlusion problem present in exact area convolution to one dimension. The continuous nature of line sampling yields more accurate results than supersampling. Images anti-aliased with line sampling and prior methods are shown in Appendices C and D for comparison.

Line sampling can fail to properly anti-alias very small polygons. Stochastic line sampling, where the orientations of the line samples are randomized for each pixel, might alleviate the effects of this undersampling.

Line sampling as presented is limited to anti-aliasing opaque polygons. Future extensions to line sampling might include transparent objects and curved surfaces.

5 Acknowledgments

I would like to thank to my mentor, Mr. Ray Jones, for his guidance on the project, and for his countless hours helping me with this paper.

To Ron Perry and Michael Callahan for their help with proofreading and improving this paper.

To my tutor, Jenny Sendova, for her time and help with the presentation.

To the Mitsubishi Electric Research Lab and the Massachusetts Institute of Technology, for providing this research opportunity.

And finally, to Mrs. Joann P. DiGennaro, Dr. Mark Saul, the Center for Excellence in Education, and the Research Science Institute for supporting this program and finding me the opportunity to do this research.

References

[1] Jim Blinn. *Jim Blinn's Corner: Dirty Pixels*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 1998.

[2] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, January 1986.

[3] Tom Duff. Polygon scan conversion by exact convolution. *Raster Imaging and Digital Typography*, pages 154–168, October 1989.

[4] Brian Guenter and Jack Tumblin. Quadrature prefiltering for high quality antialiasing. *ACM Transactions on Graphics*, 15(4):332–353, October 1996.

[5] Kenneth Joy, Charles Grant, Nelson Max, and Lansing Hatfield. *Tutorial: Computer Graphics: Image Synthesis*, volume 19. Computer Society Press, Washington, 1985.

[6] Nelson L. Max. Antialiasing scan-line data. *IEEE Computer Graphics and Applications*, 10(1):18–30, January 1990.

[7] Juan Pineda. A parallel algorithm for polygon rasterization. *Computer Graphics*, 22(4):17–20, August 1988.

[8] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. FLannery. *Numerical Recipes in C*. Cambridge University Press, New York, NY 10011, USA, 1992.

[9] Alvy Ray Smith. A pixel is not a little square. *Microsoft Tech Memo 6*, July 1995.

[10] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison-Wesley, 1992.

