Factoring 3-D Image Warping Equations into a Pre-Warp Followed by Conventional Texture Mapping

Manuel M. Oliveira and Gary Bishop
{oliveira|gb}@cs.unc.edu

Department of Computer Science
University of North Carolina at Chapel Hill
Sitterson Hall, CB# 3175
Chapel Hill, NC, 27599-3175

Technical Report TR99-002
January 15, 1999

Abstract

Conventional texture mapping is a special case of three-dimensional image warping. Therefore, all transformations embodied in the texture mapping equations are also embodied in the three-dimensional image warping equations. In this work, we show that three-dimensional image warping can be factored into a pre-warp step followed by conventional texture mapping.

1 Introduction

Three-dimensional image warping [McMillan97] provides an efficient way to compute new perspective views of scenes from reference images extended with depth on a per pixel basis. Such reference images are usually called *source* images, while the reconstructed views are usually referred to as *destination* images. Given the coordinates (u_1,v_1) of pixels in a source image, the coordinates of the corresponding pixels in the destination image are compute as [McMillan97]:

$$u_2 = \frac{\vec{a}_1 \cdot (\vec{b}_2 \times \vec{c}_2) u_1 + \vec{b}_1 \cdot (b_2 \times c_2) v_1 + \vec{c}_1 \cdot (\vec{b}_2 \times \vec{c}_2) + (\dot{C}_1 - \dot{C}_2) \cdot (\vec{b}_2 \times \vec{c}_2) \delta(u_1, v_1)}{\vec{a}_1 \cdot (\vec{a}_2 \times \vec{b}_2) u_1 + \vec{b}_1 \cdot (\vec{a}_2 \times \vec{b}_2) v_1 + \vec{c}_1 \cdot (\vec{a}_2 \times \vec{b}_2) + (\dot{C}_1 - \dot{C}_2) \cdot (\vec{a}_2 \times \vec{b}_2) \delta(u_1, v_1)}$$

$$v_2 = \frac{\vec{a}_1 \cdot (\vec{c}_2 \times \vec{a}_2) u_1 + \vec{b}_1 \cdot (\vec{c}_2 \times \vec{a}_2) v_1 + \vec{c}_1 \cdot (\vec{c}_2 \times \vec{a}_2) + (\dot{C}_1 - \dot{C}_2) \cdot (\vec{c}_2 \times \vec{a}_2) \delta(u_1, v_1)}{\vec{a}_1 \cdot (\vec{a}_2 \times \vec{b}_2) u_1 + \vec{b}_1 \cdot (\vec{a}_2 \times \vec{b}_2) v_1 + \vec{c}_1 \cdot (\vec{a}_2 \times \vec{b}_2) + (\dot{C}_1 - \dot{C}_2) \cdot (\vec{a}_2 \times \vec{b}_2) \delta(u_1, v_1)}$$

where subscript 1 identifies source image variables; 2, the destination image. Vectors $\vec{a}$ and $\vec{b}$ are orthogonal and form a basis for the plane of the image. The lengths of these vectors are the width and height of a pixel in the Euclidean space, respectively. The generalized disparity associated with pixel (u_1,v_1) is $\delta(u_1, v_1)$. $\dot{C}$ is the center of projection (COP) of the camera, and $\vec{c}$ is a vector from the COP to the origin of the image plane (Figure 1).

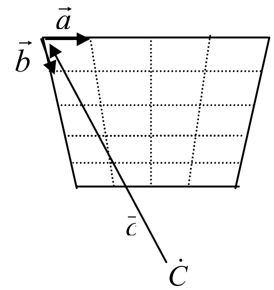

Figure 1 Perspective projection camera representation.

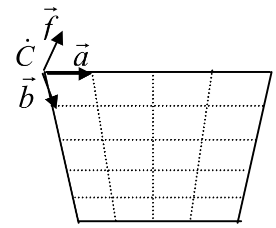

Figure 2. Parallel projection camera representation.

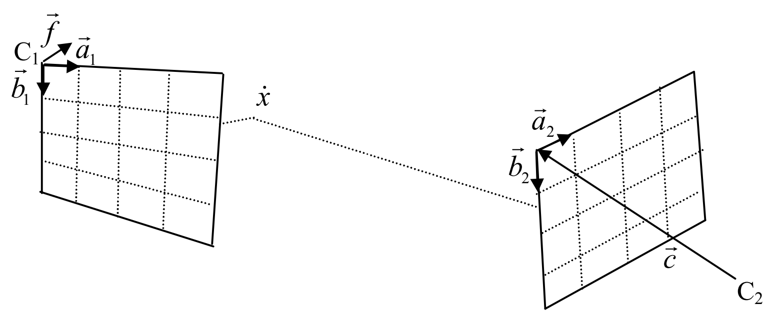

Figure 3. 3-D image warping. The source is a parallel projection image while the destination is a perspective projection image.

