Texture Potential MIP Mapping, A New High-Quality Texture Antialiasing Algorithm

R. J. CANT
Nottingham Trent University
and
P. A. SHRUBSOLE
Philips Research

A refined version of the texture potential mapping algorithm is introduced in which a one-dimensional MIP map is incorporated. This has the effect of controlling the maximum number of texture samples required. The new technique is compared to existing texture antialiasing methods in terms of quality and sample count. The new method is shown to compare favorably with existing techniques for producing high quality antialiased, texture-mapped images.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation—*Display algorithms*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Color, shading, shadowing, and texture*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Anisotropic filtering, antialiasing, texture mapping

1. INTRODUCTION

In recent years perspective texture mapping has become a standard feature of high end 3-D graphics systems, and is becoming increasingly popular as a feature of lower cost systems and games. Until comparatively recently, in low cost systems little or no effort has been made to alleviate the problems of aliasing that occur when the texture pattern becomes comparable in scale to the raster grid.

Authors' addresses: R. J. Cant, Department of Computing, Nottingham Trent University, Burton St., Nottingham, NG1 4BU, UK; email: richard.cant@doc.ntu.ac.uk; P. A. Shrubsole, User-System Interaction Technology (USIT), Philips Research, Building WAE 1.08, Prof.Holstlaan 4, 5656 AA Eindhoven, Eindhoven, 5656 AA, The Netherlands; email: paul.shrubsole@philips.com.

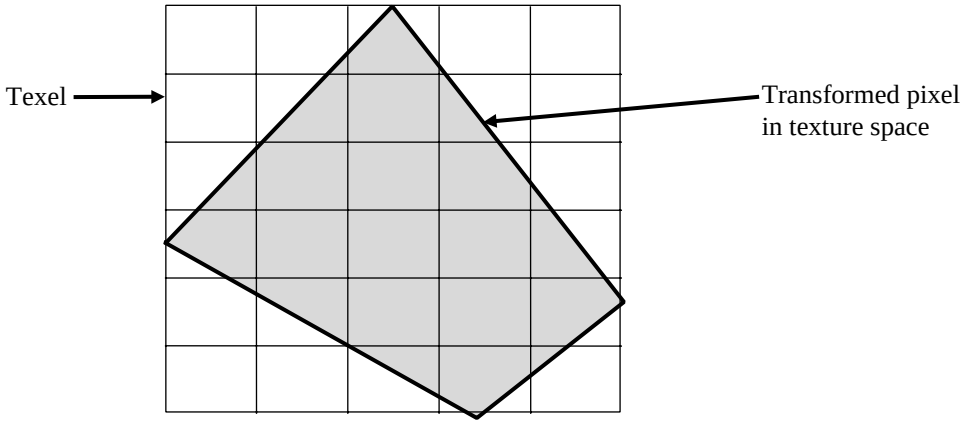

Fig. 1. The general form of a pixel once projected into texture space.

In high quality real-time systems the usual approach to the problem is MIP mapping [Williams 1983]. In all MIP mapping techniques the texture pattern is stored at a number of resolutions, the coarser ones being derived from the finer by a process of filtering and decimation that eliminates aliasing. When each pixel is drawn a suitable version of the texture pattern is selected, based on the size of the pixel's "footprint" on the texture map.

To make the most efficient use possible of the MIP map tables it is necessary for them to include frequencies right up to the Nyquist limit for their resolution. This in turn makes it desirable to bilinearly interpolate between the four nearest texels to the projected pixel in order that sharp transitions (which can reintroduce a form of aliasing) are avoided. A similar problem can arise with transitions between one resolution of texture map and the next. In this case the solution is to interpolate between the two layers. Hence, trilinear MIP mapping is the standard highest quality algorithm.

Unfortunately this algorithm is not a perfect solution because it has a significant flaw. It only allows for a variation in the pixel's overall size yet the process of perspective projection allows the pixel's footprint to be distorted into a general quadrilateral as shown in Figure 1. When the surface being displayed is at a shallow angle to the line of sight this quadrilateral will be long and thin. A possible example of this is shown in Figure 2. To prevent aliasing in the direction corresponding to the long axis of the pixel a coarse (heavily filtered) MIP map must be used. However, this will suppress the texture in the perpendicular direction, resulting in a premature fading or blurring of the pattern.

A solution to this problem which is commonly implemented in commercially available hardware is a pixel-based supersampling approach. While this can solve the problem in principle, it does so at an enormous cost. In high-end graphics workstations and flight simulator systems up to 16

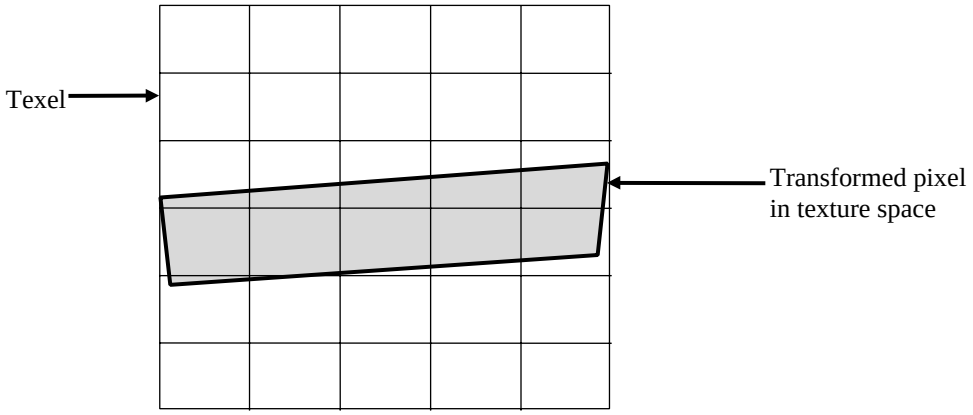

Fig. 2. A projected pixel that cannot be satisfactorily sampled using MIP mapping.

samples are commonly used, each of which requires in turn eight texels to allow for trilinear interpolation. The total is nearly 100 accesses to texture memory per pixel. Even at this cost, angles will sometimes occur that are sufficiently shallow to defeat the system.

In the early days of texture mapping several other methods of antialiasing were proposed. Some, such as the narrowband noise model [Schachter 1980] restricted the range of texture patterns that could be used, while others [Feibush et al. 1980] involved brute force integration over the pixel footprint in texture space, imposing an unacceptable computational cost for real-time systems.

A method that does not suffer from either of these problems was proposed by Crow [1984]. This method is based on the idea of storing the two-dimensional indefinite integral of the texture pattern, rather than the pattern itself. For any given pixel footprint the required definite integral can be obtained simply by looking up the value of the indefinite integral at the upper and lower bounds. This only works exactly if the pixel footprint is rectangular and aligned to the axes in texture space. Nonetheless, it still accurately covers many more situations than can be dealt with directly by using MIP mapping. The configuration shown in Figure 2 is one where this method is successful. However, Figure 3 will not be handled well because the algorithm must include contributions from a rectangular region surrounding the projected pixel. In this case all of the texels shown on the diagram will contribute and unnecessary blurring will be introduced by the large areas that lie outside the shaded region.

