Volume xx (200y), Number z, pp. 1–10

Anti-aliasing with Stratified B-spline Filters

Manuel N. Gamito† and Steve C. Maddock

Department of Computer Science, The University of Sheffield
M.Gamito@dcs.shef.ac.uk, S.Maddock@dcs.shef.ac.uk

Abstract

A simple and elegant method is presented to perform anti-aliasing in ray traced images. The method uses stratified sampling to reduce the occurrence of artifacts in an image and features a B-spline filter to compute the final luminous intensity at each pixel. The method is scalable through the specification of the filter order. A B-spline filter of order one amounts to a simple anti-aliasing scheme with box filtering. Increasing the order of the B-spline generates progressively smoother filters. Computation of the filter values is done in a recursive way, as part of a sequence of Newton-Raphson iterations, to obtain the optimal sample positions in screen space. The proposed method can perform both anti-aliasing in space and in time, the later being more commonly known as motion blur. We show an application of the method to the ray casting of implicit procedural surfaces.

Keywords: Anti-aliasing, B-spline filter, ray tracing, stratified sampling

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Picture/Image Generation]: Antialiasing I.3.7 [Three-Dimensional Graphics and Realism]: Raytracing

1. Introduction

Anti-aliasing is an important component in the rendering of ray traced images and is used to eliminate high frequencies that would otherwise show up as objectionable image artifacts. The most common approach to anti-aliasing relies on the concept of *super-sampling*, where each image pixel is subdivided into many smaller sub-pixels. The final pixel colour is averaged down from the computed luminous intensities of all sub-pixels [Gla95].

Super-sampling, by itself, pushes coherent artifacts into higher frequencies and makes them less noticeable but cannot eliminate aliasing completely. An improvement was introduced by Cook with the concept of *stratified sampling* [Coo89]. In stratified sampling, the position of each sample is slightly jittered from its original position on the node of a regular sampling grid. The placement of samples in stratified sampling mimics the placement of the photoreceptors on the human retina, which are known to obey a Poisson-disc distribution [Yel83]. This distribution of photoreceptors trades coherent aliasing artifacts for noise, most of which is later masked out by the perception mechanisms in the brain. Stratified sampling becomes a very effective anti-aliasing technique by taking advantage of this characteristic in the human visual system. Although algorithms for generating Poisson-disc distributions exist, these are expensive to compute [Mit87]. The strategy of jittering samples from their positions on a regular grid provides a reasonable and much cheaper approximation. The concept of stratified sampling is general enough that Cook was able to use it not only to perform anti-aliasing and motion blur but also to simulate depth of field.

Anti-aliasing techniques can be further enhanced with the use of low-pass filters. Instead of performing a simple arithmetic averaging to compute the pixel colour from the colour of all neighbouring samples, a weighted average is taken. The weight for each sample is given by the filters and usually depends on the distance in screen space between the sample and the pixel position.

This article will present a method for performing stratified anti-aliasing on ray traced images with a low-pass filter

† Supported by grant SFRH/BD/16249/2004 from Fundação para a Ciência e a Tecnologia, Portugal.

chosen from a family of B-spline basis functions. A similar approach has been presented by Stark *et al.* [SSA05]. Stark *et al.* present the solution for B-spline filters of up to order four (cubic B-splines) whereas our method can work with any filter order and is also simpler to implement. We also consider anti-aliasing in time for ray traced animations. By choosing the order of the B-spline filter, one can have different filter behaviours for anti-aliasing, starting with the box filter and proceeding through the tent filter, the cubic filter, and beyond.

Section 2 presents a more detailed formulation of the anti-aliasing problem. Section 3 introduces some properties of B-spline basis functions that are necessary for this work. Section 4 presents our anti-aliasing method. Section 5 explains how anti-aliasing can be extended into the time dimension. Section 6 presents some results and Section 7 gives conclusions. Two animations illustrating our anti-aliasing method in space and in time are available online at `http://www.dcs.shef.ac.uk/~mag/bspline.html`.

2. Stratified Monte Carlo Anti-aliasing

We seek to compute the luminous intensity $I(\mathbf{x})$ for each point $\mathbf{x}$ in screen space in such a way that high frequencies are removed and do not cause aliasing when $I(\mathbf{x})$ is regularly sampled onto the discrete pixel positions. Removal of high frequencies from a signal is possible by convolving it with some appropriate low-pass filter $h(\mathbf{x})$. The anti-aliased intensity $I'(\mathbf{x})$ becomes:

$$I'(\mathbf{x}) = \int I(\mathbf{u})h(\mathbf{x}-\mathbf{u})\,d\mathbf{u} \tag{1}$$

Anti-aliasing algorithms for computer graphics always try to provide some numerical approximation to this integral that is both accurate and not too expensive to compute. With no assumptions being made on the shape of the filter, the simplest approximation to (1) replaces the integral by a summation over several image positions:

$$I'(\mathbf{x}) \approx \sum_i I(\mathbf{u}_i)h(\mathbf{x}-\mathbf{u}_i) \tag{2}$$

The samples $\mathbf{u}_i$ are regularly placed around point $\mathbf{x}$, with N samples along each of the horizontal and vertical directions, for a total of N^2 intensity computations necessary to obtain a single pixel intensity. This amounts to performing a weighted average of all the computed luminous intensities, where the weights are obtained from the filter kernel.

This approach is not particularly efficient because it does not take into account the influence of the filter when placing the samples. The samples $\mathbf{u}_i$ are regularly spaced around $\mathbf{x}$, regardless of which values the filter will take there. It can happen, and usually does, that many samples will be placed in regions farthest from $\mathbf{x}$ where the values of $h(\mathbf{x}-\mathbf{u}_i)$ will be small. These farthest samples will have a negligible impact on the anti-aliased intensity $I'(\mathbf{x})$, due to the small value of their weights during averaging. This can be seen as a major source of inefficiency when one remembers that computation of the luminous intensities $I(\mathbf{u}_i)$, with ray tracing or some other rendering algorithm, is always an expensive procedure.

