A Traditionalist View of 3-D Image Warping

Hansong Zhang

TR97-043
Department of Computer Science
UNC-Chapel Hill

9/11/97

1 Introduction

The purpose of this report is to explain 3-D image warping using only transformation matrices traditional to computer graphics and understood by all graphics people. We use the OpenGL notions of spaces, transformations, the framebuffer and depth buffer, and the OpenGL camera model. It contains enough details for OpenGL users to easily warp a rendered scene into another view.

2 The Transformations

A point P_{world} in the world space is converted into a pixel on the screen and a correponding value in the depth buffer through a series of transformations: the model-view, projection and viewport transformations and a scaling of the Z values. We combine the model-view and projection transformations into a matrix $M_{composite}$,

$$M_{composite} = M_{projection} \times M_{model-view} \tag{1}$$

$M_{composite}$ transforms P_{world} into P_{NDC} in the normalized device coordinate system.

$$P_{NDC} \stackrel{\div w}{=} M_{composite} \times P_{world}$$

Symbol "$\stackrel{\div w}{=}$" indicates equality after a division by the w component on the right side, so that w is alway 1 on the left. The X, Y and Z components of P_{NDC} are in the range $\{-1, 1\}$.

Thus

$$P_{world} \stackrel{\div w}{=} M^{-1}_{composite} \times P_{NDC} \tag{2}$$

projects a point in the eye space back into the world space.

A pixel in a rendered image of width W and height H has image-space, integer coordinate (i, j), $0 \le i < W$, $0 \le j < H$, and depth z_{ij} from the depth buffer. The pixel center's coordinate in the NDC space is

$$P_{NDC} = \begin{bmatrix} -1 + (i + 0.5) * (2/W) \\ -1 + (j + 0.5) * (2/H) \\ -1 + z_{ij} * 2 \\ 1 \end{bmatrix}$$

This is just to undo the viewport transformation and scaling of Z[1]. 0.5 is added because in OpenGL the lower-left corner of a pixel has integer coordinates, and we are computing the coordinates for the center of the pixel. $2/W$ and $2/H$ are the width and height of the pixels, respectively.

Now, using (2) the pixel can be back projected into the world space. If a second view is defined by a composite matrix $M'_{composite}$, then the pixel can be transformed into the second view's NDC space by

$$P'_{NDC} \stackrel{\div w}{=} M_{warp} \times P_{NDC}$$

where

$$M_{warp} = M'_{composite} \times M^{-1}_{composite}$$

[1]We assume the depth range, which can be set by *glDepthRange()*, is $\{0, 1\}$ (the default)

The second view is often called the destination view, while the first view the source view.

After simple viewport transformation and Z-scaling, we get the pixel coordinate in the image space of the second view, and the corresponding depth value. Or in other words, the source image has be warped into the destination view.

3 Optimization

We do not have to do a full matrix-vector multiplicaton per pixel to transform the pixels in the source image to the destination view. The fact that pixel coordinates change incrementally as we go through every pixel in the source image makes it possible for us to optimize the computation.

Let the elements of the M_{warp} matrix be m_{ij}, $0 \leq i, j < 3$:

$$M_{warp} = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix}$$

And suppose we already computed the destination coordinate (x_d, y_d, z_d, w_d) for a pixel whose NDC in the source view is $P_{NDC} = (x_s, y_s, z_s, 1)^T$, i.e.

$$\begin{pmatrix} x_d \\ y_d \\ z_d \\ w_d \end{pmatrix} = M_{warp} \begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix} = M_{warp} \begin{bmatrix} x_s \\ y_s \\ 0 \\ 1 \end{bmatrix} + M_{warp} \begin{bmatrix} 0 \\ 0 \\ z_s \\ 0 \end{bmatrix}$$

$$= V_{common} + \begin{bmatrix} m_{02} z_s \\ m_{12} z_s \\ m_{22} z_s \\ m_{32} z_s \end{bmatrix}$$

