A FAST ALGORITHM FOR GENERAL RASTER ROTATION

Alan W. Paeth

Computer Graphics Laboratory, Department of Computer Science
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
Tel: (519) 888-4534, E-Mail: AWPaeth%watCGL@Waterloo.CSNet

ABSTRACT

The rotation of a digitized raster by an arbitrary angle is an essential function for many raster manipulation systems. We derive and implement a particularly fast algorithm which rotates (with scaling invariance) rasters arbitrarily; skewing and translation of the raster is also made possible by the implementation. This operation is conceptually simple, and is a good candidate for inclusion in digital paint or other interactive systems, where near real-time performance is required.

RÉSUMÉ

La rotation d'un "raster" d'un angle arbitraire est une fonction essentielle de plusieurs logiciels de manipulation de "raster". Nous avons implémenté un algorithme rapide de rotation de "raster" conservant l'échelle de l'image. Nous d'écrivons ce système qui permet aussi le biaisage et la translation du "raster". Cette opération, d'un concept simple, se révèle un bon candidat à l'insertion dans un logiciel de "paint system" (ou autre système interactif) où une performance quasi-temps réel est nécessaire.

Keywords: *raster rotation, frame buffer, real-time.*

INTRODUCTION

We derive a high-speed raster rotation algorithm based on the decomposition of a 2-D rotation matrix into the product of three shear matrices. Raster shearing is done on a scan-line basis, and is particularly efficient. A useful shearing approximation is averaging adjacent pixels, where the blending ratios remain constant for each scan-line. Taken together, our technique rotates (with anti-aliasing) rasters faster than previous methods. The general derivation of rotation also sheds light on two common techniques: small angle rotation using a two-pass algorithm, and three-pass 90-degree rotation. We also provide a comparative analysis of Catmull and Smith's method [[Catm80]] and a discussion of implementation strategies on frame buffer hardware.

STATEMENT OF THE PROBLEM

A general 2D counter-clockwise rotation of the point (x,y) onto (x',y') by angle theta is performed by multiplying the point vector (x,y) by the rotation matrix:

$$M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

The matrix is orthogonal: it is symmetric, rows and columns are unit vectors, and the determinant is one. To rotate a raster image, we consider mapping the unit cell with center at location (i,j) onto a new location (i',j').

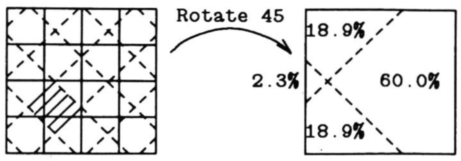

Figure 1. Rotation by Raster Sampling

The image of the input cell on the output grid is a cell with (usually) a non-integral center, and with a rotation angle theta (θ). We adopt a "box-filter" sampling criterion, so the value of the output pixel is the sum of the intensities of the covered pixels, with each contributing pixel's intensity weighted in direct proportion to its coverage (Figure 1). Note that the output pixel may take intensities from as many as six input pixels. Worse, the output pixel coverage of adjacent input pixels is non-periodic; this is directly related to the presence of irrational values in the rotation matrix. Clearly, the direct mapping of a raster by a general 2x2 matrix is computationally difficult: many intersection tests result, usually with no coherence or periodicity to speed program loops.

ROTATION THROUGH SHEARING

Now consider the simplest 2x2 matrices which may operate on a raster. These are shear matrices:

$$\text{X-shear} = \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix} \qquad \text{Y-shear} = \begin{bmatrix} 1 & 0 \\ \beta & 1 \end{bmatrix}$$

Shear matrices closely resemble the identity matrix: both have a determinant of one. They share no other properties with orthogonal matrices. To build more general matrices, we form products of shear **matrices — these correspond to a sequence of shear operations on the raster. Intuitively, consecutive** shearing along the same axis produces a conforming shear. This follows directly:

$$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & a' \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & a+a' \\ 0 & 1 \end{bmatrix}$$