A One-dimensional aliasing

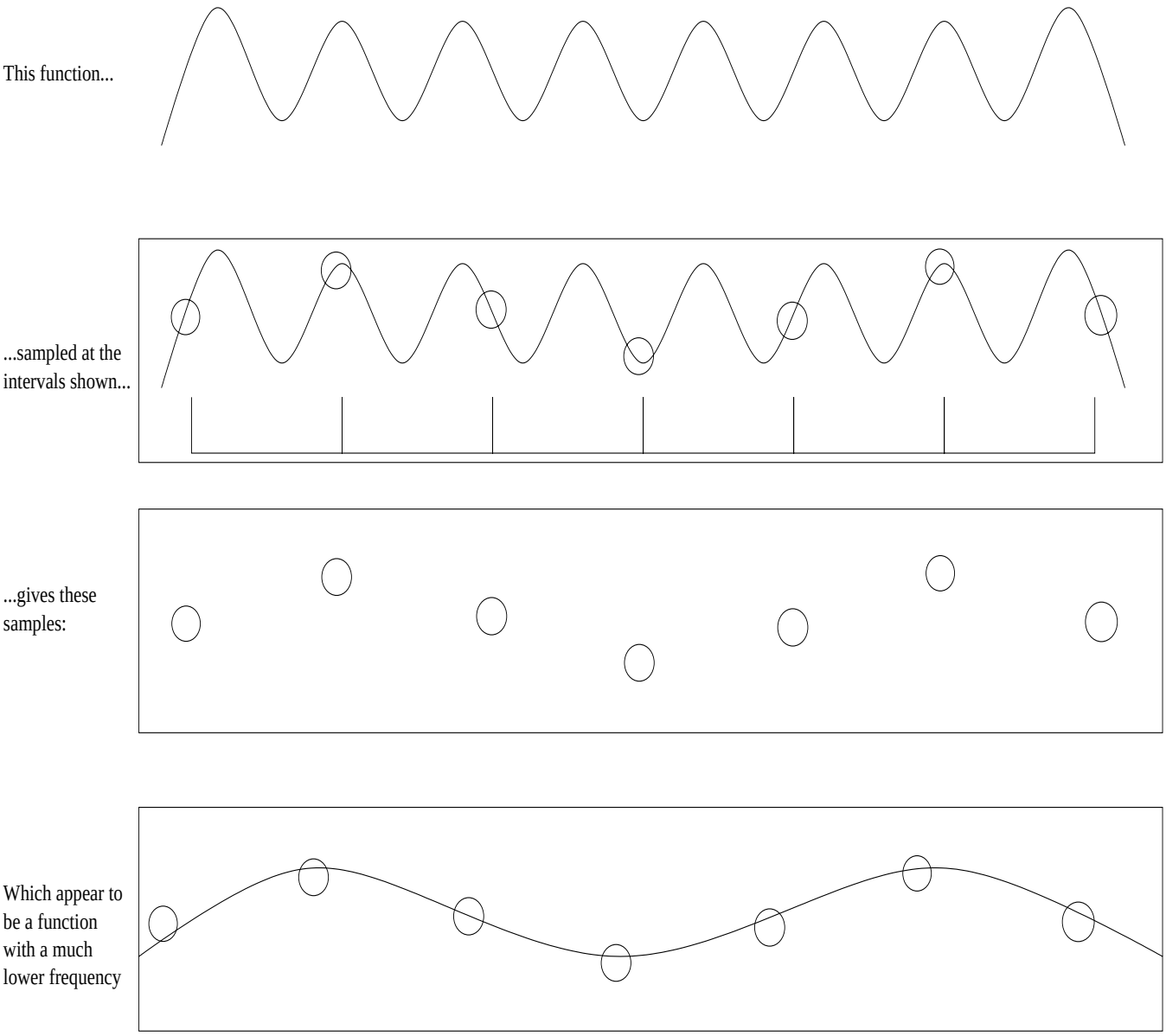

Figure 9: An example of a one-dimensional alias

Figure 9 shows how a high frequency signal can alias as a low frequency signal when the high frequency is above the Nyquist limit.

B The sinc filter

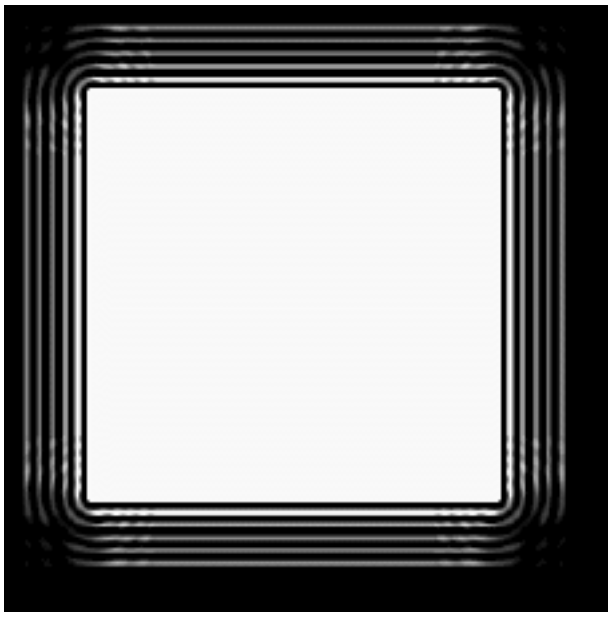

Figure 10: The sinc filter causes ringing in images

Figure 10 is the result of convolving the sinc filter with a square.

C Comparason of different anti-aliasing methods

Figures 11-15 compare different anti-aliasing methods applied to single pixel-wide lines. (Figture 14 was generated with a different program than the others, which is why the lines do not extend to the edge of the image).

Figure 11 is a point sampled representation of the image. Figure 12 was created with regular supersampling with 16 samples per pixel. Figure 13 was created with stochastic supersampling with 16 samples per pixel. Figure 14 was created with line sampling. Two line samples were used, oriented horizontally and verticially, as described in section 3.1. Figure 15 is the exact area convolution of the image. (It was not actually created with exact area convolution, but with 400 stochastic samples per pixel. This number is large enough that the difference between this and exact area convolution is negligible).

D Line sampling and occlusion

Figure 16 shows a single sampled image of one triangle intersecting another. Figure 17 is the same image as Figure 16 anti-aliased with line sampling.

