Projective Mappings for Image Warping

Paul Heckbert

15-869, Image-Based Modeling and Rendering
13 Sept 1999

Excerpted from pages 17-21 of *Fundamentals of Texture Mapping and Image Warping*, Paul Heckbert, Master's thesis, UCB/CSD 89/516, CS Division, U.C. Berkeley, June 1989, `http://www.cs.cmu.edu/~ph`, with corrections and a change to column vector notation.

1 Homogeneous Notation

The homogeneous representation for points provides a consistent notation for affine and projective mappings. Homogeneous notation was used in projective geometry [Maxwell46], [Coxeter78] long before its introduction to computer graphics [Roberts66]. The homogeneous notation is often misunderstood so we will take a moment to clarify its use and properties.

In familiar Euclidean geometry we represent points of the real plane $\mathcal{R}^2$ by vectors of the form $(x, y)^T$. The superscript T denotes matrix transpose; that is, a column vector. We will use column vectors for points and vectors. Projective geometry deals with the projective plane, a superset of the real plane, whose *homogeneous coordinates* are $(x', y', w)^T$. In projective geometry the 2-D real point $(x, y)^T$ is represented by the homogeneous vector $\mathbf{p} = (x', y', w)^T = (xw, yw, w)^T$, where w is an arbitrary nonzero number. Vectors of the form $(xw, yw, w)^T$ for $w \neq 0$ form the equivalence class of homogeneous representations for the real point $(x, y)^T$. To recover the actual coordinates from a homogeneous vector, we simply divide by the homogeneous component; e.g., the homogeneous vector $\mathbf{p} = (xw, yw, w)^T = (x', y', w)^T$ represents the actual point $(x, y)^T = (x'/w, y'/w)^T$. This division, a projection onto the $w = 1$ plane, cancels the effect of scalar multiplication by w. When representing real points with homogeneous notation we could use any nonzero w, but it is usually most convenient to choose $w = 1$ so that the real coordinates can be recovered without division. Projective space also includes the *points at infinity*: vectors of the form $(x', y', 0)^T$, excluding $(0, 0, 0)^T$. The points at infinity lie on the *line at infinity*. We will see later how augmentation of the real plane by these points simplifies projective geometry.

In homogeneous notation, 2-D points are represented by 3-vectors and 3-D points are represented by 4-vectors. For affine and projective mappings, we denote points in source space by $\mathbf{p}_s = (u', v', q)^T$ and points in the destination space by $\mathbf{p}_d = (x', y', w)^T$.

2 Projective Mappings

The *projective mapping*, also known as the perspective or homogeneous transformation, is a projection from one plane through a point onto another plane [Maxwell46]. Homogeneous transformations are used extensively for 3-D affine modeling transformations and for perspective camera transformations [Foley-van Dam82]. The 2-D projective mappings studied here are a subset of these familiar 3-D homogeneous transformations.

The general form of a projective mapping is a rational linear mapping:

$$x = \frac{au + bv + c}{gu + hv + i}, \quad y = \frac{du + ev + f}{gu + hv + i} \tag{1}$$

Manipulation of projective mappings is much easier in the homogeneous matrix notation:

$$\begin{aligned} \mathbf{p}_d &= \mathbf{M}_{sd}\mathbf{p}_s \\ = \begin{pmatrix} x' \\ y' \\ w \end{pmatrix} &= \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} u' \\ v' \\ q \end{pmatrix} \end{aligned}$$

where $(x, y)^T = (x'/w, y'/w)^T$ for $w \neq 0$, and $(u, v)^T = (u'/q, v'/q)^T$ for $q \neq 0$.

Although there are 9 coefficients in the matrix above, these mappings are homogeneous, so any nonzero scalar multiple of these matrices gives an equivalent mapping. Hence there are only 8 degrees of freedom in a 2-D projective mapping. We can assume without loss of generality that $i = 1$ except in the special case that source point $(0, 0)^T$ maps to a point at infinity. A projective mapping is affine when $g = h = 0$.

Projective mappings will in general map the line at infinity to a line in the real plane. We can think of this line as the horizon line of all vanishing points, by analogy to the perspective projection. Affine mappings are the special case of projective mappings that map the line at infinity into itself. By defining the projective mapping over the projective plane and not just the real plane, projective mappings become bijections (one-to-one and onto), except when the mapping matrix is singular. For nondegenerate mappings the forward and inverse transforms are single-valued, just as for an affine mapping.