A better approach than (2) is to numerically compute the anti-aliasing convolution (1) with Monte Carlo integration [SM03]. In Monte Carlo integration, a simple arithmetic average is used to compute the filtered intensity value:

$$I'(\mathbf{x}) \approx \frac{1}{N^2}\sum_i I(\mathbf{u}_i) \tag{3}$$

With Monte Carlo the positions $\mathbf{u}_i$ are taken to be the outcome of a random variable, whose probability density function (PDF) is the low-pass filter itself. There is now a very low probability that samples will be placed in regions where $h(\mathbf{x}-\mathbf{u}_i)$ is small. By treating the filter as the PDF of some random process, we are assured that samples will be placed in regions where they are more likely to contribute to the anti-aliased intensity. We must now deal with the issue of how to randomly place samples in screen space, according to some arbitrary probability density $h(\mathbf{x})$.

First we consider $h(x)$ itself. It must obey certain constraints if it is to be a valid PDF. The most obvious constraint is that $h(\mathbf{x})$ must be strictly positive, since it does not make sense to talk about negative probabilities. Secondly, the integral $\int_\infty h(\mathbf{x})d\mathbf{x}$ must be unity. This means that there is a probability of one (a certainty) that a sample will be located somewhere in screen space. If this later constraint does not hold but the filter still has a finite positive integral, it is possible to obtain a normalized PDF by using $h(\mathbf{x})/\int_\infty h(\mathbf{x})d\mathbf{x}$ as the filter kernel.

With a valid PDF, we can now obtain another important measure for a random process: the cumulative density function (CDF). If $h(\mathbf{x})$ is a PDF, then its CDF is given by:

$$H(\mathbf{x}) = \int_{-\infty}^{\mathbf{x}} h(\mathbf{t})\,d\mathbf{t} \tag{4}$$

The CDF $H(\mathbf{x})$ gives us the probability that the random variable will take values below $\mathbf{x}$. In the case of two-dimensional random variables, as is the case in this work, it gives us the probability that a screen sample will be inside the rectangle that has a lower left vertex at minus infinity, in both horizontal and vertical coordinates, and an upper right vertex at point $\mathbf{x}$. If $h(\mathbf{x})$ is a proper PDF, then we have again $H(+\infty) = 1$, meaning the sample is bound to be somewhere on the screen.

To numerically compute a sample from a random process, knowing its CDF, we use the method of *function inversion* [PTVF92]. We begin by generating a uniform random variable $\mathbf{y}$ that takes values in the unit rectangle $[0,1[\times[0,1[$. The sample having our desired CDF, $H(\mathbf{x})$, and PDF, $h(\mathbf{x})$, is now obtained by finding the solution to the equation:

$$\mathbf{y} = H(\mathbf{x}) \tag{5}$$

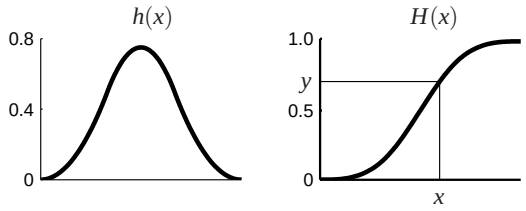

Figure 1: *A probability density function (left) and its corresponding cumulative density function (right). To obtain a sample x with this PDF, sample uniformly with y and obtain x by inverting the CDF.*

Figure 1 exemplifies this process for a typical low-pass filter in one dimension. It is possible to see that $H(\mathbf{x})$ is a monotonically increasing function from 0 to 1. This is true of any CDF and it is a property that will be used to advantage in Section 4. To distribute sample points around the point $\mathbf{x}$ for Monte Carlo integration we create a regular grid of samples, not in screen space, but in the unit square space of the $\mathbf{y}$ variable. These samples become stratified by the addition of a random component ξ that breaks up the regularity of the grid:

$$\mathbf{v}_i = \Delta\mathbf{v}\, i + \xi \tag{6}$$

If we use N samples, along both the horizontal and vertical coordinates, then the vector $\Delta\mathbf{v}$ has equal components $1/N$ and the random vector ξ has components taking uniform random values in the interval $[0, 1/N[$. We thus obtain a stratified distribution of samples $\mathbf{u}_i$ to use in the Monte Carlo approximation (3) to the anti-aliasing integral. The samples obey a PDF equal to the filter $h(\mathbf{x})$ by having them be the solution of $\mathbf{v}_i = H(\mathbf{u}_i)$ for all i.

3. A Family of Uniform B-splines

A family of B-spline bases $n_m(x)$, with knots placed at integer positions, can be obtained by performing consecutive convolutions with the characteristic function on the interval $[0, 1[$. We start off with the B-spline of order one, which is the aforementioned characteristic function itself:

$$n_1(x) = \begin{cases} 1 & \text{if } x \in [0,1[, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The convolution of a B-spline of order $m-1$ with the characteristic function gives the next B-spline in the sequence:

$$\begin{aligned} n_m(x) &= (n_{m-1} * n_1)(x) \\ &= \int_0^1 n_{m-1}(x-t)\, dt, \quad \text{for } m > 1. \end{aligned} \tag{8}$$

Because of the consecutive convolutions, the B-spline curves become progressively smoother as m increases.