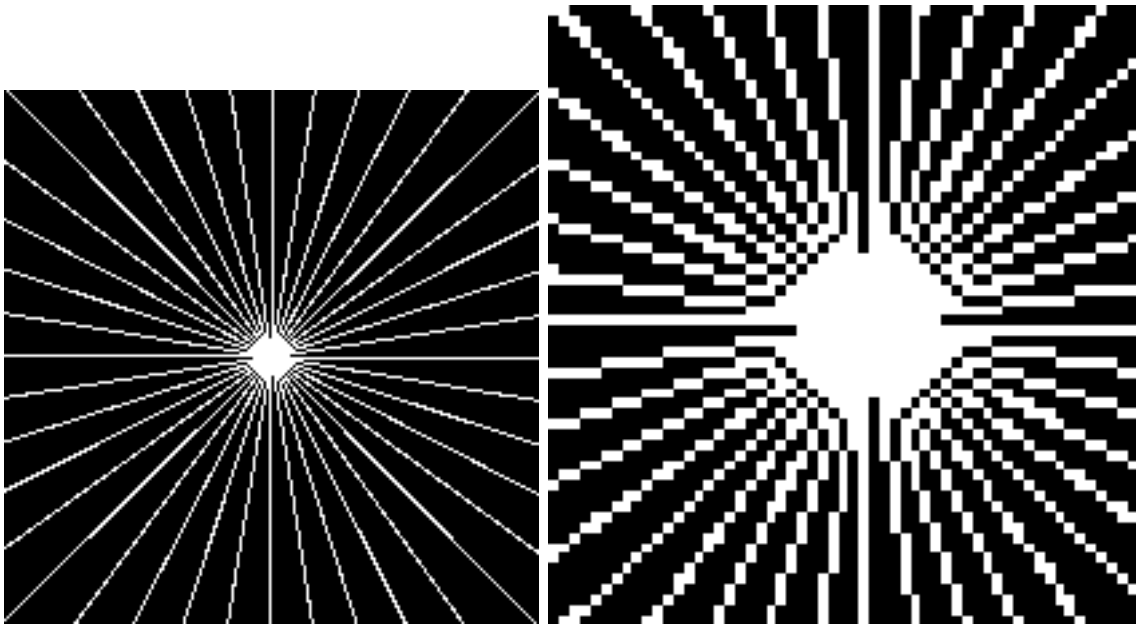

Figure 11: Single Sampling

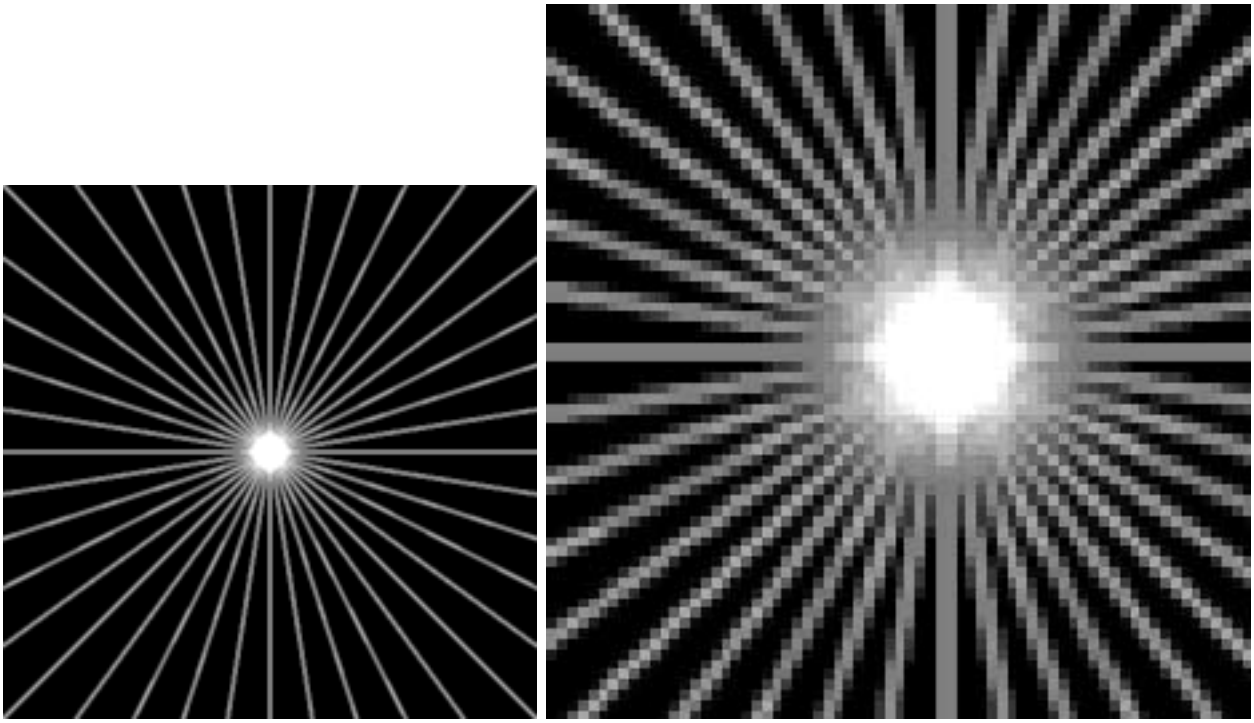

Figure 12: Regular