Refinements to MIP mapping [Schilling et al. 1996] and Crow's method [Glassner 1986] have been proposed to overcome this problem. The present authors have proposed texture potential mapping (TPM) [Cant and Shrubsole 1996; 1997] for the same purpose. An extension of this method, which substantially limits its computational cost, forms the subject of the present article.

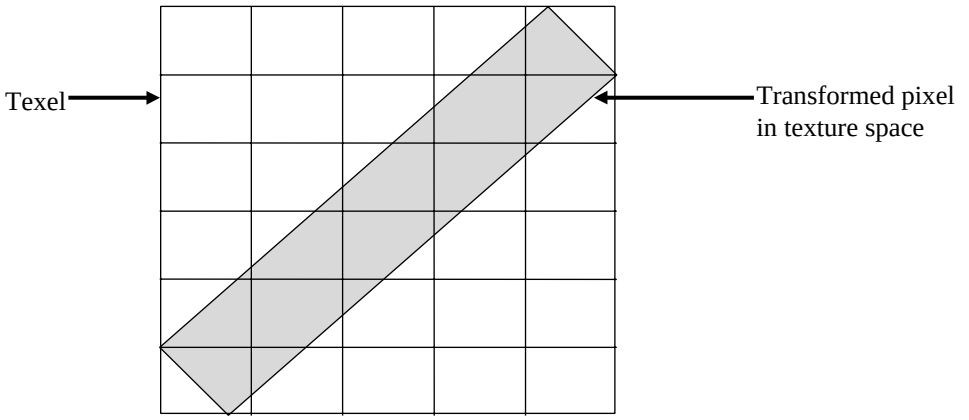

Fig. 3. A projected pixel which cannot be satisfactorily sampled using Crow's summed area table technique.

2. REVIEW OF TEXTURE POTENTIAL MAPPING—PROBLEM OF LARGE NUMBER OF SAMPLES FOR DISTANT PIXELS

Unlike the algorithms mentioned previously, TPM follows the form of each projected pixel faithfully in order to overcome the problem of texture "over-blurring" that is evident when approximate pixel footprints are applied.

In order to produce antialiased texture, an average texture intensity (or color) over all the texels that lie within the quadrilateral must be derived. A brute force approach of looking at every texel value is too time-consuming. TPM overcomes this by finding the average texture value within a pixel footprint by integrating the texture within it using values that lie around its edges only. This concept is analogous to Gauss' theorem in physics, in which the charge contained within a volume can be found by examining the electric field on the surface of that volume. Thus, each texture coordinate is forced to have a corresponding "field" value which is stored as a running total of its previous intensities down a column in texture space. The total texture intensity bounded by the quadrilateral within that column is then found simply by the subtraction of the upper and lower "field" values. If we integrate this over all the columns within the quadrilateral, an average intensity value can be deduced. Since the "field" values only need to be calculated once, they can be precomputed and stored, leaving only one subtraction per texture column and only one division per transformed screen pixel.

This algorithm produces excellent qualitative results, as shown on the left-hand side of Figure 4, when compared with trilinear MIP mapping of an identical image shown on the right-hand side of the figure. For the majority of angles the speed overhead of TPM is only a factor of two to four relative to trilinear MIP mapping. However, unlike that method, or the summed area table, the time taken per pixel by TPM will vary according to the scale and orientation of the texture. This is due to the fact that

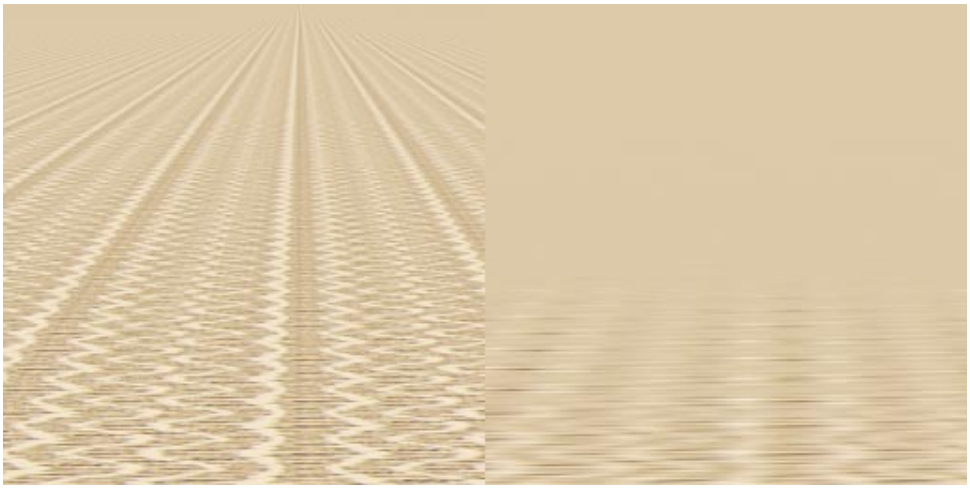

Fig. 4. Texture potential mapping of a pattern at an extremely shallow angle compared to trilinear MIP mapping of the same pattern at the same angle.

integration occurs in one direction only, leading to an unpredictable variation in the number of samples that need to be taken in the other direction. TPM thus has a weakness when the projected pixels are very wide. In this situation the number of samples required, although smaller than needed by the brute force method, is still excessive. In many of these cases quite adequate results can be produced by the summed area table, or even conventional MIP mapping.

This suggests that there should be a way to reduce the overhead of TPM in these cases while retaining most of the quality improvement when it is needed.

3. COMBINING TEXTURE POTENTIAL MAPPING AND MIP MAPPING — PRINCIPLES

To overcome this problem we propose the combination of TPM with a MIP map approach in the nonsummed direction. The resulting algorithm is referred to as texture potential MIP mapping (TPMM). We work with a set of potential maps, each one-half the size of the preceding one in the x direction only. The summation principle is used in the y direction. The resulting table is twice the size of the original potential map. The generation of this table from the original texture pattern can be done in either order but it is more efficient to filter and decimate for the MIP map before constructing the potential since it means that the filtering process can be done with a lower resolution number format.

In our experimental programs the filtering has been done by Fourier transforming the original texture, deleting frequency components above the Nyquist frequency for each MIP map, and then transforming back to the x space representation. This method of filtering is time-consuming but has the advantage of guaranteeing optimal results. To achieve similar results

with an FIR filter would require a convolution window as large as the texture pattern.

To generate an antialiased textured pixel we first apply the texture transformation to the corners of the pixel. In an optimized implementation this would be done incrementally and the results from corners that are shared between pixels can be reused. The workload for this operation is thus only marginally greater than it would be for a nonantialiased texture implementation.

Next we determine the position of the extremes of the pixel in the x direction of texture space. This information is helpful later in the process of edge tracing but its immediate purpose is to determine the width of the pixel. From the logarithm of the pixel width we can determine which of the MIP maps to use. The logarithm need not be computed exactly and so all we need to do is to find the most significant bit that is set. This would be easy in a hardware implementation, while in software, if the width of the pixel were expressed in floating point format, the exponent would contain the information required.

At this point the procedure varies from that employed by normal MIP mapping, because in the present case there is no unique "correct" choice of table to use. The decision must be based on a trade-off between the accuracy with which we follow the edge of the pixel and the number of samples that we take. The intuitive answer to this would probably be to choose the table so that the number of pairs of samples taken lies between, say, 4 and 8 or 8 and 16. This can be done using a parameter n with the maximum number of sample pairs being limited to $2n$. Clearly if we were to allow fewer than 4 pairs of samples there would not be much chance of following an angled pixel footprint with any accuracy, while a number much greater than 16 implies an excessive amount of work.

