to appear in IEEE Visualization'99 conference proceedings

Forward Image Warping

Baoquan Chen, Frank Dachille and Arie Kaufman*

Center for Visual Computing
and Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400

Abstract

We present a new forward image warping algorithm, which speeds up perspective warping – as in texture mapping. It processes the source image in a special scanline order instead of the normal raster scanline order. This special scanline has the property of preserving parallelism when projecting to the target image. The algorithm reduces the complexity of perspective-correct image warping by eliminating the division per pixel and replacing it with a division per scanline. The method also corrects the perspective distortion in Gouraud shading with negligible overhead. Furthermore, the special scanline order is suitable for antialiasing using a more accurate antialiasing conic filter, with minimum additional cost. The algorithm is highlighted by incremental calculations and optimized memory bandwidth by reading each source pixel only once, suggesting a potential hardware implementation.

CR Categories: I.3.1 [Computer Graphics]: Hardware Architecture – Graphics Processors, Raster Display Devices; I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Image Processing]: Enhancement – Geometric correction, filtering;

Keywords: image warping, forward mapping, texture mapping, antialiasing, anisotropic filtering, Gouraud shading, hardware

1 Introduction

Image warping [13] deals with the geometric transformation between two images, a source image and a target image. The geometric transformation defines the relationship between source pixels and target pixels. Among its practical applications in medical imaging, remote sensing and computer vision, image warping has played an important role in computer graphics, such as in texture mapping, image morphing, image based rendering [2], plenoptic modeling [9], light field rendering [8], and lumigraph [5]. In Talisman [12], image layers are manipulated based on the updated viewing parameters to create new scenes. Efficiency and high quality are equally critical issues in these applications and are the foci of this paper.

Distinguished by the data flow of the transformation, image warping methods are classified as forward or backward warping. In forward warping, the source pixels are processed in scanline order and the results are projected onto the target image; while in backward warping, the target pixels in raster order are inversely mapped to the source image and sampled accordingly. Most existing algorithms are backward warping.

Compared with affine transformations (translation, rotation, scaling, shearing, etc.), a perspective transformation is considered more expensive and challenging. Smith [11] has proven that at least one division per pixel is required for perspective transformation due to its non-linearity. Almost all one-pass incremental algorithms follow Smith's method. Division is expensive in either software or hardware implementations. For example, on a Pentium Pro 180MHz, one division costs the same as seven multiplications or nine additions. Other research has been conducted on decomposing the perspective transformation into several simpler transformations. For example, Catmull and Smith [1] have proposed a decomposition of 2D perspective mapping into two orthogonal passes of 1D resampling operations. Gangnet et al. [4] have presented another decomposition of perspective warping into a concatenation of "rotation-homology-rotation." The primary inherent problem of a multi-pass algorithm is that the combination of two 1D filtering operations is not as flexible as true 2D filtering. Furthermore, multiple passes introduce additional filterings which degrade image quality. The algorithm introduced in this paper is a one-pass forward warping algorithm which can be implemented with nearly the same efficiency as affine transformations. The costly divisions are reduced to only one per scanline, compared to the usual one per pixel.

Image quality is another important issue in image warping. For perspective warping, aliasing due to the perspective foreshortening is the main reason for image quality degeneration. Antialiasing [3, 4] is expensive because of the variation in source pixel contribution regions to target pixels. As pointed out by Greene and Heckbert [6], circular pixels in the target image correspond to conic areas, called footprints, in the source image. Unlike affine image warping, the size

*Email:{baoquan,dachille,ari}@cs.sunysb.edu

and the orientation of the footprint varies from pixel to pixel. Greene and Heckbert [6] have described a method in which the Jacobian of the pixel is calculated and treated as two basis vectors in the texture image; the shape of the footprint is then approximated by a locally affine transformation, resulting in an elliptical footprint instead of a conic one. Schilling et al. [10] have pointed out that the computational expense of finding the two main directions of the ellipse are too high for real-time operation, so they approximated them.