2 Pre-Warping Equations

Figure 2 shows the representation we use for a parallel projection camera. Vectors $\vec{a}$ and $\vec{b}$ have the same definition as in the projective pinhole camera shown in Figure 1. Vector $\vec{f}$ is a unit vector orthogonal to the plane spanned by $\vec{a}$ and $\vec{b}$. The tails of all these vectors are at $\dot{C}$, the origin of the image plane. This representation is compatible with the projective pinhole camera representation used in [McMillan97]. The coordinates of a point $\dot{x}$ in Euclidean space (Figure 3) can be expressed as:

$$\dot{x} = \dot{C}_1 + \begin{bmatrix} a_{1i} & b_{1i} & f_i \\ a_{1j} & b_{1j} & f_j \\ a_{1k} & b_{1k} & f_k \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ displ(u_1, v_1) \end{bmatrix} = \dot{C}_1 + P_1 \vec{X}_1 \tag{1}$$

where *displ(u₁, v₁)* is the orthogonal displacement or height associated with the pixel whose coordinates are (u_1 , v_1). Alternatively, using the original formulation for perspective projection cameras [McMillan97] the coordinates of point $\dot{x}$ can be written as:

$$\dot{x} = \dot{C}_2 + t_2 \begin{bmatrix} a_{2i} & b_{2i} & c_i \\ a_{2j} & b_{2j} & c_j \\ a_{2k} & b_{2k} & c_k \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \dot{C}_2 + t_2 P_2 \vec{X}_2 \tag{2}$$

where t_2 is a scalar value defined on a per pixel basis. Solving for $\vec{X}_2$, we get:

$$\begin{aligned} &\dot{C}_2 + t_2 P_2 \vec{X}_2 = \dot{C}_1 + P_1 \vec{X}_1 \\ &t_2 P_2 \vec{X}_2 = P_1 \vec{X}_1 + (\dot{C}_1 - \dot{C}_2) \\ &\vec{X}_2 \doteq P_2^{-1} (P_1 \vec{X}_1 + (\dot{C}_1 - \dot{C}_2)) \end{aligned} \tag{3}$$

where $\doteq$ is projective equivalence, that is, the same except for a scalar multiple. In matrix notation, we have:

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \doteq \begin{bmatrix} \vec{a}_2 & \vec{b}_2 & \vec{c} \end{bmatrix}^{-1} \left(\begin{bmatrix} \vec{a}_1 & \vec{b}_1 & \vec{f} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ displ(u_1, v_1) \end{bmatrix} + \begin{bmatrix} (\dot{C}_1 - \dot{C}_2) \end{bmatrix} \right) \tag{4}$$

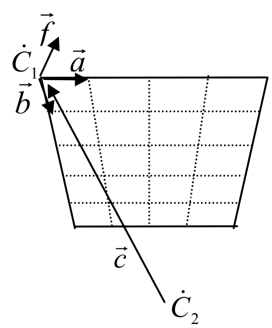

Figure 4. Parallel and perspective projection cameras that share the same image plane (origin, $\vec{a}$ and $\vec{b}$ vectors).

By making both image planes coincide (including their origins - Figure 4), $\vec{a}_1 = \vec{a}_2 = \vec{a}$, $\vec{b}_1 = \vec{b}_2 = \vec{b}$, $\vec{c} = (\dot{C}_1 - \dot{C}_2)$ and Equation (4) then becomes:

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \doteq \begin{bmatrix} \vec{a} & \vec{b} & \vec{c} \end{bmatrix}^{-1} \left(\begin{bmatrix} \vec{a} & \vec{b} & \vec{f} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ displ(u_1, v_1) \end{bmatrix} + \begin{bmatrix} \vec{c} \end{bmatrix} \right) \tag{5}$$