In order to choose the best compromise we need to understand how the image will be degraded if too few samples are chosen. There are two possible effects that could arise, aliasing or blurring of the texture. Clearly blurring is more acceptable than aliasing but unfortunately a naive implementation of our algorithm gives rise to aliasing in the middle distance at critical angles as shown in Figure 5.

To understand why this is so, consider the way in which the tracing process is affected by going to one of the coarser maps and taking just a few samples as shown in Figure 6.

Where the traced pixel lies outside the ideal pixel the result will be blurring because the sampling footprint is too large and hence the frequency cut-off too low. These areas are shown shaded on the diagram. The hashed areas show where the ideal pixel outline is outside the traced pixel. This is much more significant because in this case the footprint is too small and hence the frequency cut-off too high, potentially allowing in frequencies above the Nyquist frequency and causing aliasing.

The left-hand side of the diagram shows what happens if we use the original (basic) resolution of the potential map. In this case the errors are very small and the resulting image shows neither aliasing nor blurring.

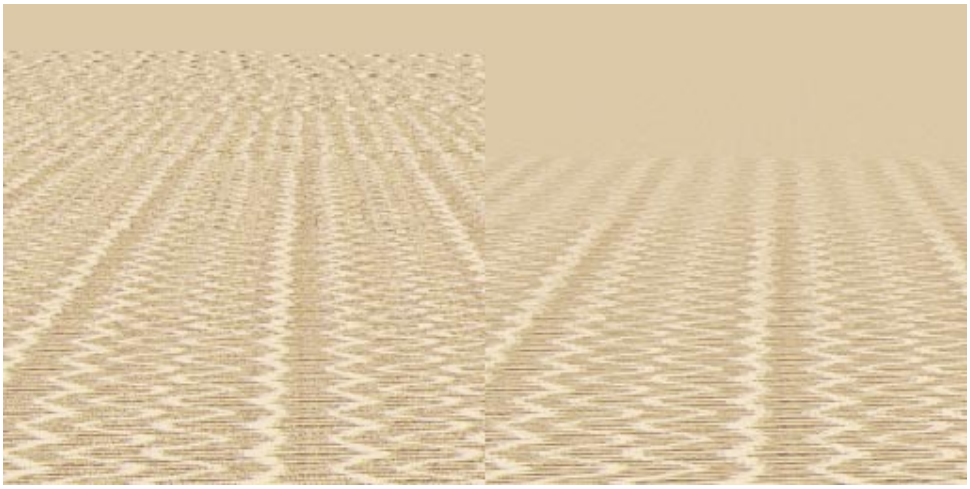

Fig. 5. Illustration of the aliasing effect in a naive implementation of texture potential MIP mapping (left-hand image) compared to the more acceptable blurring effect (right-hand image) that replaces it when the traced footprint is enlarged to guarantee the inclusion of the whole of the pixel. Both images were created using a very low sample count to illustrate the effect clearly.

The right-hand side shows the situation when the texture value for the same pixel is calculated using one of the coarser MIP maps. The errors are much larger now and the size of the striped areas is significant. The striped regions cause aliasing because they reduce the filter kernel width so that it passes frequencies above the Nyquist frequency. This has been marked in Figure 6.

To correct this situation we must expand the traced pixel to guarantee that it includes all of the ideal pixel. This arrangement is shown in Figure 7. The total sample footprint is increased in the vertical direction. When this is done, the aliasing that was evident in the left-hand side of Figure 5 is replaced by blurring as seen on the right.

The coordinates that are used to access the table can be derived by any line-drawing technique following the edges of the pixel. Either the DDA algorithm or Bressenham's algorithm can be used, depending on available hardware and bearing in mind that the slope calculation can be shared along a raster line. To expand the footprint to include the whole of the ideal footprint we calculate the positions of the intercepts at both edges of each column in the potential map and choose the largest value at the top and the smallest value at the bottom. Note that these intercept calculations are shared between neighboring columns of texels.

At this point it would seem that only one task remains—to find the optimal value of n for a given application. This is not the best way to proceed however because there is an observed relationship between the ideal value of n for a given image and the angle that the pixels' long axis makes to the grid of the texture map. Very steep and very shallow angles allow a much smaller value of n to be used while near-45 degree angles

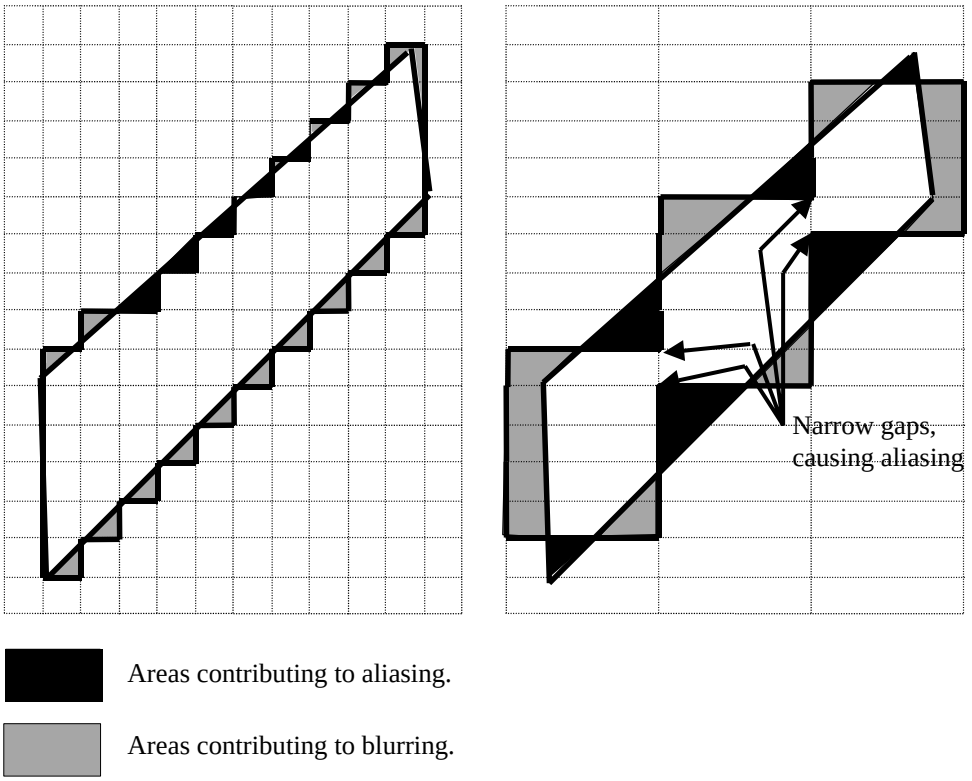

Fig. 6. The effect of different MIP map resolutions in the naive implementation of texture potential MIP mapping.

require a larger value. The reason for this is that in the former situations the ratio between the area of the ideal projected pixel footprint and that of the expanded footprint is close to 1. A fatter pixel footprint also allows n to be smaller because the errors, although they have the same absolute values, are reduced as a proportion of the total pixel area. Figure 8 illustrates this point.