We propose a new forward warping algorithm, that uses a scanline approach to perform perspective warping. Instead of scanning in normal raster scanline order, the algorithm is processed in a special scanline direction in the source image. This direction has the property that parallel scanlines in the source image remain parallel in the target image.

There are several advantages to our algorithm by scanning in the special direction. It

- reduces the complexity of perspective-correct image warping by eliminating the division per pixel and replacing it with a division per scanline, thus theoretically making perspective warping costs nearly the same as parallel warping;
- performs accurate antialiasing by incorporating anisotropic filtering without significant additional cost;
- corrects flaws in Gouraud shading caused by bilinear interpolation;
- optimizes the memory bandwidth by reading each source pixel exactly once. It also regularizes the memory access pattern which simplifies potential hardware implementation.

The remainder of the paper is organized as follows. We discuss our forward image warping algorithm in Section 2. Gouraud shading correction using our special scanline is then presented in Section 3, high-quality image warping in Section 4, and results and discussion in Section 5.

2 Forward Warping Algorithm

Our forward warping algorithm is performed in two stages: (1) calculating the special scanline direction, and (2) forward mapping the source image to the target image along the special scanlines, incrementally within each scanline.

2.1 Algorithm Overview

Our algorithm is a forward warping algorithm. The advantage of forward mapping is that it can process in source image order, and every source pixel is read only once. In the previous forward warping algorithm [13], the source pixel is irregularly projected to the target image due to the nonlinearity of perspective warping. Thus, an entire image size

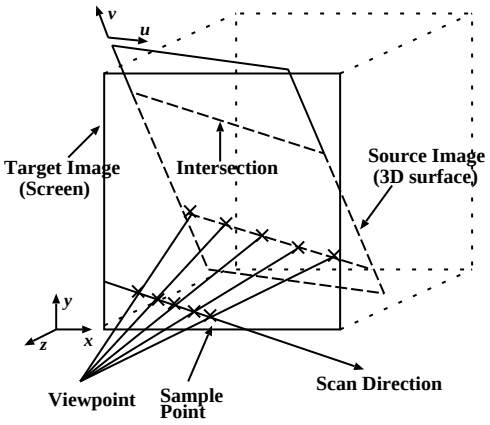

Figure 3: *Special scan direction for perspective projection*

accumulation buffer is needed to accumulate all the contributions to every target pixel. Another disadvantage of this method is the random access to target pixels and the multiple read and write of target pixels. Unlike the previous method that processes in source raster scanline order, our forward warping algorithm processes in a special scanline direction and overcomes the shortcomings of the previous method. This scanline direction has the property that parallel scanlines in the source image remain parallel in the target image, and the equi-distant sample points along a source scanline remain equi-distant in the target scanline.

The intuition of this special scanline direction comes from projection geometry as shown in Figure 3. The source image is placed on a 3D planar surface and the target image is placed on the screen. As in typical texture mapping, to obtain the pixel on screen, a ray is cast from the viewpoint to 3D space and intersected with the screen and 3D surface. The intersection points are the sample points. Now, when the scan direction in screen space is parallel to 3D planar surface, the scanlines in both images are parallel to each other, and equi-distant sample points along the scanline remain equi-distant in the 3D surface plane. In the following, we call this special parallel-preserving scanline the PP scanline. From projection geometry, we know that this PP direction exists and is unique for a given perspective transformation, because this special PP scanline is the intersection line between the screen and the 3D planar surface (see Figure 3). We can extend this intuition directly to 2D image warping. Notice that for parallel projection, any direction preserves this parallelism on both images and thus, a raster scanline direction can be used due to its simplicity.

Figure 2 shows the PP scanlines (red lines) in both images. Once we have the parallelism property, the pixel access becomes regular and spatial coherency can be utilized in both images. Furthermore, the PP scanline allows us to apply a pure incremental algorithm without division to each scanline for calculating the projection of source samples. Notice, however, that one division is still needed for the two end-

points of every scanline due to the non-linear projection.