Thus, shear products may be restricted to products of alternating x and y shears, without loss of generality. The product of three shears gives rise to a general 2x2 matrix in which three arbitrary elements may be specified. The fourth element will take on a value which insures that the determinant of the matrix remains one. This "falls out" because the determinant of the product is the product of the determinants (which are always one for each shear matrix). Orthogonal 2x2 matrices also have unit determinant, and may thus be decomposed into a product of no more than three shears.

$$\begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta & 1 \end{bmatrix} \begin{bmatrix} 1 & \gamma \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Solving the general equation, we have $\alpha=\gamma=1-\cos\theta/\sin\theta$; $\beta=\sin\theta$. The first equation is numerically unstable near zero, but can be replaced by the half-angle identity: $\alpha=\gamma=-\tan(\theta/2)$. Program code to shear and update the point (x,y) with (x',y') is then:

```
/* X Shear */              /* Y Shear */
x' := x - sinθ * y;        x' := x;
y' := y;                   y' := y + tan(θ/2) * x;
```

When the output vector replaces the input, x≡x' and y≡y', so the second line of the sequence may be optimized out. Consecutive shears yield sequential program steps. Thus, a three-shear rotation is achieved by the three program statements:

```
x := x + α * y;      [1]
y := y + β * x;      [2]
x := x + α * y;      [3]
```

With $\theta \simeq 0$, Cohen [Newm79] uses steps [1] and [2] to generate circles by plotting points incrementally. His derivation begins be choosing α and β to approximate the conventional rotation matrix, and then points out that by reassigning x+α*y to the original variable x in [1], and not to a temporary value x', the **determinant becomes one, and the circle eventually closes. Our analysis demonstrates formally why this is true:** rewriting the variables constitutes a shear, and the sequence of shears always maintains a determinant of one. Augmenting the code with line [3] would convert the two-axis shear into a true rotation: the circle generator would then produce points rotated through a constant angle relative to the preceding point. This is important should the algorithm be used to produce circle approximations as n-gons (and not point drawings), where $\theta=360/n$ is no longer small.

RASTER SHEARING

Raster shearing differs from point transformation because we must consider the area of the unit cell which represents each pixel. Fortunately, the shear operation modifies the pixel location with respect to only one axis, so the shear can be represented by skewing pixels along each scan-line. This simplifies the intersection testing that must go on to recompute the intensity of each new output pixel.

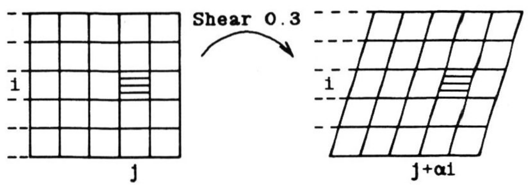

Figure 2. Raster Shearing Along the X Axis

In general, the unit square P(i,j) on row i is rewritten as a unit parallelogram with side of slope $1/\alpha$ on row i, with the former displaced by αy pixel widths. This displacement is not usually integral, but remains invariant for all pixels on the ith scan-line. For illustration and implementation, it is represented as the sum of an integral and a fractional part ("f" in Figure 3; "skewf" in Figure 6). Those pixels covered by this parallelogram are written with fractional intensities proportional to their coverage by the parallelogram. The sum of all these pixels must equal the value of the original input pixel, as they represent this input pixel after shearing.

We next approximate this parallelogram of unit area with a unit square. Placing the edges of the square through the midpoints of the parallelogram, we produce an exact approximation when the parallelogram covers two pixels, but not when it covers three. This approximation is the basis for our rotation algorithm. As we shall see, it can be implemented as a very efficient inner-most pixel blending loop, thus offsetting the cost of making three shearing passes, as compared to previous techniques, which employ two less efficient (though more general) passes.

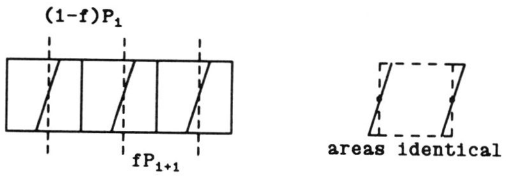

Figure 3. The Parallelogram Approximation

Based on this filtering strategy, we consider two approaches to rotation. First, we seek angles θ for which the filtering is exact. Second, we analyze the filter for arbitrary values of θ where the filter may not be exact.