Now we consider the transformation for the next source pixel. We assume the next pixel's position is an increment of 1 to the current pixel's in the X direction of the source image (the discussion for an increment in the Y direction is similar). So the next pixel's NDC, say Q_{NDC}, in the source view is

$$Q_{NDC} = \begin{bmatrix} x_s + (2/W) \\ y_s \\ z_Q \\ 1 \end{bmatrix}$$

And to transform it into the destination NDC space, we have

$$M_{warp} \begin{bmatrix} x_s + (2/W) \\ y_s \\ z_Q \\ 1 \end{bmatrix} = M_{warp} \begin{bmatrix} x_s \\ y_s \\ 0 \\ 1 \end{bmatrix} + M_{warp} \begin{bmatrix} 2/W \\ 0 \\ z_Q \\ 0 \end{bmatrix}$$

$$= V_{common} + \begin{bmatrix} m_{00}\frac{2}{W} + m_{02}z_Q \\ m_{10}\frac{2}{W} + m_{12}z_Q \\ m_{20}\frac{2}{W} + m_{22}z_Q \\ m_{30}\frac{2}{W} + m_{32}z_Q \end{bmatrix}$$

Apparently, $m_{i0}\frac{2}{W}$, $0 \leq i < 4$, are constants and thus can be precomputed. So the amount of computation for warping Q is only 4 multiplications and 8 additions.

4 Derivatives

The derivatives of the output coordinates to the input are useful for estimating the "stamp size" of the warped pixel, i.e. how big the pixel becomes in the destination image. For transformation,

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = M_{warp} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

the partial derivatives of the destination coordinates to x are:

$$\begin{pmatrix} \partial x'/\partial x \\ \partial y'/\partial x \\ \partial z'/\partial x \\ \partial w'/\partial x \end{pmatrix} = M_{warp} \begin{pmatrix} 1 \\ 0 \\ \partial z/\partial x \\ 0 \end{pmatrix} = \begin{pmatrix} m_{00} + m_{02}\frac{\partial z}{\partial x} \\ m_{10} + m_{12}\frac{\partial z}{\partial x} \\ m_{20} + m_{22}\frac{\partial z}{\partial x} \\ m_{30} + m_{32}\frac{\partial z}{\partial x} \end{pmatrix}$$

And derivatives to y can be obtained similarly. Now, in the destination NDC space where $(x'', y'', z'') = (x'/w', y'/w', z'/w')$, the derivative of x'' to x is

$$\frac{\partial x''}{\partial x} = (\frac{\partial x'}{\partial x} w' + \frac{\partial w'}{\partial x} x')/w'^2$$

and likewise for other derivatives.

5 Frame-To-Frame Coherence

Suppose the warping matrix of frame n is M_{warp_n} and that of frame $n+1$ $M_{warp_{n+1}}$. Let their difference be $\Delta M_{warp} = M_{warp_{n+1}} - M_{warp_n}$. Then when transforming point p in the source NDC space, we obviously have:

$$M_{warp_{n+1}} p = M_{warp_n} p + \Delta M_{warp} \quad p$$

So if ΔM_{warp} is sparse, $\Delta M_{warp} p$ can be less expensive to compute than $M_{warp_{n+1}} p$. For example, if the motion of the eye from frame n to frame $n+1$ is simply a translation, then only 4 elements in ΔM are non-zero. To see this, let the composite transformation matrix that produces the source image be M, and the matrices that produces frame n and $n+1$ be $M_n^{'}$ and $M_n + 1^{'}$, respectively, then from discussions in section 1 we have $M_{warp_n} = M^{-1} M_n^{'}$ and $M_{warp_{n+1}} = M^{-1} M_{n+1}^{'}$, so

$$\Delta M_{warp} = M^{-1}(M_{n+1}^{'} - M_n^{'})$$

Thus if the eye motion is a translation, then only 3 elements in the right column of $M_{n+1}^{'} - M_n^{'}$ will be non-zero, and so only 4 elements in ΔM_{warp} can be non-zero.

6 Acknowledgement

Thanks to Bill Mark for discussions and suggestions on incremental matrix-vector multiplication, and Kenny Hoff for reading the draft.

7 reference

[1] Mason Woo, Jackie Neider, Tom Davis, *OpenGL Programming Guide*, Second Edition, 1997