As we scan in the PP scanline direction rather than the raster direction, sample points on the target scanline do not necessarily coincide with the target pixels. However, we can align the sample points on the x grid lines of the target image (see Figure 2), thus the sample points are only off the y grid lines (obviously, they are equi-distant along the scanline). Placing the sample value in the nearest-neighbor target pixel is then a reasonable approximation, as a half pixel is the maximum error. Figure 3 shows the comparison between our method with the sample position approximation and the traditional raster scanline method, which samples on the exact grid points. The two images are practically indistinguishable. Below in Section 4.1 we present a variation of our algorithm that also samples on the exact grid points.

In general, a reduction in the number of divisions from $O(n^2)$ to $O(n)$ is obtained by our algorithm (n is the linear resolution). For our algorithm, only two additions are needed to calculate each sample point, while three additions, one division and two multiplications per pixel are required for each pixel in the traditional raster scanline algorithm. Since fixed point calculation is usually faster than floating point calculations, we assume all calculations are in fixed point. Based on our observations on a Pentium Pro 180MHz that one division costs the same as seven multiplications or nine additions, the comparison of the calculations per pixel of the two methods is approximately between two and 14 additions, which shows that our algorithm is theoretically seven times faster than the traditional algorithm. Using a similar analysis on an R10000 workstation our method is about theoretically four times faster than the traditional one, but practically, we get three times improved performance. We want to point out that in many graphics applications such as computer games, when speed is the dominant factor, our algorithm is extremely useful. Our algorithms, however, also caters to higher quality applications and offers additional advantages as discussed below.

2.2 Calculate the PP scanline

As described above, the PP scanline is the intersection line between the 3D planar surface and the screen. However in a 2D problem, the PP scanline must be calculated based on a 2D matrix. In general, a perspective transformation can be presented as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

where (u, v) is the coordinate of the source pixel, (x, y) is the coordinate of the target pixel, and M is the perspective transformation matrix. The (u, v) coordinate can be expressed in terms of (x, y) as

$$(uv) = F(x, y) = C \begin{bmatrix} ax + dy + g \\ bx + ey + h \end{bmatrix} \quad (2)$$

where

$$C = \frac{1}{(cx + fy + 1)}$$

A line in the target image can be expressed as $y = kx + B$, where slope k denotes a line direction. To calculate k for the special PP scanline, we first define two parallel lines with identical slope k and intercept B of 0 and 1, represented by point pairs of $(0, 0), (1, k)$ and $(0, 1), (1, k + 1)$, respectively. Then, we calculate the coordinates of these points in the source image. As perspective transformation preserves straight lines, these two lines are still straight lines and their slopes can be calculated from two point pairs. Assuming that the slopes of the two mapped lines are equal, we have an equation in k. Solving this equation, we get k as,

$$k = -\frac{c}{f} \quad (3)$$

The corresponding slope in the source image is then

$$k' = \frac{bf - ec}{af - dc} \quad (4)$$

Note that when $k = -c/f$, the denominator of the homogenous coordinates becomes a constant value of $Bf + 1$, where B is the intercept in $y = kx + B$. We further analyze k in Section 4.2, and discuss additional advantages in antialiasing of the PP scanline.

2.3 Scanline Processing

The algorithm sweeps the scanlines through the source image. The scanlines have the slope k'. The samples along each scanline are incrementally calculated. First, for each scanline, we calculate the projection of the endpoints from the target image onto the source image, and then based on the number of sample points on the scanline, increments are calculated in both x and y directions.

Considering the traditional bilinear interpolation of samples in the source image, every sample needs the contribution of four surrounding source pixels. If pixels are read every time for every sample, each pixel ought to be read four times. This leads to a memory bandwidth of four times the target image size. Since all the scanlines are parallel, samples on neighboring scanlines usually share contributing source pixels. Thus, we can buffer the read pixels so that common pixels are read from the buffer instead of from the image.