There are thus many situations in which a smaller value of n can be chosen than that which is required to cope with the most awkward cases. This is not surprising because these are the configurations where Crow's method or (in the fat pixel case only) traditional MIP mapping also work quite adequately.

5. CHOOSING THE BEST NUMBER OF SAMPLES

From the above we see that the system would be most efficient if we could control the number of samples as a function of the size, shape, and angle of the pixel footprint, rather than just keep it constant using the width. To do this we need a simple shape dependent parameter that can be easily calculated and that reflects the way in which the number of samples needed changes with all of these factors.

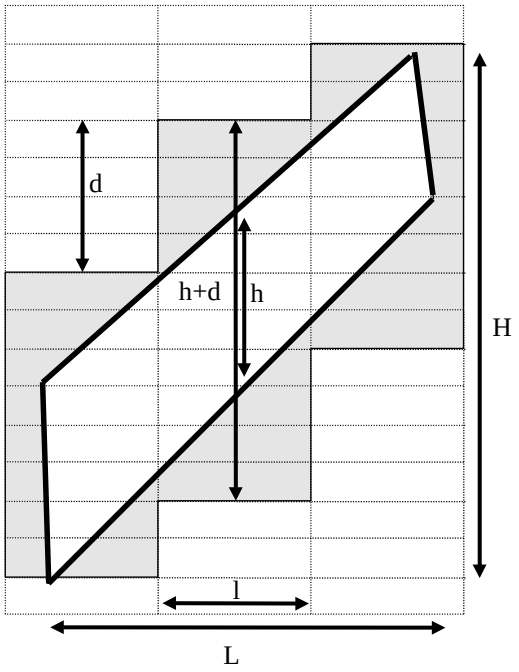

Fig. 7. Expanding the sampled pixel to cover the actual pixel.

The ratio of area of ideal pixel to that of actual sampled pixel would be suitable but requires far too much calculation to be usable in the present situation. It is possible, however, to find a simple parameter that behaves in an equivalent way. To make this calculation we consider the pixel to be a thin parallelogram and ignore what happens at the ends. This assumption is certainly accurate in those cases where a large number of samples need to be taken and so it follows that we will never misclassify such a case. If we make a mistake that causes *too many* samples to be taken, then the consequence is only a slight decrease in efficiency, which is outweighed by the time saved in making the test itself.

If we consider an individual column in the pixel footprint in Figure 7 then the area of the ideal slice is lh and that of the sample actually taken is $l(h + d)$. For the pixel as a whole these values become Lh and $L(h + d)$, respectively. The ratio of the areas is thus:

$$R = \frac{(h + d)}{h} = 1 + \frac{d}{h} \tag{1}$$

Now if there are n columns in the pixel it follows that $d = (H - h)/n$ and so we have:

$$R = 1 + \frac{H}{nh} - \frac{1}{n} \tag{2}$$

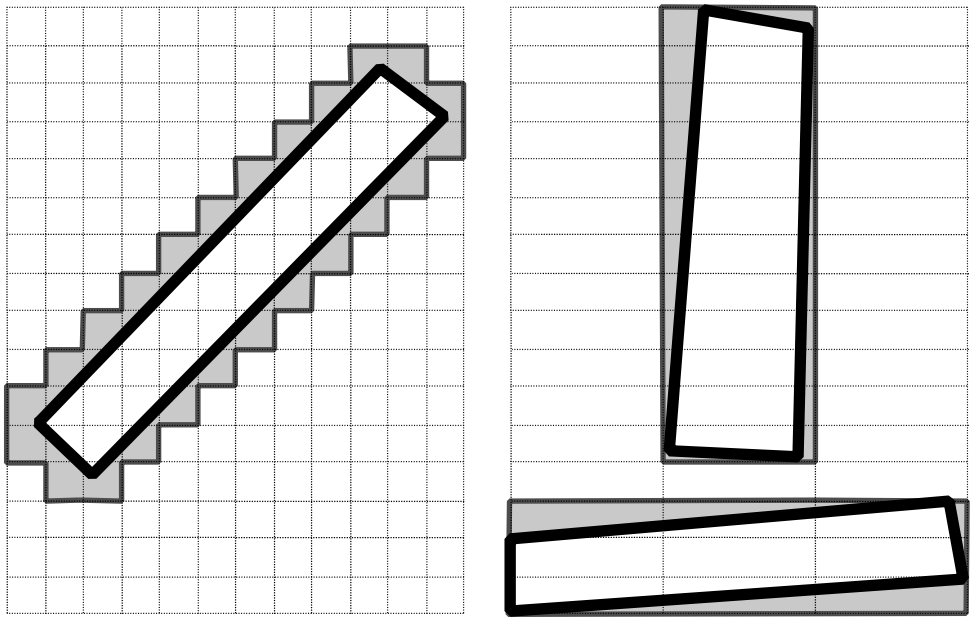

Fig. 8. The effect of pixel angle on choice of MIP map.

so we can choose an acceptable value for R and then derive n by the equation:

$$n = \frac{1}{(R - 1)}\left(\frac{H}{h} - 1\right) \tag{3}$$

which is reasonably tractable since it only requires the total height of the pixel and a sample of its "thickness" in the middle. If the equation is used in this form then there is a problem where $H \approx h$ since a value of $n = 0$ can result (or even a negative value of n, given the possibility of rounding errors). To prevent this happening, with a margin of safety, we have modified the equation to read:

$$n = \frac{1}{(R - 1)}\left(\frac{H}{h} + 1\right) \tag{4}$$

This modification has the advantage that it allows the minimum value of n to be controlled by R and it introduces no extra computation compared to the original form (Equation (3)). To save computational time this quantity could be estimated on a per-polygon basis. However it is most effective to calculate it for each pixel as we have done in our experimental programs.

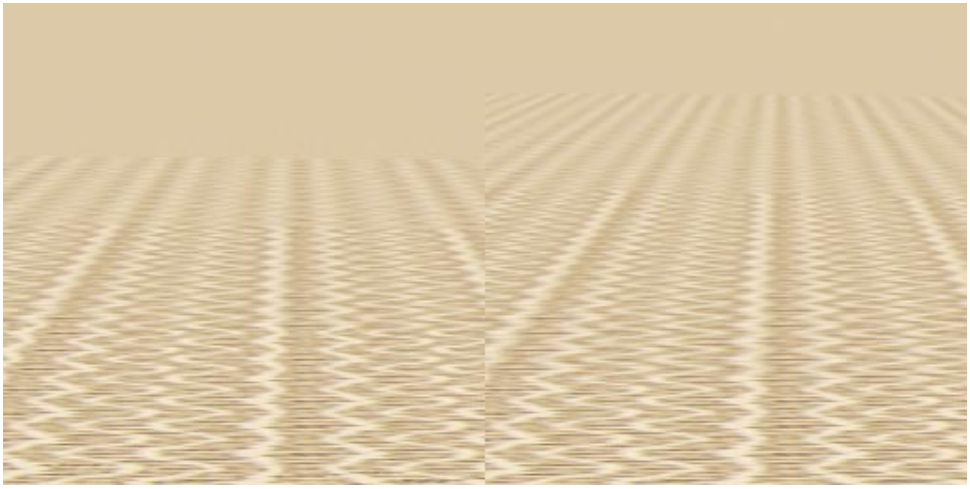

Fig. 9. Comparison between the effect of a fixed value for n (left-hand image) and individual calculation for each pixel (right-hand image). The average number of samples per pixel is 12 in both images.

