Relief Texture Mapping

Manuel M. Oliveira[†] Gary Bishop[‡] David McAllister[‡]

University of North Carolina at Chapel Hill

Figure 1. Town rendered using conventional texture mapping. The façades and brick walls represented with one texture each.

Figure 2. Same view as in Figure 1 rendered using relief texture mapping. Both scenes contain the same number of polygons. Notice the bricks standing out and the protruding dormers.

ABSTRACT

We present an extension to texture mapping that supports the representation of 3-D surface details and view motion parallax. The results are correct for viewpoints that are static or moving, far away or nearby. Our approach is very simple: a *relief texture* (texture extended with an orthogonal displacement per texel) is mapped onto a polygon using a two-step process: First, it is converted into an ordinary texture using a surprisingly simple 1-D forward transform. The resulting texture is then mapped onto the polygon using standard texture mapping. The 1-D warping functions work in texture coordinates to handle the parallax and visibility changes that result from the 3-D shape of the displacement surface. The subsequent texture-mapping operation handles the transformation from texture to screen coordinates.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation I.3.6 [Computer Graphics]: Methodologies and Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

[†] Now at the Computer Science Department
SUNY at Stony Brook, Stony Brook, NY, 11794-4400
oliveira@cs.sunysb.edu http://www.cs.sunysb.edu/~oliveira

[‡] UNC Department of Computer Science
CB #3175, Sitterson Hall, Chapel Hill, NC, 27599-3175
{bishop | davemc}@cs.unc.edu http://www.cs.unc.edu/~ibr

Additional Keywords: Image-Based Rendering, Texture Mapping, Range Images, Rendering.

1. INTRODUCTION

Texture mapping has long been used to enhance the realism of computer-generated images by adding 2-D details to object surfaces [1]. For instance, it can be used to correctly simulate a picture on a wall, or the label on a can. Unfortunately, texture mapping is not as effective for adding 3-D details to a surface. When seen by a moving observer, the absence of parallax reveals the flatness of the surface. Such flatness is also evidenced when the surface is observed from an oblique angle (Figure 1).

A much more convincing illusion of 3-D surface detail can be achieved by using a height field in conjunction with a texture map. A height field is a scalar field of distances between surface points and their orthogonal projections onto a plane that forms its algebraic basis. Unfortunately, rendering height fields is much more difficult than texture mapping. The planar-projective transform of texture mapping has a very convenient inverse formulation. This allows direct computation of texture coordinates from screen coordinates, thus allowing efficient implementation as well as accurate resampling and filtering. Height-field rendering allows no such inverse formulation directly. Multiple samples from the height field may be mapped to the same pixel in the final image. Assuring correct visibility requires either a search for the closest surfaces (essentially a ray-tracing strategy) or a direct forward mapping [11].

We present an extension to texture mapping for representing three-dimensional surface details and view motion parallax. This new approach, called *relief texture mapping*, results from a factorization of the 3-D image-warping equation of McMillan and

Bishop into a pre-warp followed by standard texture mapping. The pre-warp is applied to images with per-texel displacements and handles only the parallax effects resulting from the direction of view and the displacements of texture elements; the subsequent texture-mapping operation handles scaling, rotation, and the remaining perspective transformation.

The pre-warping equations have a very simple 1-D structure that enables the pre-warp to be implemented using only 1-D image operations along rows and columns and requires interpolation between only two adjacent texels at a time. This allows efficient implementation in software and should allow a simple and efficient hardware implementation. The texture-mapping hardware already very common in graphics systems efficiently implements the final texture mapping stage of the warp.

In recent years, image-based modeling and rendering (IBMR) techniques have gained considerable attention in the graphics community because of their potential to create very realistic images. We hope to help to bridge the gap between IBMR and conventional polygonal rendering techniques by casting a subset of IBMR as an extension of texture mapping. Such a hybrid system can offer much of the photo-realistic promise of IBMR while retaining the advantages of polygonal rendering. In section 4.4, we present an example of a real environment modeled and rendered using relief texture mapping.

