An Efficient Antialiasing Technique

Xiaolin Wu
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract– An intuitive concept of antialiasing is developed into very efficient antialiased line and circle generators that require even less amount of integer arithmetic than Bresenham's line and circle algorithms. Unlike its predecessors, the new antialiasing technique is derived in spatial domain (raster plane) under a subjectively meaningful error measure to preserve the dynamics of curve and object boundaries. A formal analysis of the new antialiasing technique in frequency domain is also conducted. It is shown that our antialiasing technique computes the same antialiased images as Fujimoto-Iwata's algorithm but at a fraction of the latter's computational cost. The simplicities of the new antialiased line and circle generators also mean their easy hardware implementations.

CR Category: I.3.3 [**Computer Graphics**]: Picture/Image Generation - *display algorithms.*

Key Words: Antialiasing, curve digitization, digital geometry, convolution.

1 Introduction

Curve-rendering on raster devices, a fundamental operation in computer graphics, is essentially a process of quantizing (digitizing) continuous two-dimensional visual signals at the sampling rate of device resolution. This sampling rate is usually significantly lower than twice the maximum frequency of object boundaries and curve edges, [1] resulting in loss of information as explained by the Shannon sampling theorem. This information loss is the reason for the existence of visually unpleasant "aliasing" (staircasing effect) on digitized object boundaries and curves. There are two ways to attack the problem: increasing the sampling rate and removing high frequency components of the image. The first approach calls for increasing the resolution of the raster device. But the size of frame buffer and consequently the rendering costs increase quadratically in the resolution. Even at a resolution of 1024 × 1024, objectionable staircasing effects still exist. High-resolution alone is not an economic solution to the problem. The second approach of filtering high frequency components of the image was adopted by many researchers [1, 4, 5, 6, 7, 8] to combat aliasing. These techniques utilize grayscales to increase the effective spatial resolution. The disadvantages of the second approach are high computational cost involved in low-pass filtering operations, and fuzzy object edges.

Proposed in this paper is a new concept of antialiasing that leads to efficient smooth curve rendering algorithms. Our antialiasing research is done in both spatial and frequency domains. The new algorithms achieve exactly the same antialiasing effects as Fujimoto-Iwata's algorithm for line segments but at a fraction of the latter's cost. A new antialiased line generator is designed for smooth line generation that requires only half as much integer arithmetic as Bresenham's line algorithm [2]. And the antialiased line generation can be easily implemented by hardware. Smooth circles can also be generated by the new technique

[1] For physical displays a curve should be modeled as a narrow 2-dimensional image rather than a 1-dimensional mathematical entity of no area.

 ACM-0-89791-436-8/91/007/0143 $00.75

at a lower cost than Bresenham's circle algorithm [3]. The paper is organized as follows. In the next section a dynamic error measure for the quality of digitized curves is introduced, and the correspondence between the measure and the image quality is demonstrated. Then based on this error measure the new antialiasing concept is introduced in section 3. The rationale for the new antialiasing algorithm is also established using convolution theorem, hence it is in principle congruent to the current antialiasing algorithms. In section 4 we prove the equivalence between our algorithm and Fujimoto-Iwata's algorithm. In sections 5 and 6 the high efficiency of the new antialiasing technique is demonstrated by the development of fast antialiased line and circle generators. Section 7 deals with the generalization of the new antialiasing technique to general curves and to antialiased object boundaries blent in colorful background.

2 Dynamic Error in Curve Digitization

Previously image aliasing was investigated in the frequency domain. In this section we study image aliasing in the spatial domain (raster plane). Some of our previous results in digital geometry [10, 12] are used to study the quality of digitized curves. Let $y = f(x)$ be a differentiable curve to be digitized in the raster plane, and partition the curve into segments where either $0 \leq |f'(x)| \leq 1$ or $1 < |f'(x)| < \infty$, called x-dominant and y-dominant segments, respectively. Now consider an x-dominant curve segment without loss of generality (the discussion on y-dominant curve segments is the same through symmetry). Then the digitization of this curve segment is defined to be an ordered point set $\{(i, Y_i) : 1 \leq i \leq N\}$. This definition means that the curve segment is sampled in unit raster steps along the x axis, and the sample value $f(i)$ is quantized to Y_i. Due to the finite precision of the raster plane Y_i must be an integer, resulting in the commonly used quantization scheme

$$Y_i = \left\lfloor f(i) + \frac{1}{2} \right\rfloor \qquad (1)$$

to minimize the y distance between the sampled value $f(i)$ and its image point in the raster plane. But how meaningful in terms of human perception is this simple criterion Eq(1)? Let us consider the geometry of Fig.1 where the three pixels indicated by solid dots are chosen

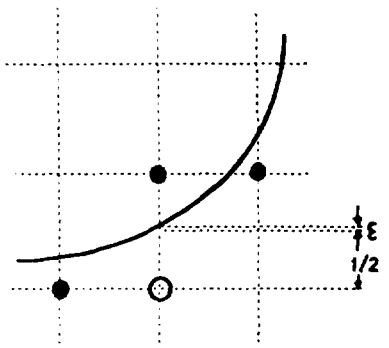

Figure 1: Dynamic error in curve digitization.