We have found that it gives very good discrimination when used in this way. Quite large values of R can be chosen to give qualitatively excellent results without compromising speed. The improvement obtained by calculating n on a per-pixel basis compared with simply choosing a value for the polygon is shown in Figure 9. To illustrate this effect clearly it is necessary to use a rather lower number of samples than normal. The left-hand side of the image uses a fixed n value of 2 whereas the right-hand side has been created using a similar average number of samples but with n being derived individually for each pixel from an R value of 6. The average number of samples per pixel is 12 in both cases.

No doubt those who wish to optimize the algorithm will find ways to allow the number of calculations of n to be reduced without affecting quality significantly.

6. PERFORMANCE COMPARISONS

There are two algorithms that have been put forward as solutions to the problems that we address here. These are a modification of Crow's algorithm [Crow 1984] which was proposed by Glassner [1986] which is known as the adaptive precision method and an algorithm proposed by Schilling et al. [1996] which they call the footprint assembly method.

In estimating the performance likely to be achieved by any of these methods a number of different cost factors need to be considered. The true performance in a given situation (hardware and software environment together with the user requirement) can then be estimated by taking appropriate account of each of them as the situation demands.

We can identify these different performance criteria for texture antialiasing systems:

(i) computational cost,

(ii) number of samples needed per pixel,

(iii) bandwidth to texture memory, and

(iv) table size.

A comparison based on computational cost at the level of individual arithmetic operations would require an optimal implementation of each algorithm. Since all the algorithms in question are quite complicated, this would be difficult to do with any certainty. The result would also be machine dependent and, given the rate of technological change, time dependent. In contrast, the slowest moving performance factor in recent years has been from chip communication. For example, PC main memory bus speed has improved by only a factor of 3 between 1989 and 1999, while processor speeds have gone up by a factor of 12. This makes the number of texture samples required to generate a pixel a significant performance factor since it determines the number of independent memory accesses that will be required. It will also give an estimate of computational cost, since all the algorithms have a processing requirement that is proportional to the number of samples.

Of course the *number* of accesses is not the only factor to be considered. The *size* of each data item is also important because that will determine the total *bandwidth* that will be needed for this traffic. Bandwidth is both a better and a worse measure than number of samples. It is a better measure because it is clearly incorrect to equate the retrieval of a 64-bit word with that of an 8 bit byte. It is also a worse measure because a byte that is retrieved as part of a 64-bit word clearly costs less than one read individually. Consequently, although we have presented our results in terms of the number of samples taken, we have included bandwidth information also.

The effect of both of these numbers will, in turn, be modified by any caching of samples that is taking place, but this effect is very sensitive to the cache size relative to the size of the texture map.

COMPARISON WITH GLASSNER'S ADAPTIVE PRECISION ALGORITHM

Glassner's paper [Glassner 1986] explores a number of different techniques for refining Crow's summed area table algorithm. Many of these proposals are not particularly practical since they require multiples of the full summed area tables to be stored. Glassner's most practical method is similar to TPMM in its sampling patterns for long thin pixels (although the number and size of samples differs). It is therefore possible to directly compare the two at a theoretical level, without the need to compare results visually.

Glassner's idea is to improve on the rectangular approximation of the pixel footprint used in Crow's technique by subtracting further rectangular pieces from it. Because each piece is a rectangle its contribution can also be evaluated using the same summed area table.

Both our method and Glassner's require a rather larger version of the texture map than MIP mapping or basic (nonantialiased) texture mapping, although the summed maps do not need to be quite as large as some have suggested [Schilling et al. 1996]. For a 256 x 256 x 8 bits per color texture pattern a texture potential map could theoretically require 16 bits per color and a summed area table as many as 24. However this worst case is rarely met with practical textures and provided the color values are calculated relative to an average value (which can be calculated separately for each column in TPM with little extra cost) 13 bits will almost always suffice. Indeed, if a small amount of compression of dynamic range is accepted this can be reduced to 12. The required table sizes for each of the techniques are summarized in Table I.

When comparing sample count and bandwidth, the range of possible pixel footprint configurations makes a direct discussion of the general case quite difficult. Because of this we first examine first the simple cases when small values of n suffice and then the limit of large n, in which simplifying approximations may be made. The behavior between these two points can then be inferred. In the simplest case, Glassner's method reverts to the basic summed area technique and simply requires four samples per pixel. TPMM also requires a minimum of four samples in general since each pixel will span two columns. However, the summed area table samples require 16 to 24 bits per color each while TPMM needs only 12 to 16 bits per color. In addition the TPMM approach is, at this stage, already able to follow a diagonally oriented pixel to some extent since the two pairs of samples can be vertically displaced relative to one another. In the horizontal direction the sampling methods are rather different and hence a direct comparison is difficult.

Superficially it might seem that the summed area method has an advantage in that its sampling window can be positioned at the resolution of the original texture map whereas TPMM is limited by the fixed sampling points of the current MIP map. However, this only helps when the texture pattern contains a frequency component with a particular phase relationship to the display pixel grid. This frequency component will then exhibit aliasing if the image moves.

For large numbers of samples the same simplifying assumptions can be made about the pixel ends as were used in the discussion of the R parameter above. For TPMM the number of samples required for a given factor is double the value of n as given by:

$$n = \frac{1}{(R - 1)}\left(\frac{H}{h} - 1\right)$$

For Glassner's method consider a section from the middle of a diagonal pixel as shown in Figure 10.

The variables H, h, and d have the same meanings as in the TPMM case. Glassner's method requires samples to be taken at the points marked A

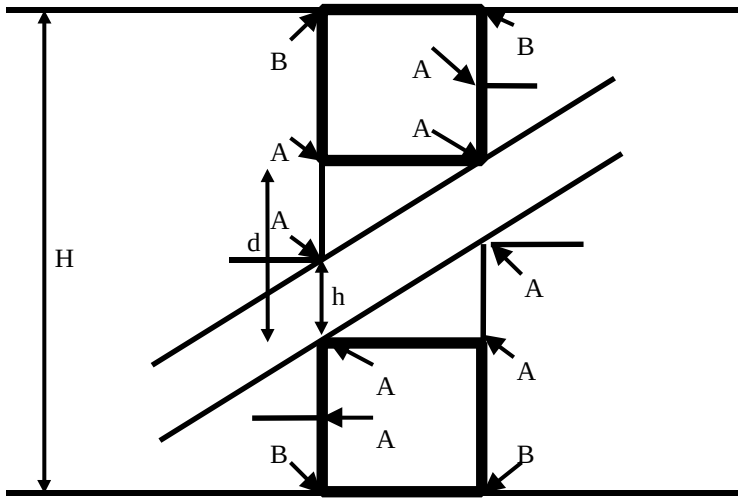

Fig. 10. The sampling of a pixel using Glassner's adaptive precision technique.

and, naively, two at each point marked B also. However these multiple samplings will cancel, leaving a total of eight samples. These samples are of course shared between two columns, so the final result is four samples per column. This means that for a given R the number of samples will be four times the value of n given by

$$n = \frac{1}{(R-1)}\left(\frac{H}{h} - 1\right)$$

Each of these samples will require 54 bits to be retrieved for a 24-bit color texture. TPM needs 39 bits per sample on the same assumptions.T-PMM thus has an asymptotic advantage of a factor 2 for number of samples and better than 2.5 for bandwidth when compared with Glassner's method for similar image quality. For the less critical pixels this will fall away progressively to equality for sample count and an advantage of 18:13 for bandwidth.