Indeed, the pixels are read in a fixed pattern, called the pixel read template, calculated based on the Bresenham algorithm (Figure 5a). The binary digits at the bottom of Figure 5a are one way of encoding this template. This code indicates the increase in v direction, 0 means no increase and 1 denotes an increase by 1, while u is always increased by 1.

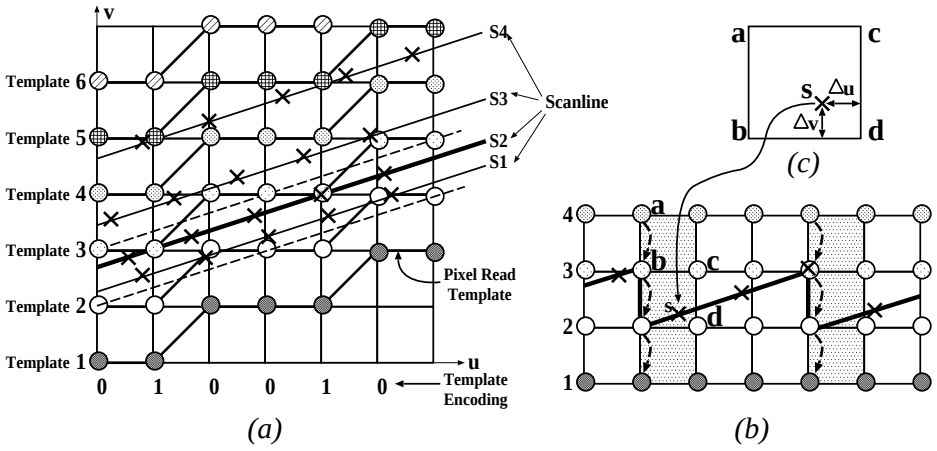

Figure 5: *Scanline processing: (a) Pixel read templates in the source image, (b) pixels are read into buffers as the scanline sweeps through the source image, four scanline buffers are needed to perform sampling for one output scanline, (c) bilinear interpolation of samples in the shaded region.*

In this case, we call axis u the primary processing axis. The template always starts from the left-most pixel and moves in the vertical direction so that all pixels are read and placed into the buffer for subsequent use in the sampling. How much buffering do we need and how to address the samples in the buffer? We can see from Figure 5a that in order to provide pixels for sampling on any scanline between the two dotted lines, four pixel templates are needed, even though for a specific scanline, only three pixel templates might be sufficient (for example, only templates 2, 3 and 4 are necessary for the current scanline $S2$). Thus, the buffer size is four scanlines.

Figure 5b shows the addressing of samples in the buffer. Whenever the template code value is 1, the sample decreases by 1 in v. The thick zig-zag line represents the output scanline in the buffer. When the sample falls in the shaded region, in which the pixels in the buffer are sheared, care must be taken to read the correct pixels for sampling. Figure 5c shows how to bilinearly interpolate one of the samples, s, in this region.

The contents of the buffer are updated based on the scanline position. For the example in Figure 5a, templates 1, 2, 3, and 4 are in the buffer when processing scanline $S1$. For $S2$, the buffer remains the same. For $S3$, template 5 is read into the buffer, and template 1 is discarded. Template 6 replaces template 2 for scanline $S4$, and so on.

3 Gouraud Shading Correction

Gouraud shading is a popular intensity interpolation algorithm used to shade polygonal surfaces. Given color only at the vertices, Gouraud shading bilinearly interpolates the intensities for the entire rasterization of a polygon in a raster scanline order. Flaws of this approach have been pointed out and a solution of subdivisions have been analyzed [13]. Figure 5a shows a rectangle with a top-left red vertex, a bottom-right green vertex, and the other two vertices having the same color of half yellow, $(0.5, 0.5, 0)$ in RGB. Thus, the diagonal line connecting the top-right and bottom-left is denoted by the color of half yellow. However, when these four points are perspectively projected onto the screen, shown in Figure 5b, Gouraud shading converts this diagonal line into a curve, which violates the property of preserving lines in perspective transformation.