Filtering is exact when all parallelograms overlap no more than two pixels. This will always occur when the shear offset is of length $1/n$, as a periodic cycle of n parallelograms results, in which each spans exactly two pixels. Choosing this ideal filter for the first and third passes, we derive the second pass shear value. Setting $\alpha=1/n$, we have $\theta=2\cdot\tan^{-1}(1/n)$ and thus by manipulation of inverse trigonometric functions, $\beta=2n/1+n^2$. Setting n=1 yields $\alpha=-1$, $\beta=1$ and thus rotations with $\theta=90$ degrees are exact (not possible with the Catmull-Smith approach). Moreover, this shearing matrix never generates fractional values (implying no anti-aliasing must take place), so 90 degree rotation may be coded as a three pass pixel "shuffle".

Other choices of n yield exact sampling in the two α passes, as $\alpha=1/n$. Here $\beta=2n/1+n^2$ (in the middle pass) will never be of the form $1/m$, so some filtering artifacts will be present. However, we can form small rational values for α and β corresponding to various angular rotations, and create specialized filters, in which only the β pass generates small errors on a periodic basis. When α and β are small rationals of the form $1/j$, then the shear values (which are used as blending coefficients by our algorithm) will recur every j scan-lines, leading to customized algorithms. Solving for general rational values of α and β, we find that $\alpha=1/j$ and $\beta=2ij/i^2+j^2$. These tabulated values give rise to highly efficient filters, with approximation errors minimized:

α	β	θ
−1	1	90.00
−3/4	24/25	73.74
−2/3	12/13	67.38
−1/2	4/5	53.13
−1/3	3/5	36.87
−1/4	8/17	28.07
−1/5	5/13	22.62

Figure 4. Rotation by a Rational Shear

We now consider arbitrary choices of θ, and then the precision of the rotation. For $\theta>90$ degrees, our shear parallelogram may span four pixels, and the filtering rapidly breaks down. Based on the four-fold symmetry of a raster, we may restrict our attention to rotations of no more than 45 degrees, where our approximation has worst-case performance (because α and β grow monotonically with $0\le\theta<90$ degrees). Here $\alpha=1-\sqrt{2}\approx-.4142$; and $\beta=\sqrt{2}/2\approx.7071$. The second β pass is the most error-prone.

Probabilistically, its filter is exact 29.3% of the time. Otherwise, the parallelogram spans three pixels, and the error, as a function of fractional pixel skew, grows quadratically to a cusp, reaching its worst-case error when the parallelogram is symmetric about the output pixel. This error is $\sqrt{2}/8$ or 17.7%. However, the sampling tile shifts as the shear value changes with each scan-line, so average degradation to the sheared raster is computed by integrating over parallelograms of all possible skew. Solving the equations, we find that the worst-case shear filter approximates intensities to within 4.2% of the actual intensity. For rotations less the 45 degrees, the approximation is even closer, as the the probability of the parallelogram spanning three pixels decreases. Where it does, the error terms are also smaller.

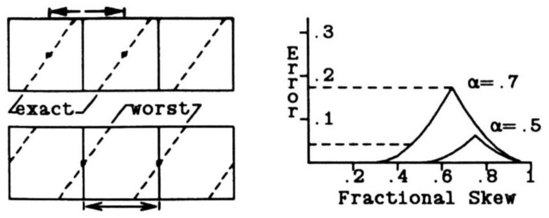

Figure 5. Approximation Error

The nature of the error is to concentrate intensities from a center pixel whereas the true box-filter approximation calls for contributing coverages from two neighboring pixels. Thus, the approach "peaks" the data: the nature of the degradation is not random. Further, a reasonable implementation of the filter guarantees that when any scan-line is skew-sheared by a fractional amount, the contributing intensities of each input pixel sum to 1.0 — the filter parallelograms never overlap. If we consider the sum of the pixel intensities along any scan-line, this sum remains unchanged after the shear operation. Thus, the algorithm produces no visible shifts in intensity, and introduces no "holes" during rotation. The only rotation artifacts discernible appear with high-frequency data (such as lines of single pixel width), and even then only after magnification. This property is shared generally with rotation and translation algorithms which must resample such "sharp" rasters onto non-integral pixel boundaries.