The above analysis was based on the situation for individual pixels. However the existence of a discrete set of MIP maps in TPMM means that a demanded value of R and n will not normally be achieved exactly. Consequently, if a given *maximum* value for R is requested then, on average, the value of n actually used will be 50% larger than that given by the equation. This will result in a 50% reduction in the advantage of TPMM over the adaptive precision method if a "worst-case" performance criterion is used.

COMPARISION WITH FOOTPRINT ASSEMBLY

The idea of footprint assembly [Schilling et al. 1996] is to use a standard MIP-map table but instead of allowing the scale to be determined by the

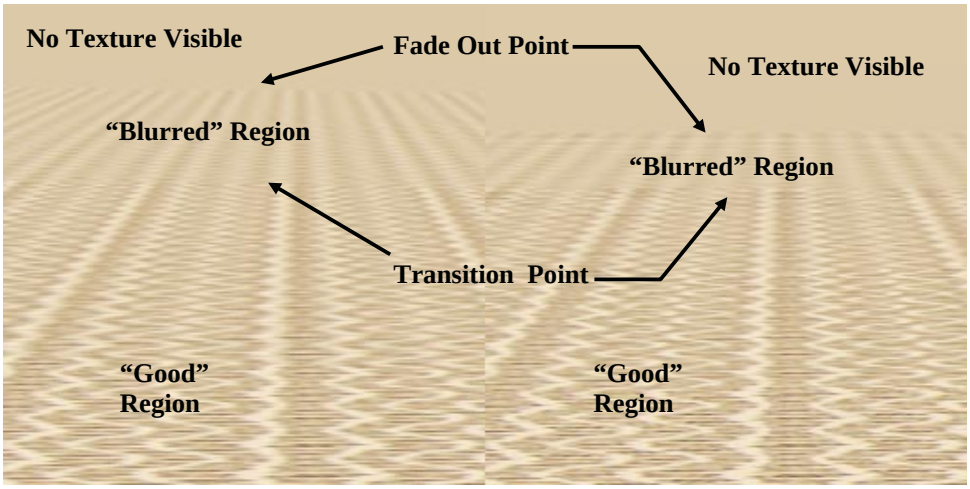

Fig. 11. Comparison of texture potential mapping with $R = 5.76$ and footprint assembly with $N\max = 4$.

largest dimension of the projected pixel the smallest dimension is used and multiple samples are taken to avoid aliasing. This technique can be used with single sample, bilinear, or trilinear MIP mapping. We have compared these techniques with TPMM in terms of image quality, number of samples taken, and memory bandwidth.

Experiments were done using TPMM and our own implementation of footprint assembly mapping. The texture pattern chosen was selected because it contains a range of frequencies but is still intelligible to the human eye. Patterns such as checkerboards can display certain problems very clearly but only contain a limited set of frequencies and hence do not exercise the algorithms fully. The optimum pattern from the point of view of frequency coverage would be a noise texture, but this cannot be used for visual comparison for obvious reasons. In each case the image created was of a complete "ground plane" of texture. The view of this ground plane was varied in certain ways. First, the angle of the plane itself relative to the line of sight was varied between 0 (looking straight down at the surface) and 1.5 radians. This angle is henceforth referred to as α. It corresponds to a rotation about the x axis in most conventional viewing coordinate systems. Since the angle of view chosen was 0.14 radians, 1.5 radians is the angle at which the point at infinity becomes just visible at the top of the screen. Any further rotation would simply add extra empty space at the top of the image. An α value of 1.5 is thus the most demanding test for any texture antialiasing system. The second angle is that of the texture pattern itself, rotating in its own plane. We have referred to this as β . When β is close to 0 or a multiple of $\pi/2$ the texture grid is aligned with the pixel grid. In this situation Crow's method will work well and TPMM will require fewer samples. In contrast, a β value near to a multiple of $\pi/4$ will be the most demanding case. The example images in Figures 4, 5, 9, and 11 come

from this set. They used $\alpha = 1.5$ radians and $\beta = \pi/4$ radians. All the images were originally created at a resolution of 768 x 768, but the figures as shown here comprise only the top, central 256 x 256 pixel portion of each image so that the aliasing and blurring effects can be seen clearly. The sample counts and bandwidth requirements have been calculated using the full 768 x 768 image to make the results more representative of real world requirements. While the image quality of the methods needs to be assessed in those domains where significant differences can be detected, the speed must be evaluated as an average over a representative range of different configurations.

The critical set of angles described previously creates long thin projected pixels that lie diagonally across the texture map. To make visual comparisons between the two methods unambiguous we have implemented both of them without interpolation between the different MIP maps. This creates clear discontinuities in the image in both techniques. Comparisons can be made between the methods by matching the positions of these discontinuities. It would not be practical to include all the images that were used in this process within the present article. However we have included a particular example in Figure 11 to make the objective nature of the process clear. The left-hand side of Figure 11 shows the results of TPMM using an R value of 5.76 at the angle specified above. For comparison the right-hand side shows the results of footprint assembly mapping using bilinear interpolation and a maximum MIP map sample count of 4 at the same set of angles. In the foreground of each half of the image there is a region where the texture is clear and well defined. At some point in the distance this gives way to a more blurred effect and finally the texture fades out altogether. These features are annotated in Figure 11 and are present in all the images created. Two sets of comparisons were done. One set is based on the matching between the two algorithms of the point of transition to the initial blur. The other is the same except that it uses the final fadeout point. The comparisons were performed by an observer who is not involved in this program of research. The observer was presented with the images in pairs. In each case one of the images was generated by TPMM and the other by footprint assembly but the observer was unaware as to which was which. The observer was asked to specify which of each pair of images had its “fadeout point” and “transition point” farthest up the screen. The meanings of these terms are as illustrated in Figure 11. For reference, the “transition point” and “fadeout point” are both farthest up the screen on the left-hand image of that figure. Different pairs of images were presented with the object of finding pairs that were matched. We defined the quality to be matched when the transition in question was at exactly the same distance up the screen. Image magnification and a cursor with pixel position readout were available to clarify this, so the process was one of image measurement rather than subjective assessment. If an exact match was not found then the nearest pair of images with the TPMM one having a higher cut-off or transition point was used.