by Eq(1) as the discrete image of the continuous curve. If, however, the pixel labeled by o replaces the one just above it, then the so-called dynamic error defined by

$$E_{i,j} = f(j) - f(i) - [Y_j - Y_i] \qquad (2)$$

is minimized. The above dynamic error relates to the first-order difference and hence characterizes the digitization error in curve dynamics. Visually, the new pixel configuration obtained by pulling the pixel in the middle column down by one raster unit presents a better approximation to the original curve. This improvement results because the pixel pattern of the solid dots distorts the dynamic context of the original curve segment. Namely, the convex curve $f(x)$ is mapped to a concave pixel pattern. By moving the middle pixel down by a raster unit, the convexity is preserved, resulting in a more pleasant rendering. Human eyes are more sensitive to the dynamic context of a curve than to its absolute spatial position. It is difficult for viewers to detect a translation of an object if the amount of shift is relatively small compared with the size of background, but easy to catch a slight distortion of the dynamic context of the object as a disturbing image alias. This observation suggests that the error measure Eq(2) is subjectively more meaningful than Eq(1), and antialiasing should aim for minimizing the loss of dynamic information of original curves due to digitization. Given an x-dominant curve segment $f(x)$ and its digitization $\{(i, Y_i) : 1 \leq i \leq N\}$, an $N \times N$ matrix of dynamic errors $\{E_{i,j}\}$, $1 \leq i, j \leq N$, is defined (see [12] for more detailed discussions on the dynamic error matrix $\mathbf{E}$). Our goal is to minimize $\| \mathbf{E} \|$, the norm of the error matrix. For binary raster displays minimizing $\| \mathbf{E} \|$ is a very difficult optimization problem [12]. Fortunately, for grayscale devices we can have a simple solution to the problem.

3 Two-Point Anti-Aliasing Scheme

The dynamic error is caused by rounding $f(i)$ to an integer Y_i. The dynamic error matrix $\mathbf{E}$ becames a zero matrix, i.e., $\| \mathbf{E} \| = 0$, if Y_i were chosen to be $f(i)$. We would like to have an addressable pixel centered at the coordinates $(i, f(i))$. Let $I[i,j]$ be the intensity of the pixel (i,j) and I_0 be the intended intensity for the curve. Then the imaginary pixel $(i, f(i))$ may be visually simulated by setting

$$\begin{cases} I[i, \lfloor f(i) \rfloor] = I_0(\lceil f(i) \rceil - f(i)) \\ I[i, \lceil f(i) \rceil] = I_0(f(i) - \lfloor f(i) \rfloor) \end{cases} \quad (3)$$

If we consider the pixel (i,j) as a unit square centered at (i,j) containing light energy $I[i,j]$, then point $\mathbf{p}_\mu = (i, f(i))$ is the center of gravity of the two lit points $\mathbf{p}_0 = (i, \lfloor f(i) \rfloor)$ and $\mathbf{p}_1 = (i, \lceil f(i) \rceil)$, because

$$I_0 \mathbf{p}_\mu = I[i, \lfloor f(i) \rfloor] \mathbf{p}_0 + I[i, \lceil f(i) \rceil] \mathbf{p}_1. \quad (4)$$

Therefore, the overall effect of Eq(3) is a lit area of energy I_0 focused at the real point $\mathbf{p}_\mu = (i, f(i))$ which is a perceived pixel exactly on the original curve $f(x)$. The ordered set $\{(i, f(i)) : 1 \le i \le N\}$ of those perceived pixels renders a perceived curve. Clearly the dynamic error $\| \mathbf{E} \|$ for this perceived curve is zero, eliminating the loss of dynamic information. The practical significance of Eq(3) is its simplicity which leads very fast anti-aliasing algorithms as we will see later. Eq(3) is a two-point antialiasing scheme. We plot all pixels in the two-pixel wide band that bounds the true curve $y = f(x)$ with their intensities inversely proportional to the distances between these pixels to the curve. The closer is a pixel to the line, the brighter it is, then the overall visual effect of this band will be the illumination area of the lit curve at its real position after our eyes integrate the contributions of all pixels in the band.

In addition to being intuitively appealing the antialiasing scheme Eq(3) can also relates to removing high frequency components of sharp intensity jumps at the image edges. In order to apply a filter to the image we no longer treat $y = f(x)$ as a mathematical curve of no width; instead we model the curve by a two-dimensional grayscale signal $g(x,y)$ with interior intensity I_0 and exterior intensity 0. The curve $y = f(x)$ is the center line of the two-dimensional signal $g(x,y)$. The image intensity $I(i,j)$ after applying a low-pass filter to $g(x,y)$ is given by

$$I[i,j] = \int\int \delta(u,v) g(i-u, j-v) du dv. \quad (5)$$

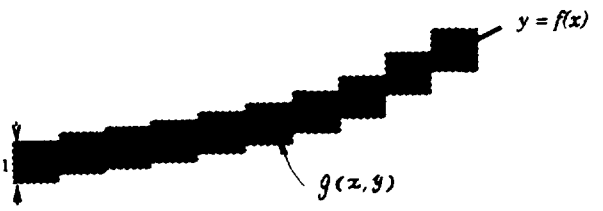

Figure 2: The two-dimensional signal $g(x,y)$ modeling the physical image of the curve $y = f(x)$ before filtering.

The convolution kernel $\delta(u,v)$ is determined by the intensity density of a pixel in its neighborhood. It is easy to verify that if we choose the box filter

$$\delta(u,v) = \begin{cases} 1 & |u| \le \frac{1}{2}, |v| \le \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

and model the curve $y = f(x)$ in raster plane by the two-dimensional signal

$$g(x,y) = \begin{cases} I_0 & |y - f(\lfloor x + \frac{1}{2} \rfloor)| \le \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

then the solution of the convolution Eq(5) is the simple expression of Eq(3). The signal $g(x,y)$ is a chain of two-dimensional unit square impulses as depicted by Fig. 2. The above analysis reveals that the two-point anti-aliasing scheme Eq(3) is a two-step process. First the image of the curve $y = f(x)$ is modeled by the two dimensional impulse signal signal $g(x,y)$ of Eq(7), then the impulse signal is put through the box filters $\delta(u,v)$ centered at individual pixels. The additive responses of these atomic filters yields the antialiased digital curve.