The coordinates of the pixels in the destination image are then given by

$$u_2 = \frac{\vec{a}\cdot(\vec{b}\times\vec{c})u_1 + \vec{b}\cdot(\vec{b}\times\vec{c})v_1 + \vec{c}\cdot(\vec{b}\times\vec{c}) + \vec{f}\cdot(\vec{b}\times\vec{c})displ(u_1,v_1)}{\vec{a}\cdot(\vec{a}\times\vec{b})u_1 + \vec{b}\cdot(\vec{a}\times\vec{b})v_1 + \vec{c}\cdot(\vec{a}\times\vec{b}) + \vec{f}\cdot(\vec{a}\times\vec{b})displ(u_1,v_1)} \qquad (6a)$$

$$v_2 = \frac{\vec{a}\cdot(\vec{c}\times\vec{a})u_1 + \vec{b}\cdot(\vec{c}\times\vec{a})v_1 + \vec{c}\cdot(\vec{c}\times\vec{a}) + \vec{f}\cdot(\vec{c}\times\vec{a})displ(u_1,v_1)}{\vec{a}\cdot(\vec{a}\times\vec{b})u_1 + \vec{b}\cdot(\vec{a}\times\vec{b})v_1 + \vec{c}\cdot(\vec{a}\times\vec{b}) + \vec{f}\cdot(\vec{a}\times\vec{b})displ(u_1,v_1)} \qquad (6b)$$

Note that many of the scalar triple products in equations (6*a*) and (6*b*) are of the form $\vec{v}\cdot(\vec{v}\times\vec{w})$ or $\vec{w}\cdot(\vec{v}\times\vec{w})$ and therefore reduce to zero. Thus,

$$u_2 = \frac{\vec{a}\cdot(\vec{b}\times\vec{c})u_1 + \vec{f}\cdot(\vec{b}\times\vec{c})displ(u_1,v_1)}{\vec{c}\cdot(\vec{a}\times\vec{b}) + \vec{f}\cdot(\vec{a}\times\vec{b})displ(u_1,v_1)} \qquad (7a)$$

$$v_2 = \frac{\vec{b}\cdot(\vec{c}\times\vec{a})v_1 + \vec{f}\cdot(\vec{c}\times\vec{a})displ(u_1,v_1)}{\vec{c}\cdot(\vec{a}\times\vec{b}) + \vec{f}\cdot(\vec{a}\times\vec{b})displ(u_1,v_1)} \qquad (7b)$$

But $\vec{a}\cdot(\vec{b}\times\vec{c}) \neq 0$ is the determinant of the 3x3 matrix whose rows are respectively $\vec{a}$, $\vec{b}$, and $\vec{c}$. Also, $\vec{c}\cdot(\vec{a}\times\vec{b})$ is the determinant of the same matrix after two permutations of rows, and therefore has the same value. The same observation holds for $\vec{b}\cdot(\vec{c}\times\vec{a})$. Thus, dividing both numerators and denominators of equations (7*a*) and (7*b*) by $\vec{a}\cdot(\vec{b}\times\vec{c})$, we get

$$r = u_1 + k_1 displ(u_1,v_1) \qquad (8)$$
$$s = v_1 + k_2 displ(u_1,v_1) \qquad (9)$$
$$t = 1 + k_3 displ(u_1,v_1) \qquad (10)$$

$$u_2 = \frac{r}{t} \qquad (11a)$$

$$v_2 = \frac{s}{t} \qquad (11b)$$

where $k_1 = \frac{\vec{f}\cdot(\vec{b}\times\vec{c})}{\vec{a}\cdot(\vec{b}\times\vec{c})}$, $k_2 = \frac{\vec{f}\cdot(\vec{c}\times a)}{\vec{a}\cdot(\vec{b}\times\vec{c})} = \frac{\vec{f}\cdot(\vec{c}\times a)}{\vec{b}\cdot(\vec{c}\times\vec{a})}$ and $k_3 = \frac{\vec{f}\cdot(\vec{a}\times\vec{b})}{\vec{a}\cdot(\vec{b}\times\vec{c})} = \frac{\vec{f}\cdot(\vec{a}\times\vec{b})}{\vec{c}\cdot(\vec{a}\times\vec{b})}$ are constants across the entire source image and determine the amount of change in the coordinates of corresponding pixels on both images (optical flow [Faugeras93]). Notice that if the displacement displ(u_1, v_1)= 0, then (u_2, v_2) = (u_1, v_1), *i.e.*, the pre-warping operation is the identity function. Equations (11a) and (11b) are called *pre-warping equations*.

3 3-D Image Warping as a Pre-Warp Followed by Conventional Texture Mapping

Theorem: 3-D image warping can be factored into a pre-warp followed by conventional texture mapping.