supersampling (16 samples per pixel)

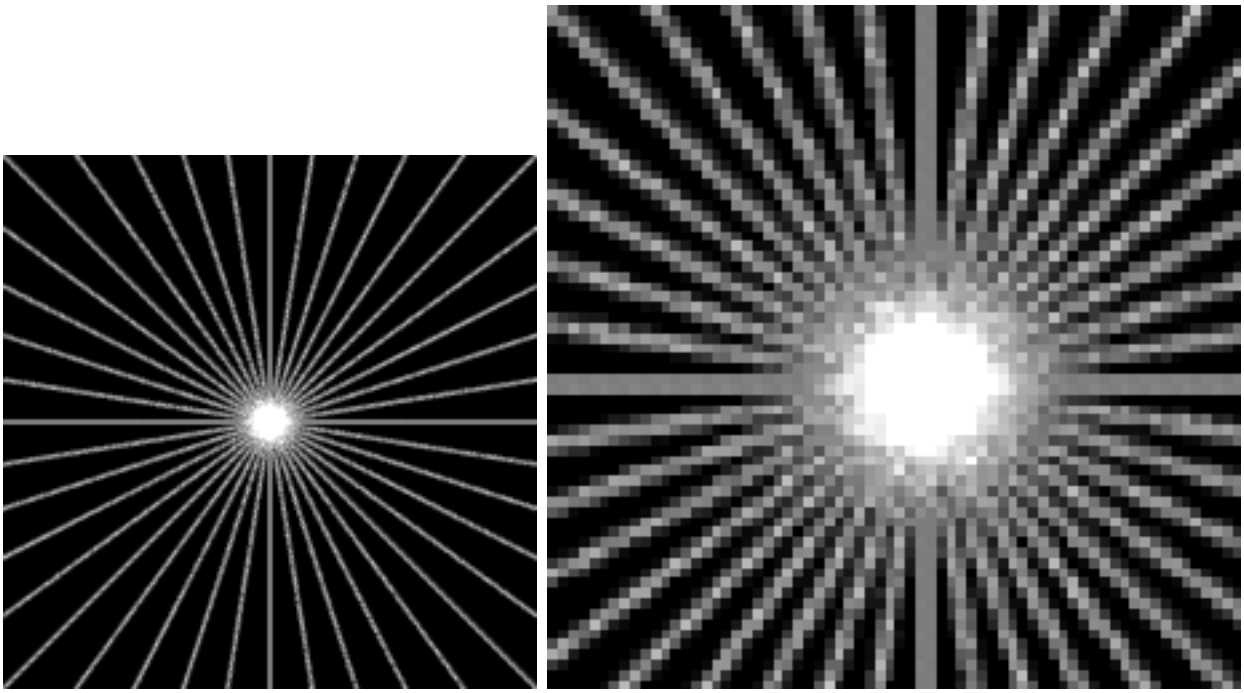

Figure 13: Stochastic supersampling (16 samples per pixel)