Admittedly the above anti-aliasing model is far from ideal. The box filter does not reflect the fact that the intensity density of a pixel has Gaussian-like rather than uniform shape. Moreover, the curve $y = f(x)$ is modeled by a stripe image $g(x,y)$ whose edge is not smooth. But aliasing is primarily caused by sharp intensity changes (high frequency components at the intensity transition from $g(x,y) = 0$ to $g(x,y) = I_0$). The low-pass filtering aims at smoothing the steep intensity jump not at smoothing the geometric shape of the input signal. The tendency of $g(x,y)$ to preserve the dynamic information of $y = f(x)$ is far more important than its geometric smoothness in our principle of antialiasing.

The staircase appearance of the $g(x, y)$ will be eventually subdued since the low-pass filter will blur the input image $g(x, y)$ anyway.

For comparison Fig.3 gives three groups of lines with various orientations done by Bresenham's, Gupta-Sproull's and the two-point antialiasing scheme Eq(3). Gupta-Sproull's antialiased line algorithm [7], generally regarded as a better performed one, uses a cone-shaped low-pass filter as an approximation of Gaussian filter to suppress the jaggies. The algorithm understandably is quite computationally demanding. The photos show that the line images produced by the new technique are not inferior in quality to those produced by Gupta-Sproull's algorithm in quality. Note that Gupta-Sproull's algorithm is a three-point antialiasing scheme in the sense that in each column three pixels are usually set to different intensities. Consequently, the lines generated by this algorithm look fuzzier than those done by the two-point scheme.

Our real motive for developing the model Eq(7) is to convert the convolution integration of Eq(5) to the simple intensity interpolation between two adjacent pixels in Eq(3), gaining computational efficiency of antialiasing as we will see in sections 5 and 6.

4 Equivalence to Fujimoto-Iwata's Algorithm

Interestingly, we can prove the equivalence between the simple formula Eq(3) and the seemingly more complicated antialiasing operation by Fujimoto and Iwata [6]. Indeed, after some intricate derivation, Fujimoto and Iwata arrived at

$$\begin{aligned} I[i, \lfloor f(i) \rfloor] &= I(d - 2d_1)/d \\ I[i, \lceil f(i) \rceil] &= I(d - 2d_2)/d, \end{aligned} \tag{8}$$

where, as marked in Fig. 4, $d = 2\cos\alpha$, d_1 and d_2 are the distances from the pixels $(i, \lfloor f(i) \rfloor)$ and $(i, \lceil f(i) \rceil)$ to the true line. Eq(8) is the formula for antialiased lines using the smallest Fourier window. It is apparent from the figure that

$$\begin{aligned} d_1 &= (f(i) - \lfloor f(i) \rfloor)\cos\alpha \\ d_2 &= (\lceil f(i) \rceil - f(i))\cos\alpha. \end{aligned} \tag{9}$$

Plugging d_1 and d_2 into Eq(8) we can simplify Eq(8) to Eq(3).

The above simplification gives Fujimoto-Iwata's antialiasing algorithm a more intuitive interpretation of

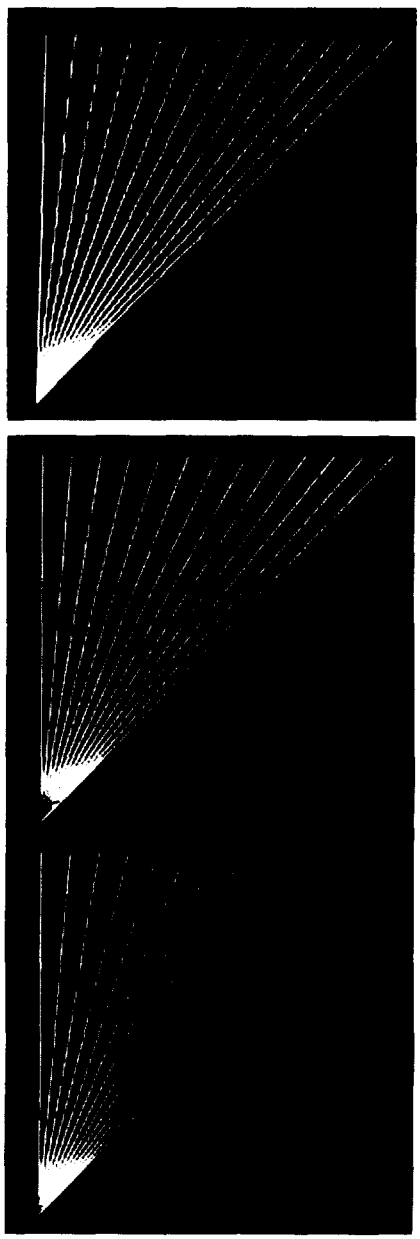

Figure 3: Lines generated by Bresenham's (above), Gupta-Sproull's (middle) and the two-point antialiasing (bottom) algorithms.

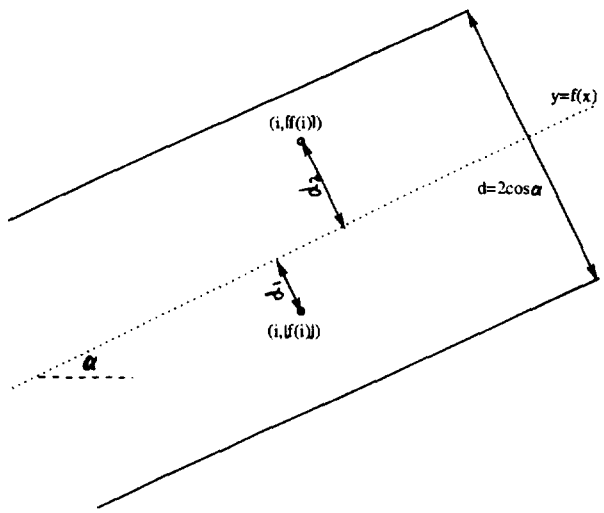

Figure 4: The geometry of Fujimoto-Iwata's antialiasing algorithm.