Proof: Let (u_1, v_1), (u_2, v_2) and (u_3, v_3) represent the coordinates of pixels in the source, destination, and intermediate pre-warped image, respectively. According to Equations (8) to (11*b*), the (u_3, v_3) coordinates of a pre-warped sample are given by:

$$u_3 = \frac{u_1 + k_1 displ(u_1, v_1)}{1 + k_3 displ(u_1, v_1)} \qquad (12a)$$

$$v_3 = \frac{v_1 + k_2 displ(u_1, v_1)}{1 + k_3 displ(u_1, v_1)} \qquad (12b)$$

Texture mapping is a projective mapping defined as [Heckbert89]:

$$u_2 = \frac{Au_1 + Bv_1 + C}{Iu_1 + Jv_1 + K} \qquad (13a)$$

$$v_2 = \frac{Eu_1 + Fv_1 + G}{Iu_1 + Jv_1 + K} \qquad (13b)$$

where A, B, C, E, F, G, I, J and K are constants for a particular mapping[1].

The 3-D image warping equations that use parallel projection source images (equations (6*a*) and (6*b*)) can be rewritten as:

$$u_2 = \frac{Au_1 + Bv_1 + C + Ddispl(u_1, v_1)}{Iu_1 + Jv_1 + K + Ldispl(u_1, v_1)} \qquad (14a)$$

$$v_2 = \frac{Eu_1 + Fv_1 + G + Hdispl(u_1, v_1)}{Iu_1 + Jv_1 + K + Ldispl(u_1, v_1)} \qquad (14b)$$

If $displ(u_1, v_1) = 0$ for all pixels, equations (14*a*) and (14*b*) reduce to equations (13*a*) and (13*b*), respectively, and 3-D image warping reduces to texture mapping. In other words, texture mapping is a special case of the 3-D image warping for which all samples happen to be on the image plane [McMillan97]. Therefore, it is not surprising that coefficients A, B, C, E, F, G, I, J, and K in equations (14*a*) and (14*b*) are exactly the same as the ones in equations (13*a*) and (13*b*).

[1] Notice that Equations (13*a*) and (13*b*) used here represent forward texture mapping, since both the pre-warping equations (Equations (11*a*) and (11*b*)) and the full warping equation (Equation (4)) are forward operations. An actual implementation of the two-step process uses inverse texture mapping [Oliveira99].

Let pre-warped images be used as input for texture mapping operation. By substituting equations (12*a*) and (12*b*) into equations (13*a*), we get

$$u_2 = \frac{A\left(\dfrac{u_1 + k_1 displ(u_1, v_1)}{1 + k_3 displ(u_1, v_1)}\right) + B\left(\dfrac{v_1 + k_2 displ(u_1, v_1)}{1 + k_3 displ(u_1, v_1)}\right) + C}{I\left(\dfrac{u_1 + k_1 displ(u_1, v_1)}{1 + k_3 displ(u_1, v_1)}\right) + J\left(\dfrac{v_1 + k_2 displ(u_1, v_1)}{1 + k_3 displ(u_1, v_1)}\right) + K}$$

$$u_2 = \frac{A(u_1 + k_1 displ(u_1, v_1)) + B(v_1 + k_2 displ(u_1, v_1)) + C(1 + k_3 displ(u_1, v_1))}{I(u_1 + k_1 displ(u_1, v_1)) + J(v_1 + k_2 displ(u_1, v_1)) + K(1 + k_3 displ(u_1, v_1))}$$

$$u_2 = \frac{Au_1 + Bv_1 + C + (Ak_1 + Bk_2 + Ck_3)displ(u_1, v_1)}{Iu_1 + Jv_1 + K + (Ik_1 + Jk_2 + Kk_3)displ(u_1, v_1)} \tag{15a}$$

Likewise for v_2:

$$v_2 = \frac{Eu_1 + Fv_1 + G + (Ek_1 + Fk_2 + Gk_3)displ(u_1, v_1)}{Iu_1 + Jv_1 + K + (Ik_1 + Jk_2 + Kk_3)displ(u_1, v_1)} \tag{15b}$$