Our special scan direction fixes the perspective distortion in Gouraud shading. The perspective distortion is present because the linear interpolation along a raster in screen space is generally non-linear when transformed into polygonal coordinates. With the special scan direction, however, linearity *is* preserved by the mapping. Thus, interpolation is linear in both image and polygonal space – fixing the distortion of Gouraud shading. Note that interpolation along the edges is still non-linear, so the scanline endpoints must be transformed into polygonal space for correct interpolation. The result of shading using our algorithm is shown in Figure 5c in contrast to Gouraud shading in raster order (Figure 5b).

4 High Quality Image Warping

Our forward mapping algorithm with nearest-neighbor approximation generates a target image that is practically indistinguishable from an image generated with tradition methods. However, when a higher image quality is desired, our algorithm can calculate the pixel value at the exact grid points. A simple scheme is introduced to perform this correction. Our algorithm can further improve on image quality by antialiasing; the PP scanline promises a cheaper and higher-quality method of antialiasing.

4.1 Target Pixel Correction

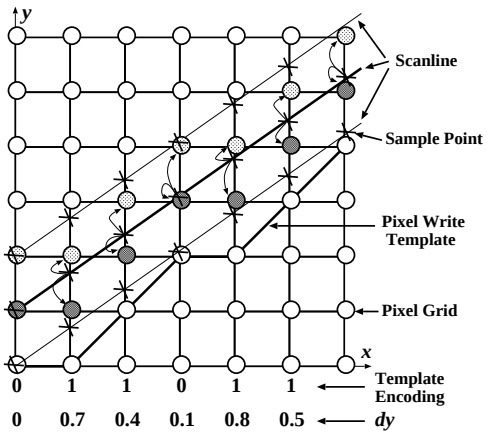

Figure 7: *Linear interpolation on samples to obtain pixels*

The sample points in the target image are aligned on integer x coordinates. In order to obtain the pixel value at the exact pixel grid locations, we need to linearly interpolate the two samples immediately above and below every pixel. Performing this linear interpolation simply as a second pass may increase the cost, since we have to read the samples all over again. Instead, as each sample is generated, we spread its contribution to the upper and lower pixels with no intermediate buffering.

In Figure 6, the samples on the thick inclined scanline contribute to the shaded pixels neighboring them. The arrows show that each sample is contributing to two pixels. We cannot write out a pixel until both contributions are collected, so we need one scanline buffer to store the intermediate pixel values.

To write out pixels correctly and efficiently, a pixel write pattern, called the pixel write template is pre-calculated. Unlike the pixel read template, this template is calculated by truncating the y coordinate value of samples along a scanline. The template is encoded as a series of integer y steps and fractional distances dy from the true scanline. The weights used for the final linear interpolation are dy and 1-dy for the upper and lower pixels, respectively. Since all scanlines are one unit apart in the vertical direction, the template is calculated only once per projection.

4.2 Antialiasing

As shown in Figure 2, the sample points on the upper scanlines are sparser than on the lower scanlines, resulting in a transition from under-sampling to normal sampling. Thus, an appropriate resampling filter must be used to avoid aliasing on the upper scanlines. Isotropic filtering results in clearly incorrect and blurry images. The need for anisotropic filters has been addressed in [7] and recently in [10]. Each filter is defined by its footprint and it profile. Taking a target sample as a circle, its projection in the source image is its footprint. In general, this footprint should be neither circular (isotropic) nor squarish (as in mip-mapping), but conic in shape (see Figure 7). The profile of the filter decides the weights of the contributing pixels within the footprint. Although the sinc filter is optimal, we choose to use a gaussian filter because of its finite footprint and good low-pass characteristics. Our new perspective warping algorithm is very suitable for antialiasing; it offers more accuracy in calculating the anisotropic footprint, producing higher image quality at a lower cost.