Eq(3), another analytical basis formed by Eqs(5)-(7), and more importantly, a simpler and more efficient implementation.

5 Fast Anti-Aliased Line Generator

In this section we convert the simple antialiasing scheme Eq(3) to a fast antialiased line generator. Without loss of generality only lines in the first octant are considered. Other cases follow trivially through symmetry. Let $(x0, y0)$, $(x1, y1)$, $x2 > x1$, $y2 > y1$, be the two points in the raster plane defining a line. We translate the point $(x0, y0)$ to the origin, so the equation of the line becomes $y = kx$, $0 \leq k = \frac{y1-y0}{x1-x0} \leq 1$. Then Eq(3) can be rewritten for $f(x) = kx$ as

$$\begin{aligned} I(x, \lceil kx \rceil) &= I_0(kx - \lfloor kx \rfloor) \\ I(x, \lfloor kx \rfloor) &= I_0 - I(x, \lceil kx \rceil). \end{aligned} \quad (10)$$

where $(x, \lfloor kx \rfloor)$ and $(x, \lceil kx \rceil)$ are the two adjacent pixels in the x column that are immediately below and above the true line. Clearly, the total intensity in a column is the constant I, so the even brightness of the band can be achieved.

To implement the antialiasing scheme Eq(10), we need to determine for a given x the pixel positions $(x, \lfloor kx \rfloor)$ and $(x, \lceil kx \rceil)$ (they coincide if kx is an integer) and their intensities $I(x, \lfloor kx \rfloor)$ and $I(x, \lceil kx \rceil)$. These four values can be determined by an elegant incremental algorithm operating on a single integer D represented by a machine word of n bits. The integer increment involved is $d = \lfloor k2^n + 0.5 \rfloor$. As the initialization, we set $D = 0$, $I(x0, y0) = I$. Then we march x from $x0$ to $x1$ and increment D by d at unit step. The operation $D \leftarrow D + d$ is a module 2^n addition with the overflow recorded. Whenever D overflows the two-point high pixel band pixel moves diagonally; otherwise it moves horizontally. This is essentially a classical DDA method. The only difference is in that both the x and y increments, namely, $\Delta x = 1$ and $\Delta y = d$, are integer rather than real values. For the following analysis we may consider D as a fixed point number with the decimal point before its most significant bit, or conceptually perceive the proposed integer arithmetic as fixed point arithmetic. Thus the error between the real DDA increment and our integer DDA increment is

$$e = k - d2^{-n}. \quad (11)$$

Clearly, $|e| < 2^{-n}$, and this error will be shown to be negligible.

All gray-scale raster devices have 2^m, for some $m > 1$, discrete intensity levels from 0 (absolutely black) to $2^m - 1$ (absolute white). Thus the intensity interpolation between the two vertically adjacent pixels of Eq(10) becomes a bi-partition of the integer I, the maximum intensity. The intensity of the upper pixel for the line is

$$\begin{aligned} I(x, \lceil kx \rceil) &= I_0(kx - \lfloor kx \rfloor) \\ &= (2^m - 1)(D2^{-n} + ex) \\ &= D2^{m-n} + (2^m - 1)ex - D2^{-n}. \end{aligned} \quad (12)$$

Since the intensity $I(x, \lceil kx \rceil)$ must be an integer, we approximate it by the first term of Eq(12), $D2^{m-n}$, assuming $n > m$. This approximation gains great computational efficiency while the error incurred (the last two terms of Eq(12) has no or little impact on image quality as we will analyze later.

The approximated $I(x, \lceil kx \rceil) \approx D2^{m-n}$ is simply presented by the m most significant bits of D. Moreover, it is evident that the intensity of the lower pixel $I(x, \lfloor kx \rfloor) = I_0 - I(x, \lceil kx \rceil) = \overline{I(x, \lceil kx \rceil)}$, where $\overline{I(x, \lceil kx \rceil)}$ is the integer obtained by the bitwise-inverse operation on $I(x, \lceil kx \rceil)$. This is because the bit pattern for the integer $2^m - 1 - D2^{m-n}$ is the inverse of that for $D2^{m-n}$ due to the fact $I_0 = 2^m - 1$. Now we can see that the integer D controls both pixel positions