Note the similarities between the 3-D warping equations (14*a*) and (14*b*) and equations (15*a*) and (15*b*) that result from texture mapping the pre-warped version of the source image onto its own image plane. In order to show that these equations are equal, we have to verify that $D = (Ak_1 + Bk_2 + Ck_3)$, $H = (Ek_1 + Fk_2 + Gk_3)$, and $L = (Ik_1 + Jk_2 + Kk_3)$. Comparing equations (6*a*), (6*b*), (14*a*) and (14*b*) we have:

$$\begin{array}{llll}
A = \vec{a} \cdot (\vec{b} \times \vec{c}), & B = \vec{b} \cdot (\vec{b} \times \vec{c}) = 0, & C = (\dot{C}_1 - \dot{C}_2) \cdot (\vec{b} \times \vec{c}) = \vec{c} \cdot (\vec{b} \times \vec{c}) = 0, & D = \vec{f} \cdot (\vec{b} \times \vec{c}) \\
E = \vec{a} \cdot (\vec{c} \times \vec{a}) = 0, & F = \vec{b} \cdot (\vec{c} \times a), & G = (\dot{C}_1 - \dot{C}_2) \cdot (\vec{c} \times \vec{a}) = \vec{c} \cdot (\vec{c} \times \vec{a}) = 0, & H = \vec{f} \cdot (\vec{c} \times \vec{a}) \\
I = \vec{a} \cdot (\vec{a} \times \vec{b}) = 0, & J = \vec{b} \cdot (\vec{a} \times \vec{b}) = 0, & K = (\dot{C}_1 - \dot{C}_2) \cdot (\vec{a} \times \vec{b}) = \vec{c} \cdot (\vec{a} \times \vec{b}), & L = \vec{f} \cdot (\vec{a} \times \vec{b})
\end{array}$$

Recalling the expressions for k_1, k_2 and k_3 computed at the end of section 2

$$Ak_1 + Bk_2 + Ck_3 = Ak_1 = \vec{a} \cdot (\vec{b} \times \vec{c})\left(\frac{\vec{f} \cdot (\vec{b} \times \vec{c})}{a \cdot (\vec{b} \times \vec{c})}\right) = \vec{f} \cdot (\vec{b} \times \vec{c}) = D$$

$$Ek_1 + Fk_2 + Gk_3 = Fk_2 = \vec{b} \cdot (\vec{c} \times \vec{a})\left(\frac{\vec{f} \cdot (\vec{c} \times a)}{\vec{b} \cdot (\vec{c} \times \vec{a})}\right) = \vec{f} \cdot (\vec{c} \times a) = H$$

$$Ik_1 + Jk_2 + Kk_3 = Kk_3 = \vec{c} \cdot (\vec{a} \times \vec{b})\left(\frac{\vec{f} \cdot (\vec{a} \times \vec{b})}{\vec{c} \cdot (\vec{a} \times \vec{b})}\right) = \vec{f} \cdot (\vec{a} \times \vec{b}) = L$$

(q.e.d.)

The use of a desired image plane that coincides with both the source image plane and the polygon to be texture-mapped is just a trick that greatly simplifies the verification of the identity. Arbitrary desired view plane could have been used instead, and the texture-mapped polygon mapped onto them, producing correct results. Intuitively, since the source image has been (pre-) warped to its own image plane, the resulting image has the correct perspective for the desired viewpoint, and thus can be used to texture map a polygon that matches the dimensions, position and orientation of the original image plane. The limitations of this technique are discussed in [Oliveira99].

The coefficients *C* and *G* in Equations (13*a*), (13b), (14*a*) and (14*b*) are zero. This is due to the fact that the image planes of the source and pre-warped images share the same origin, making $\vec{c} = (\dot{C}_1 - \dot{C}_2)$. While such equality led to some simplification in the pre-warping equations, it has no further meaning.

4 Example

This section presents a complete example illustrating the use of the two-step process. Figures 5 shows a top view of a scene sampled using a parallel projection image with depth (Figure 5(b)) that is then used as source image in the configuration shown in Figure 6. While one can use Equation (4) to perform conventional 3-D image warping from parallel projection to perspective projection images, the two-step process illustrated in Figure 7 has several reconstruction and filtering advantages over the traditional approach [Oliveira99].

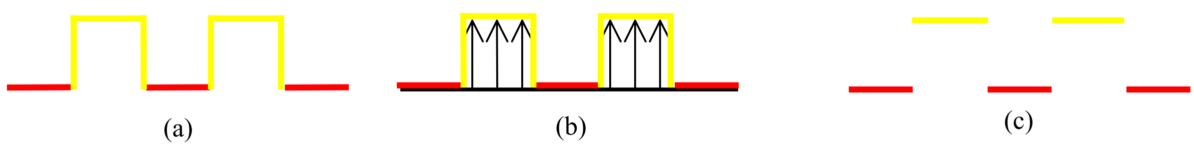

Figure 5. (a) Actual scene. (b) Sampling of the geometry with a parallel projection image with depth. (c) Re-projection of the sampled surfaces to 3-D.