From [Chu92], we can state the properties of the family of B-spline functions that make them particularly attractive for use as low-pass filters in anti-aliasing:

$$n_m(x) > 0, \quad \text{for } 0 < x < m. \tag{9a}$$

$$\int_{-\infty}^{+\infty} n_m(x)\, dx = 1, \quad \text{for all } m. \tag{9b}$$

$$\operatorname{supp} n_m = [0, m]. \tag{9c}$$

$$n_m(x) = \frac{x}{m-1} n_{m-1}(x) + \frac{m-x}{m-1} n_{m-1}(x-1), \quad \text{for } m > 1. \tag{9d}$$

Properties (9a) and (9b) together tell us that any $n_m(x)$ is a valid probability density function for Monte Carlo anti-aliasing. The support of a B-spline $n_m(x)$, when defined according to (8), is the interval $[0, m]$. For anti-aliasing, however, we require that random samples be centered around some desired pixel position. This can be achieved for B-spline filters by generating the samples, as explained in the previous section, and performing a simple offset of $-m/2$ along the horizontal and vertical coordinates. Property (9d) is the most important. It gives us an algebraic relation between the B-spline of order m and the B-spline of order $m-1$. With this knowledge and with knowledge of the shape of $n_1(x)$ (7) we can compute $n_m(x)$ recursively, at any point x and for any order m.

We now know that B-splines can be used as filters for anti-aliasing and that a simple recursive procedure exists to evaluate them. To study the behaviour of B-splines as low-pass filters, we must also study their spectra $\hat{n}_m(f)$, as given by the application of the Fourier transform to $n_m(x)$:

$$\hat{n}_m(f) = \mathcal{F}\{n_m(x)\} = \left(\frac{\sin \pi f}{\pi f}\right)^m e^{-i\pi f m/2} \tag{10}$$

If we admit a unit distance between pixels on the screen, we then have a sampling frequency of 1 Hz along the horizontal and vertical directions. The Nyquist Sampling Theorem tells us that we must filter out all frequencies above 0.5 Hz if no aliasing is to occur. Ideally, we would like to have a perfect low-pass filter with a sharp frequency cut-off at 0.5 Hz. Such an ideal filter, however, would have an infinite support and would be intractable under any of the approximations to the anti-aliasing integral. The family of B-spline basis functions provides a sequence of approximations to this ideal low-pass filter.

Figure 2 shows the shape of four B-splines with orders of 1, 2, 4 and 20, on the left, together with the modulus of their respective spectra, on the right. These spectra are symmetric about the origin, since they are the transform of real functions, and only the positive frequencies are shown. The spectrum of the ideal low-pass filter for anti-aliasing has been superimposed as a dashed rectangle. The basis $n_1(x)$ gives the well-known box filter for anti-aliasing. It is quite simple to implement but it allows too many high frequencies to pass through, as evidenced by the significant lobes that fall outside of the spectrum for the perfect filter. The basis $n_2(x)$ is

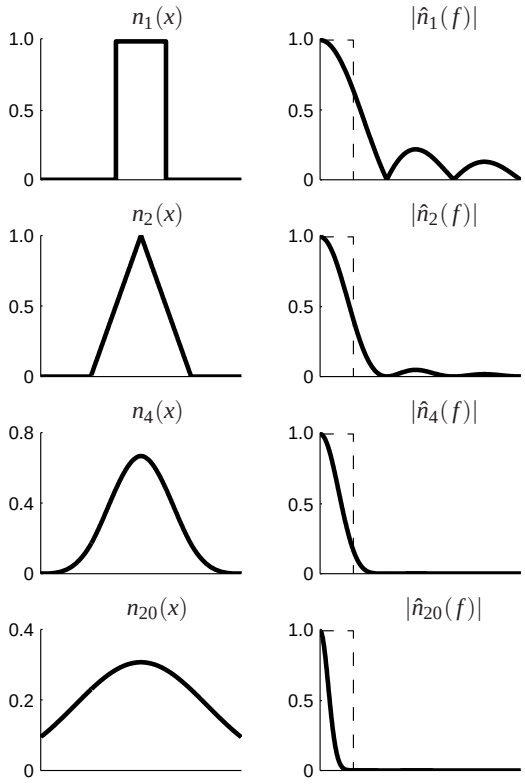

Figure 2: *B-spline basis functions (on the left), for m equal to* 1, 2, 4 *and* 20, *and their respective spectra (on the right). The spectrum for the ideal anti-aliasing filter is superimposed as a dashed rectangle on the later.*

also known as the tent or triangle filter because of its shape. It has a better spectral behaviour than the box filter, since the lateral spectral lobes are now more attenuated. However, the trend of increasing m in order to block more of the high frequencies cannot continue indefinitely. For too large a value of m, the low frequencies also become excessively attenuated. This is exemplified by the B-spline filter of order 20 in Figure 2. Visually, this distortion in the low frequencies translates into a blurry image, where the amount of blurring is far greater than necessary to eliminate aliasing. It is generally considered that a good compromise between blocking the high frequencies and not distorting the low frequencies is achieved with the cubic B-spline filter $n_4(x)$, also shown in Figure 2.

Once a particular order for the B-spline is chosen, a two-dimensional anti-aliasing filter is made from the cartesian product of the $n_m(x)$ kernel with itself:

$$h(\mathbf{x}) = h(x_1, x_2) = n_m(x_1 - m/2)\, n_m(x_2 - m/2) \quad (11)$$

As previously explained, the offset by $-m/2$ properly centres the kernels around the origin. It would be equally as simple to have different orders for the horizontal and vertical kernels but there does not seem to be, however, any significant advantage in doing so.