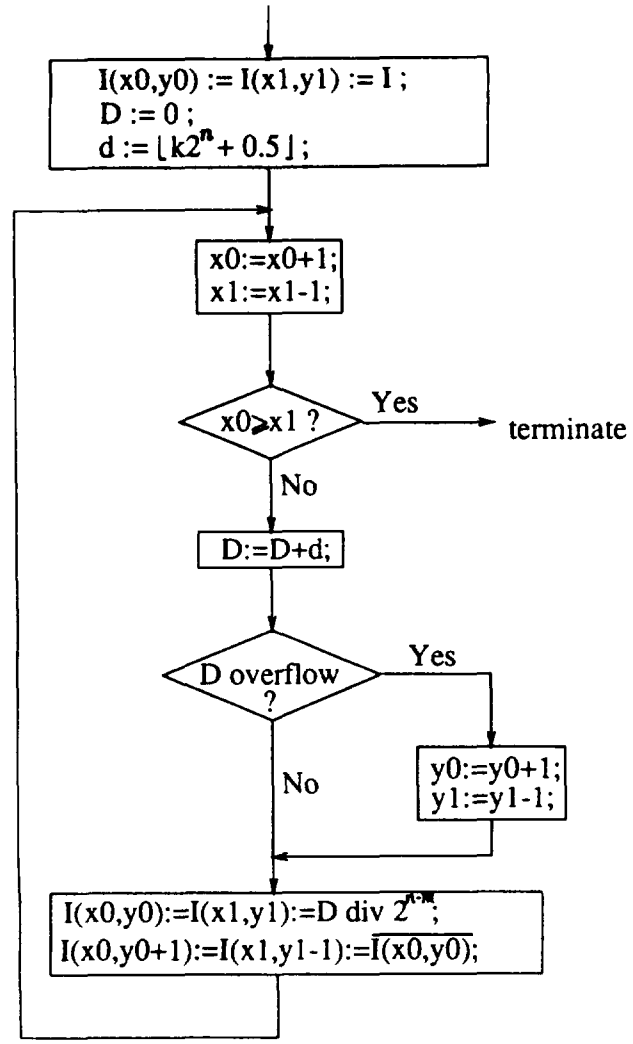

Figure 5: The antialiased line generator ($0 \leq k \leq 1$).

and intensities, and the inner loop of the algorithm only requires an integer addition to D.

Furthermore, since the line segment has mirror symmetry with respect to its center, we can plot it from the two ends toward the center using the same logic [13], saving half of the computations. The new algorithm for lines with $0 \leq k \leq 1$ is described by the flowchart in Fig. 5.

Unlike all its predecessors, the above antialiased line generator requires only integer addition and bit manipulations. While producing smooth lines, the new algorithm requires only half as many operations as Bresenham's algorithm because it propagates two pixels per iteration while using the same amount of computations per iteration. The numbers of different operations for plotting a line of length N required by Bresenham's algorithm and the new antialiased line algorithm are tabulated in Table 1.

	Bresenham	New
Addition	N	$N/2$
Sign Test	N	$N/2$
Comparison	N	$N/2$
Buffer Writing	N	$< 2N$

Table 1: The number of different operations required by Bresenham's algorithm and the new antialiased line algorithm.

The new algorithm requires twice as many buffer writes as Bresenham's algorithm, but still its buffer access, a bottleneck in rendering, is a minimum (tie with Fujimoto and Iwata's algorithm with the smallest Fourier window) among all current antialiased line generators.

An attractive feature of the new antialiased line generator is that it simultaneously meets two usually mutually exclusive criteria: good image quality and high computational efficiency. At the same time the logic of the new line algorithm remains simple and its hardware implementation is straightforward. An integer adder for D is all we need with its overflow controlling the pixel positioning and its original and inverse values being the required intensities. Historically, we used a very crude fifty-fifty intensity split scheme as a trick to speed up curve scan-conversion [9, 11] under a guise of antialiasing. The above work drew a satisfactory conclusion to our attempt to unify antialiasing and scan-conversion.

The new algorithm is not complete without an error bound for the approximation it employs. The error in approximating $I(x, \lceil kx \rceil)$ by $D2^{m-n}$ is determined by the magnitude of x and the difference $n - m$. Let L be the line length. If $2^{t-1} < L \leq 2^t$, $t > 0$, then the truncation error can be bounded by

$$\begin{aligned} \left| I_0(kx - \lfloor kx \rfloor) - D2^{m-n} \right| &< (2^m - 1)2^{-n}2^{t-1} + D2^{-n} \\ &< 2^{t-n-1}(2^m - 1) + 1. \quad (13) \end{aligned}$$

In the above inequality we used the facts that $D < 2^n$, $|e| < 2^{-n}$, and $x \leq L/2$ due to symmetric generation of the antialiased line. For $L < 2^{n-m+1}$ the error in $I(x, \lceil kx \rceil)$ has a magnitude less than 2. In our experiments, a 10% relative error in distributing I between two adjacent pixels does not lead to noticeable degradation in image quality. It was also observed that 32

different gray scales are sufficient to eliminate the most of aliasing. For a 1024 x 1024 display, the maximum $L < 2^{11}$ ($t = 11$). Suppose that 32 gray scales ($m = 5$) is used for antialiasing. Then we need $n \geq 15$ to bound the error in $I(x, \lceil kx \rceil)$ by 2, or the relative error by 0.063. This only requires D to be a two-bytes integer.

Our recent research revealed that the proposed antialiased line algorithm is particularly suitable to be incorporated into a logic-enhanced frame buffer to solve the bottleneck of frame buffer access [14]. An intelligent frame buffer architecture in the form of wavefront array processors was designed to scan-convert lines right inside the frame buffer. This design achieves extremely high rendering throughput with very low frame buffer bandwidth requirement.

6 Fast Anti-Aliased Circle Generator

Due to the 8-way symmetry of the circle, it suffices to consider the circle $x^2 + y^2 = r^2$ in the first octant. For the circle equation, the two-point antialiasing scheme Eq(3) becomes

$$\begin{aligned} I\left(\left\lfloor\sqrt{r^2-j^2}\right\rfloor, j\right) &= I\left(\left\lceil\sqrt{r^2-j^2}\right\rceil - \sqrt{r^2-j^2}\right) \\ I\left(\left\lceil\sqrt{r^2-j^2}\right\rceil, j\right) &= I - I\left(\left\lfloor\sqrt{r^2-j^2}\right\rfloor, j\right), \\ 1 \leq j \leq \frac{r}{\sqrt{2}}. & \end{aligned} \tag{14}$$

Now we derive the algorithm to compute Eq(14) as j marches in the y axis from 0 to $\frac{r}{\sqrt{2}}$ in scan-converting the first octant circular arc. The first issue is to determine when the integer-valued function $\left\lceil\sqrt{r^2-j^2}\right\rceil$ decreases by 1 as j increases. We need the critical values t such that $\left\lceil\sqrt{r^2-(t-1)^2}\right\rceil - \left\lceil\sqrt{r^2-t^2}\right\rceil = 1$ to move the pixel band being plotted to the left by one step. This computation can be simplified by the following lemma.