In Figure 7, the source image is pre-warped to its own image plane using a perspective projection camera whose COP is at the desired viewpoint (C_2) and that shares the image plane of the source image. Notice the introduction of the vector $\vec{c}\,'$ in Figure 7. This configuration is similar to the one represented in Figure 4. The original vectors $\vec{c}_2$, $\vec{a}_2$ and $\vec{b}_2$ are not used. Visibility is solved using an occlusion compatible order algorithm described in [Oliveira 99]. The resulting pre-warped image has correct perspective for the desired viewpoint and can, therefore, be used as a texture to be mapped onto a quadrilateral that matches the source image plane in 3-space. The texture-mapping step takes care of the final planar perspective projection from the texture-mapped polygon onto the desired view plane (using an inverse mapping). Because the pre-warping equations present very simple 1-D structure, reconstruction can be performed using 1-D image operations along rows and columns and requiring interpolation between only two pixels at any time. Examples involving the pre-warp of actual 2-D textures are presented in [Oliveira99].

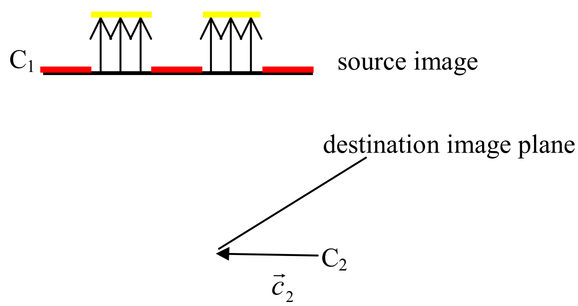

Figure 6. Configuration showing a source parallel projection image and a destination perspective projection image.

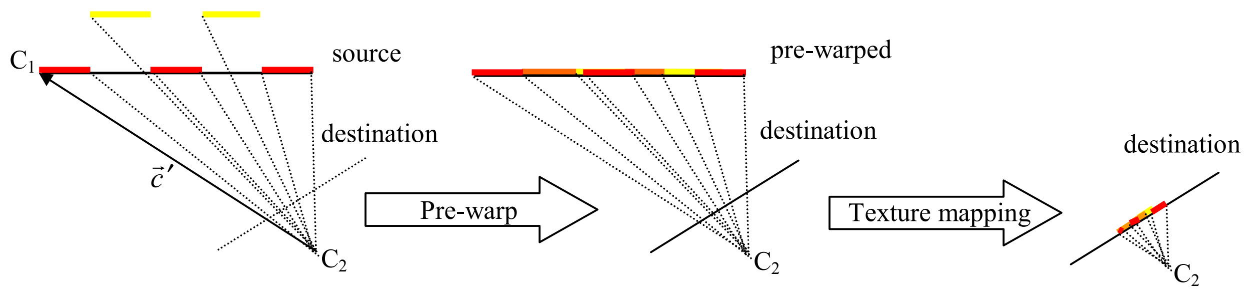

Figure 7. Two-step process for the configuration shown in Figure 6. The source image is pre-warped to its own image plane by defining a perspective projection camera with COP at the desired viewpoint (C_2) and whose image plane is shared with the source image. The resulting pre-warped image is then texture-mapped onto a quadrilateral that matches the source image plane in 3-D, producing a correct view of the represented surface. The orange regions were interpolated between red and yellow regions during the 1-D reconstruction process.

Acknowledgements

This work was sponsored by CNPq/Brazil – Process # 200054/95. Additional support provided by DARPA under order # E278 and NFS under grant # MIP-961.

References

[Faugeras93] Faugeras, Olivier. Three-Dimensional Computer Vision: A Geometric Viewpoint. The MIT Press, 1993.

[Heckbert89] Heckbert, Paul. Fundamentals of Texture Mapping and Image Warping. Master's Thesis. Computer Science Division, University of California, Berkeley. Report No. UCB/CSD 89/516, June 1989.

[McMillan97] McMillan, Leonard. *An Image-Based Approach to Three-Dimensional Computer Graphics*. Ph.D. Dissertation. UNC Computer Science Technical Report TR97-013, University of North Carolina, April 1997.

[Oliveira99] Oliveira, Manuel and Gary Bishop. *Relief Textures.* UNC Computer Science Technical Report TR99-015, University of North Carolina, March 1999.