IMPLEMENTATION

Scan line shearing is approximated by a blending of adjacent pixels. In the following code segment, the "pixmult" function returns a pixel scaled by a value `skewf`, where $0 \le$ `skewf` < 1, is a constant parameter for all "width" passes through the inner-most loop:

```
PROCEDURE xshear(shear, width, height)
    FOR y := 0 TO height-1 DO
        skew  := shear * (y+0.5);
        skewi := floor(skew);
        skewf := frac(skew);
        oleft := 0;
        FOR x := 0 TO width-1 DO
            pixel := P(width-x, y);
            left  := pixmult(pixel, skewf);
                     /* pixel - left = right */
            pixel := pixel - left + oleft;
            P(width-x+skewi, y) := pixel;
            oleft := left;
        OD
        P(skewi, y) := oleft;
    OD
```

Figure 6. Shearing Algorithm for X Axis

This operation shears a raster of size (width, height) by the value present in "shear", so the data matrix `P` must be of sufficient width to accommodate the shifted output data. Note that only "width" output entries are written, so the skewed output line may be written to frame buffer memory modulo the frame buffer pixel width, thus requiring no additional memory, but complicating the specification of data to the three shear passes. A virtual frame buffer implementation which provided a notion of "margins" to active picture detail can maintain this offset information implicitly.

A shear operation always has an axis of shear invariance (it is an fact an eigenvector). In this implementation, the axis is the pixel boundary "below" the final row of pixel data at a distance "height". This gives rise to rotation about the interstices between pixel centers. To rotate rasters about pixel centers, the "0.5" half-pixel offset may be removed.

The code splits each pixel into a "left" and "right" value using one multiply per pixel; left and right always sum exactly to the original pixel value, regardless of machine rounding considerations. The output pixel is then the sum of the remainder of the left-hand pixel, plus the computed fractional value for the present (right-hand) pixel. The "pixmult" function reduces to a fractional multiply or table lookup operation with monochromatic images. More generally, it may operate on an aggregate pixel which might contain three color components, or an optional coverage factor [Port84]. Because read and write references to `P` occur at adjacent pixel locations during the course of the inner-most loop, pixel indexing can be greatly optimized.

On machines lacking hardware multiply, code to shear a large (512x512) image may build a multiply table at the beginning of each scan-line, and then use table lookup to multiply. By skew symmetry, x-shearing of line `-n` and line `n` are identical, save for shear direction, so one table may be used for two scan-lines, or for every 1024 pixels. With a pixel consisting of three 8-bit components, the table length is 256, and table fetches will exceed table loads by a factor of 12. Since the table can be built with one addition per (consecutive) entry, its amortized cost per lookup is low, and decreases linearly with raster size.

Framebuffers are beginning to incorporate integer multiply hardware, often targeted to pixel blending applications (The Adage/Ikonas frame buffers at Waterloo's Computer Graphics Laboratory provide a 16-bit integer multiply in hardware). This speeds the evaluation of the pixel blending; the majority of the inner-loop overhead is in (un)packing the 24-bit RGB pixel to provide suitable input for the multiplier. Fortunately, the addition used to complete the blend may be done as a 24-bit parallel add, because the values to be summed, "left" and "right", have been scaled by `frac` and `1-ffrac` respectively. Thus, the blending operation is "closed", and no carry can overflow from one pixel component into the next.

Finally, the shear code may more generally be used to introduce spatial translation of the raster. By introducing an output offset in the shear code, a "BitBlt" [[Inga78]] style operation may be included at no extra cost. In this setting, "skewi" and "skewf" would have integral and fractional offsets added to them to accommodate the lateral displacement of the raster. Displacement during data passes two and three provides arbitrary displacement on the plane, with orthogonal specification of the displacement parameters.