LEMMA 1
The relation $\left\lceil\sqrt{r^2-(t-1)^2}\right\rceil - \left\lceil\sqrt{r^2-t^2}\right\rceil = 1$ *holds if and only if* $\left\lceil\sqrt{r^2-(t-1)^2}\right\rceil - \sqrt{r^2-(t-1)^2} > \left\lceil\sqrt{r^2-t^2}\right\rceil - \sqrt{r^2-t^2}$.

Proof. Since $\sqrt{r^2-j^2}$ is monotonically decreasing in j, $\left\lceil\sqrt{r^2-(t-1)^2}\right\rceil - \sqrt{r^2-(t-1)^2} > \left\lceil\sqrt{r^2-t^2}\right\rceil - \sqrt{r^2-t^2}$ implies $\left\lceil\sqrt{r^2-(t-1)^2}\right\rceil - \left\lceil\sqrt{r^2-t^2}\right\rceil > 0$. But in the first octant we have

$$\sqrt{r^2-(t-1)^2} - \sqrt{r^2-t^2} \leq 1 \tag{15}$$

prohibiting $\left\lceil\sqrt{r^2-(t-1)^2}\right\rceil - \left\lceil\sqrt{r^2-t^2}\right\rceil > 1$, hence $\left\lceil\sqrt{r^2-(t-1)^2}\right\rceil - \left\lceil\sqrt{r^2-t^2}\right\rceil = 1$.

The only-if part can be proven by contradiction. Assume that $\left\lceil\sqrt{r^2-(t-1)^2}\right\rceil - \left\lceil\sqrt{r^2-t^2}\right\rceil = 1$ but $\left\lceil\sqrt{r^2-(t-1)^2}\right\rceil - \sqrt{r^2-(t-1)^2} \leq \left\lceil\sqrt{r^2-t^2}\right\rceil - \sqrt{r^2-t^2}$. This requires $\sqrt{r^2-(t-1)^2} - \sqrt{r^2-t^2} > 1$, an impossibility in the first octant. □

For given r the values $\left\lceil\sqrt{r^2-j^2}\right\rceil - \sqrt{r^2-j^2}$, $1 \leq j \leq \frac{r}{\sqrt{2}}$, serve dual purposes: determining the pixel positions as suggested by the above lemma and determining the pixel intensities as in Eq(14). Let the intensity range for the display be from 0 to $2^m - 1$ and define the integer variable

$$D(r,j) = \left\lfloor (2^m - 1)\left(\left\lceil\sqrt{r^2-j^2}\right\rceil - \sqrt{r^2-j^2}\right) + 0.5 \right\rfloor \tag{16}$$

Then it follows from Eq(14) that

$$\begin{aligned} I\left(\left\lfloor\sqrt{r^2-j^2}\right\rfloor, j\right) &= D(r,j) \\ I\left(\left\lceil\sqrt{r^2-j^2}\right\rceil, j\right) &= \overline{D(r,j)}, \quad 1 \leq j \leq \frac{r}{\sqrt{2}}, \end{aligned} \tag{17}$$

where $\overline{D(r,j)}$ is the integer value obtained through bitwise-inverse operation on $D(r,j)$ since

$$I\left(\left\lceil\sqrt{r^2-j^2}\right\rceil, j\right) + I\left(\left\lfloor\sqrt{r^2-j^2}\right\rfloor, j\right) = I = 2^m - 1 \tag{18}$$

and since the intensity values are integers. By Eq(16) every decrement of the function $\left\lceil\sqrt{r^2-j^2}\right\rceil - \sqrt{r^2-j^2}$ as j increases is reflected by a decrement of $D(r,j)$, thus $D(r,j)$ can be used to control the scan-conversion of the circle. The new antialiased circle algorithm based on precomputed $D(r,j)$ is extremely simple and fast. The algorithm for the first octant is described by the flowchart in Fig. 6.

The inner loop of the antialiased circle algorithm requires even fewer operations than Bresenham's circle algorithm. Of course, the gains in image quality and scan-conversion speed are obtained by using the $D(r,j)$ table. If R_{max} is the maximum radius handled by the circle generator, then the table size will be $\frac{\sqrt{2}}{4}R_{max}$. It is my opinion that the rapidly decreasing memory cost makes the above simple idea a viable solution to real-time antialiased circle generation. For instance, for a 64K bytes ROM the above algorithm can display antialiased circular arcs of radius up to 430. Without the