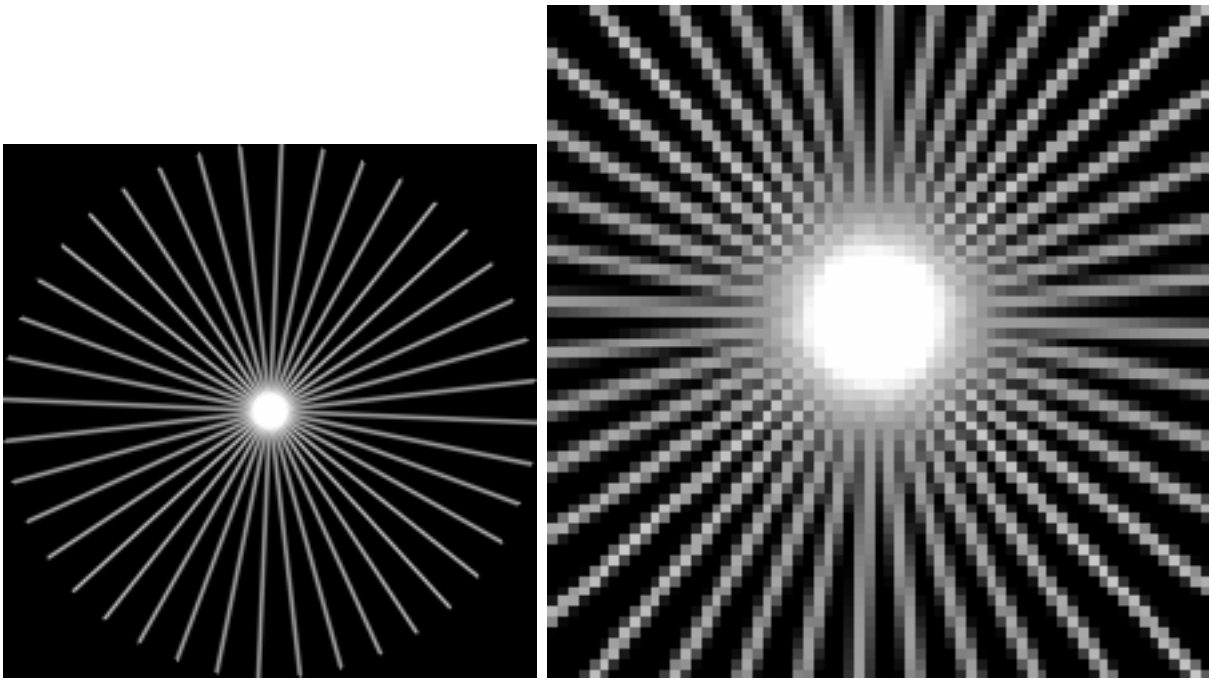

Figure 14: Line sampling

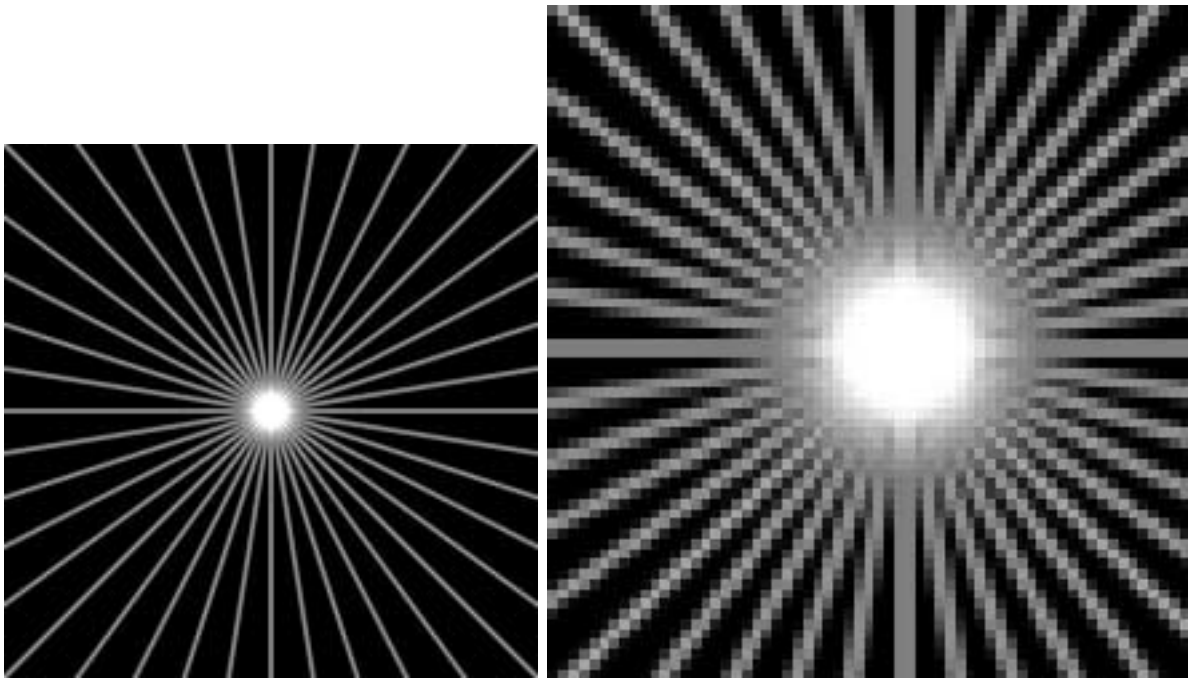

Figure 15: Exact area convolution

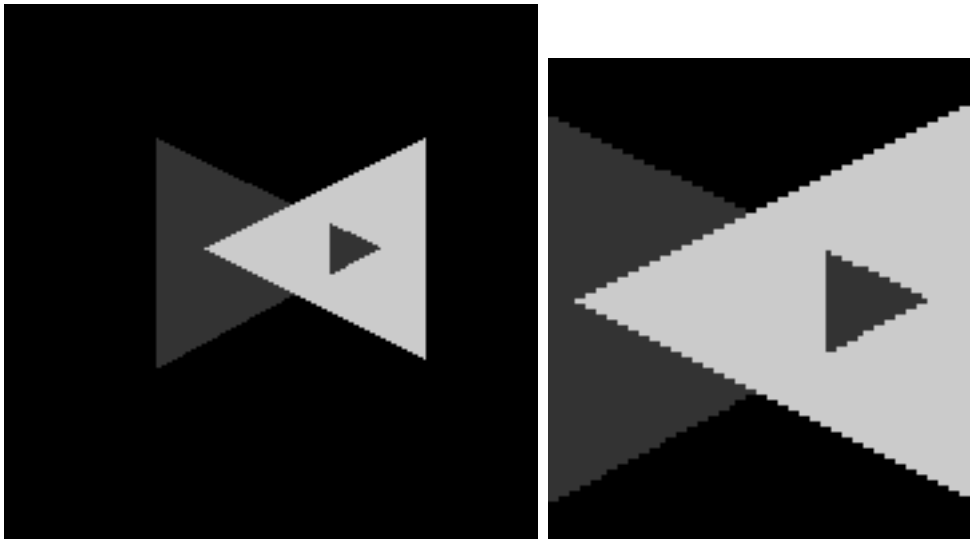