We demonstrate a software implementation of our method and show that it significantly increases the expressive power of conventional texture mapping. Our approach also dramatically reduces the polygonal count required to model a scene, while preserving its realistic look. Figure 2 shows the use of our approach for the same viewpoint used to create Figure 1. The two scenes used to render these images were modeled using the same number of polygons. In the example of Figure 2, each façade and brick wall is represented with a single relief texture. Notice the bricks standing out of the wall and the protruding dormers. In the original model of the town, each house consists of a few thousand polygons, whereas the corresponding relief texture representation uses only seven polygons per house.

The new results presented in this paper are:

- An extension to texture mapping that supports view motion parallax (Section 3);
- An exact factorization of the 3-D image warping equation [11] into a 1-D pre-warp followed by a planar projective mapping (Section 3.1);
- After rotations are factored out, 3-D warps reduce to a 2-D problem, regardless of the coordinate systems associated with the source and target images (section 3.1); and
- A 1-D image reconstruction algorithm that handles an arbitrary number of self-occlusions without requiring extra storage or depth comparison (section 3.3.2).

2. RELATED WORK

3-D Image Warping. Three-dimensional image warping [11] is a geometric transformation that maps a source image with depth i_s into a target image i_t. The geometric content of the scene is represented implicitly by combining depth information with a camera model associated with the source image. Thus, let $\dot{x}$ be a point in Euclidean space whose projection on the image plane of i_s has coordinates (u_s, v_s). The projection of $\dot{x}$ into an arbitrary target image plane, $\vec{x}_t$, is given by:

$$\vec{x}_t \doteq P_t^{-1} P_s \vec{x}_s + P_t^{-1}(\dot{C}_s - \dot{C}_t)\delta_s(u_s, v_s) \qquad (1)$$

where $\doteq$ is projective equivalence, *i.e.*, the same except for a scalar multiple, P_k is a 3x3 camera matrix associated with image i_k, $\vec{x}_s = [u_s \quad v_s \quad 1]^T$, $\dot{C}_k$ is the center of projection (COP) of the pinhole camera associated with image i_k and $\delta_s(u_s, v_s)$ is the *generalized disparity* of source pixel (u_s, v_s) [11]. Equation (1) shows that the target image can be obtained by applying a planar perspective transformation to the source image followed by a per-pixel shift proportional to $\delta_s(u_s, v_s)$ in the direction of the *epipole*[1] of the target image. Such a factorization is often referred to as *plane-plus-parallax* in the computer vision literature [15].

Texture mapping is a special case of 3-D image warping for which all pixels of the source image share a single disparity value [11]. This fact will be exploited in section 3.1 for the derivation of the pre-warping equations.

Sprites with Depth. *Sprites with depth* [17] enhance the descriptive power of traditional sprites with out-of-plane displacements per pixel. Such a technique is based on the *plane-plus-parallax* factorization [15] mentioned before. In a first step, the displacement information associated with the source image is forward mapped using a 2-D transformation to compute an intermediate displacement map. In the second pass, each pixel of the desired image is transformed by a homography (planar perspective projection) and the resulting coordinates are used to index the displacement map computed in the first pass. The retrieved displacement value is then multiplied by the epipole of the target image and added to the result of the homography. These new coordinates are used to index the color of the desired pixel.

Although such an approach may sound similar to ours at first, it differs in some fundamental aspects. Sprites with Depth are an approximation to the 3-D image warping process. Our method, on the other hand, is based on an exact factorization of the 3-D image warping equation [11], takes advantage of texture mapping hardware, uses an efficient image reconstruction strategy and naturally integrates itself with popular graphics APIs such as OpenGL [21].

View-dependent Texture Mapping. New views of a scene can be rendered by compositing multiple textures based on the observer's viewpoint, which are then mapped onto a polygonal model. In [4], a model-based stereo algorithm is used to compute depth maps from pairs of images. Once a depth map associated with a particular image has been computed, new views of the scene can be rendered using several image-based rendering techniques.