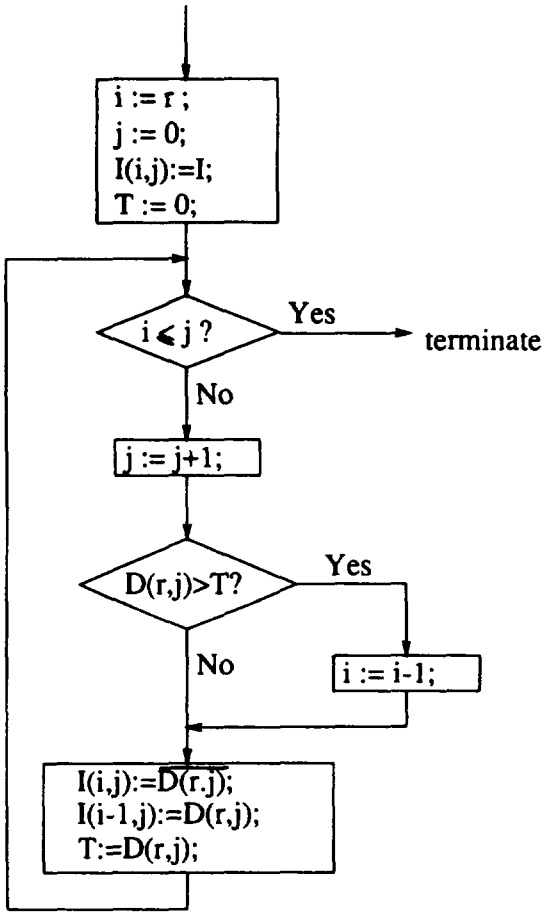

Figure 6: Antialiased circle generator (1st octant).

precomputed table $D(r,j)$ the antialiased circle algorithm can be implemented by computing the function $D(r,j)$.

The performance of the new antialiased circle algorithm is demonstrated by Fig. 7.

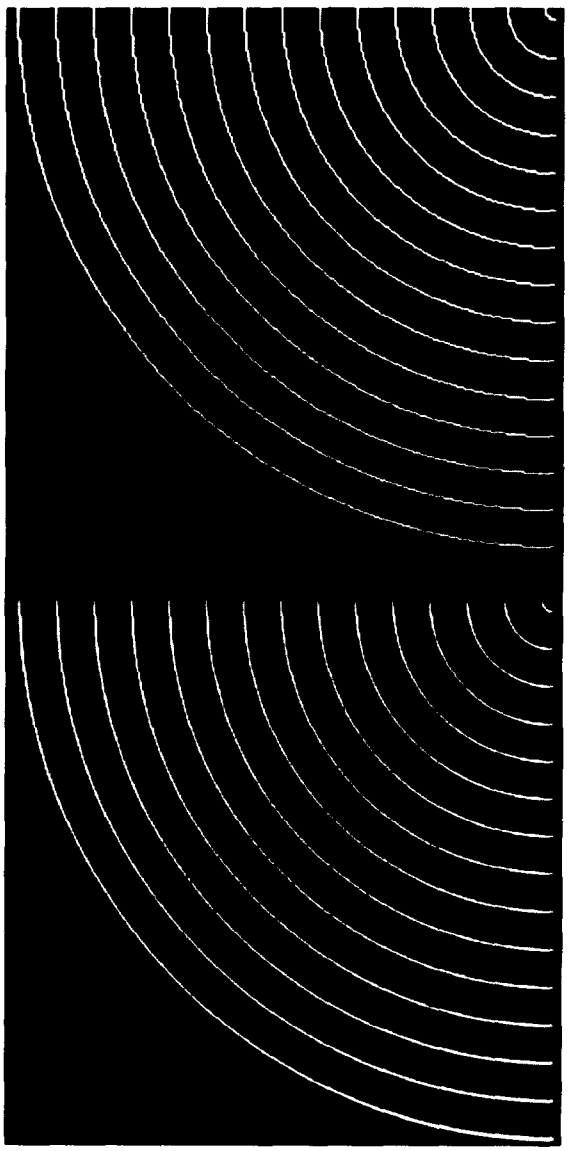

Figure 7: Circles by Bresenham's (above) and the new antialiasing algorithms (below).

7 Other Antialiasing Issues

We demonstrated that the new antialiasing technique is particularly efficient for generating antialiased lines and circles since Eq(3) can be incorporated into the classical incremental curve scan-conversion framework. But it is not restricted to those two graphics primitives. The intensity interpolation of Eq(3) applies to any curves or object edges. We should not partition a general curve into line segments and then antialias line

segments as suggested by some authors before. Instead an antialiased curve can be computed by directly scan-converting the curve, i.e., for increasing raster ordinate i, computing $f(i)$ and then interpolate the intended curve intensity I_0 between the two pixels $(i, \lfloor f(i) \rfloor)$ and $(i, \lceil f(i) \rceil)$. The main cost is to compute the real value $f(i)$, but it is required by scan-conversion anyway. So antialiasing will not be a computational burden for general curves.

Although our algorithms were presented for antialiasing curves, their extension to object boundaries is straightforward. We simply partition the object boundaries into x-dominant and y-dominant curve segments, and scan-convert them. It is easy to determine which side of such a curve segment is exterior. We use Eq(3) to interpolate the object color on two adjacent pixels at the two sides of the continuous boundary curve, and then blend the outer pixel value with the background color. Let I_0 and I_b be the intensities (colors) for object and its background, and $d < 1$ be the distance between the outer pixel and the true object boundary, then the blending formula for the boundary pixel is

$$I = dI_0 + (1 - d)I_b. \tag{19}$$

Note that unlike the blending formula by Fujimoto and Iwata [6] no division is required here. Furthermore, for antialiasing polygon edges in uniform background, we can solve Eq(19) incrementally with only integer additions and binary shifts much like our antialiased line algorithm. We will not pursue this efficiency issue any further due to the space limitation. The performance of the new technique on antialiased object edges in colorful background is shown by Fig. 8, where a filled circle with antialiasing in a complex background is compared with the one without. The antialiased filled circle appears smooth and sharp. Note that the results of Fig. 8 were obtained on an 8-bit color device, so color quantization was necessarily performed. On a 24-bit color device with more subtle shades available the antialiased disk looked even better.