Figure 16: Two triangles intersecting ond occluding one another

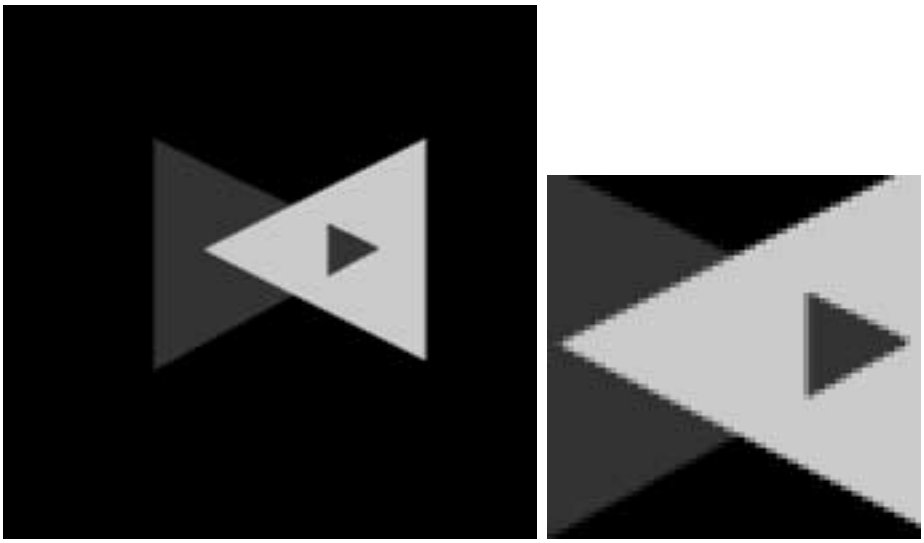

Figure 17: Image for Figure 16 with line sampling