1-D Perspective Projection. Robertson [14] showed how hidden-point removal and perspective projection of height images could be performed on scanlines or columns. This approach explores the separability of perspective projection into orthogonal components. First, the image is rotated to align its lower edge with the lower edge of the viewing window. Then, a horizontal compression is applied to each scanline so that all points that may potentially occlude each other

[1] The projection of one camera's center of projection into the image plane of another camera.

fall along the same column. 1-D vertical perspective projection is applied to the columns of the intermediate image in back-to-front order, thus performing hidden-point removal. Finally, 1-D horizontal perspective projection is applied to the resulting image, incorporating compensation for the compression applied in the second step [14].

Serial Warps. Image operations such as texture mapping and image warping involve transformations among pairs of coordinates. Catmull and Smith [2] showed how affine and perspective transformations applied to planar surfaces and to bilinear and biquadratic patches can be decomposed into a series of 1-D operations over rows and columns. Later, Smith [18] showed that texture mapping onto planar quadric and superquadric surfaces, and planar bicubic and biquadratic image warps are two-pass transformable.

Serial warps suffer from a problem commonly referred to as *bottleneck*, the collapse of the intermediate image into an area much smaller than the final image [2]. Non-injective 2-D mapping may also map multiple samples to the same pixel on the screen, a situation known as *foldover* [2]. The major sources of bottlenecks are *image rotations* and *perspective distortions* [20]. In combination with rotations, perspective distortions can cause the intermediate image to twist, leading to loss of information and introducing severe artifacts in the final image [13].

3. RELIEF TEXTURE MAPPING

A *relief texture* is a texture extended with orthogonal displacements per texel, and has some interesting properties. For instance, when the viewer is far away from the represented surface, it can be rendered as a regular texture. As the viewer approaches the surface, the relief texture can be warped before being mapped onto a polygon and, when the viewer is extremely close, the relief texture can be rendered as a mesh of micro-polygons.

Image-based rendering techniques can generate very realistic views by warping images and Equation (1) concisely describes the warping process. Ideally, from a conventional rendering point-of-view, the mapping expressed by Equation (1) should be factored so to allow conventional texture mapping to be applied after the shift in the direction of the epipole. Such an approach is the opposite of the conventional plane-plus-parallax decomposition, in the sense that shifts take place prior to the homography (Figure 3), and presents several advantages. First, it can benefit from the texture mapping hardware in graphics systems to perform the final transformation and filtering. Secondly, the warp can be implemented using 1-D image operations along rows and columns, requiring interpolation between only two adjacent texels at a time. This property greatly simplifies the tasks of reconstruction and filtering of the intermediate image [5] and should allow a simple and efficient hardware implementation. Thirdly, the approach naturally integrates itself with popular graphics APIs such as OpenGL [21].

During the warp, texels move only horizontally and vertically in texture space by amounts that depend on their orthogonal displacements and on the viewing configuration. The warp implements no rotations (which are subsequently performed as part of the conventional texture mapping operation) and the resulting serial warps do not suffer from bottlenecks or from image twists. Figure 3 shows a flowchart for the relief texture-mapping algorithm resulting from such an ideal factorization. Section 3.1 explains how such a factorization is obtained.

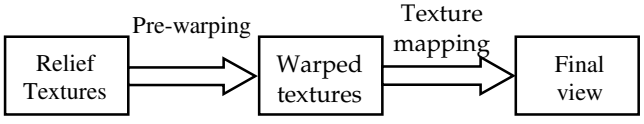

Figure 3. Relief texture mapping: pre-warping followed by standard texture mapping.

3.1 Pre-Warping Equations

In order to obtain the ideal factorization, one needs to find a pre-warp p so that the composition $m \circ p$, where m is a standard texture-mapping transformation, is equivalent to the 3-D image warp w. Thus, let $(u_i, v_i) = (u_s + \Delta u, v_s + \Delta v)$ be the intermediate coordinates obtained after shifting source pixel (u_s, v_s) by $(\Delta u, \Delta v)$. The equivalence between the composed mapping $m \circ p$ and w can be expressed as:

What coordinates (u_i, v_i) should the source pixels (u_s, v_s) have so that a view of such a flat distorted image on the source image plane from the target COP would be identical to a 3-D image warp of the source image onto the target image plane?