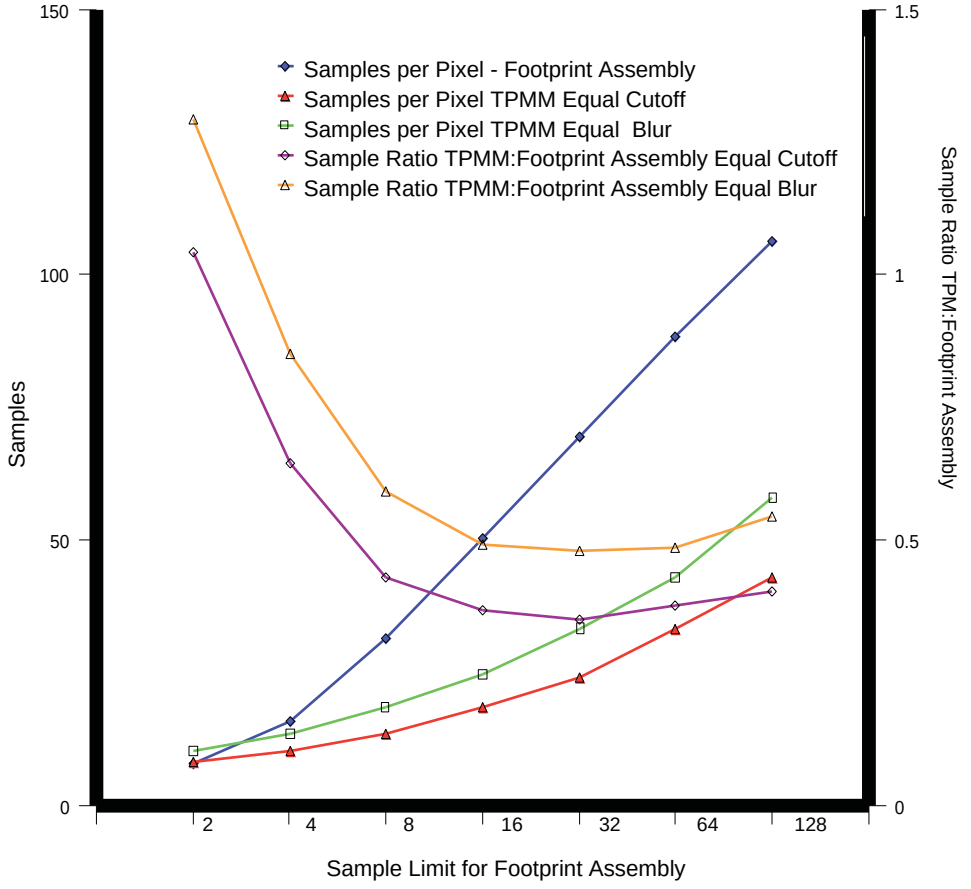

Fig. 12. Comparison of the number of samples required for texture potential MIP mapping and footprint assembly at varying levels of image quality.

Figure 12 shows the relative number of samples required for each of the methods over a range of different levels of image quality but with the quality matched in each case between the two methods as described above and illustrated in Figure 11. Two sets of curves are shown, reflecting the two comparisons made.

Overall, the results indicate that TPMM represents a better compromise between image quality and sample number/bandwidth for this kind of extreme situation. This advantage is most pronounced where footprint assembly has a sample limit in the range 8 to 32, corresponding to an average sample count in the range 10 to 100. This is also the most likely region in which either technique might be used. In addition, it is important to consider the performance over the range of possible angles. The results in Figures 13 and 14 use the "quality point" represented by $N = 16$, $R = 3.38$. Figure 13 shows the number of samples required over a range of β values but at the critical α value. Footprint assembly requires 50 samples throughout this range. The number of samples required by footprint

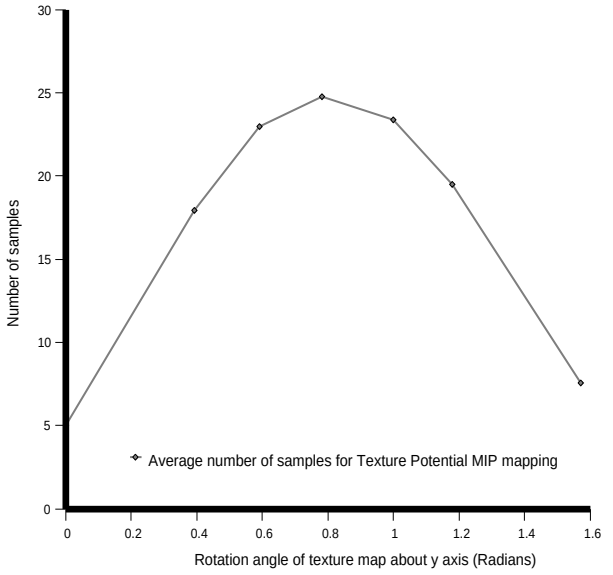

Fig. 13. Number of samples required by texture potential MIP mapping versus angle of rotation in the texture plane (β) $R = 3.38$.

assembly mapping is constant here because it is determined only by the projected pixel's shape. Conversely, that required by TPMM is reduced away from the critical β value since here the relationship between the projected pixel and the texel raster is taken into account. This can only increase the advantage of TPMM over footprint assembly. Note that this curve is not quite symmetrical about $\pi/4$ because the pure TPM method used for narrow pixels is slightly more efficient than the MIP mapping used for wide ones. We have also plotted in Figure 14 the variation in sample requirement against α while holding β fixed at its critical value. Here the bandwidth required by footprint assembly mapping also reduces away from the critical region—but less steeply than for TPMM.

These results clearly establish the greater efficiency of TPMM compared to footprint assembly with bilinear interpolation for static images. However, it is necessary to consider whether footprint assembly with bilinear interpolation is the most efficient form of footprint assembly that could be used. It is important to note that Schilling et al. [1996] introduced footprint assembly in the context of a special hardware accelerator for trilinear interpolation and hence could hardly be expected to give a lot of consideration to whether the interpolation was really necessary.

The bilinear interpolation within a layer in conventional (and footprint assembly) MIP mapping is a measure that prevents aliasing phenomena that would otherwise arise from the sharp transition between one texel and the next. If the surface is very close this aliasing would manifest itself as a patchwork of squares that, on close inspection, have jagged edges. There is room for difference of opinion as to how this problem *should* be resolved since clearly there is a lack of information in the texture map as to what