4. Stratified Anti-Aliasing with B-splines

Stratified Monte Carlo anti-aliasing with B-spline low-pass filters requires random screen samples to be computed with a PDF given by $n_m(x)$. This, in turn, requires the following CDF to be known:

$$N_m(x) = \int_0^x n_m(t)\, dt \quad (12)$$

Based on [Chu92] we can derive four properties about the integral $N_m(x)$ of a B-spline basis function that are important when writing an implementation of anti-aliasing:

$$\operatorname{supp} N_m(x) = [0, +\infty[\quad (13a)$$

$$N_m(x) \text{ increases from } N_m(0) = 0 \text{ to } N_m(+\infty) = 1. \quad (13b)$$

$$n_m(x) = N_{m-1}(x) - N_{m-1}(x-1), \quad \text{for } m > 1. \quad (13c)$$

$$N_m(x) = \frac{x}{m} N_{m-1}(x) + \left(1 - \frac{x}{m}\right) N_{m-1}(x-1), \quad \text{for } m > 1. \quad (13d)$$

Properties (13a) and (13b) are a direct consequence of (12), together with (9a), (9b) and (9c). These two properties tell us that $N_m(x)$ is a valid CDF. Properties (13c) and (13d) are derived in the Appendix. Property (13c) gives an algebraic relationship between the spline function and its integral at the next lower order. Property (13d) presents a numerical recipe for computing any value of $N_m(x)$ in a recursive way. At the end of the recursion lies the $N_1(x)$ function, whose shape has a trival expression, as given by (14).

$$N_1(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } x \in [0, 1[, \\ 1 & \text{if } x \geq 1. \end{cases} \quad (14)$$

The generation of random samples for Monte Carlo integration, with a B-spline acting as the PDF, now requires solving the following equation for a sample x in either horizontal or vertical screen coordinates, where y is a stratified uniform random variable in the $[0, 1[$ interval:

$$y = N_m(x) \quad (15)$$

The solution is immediate when $m = 1$ but, unfortunately, becomes rather involved for higher orders. Rather than try to solve (15) analytically, the best approach is to use a numerical iterative method like Newton-Raphson to find the zero of the auxiliary function $f(x) = N_m(x) - y$ (see [PTVF92] for such a method).

Newton-Raphson is a powerful root finder since it has quadratic convergence, but some care must usually be taken before its application. The root must first be bracketed inside a suitable interval. Then, $f(x)$ must be monotonic inside that interval with a non-vanishing derivative everywhere. All these conditions are naturally met in the case of a B-spline CDF. Clearly, there is one and only one root x inside the

interval between $y = 0$ and $y = 1$. The function is monotonically increasing inside that interval, as assured by property (13b), and the only points where the derivative vanishes are the two interval extremes. If we first clear the boundary case $y = 0 \Rightarrow x = 0$ (because $y \in [0,1[$, the boundary $y = 1$ need not be considered) then a series of Newton-Raphson iterations can be started, with full confidence that it will converge to the solution:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{N_m(x_i) - y}{N'_m(x_i)} = x_i - \frac{N_m(x_i) - y}{n_m(x_i)} \quad (16)$$

The cost of performing (16) is $2^m - 1$ recursive calls of $N_m(x_i)$ and $2^m - 1$ recursive calls of $n_m(x_i)$, giving a total complexity of $O(2^{m+1})$ per Newton-Raphson iteration. It is possible to do better by replacing (13c) and (13d) in (16):

$$x_{i+1} = x_i - \frac{\frac{x_i}{m} N_{m-1}(x_i) + \left(1 - \frac{x_i}{m}\right) N_{m-1}(x_i - 1) - y}{N_{m-1}(x_i) - N_{m-1}(x_i - 1)}, \quad \text{for } m > 1. \quad (17)$$

The cost is now $2^{m-1} - 1$ recursive calls to compute $N_{m-1}(x_i)$ and similarly for $N_{m-1}(x_i - 1)$. The complexity is $O(2^m)$, half of what it was before.

The Newton-Raphson iterations are started off with $x_0 = m/2$, which is at the centre of the interval (9c) where the random variable x is bound to lie.

Figure 3 shows the sampling patterns used by several anti-aliasing techniques. The boundaries between the pixels are marked with horizontal and vertical lines. The diamond shape shows the centre of the pixel for which the sampling patterns apply. In all cases, 10^2 samples per pixel was used. The pattern on the top left corresponds to super-sampling where the pixel is uniformly divided into sub-pixels. Super-sampling implies the use of a box filter which covers the extent of the pixel. The pattern on the top right uses a cubic B-spline filter whose area of support is the square $[-2,2] \times [-2,2]$. For this pattern the weighted average formula (2) is used to compute the anti-aliased pixel intensity. The pattern on the lower left applies stratification to the samples but continues to use (2). The stratification converts any aliasing artifacts that might still remain into noise. Finally, the pattern on the lower right uses Monte Carlo anti-aliasing, expressed by formula (3), with samples that were pre-stratified in the $[0,1] \times [0,1]$ domain and later converted to the screen-space domain through function inversion. The samples are now preferentially located in areas where the filter has higher importance, which is to say, closer to the centre of the pixel.

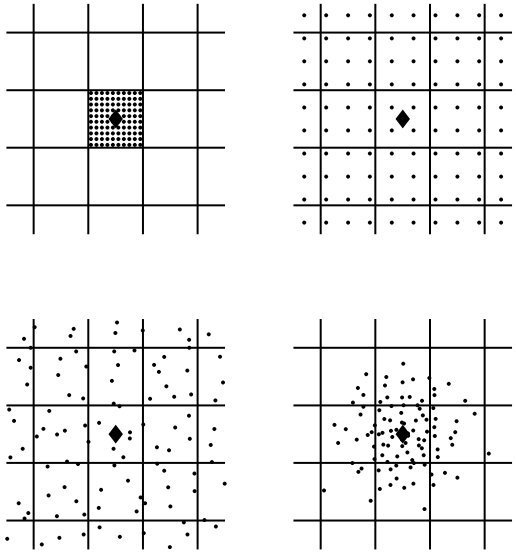

Figure 3: *From left to right, top to bottom: super-sampling, uniform anti-aliasing, stratified anti-aliasing and stratified Monte Carlo anti-aliasing. All sampling patterns use* 10^2 *samples per pixel. Except for the image on the top left, a cubic B-spline filter was used.*