More generally, when the code is incorporated into a larger package which provides arbitrary (affine) matrix operations on a raster, the composite of all intermediate image transformations are represented in one matrix. This avoids unnecessary operations to the image. Eventually, this matrix is decomposed into three operations: scaling, rotation and shearing (plus an optional translation if a 3x3 homogeneous matrix is used). The shearing, rotation and possible translation operations may be gathered into one three-shear operation. The scale pass prefaces the rotation if it scales to a size larger than `1:1`, otherwise it follows the rotation. This maximizes image quality and minimizes data to the shear (and possibly rotate) routines. Other four pass scale/shear sequences are discussed in the literature [[Weim80]].

COMPARISONS

As with the Catmull-Smith approach, the algorithm may be implemented as a pipeline for real-time video transformation. Both approaches require two "rotators" to transpose the data entering and leaving the second scan-line operator, as this step requires data in column (and not row) order.

In that approach, two scan-line passes (by x, then by y) are made upon the input raster. These may be modeled by the matrix transformation:

$$\begin{bmatrix} 1 & 0 \\ \tan\theta & \cos 2\theta \sec\theta \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

These slightly more general matricies perform a simultaneous shear and scale along one axis, while leaving the second axis unchanged. This approach saves one data pass, but incurs the penality of more complex scan-line sampling.

Because sample pixels are both sheared and scaled, no pixel-to-pixel coherence of fractional sampling location exists. Thus, each pixel must be sampled at two fractional locations, doubling the number of pixel (aggregate RGB) multiplies for each pass. Hand analysis of our microcode showed that this is already the dominant operation in the pixel loop. Further, the Catmull-Smith approach must additionally recompute the fractional sample points for each next pixel, or approximate their location using fixed-point arithmetic. In our implementation, fractional sampling points are constant per scan-line, and are calculated exactly in floating point at the beginning of each line.

Compared generally to other work, our algorithm finds application where a generalized "BitBlt" operation is needed to perform rotation and translation efficiently. More complex pixel sampling passes may justify their added expense in allowing for generalize rotation operations, such as Krieger's modified two-pass approach [[Krie84]] used to perform 3-D rotation with perspective transformation, useful in texture mapping.

CONCLUSIONS

The technique outlined here performs arbitrary high-speed raster rotation with anti-aliasing and optional translation. The mathematical derivation guarantees scaling invariance when rotating. The implementation strategy allows for particularly fast operation, while minimizing the approximation error. This algorithm is a powerful tool in the répertoire of digital paint and raster manipulation systems. Coupled with state-of-the-art raster scaling techniques, it can transform an input raster by an arbitrary 2x2 transformation matrix in near real time.

REFERENCES

[[Catm80]] Catmull, E., Smith A. R. "3-D Transformations of Images in Scanline Order" *ACM Computer Graphics* (SIGGRAPH '80) **14**(3), July 1980, pp. 279-285.

[[Newm79]] Newman, W.M., Sproull, R.F. *Principles of Interactive Computer Graphics* (2nd ed), pp. 28 McGraw-Hill, New York 1979.

[[Inga78]] Ingalls, D.H.H. "The Smalltalk-76 programming system: design and implementation" *Fifth ACM Symp. Prin. Prog. Lang.*, January, 1978, pp. 9-16.

[[Krie84]] Krieger, R.A. "3-D Environments For 2-D Animation", MMath Essay, University of Waterloo, Ontario, 1984.

[[Port84]] Porter, T., Duff, T. "Compositing Digital Images" *ACM Computer Graphics* (SIGGRAPH '84) **18**(3), July, 1984, pp. 253-259.

[[Weim80]] Weiman, C.F.R. "Continuous Anti-Aliased Rotation and Zoom of Raster Images" *ACM Computer Graphics* (SIGGRAPH '80) **14**(3), July 1980, pp. 291.