Projective mappings share many of the desirable properties of affine mappings. Unlike bilinear mappings, which preserve equispaced points along certain lines, the projective mappings do not in general preserve equispaced points. Instead they preserve a quantity called the "cross ratio" of points [Coxeter78]. Like affine mappings, *projective mappings preserve lines at all orientations*. In fact, projective mappings are the most general line-preserving bijective mappings. Projective mappings may be composed by concatenating their matrices.

Another remarkable property is that *the inverse of a projective mapping is a projective mapping*. This is intuitively explained by reversing the plane-to-plane mapping by which a projective mapping is defined. The matrix for the inverse mapping is the inverse or adjoint of the forward mapping. (The adjoint of a matrix is the transpose of the matrix of cofactors [Strang80]; $\mathbf{M}^{-1} = adj(\mathbf{M})/det(\mathbf{M})$). In homogeneous algebra, the adjoint matrix can be used in place of the inverse matrix whenever an inverse transform is needed, since the two are scalar multiples of each other, and the adjoint always exists, while the inverse does not if the matrix is singular. The inverse transformation is thus:

$$\begin{aligned} \mathbf{p}_s &= \mathbf{M}_{ds}\mathbf{p}_d \\ = \begin{pmatrix} u' \\ v' \\ q \end{pmatrix} &= \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix} \begin{pmatrix} x' \\ y' \\ w \end{pmatrix} \\ &= \begin{pmatrix} ei - fh & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{pmatrix} \begin{pmatrix} x' \\ y' \\ w \end{pmatrix} \end{aligned}$$

When mapping a point by the inverse transform we compute $(u,v)^T$ from $(x,y)^T$. If $w \neq 0$ and $q \neq 0$ then we can choose $w = 1$ and calculate:

$$u = \frac{Ax + By + C}{Gx + Hy + I}, \quad v = \frac{Dx + Ey + F}{Gx + Hy + I} \tag{2}$$

3 Inferring Projective Mappings

In an interactive image warper one might specify the four corners of source and destination quadrilaterals with a tablet or mouse, and wish to warp one area to the other. This sort of task is an ideal application of projective mappings, but how do we find the mapping matrix?

A projective mapping has 8 degrees of freedom which can be determined from the source and destination coordinates of the four corners of a quadrilateral. Let the correspondence map $(u_k, v_k)^T$ to $(x_k, y_k)^T$ for vertices numbered cyclically $k = 0, 1, 2, 3$. All coordinates are assumed to be real (finite). To compute the forward mapping matrix $\mathbf{M}_{sd}$, assuming that $i = 1$, we have eight equations in the eight unknowns a-h:

$$x_k = \frac{au_k + bv_k + c}{gu_k + hv_k + 1} \quad \Rightarrow \quad u_k a + v_k b + c - u_k x_k g - v_k x_k h = x_k$$

$$y_k = \frac{du_k + ev_k + f}{gu_k + hv_k + 1} \quad \Rightarrow \quad u_k d + v_k e + f - u_k y_k g - v_k y_k h = y_k$$

for $k = 0, 1, 2, 3$. This can be rewritten as an 8×8 system:

$$\begin{pmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0x_0 & -v_0x_0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0y_0 & -v_0y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

This linear system can be solved using Gaussian elimination or other methods for the forward mapping coefficients a-h. If the inverse mapping is desired instead, then either we compute the adjoint of $\mathbf{M}_{sd}$ or we follow the same procedure, starting from equation (2) instead of (1), and solve an 8×8 system for coefficients A-H.

In speed-critical special cases, there are more efficient formulas for computing the mapping matrix. The formula above handles the case where the polygon is a general quadrilateral in both source and destination spaces. We will consider three additional cases: square-to-quadrilateral, quadrilateral-to-square, and (again) the general quadrilateral-to-quadrilateral mapping.

Case 1. The system is easily solved symbolically in the special case where the uv quadrilateral is a unit square. If the vertex correspondence is as follows:

x	y	u	v
x_0	y_0	0	0
x_1	y_1	1	0
x_2	y_2	1	1
x_3	y_3	0	1

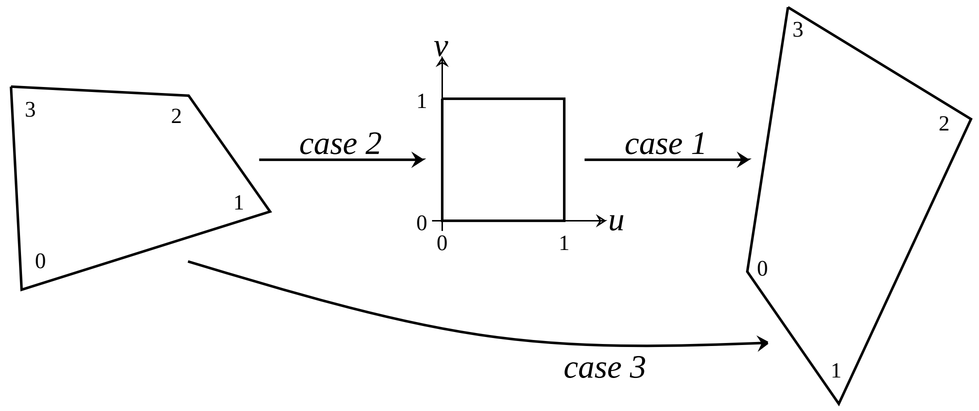

Figure 1: *Quadrilateral to quadrilateral mapping as a composition of simpler mappings.*

then the eight equations reduce to

$$\begin{aligned} c &= x_0 \\ a + c - gx_1 &= x_1 \\ a + b + c - gx_2 - hx_2 &= x_2 \\ b + c - hx_3 &= x_3 \\ f &= y_0 \\ d + f - gy_1 &= y_1 \\ d + e + f - gy_2 - hy_2 &= y_2 \\ e + f - hy_3 &= y_3 \end{aligned}$$

If we define

$$\begin{array}{lll} \Delta x_1 = x_1 - x_2 & \Delta x_2 = x_3 - x_2 & \Sigma x = x_0 - x_1 + x_2 - x_3 \\ \Delta y_1 = y_1 - y_2 & \Delta y_2 = y_3 - y_2 & \Sigma y = y_0 - y_1 + y_2 - y_3 \end{array}$$

then the solution splits into two sub-cases:

(a) $\Sigma x = 0$ and $\Sigma y = 0$. This implies that the xy polygon is a parallelogram, so the mapping is affine, and $a = x_1 - x_0$, $b = x_2 - x_1$, $c = x_0$, $d = y_1 - y_0$, $e = y_2 - y_1$, $f = y_0$, $g = 0$, $h = 0$.

(b) $\Sigma x \neq 0$ or $\Sigma y \neq 0$ gives a projective mapping:

$$\begin{aligned} g &= \begin{vmatrix} \Sigma x & \Delta x_2 \\ \Sigma y & \Delta y_2 \end{vmatrix} \Big/ \begin{vmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{vmatrix} \\ h &= \begin{vmatrix} \Delta x_1 & \Sigma x \\ \Delta y_1 & \Sigma y \end{vmatrix} \Big/ \begin{vmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{vmatrix} \\ a &= x_1 - x_0 + gx_1 \\ b &= x_3 - x_0 + hx_3 \\ c &= x_0 \\ d &= y_1 - y_0 + gy_1 \\ e &= y_3 - y_0 + hy_3 \\ f &= y_0 \end{aligned}$$

This computation is much faster than a straightforward 8×8 system solver. The mapping above is easily generalized to map a rectangle to a quadrilateral by pre-multiplying with a scale and translation matrix.

Case 2. The inverse mapping, a quadrilateral to a square, can also be optimized. It turns out that the most efficient algorithm for computing this is not purely symbolic, as in the previous case, but numerical. We use the square-to-quadrilateral formulas just described to find the inverse of the desired mapping, and then take its adjoint to compute the quadrilateral-to-square mapping.

Case 3. Since we can compute quadrilateral-to-square and square-to-quadrilateral mappings quickly, the two mappings can easily be composed to yield a general quadrilateral-to-quadrilateral mapping (figure 1). This solution method is faster than a general 8×8 system solver.

4 References

[Coxeter78] H. S. M. Coxeter, *Non-Euclidean Geometry*, U. of Toronto Press, 1978.

[Foley-van Dam82] James D. Foley, Andries van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, MA, 1982.

[Maxwell46] E. A. Maxwell, *The Methods of Plane Projective Geometry, Based on the Use of General Homogeneous Coordinates*, Cambridge U. Press, London, 1946.

[Roberts66] Lawrence G. Roberts, *Homogeneous Matrix Representation and Manipulation of N-Dimensional Constructs*, MS-1405, Lincoln Lab, Lexington, MA, July 1966.

[Strang80] Gilbert Strang, *Linear Algebra and Its Applications*, 2nd ed., Academic Press, New York, 1980.