5. Motion Blur

Our anti-aliasing method can be extended to the extra dimension of time. Anti-aliasing in the time domain is commonly referred to as *motion blur*. To obtain a combined space and time anti-aliasing for a screen intensity $I(\mathbf{x}, t)$ we perform the three-dimensional Monte Carlo integration:

$$I'(\mathbf{x}, t) \approx \frac{1}{N^2} \sum_i I(\mathbf{u}_i, v_i) \quad (18)$$

The random spatial samples $\mathbf{u}_i$ obey the PDF (11), as before. The random time sample v_i obeys a PDF given by a B-spline filter $n_l(t - l/2)$ in time. The order l of the time filter need not be the same as the order m of the spatial filter. In (18), a total of N^2 samples is still being used instead of the N^3 samples that might seem more obvious for a three-dimensional Monte Carlo integration. A table of N^2 random permutations is created before starting the anti-aliasing algorithm. Then, for each evaluation of (18), N random time samples are generated and accessed through this permutation table, based on the index of the $\mathbf{u}_i$ samples. This technique, which was originally presented in [Coo89], keeps the cost of (18) constant at N^2 evaluations of $I(\mathbf{x}, t)$ without introducing significant correlation between the spatial and the time domains.

The common example of motion blur generated by a camera with a finite exposure time can be easily modelled with our anti-aliasing method. Each frame in a movie runs from time t_i to time $t_i + \delta t$, where δt is the inverse of the frame rate. The camera shutter is only open during a smaller inter-

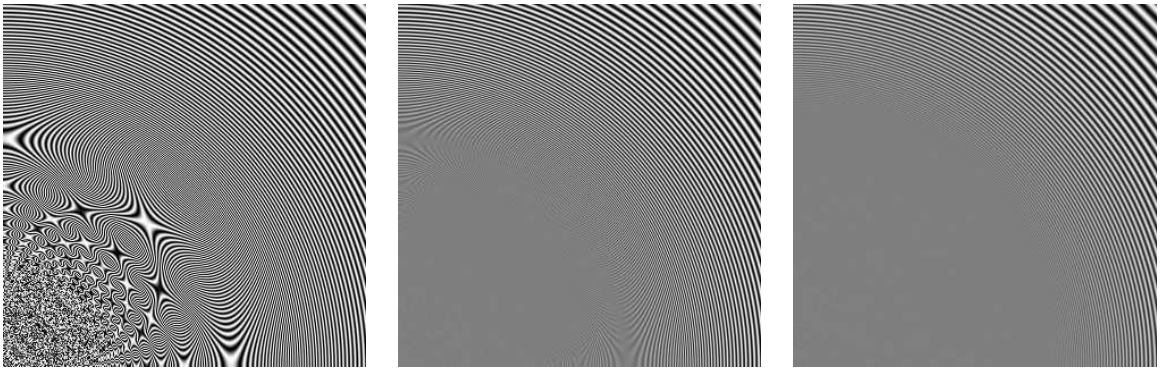

Figure 4: *Anti-aliasing test pattern. From left to right: no anti-aliasing, anti-aliasing with a box filter and anti-aliasing with a cubic B-spline filter.*

val of time, from t_i to $t_i + s$, where $s < \delta t$ is called the *shutter speed* (even though it is really a time quantity). During the final portion of the frame, the shutter must be closed while the camera's motor advances the film into the next position. The shutter, therefore, behaves as a box filter with a shape given by $\frac{1}{s} n_1(\frac{t}{s})$. The $1/s$ factor outside n_1 gives a constant unit area to the filter and ensures it is a valid PDF for any value of s.

If B-splines with $l > 1$ are used a smoother motion will be obtained even though it will not be the same as what a real camera would record. A stronger objection to using $n_l(t - l/2)$ with $l > 1$ is that such a filter, being symmetrical about the origin, would weight equally past and future values of $I(\mathbf{x}, t)$ relative to some instant t_i. No physical imaging device could possibly have such a behaviour. It would be relatively straightforward to have $n_l(t)$ obey a principle of causality (which basically states that any value of $n_l(t)$ for $t < 0$ must be zero) by manipulating the position of the spline knots and thus generating a new family of non-uniform B-splines. We have, however, decided not to pursue this since there is still much speculation about how the human retina processes time-varying information [AA93, VK03]. Without more research from the psychophysical sciences it is difficult to decide what time filter best matches the behaviour of our own visual system.

6. Results

A good test of the anti-aliasing properties of our method is shown in Figure 4. The images in this Figure where generated by directly sampling the function:

$$I(x,y) = \frac{1}{2}\left(\sin\left(\frac{1000}{x^2+y^2}\right)+1\right), \qquad (19)$$

with a resolution of 320×320. The origin $(x,y) = (0,0)$ is at the lower left corner of each image. This function was chosen to highlight the problems caused by aliasing and to show how B-spline filtering can solve them. It has a frequency content that increases without bounds for pixels progressively closer to the origin. The left image in Figure 4 does not use anti-aliasing and takes only one sample at the centre of each pixel. In this image, the outer rings are correctly rendered, the middle rings show Moiré patterns caused by aliasing and the inner rings have become totally masked by severe white noise. The image in the middle of Figure 4 uses a box filter and shows how the noise of the inner rings has been replaced by a constant grey value. The middle rings still show some Moiré patterns. The image on the right of Figure 4 uses a cubic B-spline filter. The uniform gray area is larger than in the previous image and no Moiré patterns remain.