4.2.1 Footprint Calculation

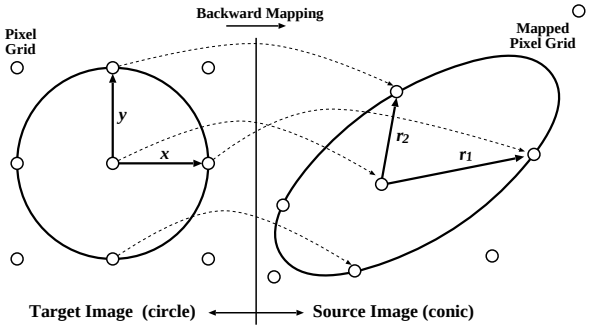

Figure 8: *Footprint geometry*

In previous methods, the main axes of the ellipse have to be calculated for every pixel [6], and even approximations have been proposed [10]. Still, this remains an expensive computation and no incremental method is available. To obtain the major axes of the ellipse, the Jacobian must be calculated. Here we present a method in which no Jacobian needs to be calculated. First we analyze the properties of the Jacobian.

The generalized backward mapping from an xy target image into a uv source image is defined in Equation 2. The Jacobian for the generalized transformation is a non-linear function of x and y,

$$J = C^2 \begin{bmatrix} y(af-cd)+a-gc & x(af-cd)-d+gf \\ y(bf-ce)+b-hc & x(bf-ce)-e+hf \end{bmatrix} \quad (5)$$

The Jacobian is used to determine the footprint of each pixel in the source image and is necessary for anisotropic filtering. The differences between screen pixels in xy raster space are projected into the source image by computing the directional derivatives in the (1,0) and (0,1) directions. These derivatives in source image space are called r_1 and r_2:

$$r_1 = J \begin{bmatrix} 1 \\ 0 \end{bmatrix} = C^2 \begin{bmatrix} y(af-cd)+a-gc \\ y(bf-ce)+b-hc \end{bmatrix} \quad (6)$$

and

$$r_2 = J \begin{bmatrix} 0 \\ 1 \end{bmatrix} = C^2 \begin{bmatrix} x(af - cd) - d + gf \\ x(bf - ce) - e + hf \end{bmatrix} \quad (7)$$

These vectors define the bounding box of an ellipse that approximates the footprint. Typically, these are calculated for every pixel, when needed, for methods of anisotropic filtering (e.g., EWA, footprint assembly). This requires one more division per pixel for calculating C.

We propose a more accurate method to determine the footprint. Because the Jacobian is a linear approximation of the non-linear mapping, it is more accurate to compute the footprint by taking the distances to neighboring samples in source image space. Because the projections of neighboring samples are already computed, this method requires no extra division.

The PP scan direction provides for greater coherency and no division to compute the Jacobian. For each pixel in the PP scanning order, the footprint is defined by r'_1 and r'_2. The directional derivative r'_1 in direction $[1, k]$ along the PP scanline is:

$$r'_1 = \nabla_{[1,k]} F = J \begin{bmatrix} 1 \\ k \end{bmatrix} = C^2 \begin{bmatrix} af - cd \\ bf - ce \end{bmatrix} \quad (8)$$

and since $y = kx + B$, $C = 1/(Bf + 1)$ is constant for every PP scanline, and thus, r'_1 is constant for every PP scanline. We exploit this fact in order to increment the source image coordinates along a scanline, with no divisions. The value of the directional derivative r'_2 in the y direction $[0, 1]$ is:

$$r'_2 = \nabla_{[0,1]} F = r_2 \quad (9)$$

r'_2 varies linearly along the scanline since it is a function of x, so it can be incremented along the scanline. The special scan direction makes it possible to compute the source image coordinates and pixel footprints simply and efficiently.