Our antialiasing technique has the same subpixel addressability as Fujimoto-Iwata's method due to their equivalence.

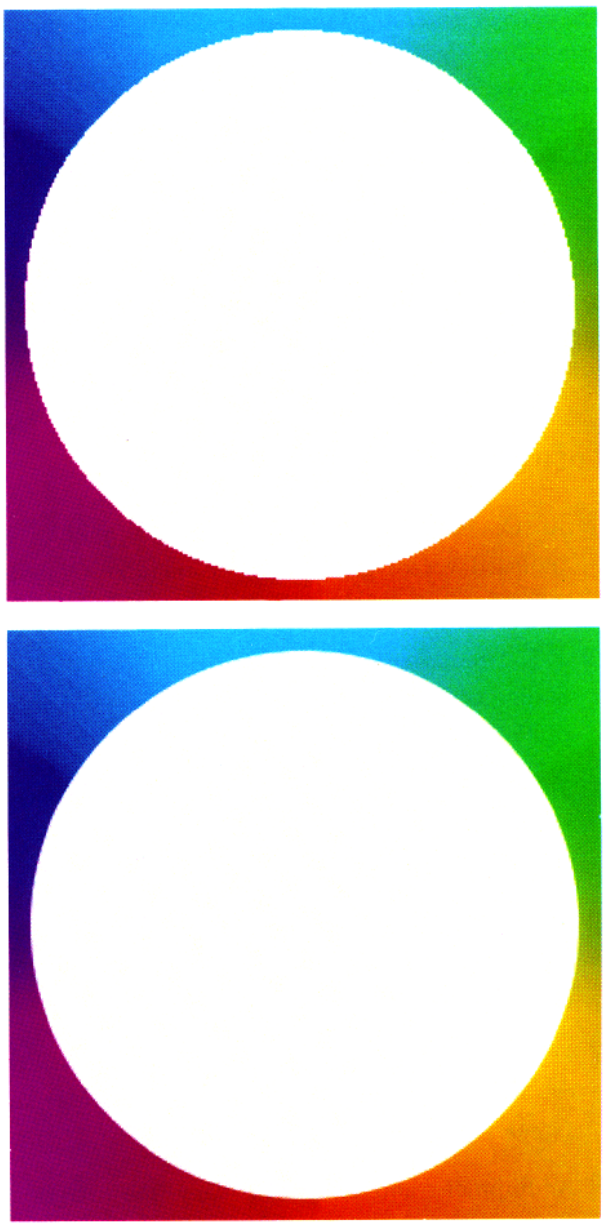

Figure 8: Filled circles embedded in colorful background without antialiasing (above) and with antialiasing (below).

8 Conclusion

Unlike all previous antialiasing research, our two-point antialiasing scheme was derived in spatial domain under a subjectively meaningful error measure to preserve

dynamic information of the original curves or object edges. The behaviour of this antialiasing scheme in frequency domain was also analyzed. It was shown that the new antialiasing technique can generate smooth line segments and circular arcs at even higher speeds than those of Bresenham's line and circle algorithms. The hardware or assembly-language realization of our new antialiasing algorithms is straightforward. These features have practical significance when antialiasing is performed on small economical graphics devices or in time-constrained applications.

Acknowledgment

The author gratefully acknowledges the financial support of the Canadian Government through NSERC grant OGP0041926 and thanks SIGGRAPH reviewers for their polishing of his original manuscript.

References

[1] A. C. Barkans, "High speed high quality antialiased vector generation," *Computer Graphics*, vol. 24, no. 4, p. 319-326, Aug. 1990.

[2] J. E. Bresenham, "Algorithm for computer control of digital plotter," *IBM Syst. J.*, vol. 4, no. 1, 1965, p. 25-30.

[3] J. E. Bresenham, "A linear algorithm for incremental digital display of circular arcs", *Comm. ACM*, vol. 20, no. 2, 1977, p. 750-752.

[4] F. Crow, "The aliasing problem in computer-generated shaded images," *Comm. ACM*, vol. 20, no. 11, Nov. 1977.

[5] D. Field, "Algorithms for drawing anti-aliased circles and ellipses," *Computer Vision, Graphics, and Image Proc.*, vol. 33, p. 1-15, 1986.

[6] A. Fujimoto and K. Iwata, "Jay-free images on raster displays," *IEEE CG&A*, vol. 3, no. 9, p. 26-34, Dec. 1983.

[7] S. Gupta and R. F. Sproull, "Filtering edges for gray-scale displays," *Computer Graphics*, vol. 15, no. 3, p. 1-5, Aug. 1981.

[8] M. Pitteway and D. Watkinson, "Bresenham's algorithm with gray scale," *Comm. ACM*, vol 23, no. 11, November 1980.

[9] X. Wu and J. Rokne, "Double-step incremental generation of lines and circles", *Computer Vision, Graphics, Image Proc.*, vol. 37, 1987, p. 331-344.

[10] X. Wu and J. Rokne, "On properties of discretized convex curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, p. 217-223, Feb. 1989.

[11] X. Wu and J. Rokne, "Double-step generation of ellipses", *IEEE CG&A*, vol. 9, no. 3. p. 56-69, May 1989.

[12] X. Wu and J. Rokne, "Dynamic error measure for curve scan-conversion," *Proc. Graphics/Interface'89*, London, Ontario, p. 183-190, June 1989.

[13] J. Rokne, B. Wyvill and X. Wu, "Fast line scan-conversion," *ACM Trans. on Graphics*, vol. 9, no. 4, p. 377-388, Oct. 1990.

[14] X. Wu, "A frame buffer architecture for parallel vector generation," *Proc. Graphics/Interface'91*, Calgary, June 1991.