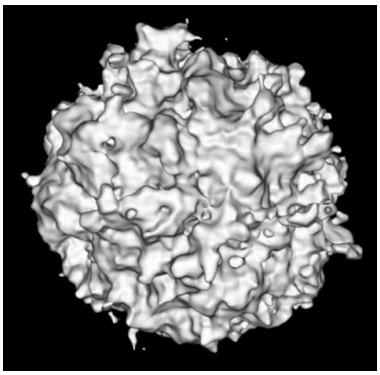

Figure 5: *A procedural implicit surface rendered by ray casting with our anti-aliasing method.*

Figure 5 shows the results of rendering an implicit surface by ray casting with our anti-aliasing method [GM05]. An animation of this surface is available from Figure 6 shows a small area of the implicit surface and illustrates the effect of changing the number of samples per pixel for a small section of the implicit surface from Figure 5. From left to right the sequence shows no anti-aliasing, and anti-aliasing with a cu-

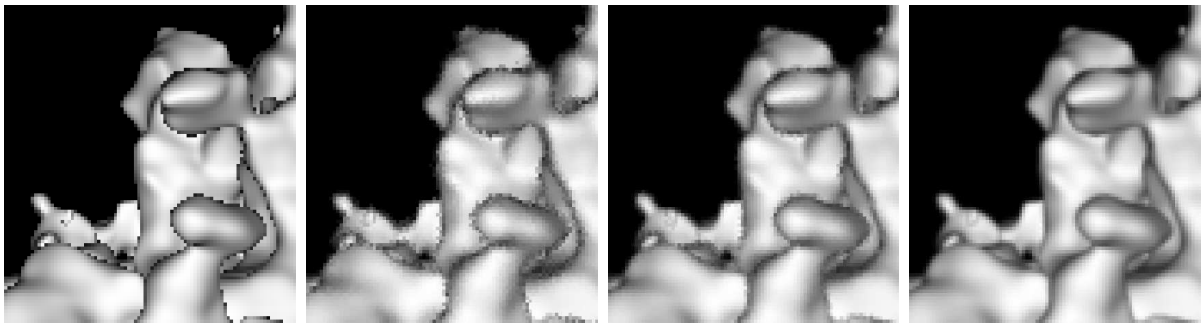

Figure 6: *From left to right: no anti-aliasing, anti-aliasing with a cubic B-spline filter with N equal to 2, 3 and 10 samples along each coordinate direction.*

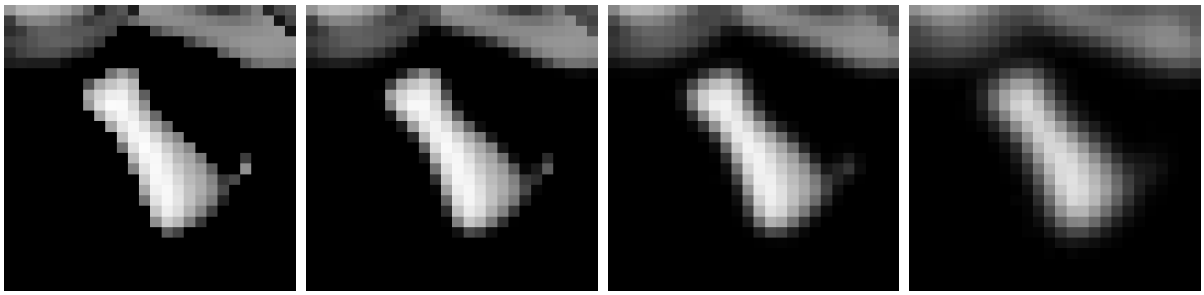

Figure 7: *From left to right: no anti-aliasing, anti-aliasing with a B-spline filter of order 1, 4, and 15. $N = 10$ samples were used along each coordinate direction for the anti-aliased images.*

bic B-spline filter with N^2 equal to 4, 9 and 100 samples per pixel. Performing anti-aliasing by taking only four random samples (two samples along each coordinate direction) is not very effective but it does show that coherent aliasing artifacts are converted into noise by the stratification of sample points. When N^2 increases to 9 and then to 100 results become progressively better.

Figure 7 shows another small area of the surface. This time, the number of samples per pixel was kept constant and the order of the B-spline filter was increased. From left to right, the sequence shows no anti-aliasing and anti-aliasing with a B-spline filter of order m equal to 1, 4 and 15. In the cases where anti-aliasing was used, $N^2 = 100$ samples per pixel were computed. It can be seen that as the filter order is increased the image becomes progressively blurred. This is because, for higher orders, the filter has a larger area of support and therefore spreads the influence of the bright pixels farther into the dark areas. The optimal order is $m = 4$, as explained in Section 3

The number of Newton-Raphson iterations, whilst rendering the images of Figures 6 and 7, was from 1 to a maximum of 8, depending on the value of y when inverting (15). If $y = 0.5$ then $x = x_0 = m/2$ and the algorithm converges after exactly one iteration. If, on the other hand, y moves away towards one of the extremities of the $[0,1[$ interval, a larger number of iterations will be required for x_i to converge to the solution. Nevertheless, the number of iterations always remains small due to the quadratic convergence properties of Newton-Raphson root finding.

Figure 8 shows, on the left, two frames from an animation of a rotating wheel at 25 frames per second ($\delta t = 0.04$) as seen by a camera with a shutter speed $s = 0.033$. This illustrates the famous *wagon wheel illusion* [PPA96]. In the lower left image, the spokes are turning counter-clockwise at high speed but the blurred features in the image are slowly turning clockwise. If we replace the box filter, which models the behaviour of the camera shutter, by a cubic B-spline filter, the images on the right of Figure 8 result. In the lower right image, the spokes are no longer discernible and a uniform circular blur results, showing no indication of wheel rotation. There is some controversy as to whether or not this is what the human eye would see under constant lighting conditions since it has been reported that the wagon wheel illusion is sometimes discernible under these circumstances [VRK05].