4.2.2 Filtering

Now that we can efficiently compute all the footprint and source pixel coordinate information, we can perform correct anisotropic filtering using a standard method, such as Greene and Heckbert's elliptical weighted average (EWA) [6] or Shilling et al.'s footprint assembly [10]. However, as pointed out before, even the elliptical footprint approximation is inaccurate. Furthermore, such methods result in redundant sampling – accessing each source pixel multiple times. For a circular filter region with a footprint radius of 1.0 source pixel, each source pixel is sampled an average of π times. By a forward mapping technique, we can eliminate redundant memory access and lower the memory bandwidth by a factor of π. Thus, we adopt a forward mapping technique in which we read once all the source pixels in pixel read template order and *splat* them onto the target image with a filter kernel.

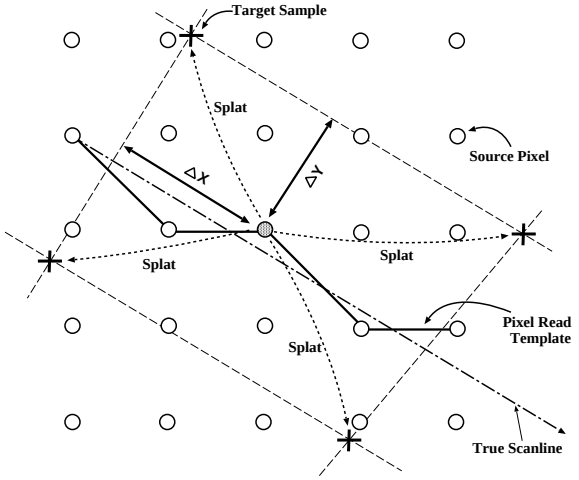

Figure 9: *Splatting source pixels onto the target samples*

As shown in Figure 8, each source pixel has a Δx,Δy relative to each of its nearest-neighbor target samples. The Δx can be computed incrementally since all samples along a scanline are equi-distant. The special scan direction guarantees that the Δy is constant along each scanline. Although the raster grid locations deviate from the true scanline, the actual distances can be estimated by adding a small correction which is stored in the template and is uniform among all scanlines. The filter kernel is pre-computed once and stored in a lookup table, and subsequently the contribution of each source pixel is indexed by its Δx and Δy into the lookup table for the four (or more) nearest-neighbor target samples. The number of target samples depends upon the footprint of the filter used and it varies from four to 16 samples. Using this method, each source pixel is read exactly once from memory, then four (or more) times modulated by a lookup table entry and accumulated in the target pixel. This way, the final pixel value is the weighted average of the nearby source pixels. This weighted average requires a division by the sum of the filter weights to normalize each final pixel intensity. This division is the major price of anisotropic filtering.

5 Results and Discussion

A straightforward implementation of our perspective warping shows nearly the same speed as affine warping, both without antialiasing. On an R5000 CPU, we can achieve 30Hz warping of a color image with a resolution of 300^2. This includes the time for calculating the warping matrices when changing the viewpoint.

In particular, we have performed experiments on a checkerboard image which is a widely used benchmark image for image warping. Greene and Heckbert's EWA method

is a competitive antialiasing method for perspective warping against which we compare our method. Figure 9a shows the EWA antialiasing method with a filter radius of 1.5 pixels in the target image. Aliasing is noticeable in the upper-left corner. A larger filter kernel with a filter radius of 2.5 pixels is suggested by Greene and Heckbert, which they call "higher quality EWA." The result of this method, depicted in Figure 9b, shows that aliasing is gone, but the image becomes blurry. When our method uses a filter radius of 1.5 pixels, better antialiasing is obtained with less blurring. The reason is that our method approximates the conic footprint better than EWA's elliptical approximation, so that even a small filter kernel offers better antialiasing. For EWA, since the filter kernel shape is not exact, there is a tradeoff between blurring and antialiasing.