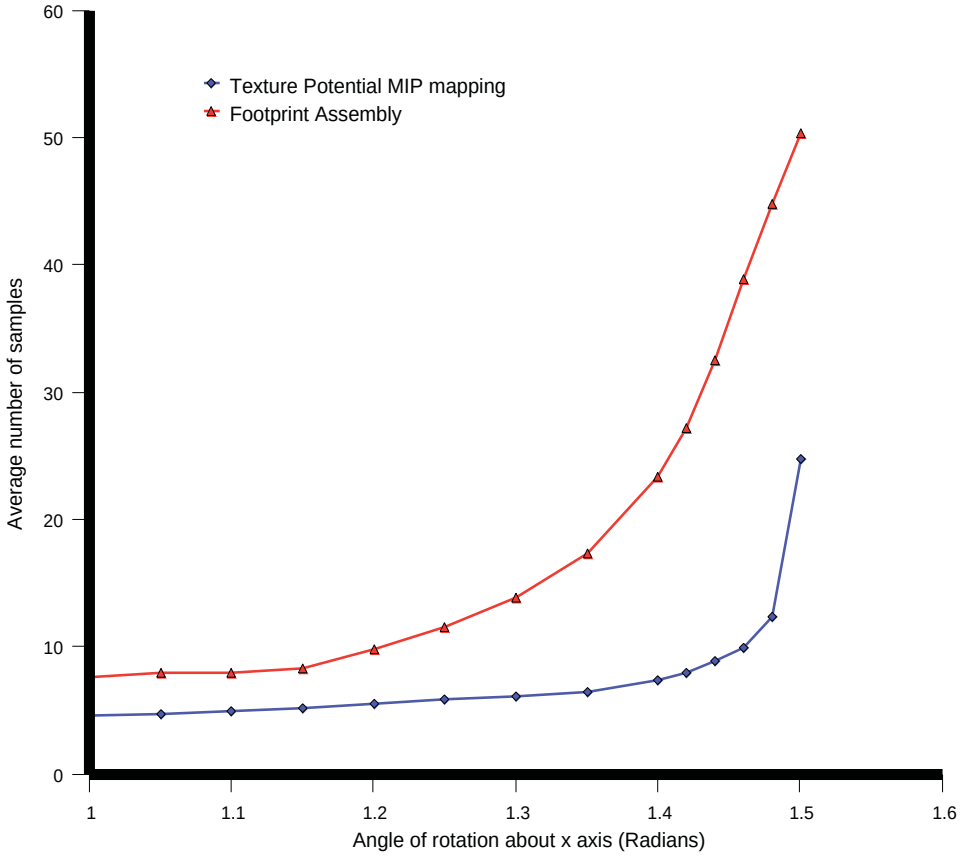

Fig. 14. Number of samples required by texture potential MIP mapping ($R = 3.38$) and footprint assembly ($N = 16$) versus angle of rotation of the texture plane α.

happens at scales finer than one texel. The naive interpretation of constant color over each texel produces the result as described above but without the jagged edges. The linear interpolation approach normally used in MIP mapping gives rise to a texture function that is continuous (C^0) but has discontinuous derivatives (not C^1). It is possible to demand a greater degree of continuity—but the interpolation required is rarely implemented at present since it is computationally expensive.

In conventional MIP mapping this aliasing problem can appear at all scales, and is particularly prominent when texture patterns are used that contain sharp edges (high frequencies). One situation in which this problem shows up is when such a texture is presented at a shallow angle. In this situation bilinear interpolation is essential with conventional MIP mapping if aliasing is to be avoided. Footprint assembly MIP mapping allows the possibility that the extra samples used for bilinear interpolation could be used to increase the value of N, but with improved results for static images. Unfortunately the problem can also appear in the form of a scintillation effect with moving images and this happens irrespective of the angle of the

textured plane. It seems therefore that the extra burden of interpolation cannot be avoided with footprint assembly if good quality results are desired.

These same problems also need to be addressed with TPMM. In this case, because of the asymmetry of the algorithm, the two directions need different treatment. In the MIP map direction this can be done by scaling the contributions of the end columns of the pixel according to their width. This approach has no effect on the number of samples or bandwidth required. A completely general treatment requires the summed direction to have the interpolation applied also. In the summed direction the problem only occurs with nearby surfaces because the texture map is always used at maximum resolution in this direction and so when the projected pixel is large it is finely sampled. If this effect is considered a problem it can be dealt with by sampling the two neighboring entries at each end of each column and interpolating between them. This will double the number of samples and bandwidth required but need only be done for certain nearby surfaces—which are otherwise undemanding in terms of required sample count. The impact on both average and worst-case performance for TPMM will thus not be very great.

If we consider bandwidth rather than sample count then the results are slightly less unfavorable to footprint assembly since it deals with 8-bit samples rather than the 12 bits (or slightly more, depending on the texture map size) required by TPMM. The sample count advantage of TPMM is about a factor of two at the critical set of angles and is larger at many other orientations so the situation should remain favorable when the bandwidth measure is used instead.

There remains the question of texture map size and here it has to be admitted that TPMM does require a larger amount of memory than either Glassner's method or footprint assembly. However, it does not require an excessive amount as do, for example, the brute force approach of separate MIP maps or summed area tables for each possible angle of the texture pattern.

Typically each MIP map layer in TPMM requires 1.5 times the memory of the original texture pattern. The multiple layers require a further factor of two, resulting in a factor of about 3 overall. It may be possible to use a factor 4 scaling between tables at some cost in sample count. In that case the overall factor would be about 2. This compares with a factor of 1.33 for conventional or footprint assembly MIP mapping and a factor of 2 to 3 for Crow's or Glassner's method.

Table I shows the memory sizes needed for the various methods described in the text. All the figures are based on a 256 x 256 texture map. The range of values given in the first row reflects the differing requirements that can occur with any of the summed area type methods depending on the content of the texture map. In the case of TPMM there is a further variation that comes from the possibility of utilizing MIP maps with a scaling factor of 4. Glassner's multiple table method allows a very wide range of possibilities

Table I. Table of Texture Map Sizes Required by the Different Methods

Texture Mapping Method	Relative Table Size (Minimum)	Relative Table Size (Typical)	Relative Table Size (Maximum)
Basic and brute force integration	1	1	1
MIP mapping and footprint assembly	1.333	1.333	1.333
Potential map	1.5	1.625	2
Summed area and Glassner's adaptive method	2	2.25	3
Potential MIP map	2	3.25	4
Glassner's multiple table method	4 (2 tables)	4.5 (2 tables)	6 (2 tables)

depending on the required quality. The figures given are for the most basic version with just two tables.

7. CONCLUSIONS

We have shown how the texture potential mapping algorithm can be modified to keep the required sample count within reasonable limits by using the principles of MIP mapping in the nonsummed direction. The resulting algorithm has a favorable combination of sample count, table size, and quality compared with competing algorithms such as footprint assembly [Schilling et al. 1996] or the adaptive precision method introduced by Glassner [1986]. The results of the algorithm are intermediate in quality between current real-time hardware systems and what can be generated offline. As such it is a candidate for implementation in future real-time hardware.

REFERENCES

CANT, R. J. AND SHRUBSOLE, P. 1997. Texture potential mapping: A way to provide antialiased texture without blurring. In *Visualization and Modelling*. Academic Press, Inc., Duluth, MN, 223–240.

CANT, R. J. AND SHRUBSOLE, P. 1996. Proposal for distributed texture mapping. In *Proceedings of the European Symposium on Simulation* (ESS'96, Genoa, Italy, Oct.). 29–33.

CROW, F. 1984. Summed area tables for texture mapping. *SIGGRAPH Comput. Graph. 18*, 4.

FEIBUSH, E., LEVOY, M., AND COOK, R. 1980. Synthetic texturing using digital filters. *SIGGRAPH Comput. Graph. 14*, 3 (July).

GLASSNER, A. 1986. Adaptive precision in texture mapping. *SIGGRAPH Comput. Graph. 20*, 4 (Aug.), 297–306.

SCHACHTER, B. 1980. Long crested wave models. *Comput. Vision Graph. Image Process. 12*, 187–201.

SCHILLING, A., KNITTEL, G., AND STRASSER, W. 1996. Texram: a smart memory for texturing. *IEEE Comput. Graph. Appl. 16*, 3 (May), 32–41.

WILLIAMS, L. 1983. Pyramidal parametrics. *SIGGRAPH Comput. Graph. 17*, 3.

Received: August 1999; revised: January 2000; accepted: July 2000