7. Conclusions and Future Work

We have presented an approach to anti-alias ray traced images using B-spline basis functions that improves upon the approach by Stark et al. [SSA05]. Our method is simpler to implement, can generate B-spline filters of any order and

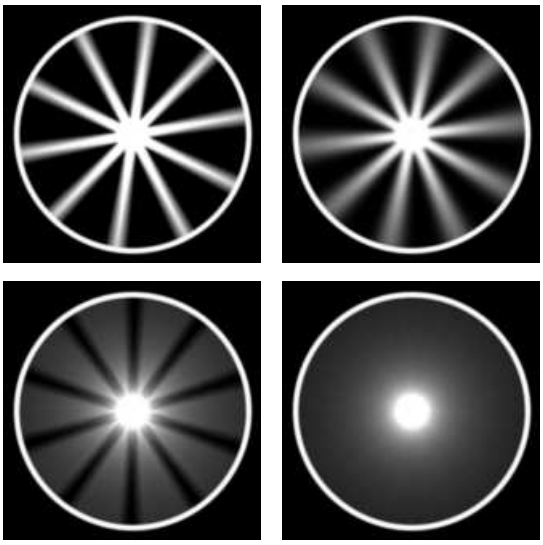

Figure 8: *A rotating wheel as seen by a camera (on the left) and filtered with a cubic B-spline (on the right). The images at the top show the wheel rotating slowly and the images on the bottom show the wheel rotating at high speed. The complete animation is available at* `http://www.dcs.shef.ac.uk/~mag/bspline.html`.

can perform anti-aliasing simultaneously in both the space and the time domains.

Our approach can generate any B-spline filter through a simple and elegant recursive procedure. This recursive procedure means we do not have to deal with the actual piecewise polynomials that describe the shape of the B-splines. Explicitly working with the polynomial representation for some B-spline $n_m(x)$ can become quite an involved procedure, even for moderate values of m. The cost of recursively evaluating $n_m(x)$ grows geometrically with m. This is not, however, a serious constraint for our method since filters of too high an order introduce excessive blurring and should be avoided. A filter of order $m = 4$ (a cubic B-spline filter) provides the best compromise between blocking the undesirable high frequencies and not distorting the low frequencies, thereby introducing minimal blurring.

Generation of random samples for Monte Carlo anti-aliasing, having $n_m(x)$ as their probability density function, requires inversion of $y = N_m(x)$ where $N_m(x)$ is the cumulative density function associated with the B-spline filter. This can be accomplished numerically with Newton-Raphson root finding in a way that is always guaranteed to converge. Experiments have shown that the number of required iterations is always less or equal to eight.

Our method was developed to perform anti-aliasing in space but it extends trivially to anti-aliasing in time. The motion blur behaviour of a camera shutter was demonstrated with a B-spline of order $m = 1$. B-spline filters of higher order are non-causal. This is not a problem for pre-scripted animations, whose behaviour in time is known a priori, but it does raise the objection of physical implausibility since a filtered intensity value will depend equally on samples from the past and from the future.

Performing anti-aliasing on computer generated images is an expensive proposition. By using N^2 samples per pixel we are doing the same amount of work as though we were rendering a virtual image with a resolution N^2 higher than the actual image. The present method does not take into account local image information when performing anti-aliasing. In the example presented in the previous section, it is quite wasteful to be casting N^2 rays for a pixel located on the inside of the implicit surface and far from any edge. A single ray would have sufficed, as the non-antialiased image on the left of Figure 6 shows. In this image, aliasing is only evident at the edges, while the internal surface features remain smooth. Techniques for turning the present anti-aliasing method with B-splines into an adaptive procedure, similar to the work by [PS89], should be further investigated.

Acknowledgments

The authors would like to express their thanks to Michael Stark and Peter Shirley for making a pre-print copy of their paper available. The authors would also like to thank Pedro Faria Lopes for his insight into how a camera works in the real world.

APPENDIX

We begin by deriving the relation (13c) between a B-spline of order m and the integral of B-splines of order $m-1$. To do so, we need to express $n_m(x)$ as an mth-order finite difference of powers of x. The reader is invited to consult [Chu92] for more details. Take the notation $x_+ = \max(x, 0)$ to represent the restriction of x to positive values only. Similarly, $x_+^m = (x_+)^m$. Consider also the finite differences of order m for some function $f(x)$. These are defined with the following recurrence relation:

$$(\Delta f)(x) = f(x) - f(x-1) \tag{A.1}$$

$$(\Delta^m f)(x) = \left(\Delta^{m-1}(\Delta f)\right)(x), \quad \text{for } m > 1. \tag{A.2}$$

Armed with this notation, we can write a B-spline $n_m(x)$ as:

$$n_m(x) = \frac{1}{(m-1)!}\Delta^m x_+^{m-1}, \quad \text{for } m > 1. \tag{A.3}$$

Taking the integral of (A.3), we arrive at a similar equation for $N_m(x)$:

$$\begin{aligned}
N_m(x) &= \int_0^x n_m(t)\,dt \;=\; \frac{1}{(m-1)!}\int_0^x \Delta^m t_+^{m-1}\,dt \\
&= \frac{1}{(m-1)!}\int_0^x \Delta^{m-1}\left\{t_+^{m-1} - (t-1)_+^{m-1}\right\} dt \\
&= \frac{1}{(m-1)!}\Delta^{m-1}\left\{\int_0^x t_+^{m-1}\,dt - \int_0^x (t-1)_+^{m-1}\,dt\right\} \\
&= \frac{1}{(m-1)!}\Delta^{m-1}\left\{\int_0^x t_+^{m-1}\,dt - \int_{-1}^{x-1} t_+^{m-1}\,dt\right\} \\
&= \frac{1}{(m-1)!}\Delta^{m-1}\left\{\int_0^x t_+^{m-1}\,dt - \int_0^{x-1} t_+^{m-1}\,dt\right\} \\
&= \frac{1}{(m-1)!}\Delta^{m}\int_0^x t_+^{m-1}\,dt \;=\; \frac{1}{m!}\Delta^m x_+^m, \text{ for } m>1.
\end{aligned} \tag{A.4}$$