Furthermore, our method is more efficient. Our results show that our method is three times faster than standard EWA and 10 times faster than 'higher quality EWA' on an R10000 workstation. There are three reasons for this result. First, our method reads each source pixel only once while EWA re-reads pixels in overlapping footprints. Second, the Jacobian is not calculated per pixel in our method, but per scanline. Third, our method considers for each output pixel a tight, oriented quadrilateral of the source image, while EWA considers an axis-aligned bounding rectangle.

A straightforward hardware implementation of our algorithm offers several advantages. Traditionally, four source pixels are read for each target pixel – making memory bandwidth a major performance bottleneck. Our algorithm boasts one-quarter the memory bandwidth of the traditional algorithm, offering obvious cost and performance savings. Also, the per-pixel computation is lower which simplifies the computational hardware. The properties of incremental calculation, memory access with spatial coherence, and the reduced memory bandwidth suggest a suitability of our algorithm for efficient hardware implementation.

6 Summary and Future Work

In this paper we have presented a new forward image warping algorithm which speeds up perspective warping and improves image quality. By choosing a special scan direction, this algorithm substantially reduces the number of divisions inherent in perspective warping. The cost of perspective warping is reduced to approximately the same as parallel warping due to this simplification. A straightforward application of this special scanline algorithm produces correct Gouraud shading. Furthermore, while antialiasing of perspective warping is usually considered an expensive process, our algorithm performs high-quality antialiasing with minimum cost. By guaranteeing one access per source pixel, the bandwidth of accessing the source image is reduced to a minimum.

One problem with our method is the low-pass filtering in one dimension. Although the samples are precise along the inclined scanline, the resampling necessary to align it to the raster grid low-pass filters each column (or row, depending on the primary processing axis) independently, resulting in some artifacts.

In future work, we will study the use of an image pyramid to reduce the complexity of antialiasing. Currently, the work in antialiasing is on the order of the visible source image. We can trade anisotropy for speed by sampling pre-filtered images along each scanline. The number of samples and scanlines can be continuously varied to adjust the degree of anisotropy for the desired rendering rate.

References

[1] E. Catmull and A. R. Smith. 3-D transformations of images in scanline order. *Computer Graphics (SIGGRAPH 80)*, pages 279–285, July 1980.

[2] S. E. Chen. Quicktime VR - an image-based approach to virtual environment navigation. *Computer Graphics (SIGGRAPH 95)*, pages 29–38, August 1995.

[3] E. A. Feibush, M. Levoy, and R. L. Cook. Synthetic texturing using digital filters. *Computer Graphics (SIGGRAPH 80)*, pages 294–301, July 1980.

[4] M. Gangnet, D. Perny, and P. Coueignoux. Perspective mapping of planar textures. *Computers and Graphics*, 8(2):115–123, 1984.

[5] S. J. Gortler, R. G., R. Szeliski, and M. F. Cohen. The lumigraph. *Computer Graphics (SIGGRAPH 96)*, pages 43–54, August 1996.

[6] N. Greene and P. S. Heckbert. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6):21–27, June 1986.

[7] P. S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.

[8] M. Levoy and P. Hanrahan. Light field rendering. *Computer Graphics (SIGGRAPH 96)*, pages 31–42, August 1996.

[9] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics (SIGGRAPH 95)*, pages 39–46, August 1995.

[10] A. Schilling, G. Knittel, and W. Strasser. Texram: Smart memory for texturing. *IEEE Computer Graphics and Applications*, 16(3):32–41, May 1996.

[11] A. R. Smith. Incremental rendering of textures in perspective. July 1980.

[12] J. Torborg and J. Kajiya. Talisman: Commodity Real-time 3D graphics for the PC. *Computer Graphics (SIGGRAPH 96)*, pages 353–364, August 1996.

[13] G. Wolberg, editor. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.