Using (A.4) twice, we have:

$$\begin{aligned}
&N_{m-1}(x) - N_{m-1}(x-1) \;= \\
&= \frac{1}{(m-1)!}\Delta^{m-1}x_+^{m-1} - \frac{1}{(m-1)!}\Delta^{m-1}(x-1)_+^{m-1} \\
&= \frac{1}{(m-1)!}\Delta^{m-1}\left\{x_+^{m-1} - (x-1)_+^{m-1}\right\} \\
&= \frac{1}{(m-1)!}\Delta^{m}x_+^{m-1} \;=\; n_m(x), \quad \text{for } m>1.
\end{aligned} \tag{A.5}$$

This completes the derivation of (13c).

We now take the recurrence relation that exists for $n_m(x)$ and arrive at a similar relation for $N_m(x)$, expressed in (13d). We place an integral on both sides of (9d) and perform integration by parts:

$$\begin{aligned}
N_m(x) &= \int_0^x n_m(t)\,dt \\
&= \int_0^x \frac{t}{m-1}n_{m-1}(t)\,dt \\
&\quad + \int_0^x \frac{m-t}{m-1}n_{m-1}(t-1)\,dt \\
&= \frac{t}{m-1}N_{m-1}(t)\Big|_0^x + \frac{m-t}{m-1}N_{m-1}(t-1)\Big|_0^x \\
&\quad - \frac{1}{m-1}\int_0^x N_{m-1}(t)\,dt \\
&\quad + \frac{1}{m-1}\int_0^x N_{m-1}(t-1)\,dt \\
&= \frac{x}{m-1}N_{m-1}(x) + \frac{m-x}{m-1}N_{m-1}(x-1) \\
&\quad - \frac{1}{m-1}\int_0^x \{N_{m-1}(t) - N_{m-1}(t-1)\}\,dt,
\end{aligned}$$

for $m>1$. (A.6)

Replacing (13c) in (A.6), we get:

$$\begin{aligned}
N_m(x) &= \frac{x}{m-1}N_{m-1}(x) + \frac{m-x}{m-1}N_{m-1}(x-1) \\
&\quad - \frac{1}{m-1}\int_0^x n_m(t)\,dt \\
&= \frac{x}{m-1}N_{m-1}(x) + \frac{m-x}{m-1}N_{m-1}(x-1) \\
&\quad - \frac{1}{m-1}N_m(x), \quad \text{for } m>1.
\end{aligned} \tag{A.7}$$

Sending the $N_m(x)$ term on the right to the left hand side of the equation and rearranging terms, we finally arrive at:

$$N_m(x) \;=\; \frac{x}{m}N_{m-1}(x) + \left(1-\frac{x}{m}\right)N_{m-1}(x-1), \text{ for } m>1. \tag{A.8}$$

This completes the derivation of (13d).

References

[AA93] ANDERSON J., ANDERSON B.: The myth of persistence of vision revisited. *Journal of Film and Video 45*, 1 (1993), 3–12.

[Chu92] CHUI C. K.: *An Introduction to Wavelets*, vol. 1 of *Wavelet Analysis and its Applications*. Academic Press, 1992, ch. 4, pp. 81–117.

[Coo89] COOK R. L.: Stochastic sampling and distributed ray tracing. In *An Introduction to Ray Tracing*, Glassner A. S., (Ed.). Academic Press, 1989, ch. 5, pp. 161–199.

[Gla95] GLASSNER A. S.: *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., 1995.

[GM05] GAMITO M. N., MADDOCK S. C.: *Ray Casting Implicit Procedural Noises with Reduced Affine Arithmetic*. Memorandum CS – 05 – 04, Dept. of Comp. Science, The University of Sheffield, Apr. 2005.

[Mit87] MITCHELL D. P.: Generating antialiased images at low sampling densities. In *Computer Graphics (SIGGRAPH '87 Proceedings)* (July 1987), Stone M. C., (Ed.), vol. 21 of *Annual Conference Series*, ACM Press, pp. 65–72.

[PPA96] PURVES D., PAYDARFAR J. A., ANDREWS T. J.: The wagon wheel illusion in movies and reality. *Proc. Natl. Acad. Sci. 93* (April 1996), 3693–3697.

[PS89] PAINTER J., SLOAN K.: Antialiased ray tracing by adaptive progressive refinement. In *Computer Graphics (SIGGRAPH '89 Proceedings)* (July 1989), Lane J., (Ed.), vol. 23, ACM Press, pp. 281–288.

[PTVF92] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, 1992.

[SM03] SHIRLEY P., MORLEY R. K.: *Realistic Ray Tracing*, second ed. A K Peters Ltd., 2003.

[SSA05] STARK M., SHIRLEY P., ASHIKMIN M.: Generation of stratified samples for B-spline pixel filtering. *Journal of Graphics Tools 10*, 1 (2005), 61–70.

[VK03] VANRULLEN R., KOCH C.: Is perception discrete or continuous? *TRENDS in Cognitive Science 7*, 5 (May 2003), 207–213.

[VRK05] VANRULLEN R., REDDY L., KOCH C.: Attention-driven discrete sampling of motion perception. *Proc. Natl. Acad. Sci. 102*, 14 (April 2005), 5291–5296.

[Yel83] YELLOT JR J. I.: Spectral consequences of photoreceptor sampling in the rhesus retina. *Science 221* (July 1983), 382–395.