While perspective projection images with depth can be used as source images for such a purpose, the use of parallel projection images with depth presents some advantages. For instance, they have constant sampling density across the entire image. Also, the perpendicular relationship between sampling rays and image plane can be exploited to produce a simple and efficient rendering algorithm (Section 4.1).

Given a parallel projection camera model (Figure 4), the coordinates of a point $\dot{x}$ in Euclidean space are given by:

$$\dot{x} = \dot{C}_s + \begin{bmatrix} a_{si} & b_{si} & f_{si} \\ a_{sj} & b_{sj} & f_{sj} \\ a_{sk} & b_{sk} & f_{sk} \end{bmatrix} \begin{bmatrix} u_s \\ v_s \\ displ(u_s, v_s) \end{bmatrix} = \dot{C}_s + P'_s \vec{x}'_s$$

where vectors $\vec{a}_s$ and $\vec{b}_s$ form a basis for the plane of the source image. The lengths of these vectors are the horizontal and vertical sample spacing in the Euclidean space, respectively. $\vec{f}_s$ is a unit vector perpendicular to the plane defined by $\vec{a}_s$ and $\vec{b}_s$, $\dot{C}_s$ is the origin of the source image plane, and *displ*(u_s, v_s) is the orthogonal displacement, or height, associated with source pixel (u_s, v_s). The reprojected coordinates of $\dot{x}$ into a target perspective projection camera (Figure 5) are given by[2]

$$u_t = \frac{Au_s + Bv_s + D + C' displ(u_s, v_s)}{Iu_s + Jv_s + L + K' displ(u_s, v_s)} \quad (2a)$$

$$v_t = \frac{Eu_s + Fv_s + H + G' displ(u_s, v_s)}{Iu_s + Jv_s + L + K' displ(u_s, v_s)} \quad (2b)$$

where $A = \vec{a}_s \cdot (\vec{b}_t \times \vec{c}_t)$, $B = \vec{b}_s \cdot (\vec{b}_t \times \vec{c}_t)$, $C' = \vec{f}_s \cdot (\vec{b}_t \times \vec{c}_t)$, $D = (\dot{C}_s - \dot{C}_t) \cdot (\vec{b}_t \times \vec{c}_t)$, $E = \vec{a}_s \cdot (\vec{c}_t \times \vec{a}_t)$, $F = \vec{b}_s \cdot (\vec{c}_t \times \vec{a}_t)$, $G' = \vec{f}_s \cdot (\vec{c}_t \times \vec{a}_t)$, $H = (\dot{C}_s - \dot{C}_t) \cdot (\vec{c}_t \times \vec{a}_t)$, $I = \vec{a}_s \cdot (\vec{a}_t \times \vec{b}_t)$, $J = \vec{b}_s \cdot (\vec{a}_t \times \vec{b}_t)$, $K' = \vec{f}_s \cdot (\vec{a}_t \times \vec{b}_t)$, $L = (\dot{C}_s - \dot{C}_t) \cdot (\vec{a}_t \times \vec{b}_t)$

[2] We preserved the original notation used in [13] for easy reference by interested readers.

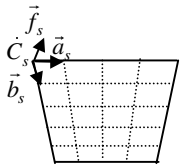

Figure 4. Parallel projection camera model.

and $\vec{c}_t$ is a vector from the target COP to the origin of the target image plane (Figure 5).

The corresponding texture mapping expressions are obtained from Equations (2a) and (2b) by letting $displ(u_s, v_s) = 0$ for all source pixels. Thus, the problem of finding the desired warp can be modeled as

$$\frac{Au_i + Bv_i + D}{Iu_i + Jv_i + L} = \frac{Au_s + Bv_s + D + C'displ(u_s, v_s)}{Iu_s + Jv_s + L + K'displ(u_s, v_s)} \quad (3a)$$

$$\frac{Eu_i + Fv_i + H}{Iu_i + Jv_i + L} = \frac{Eu_s + Fv_s + H + G'displ(u_s, v_s)}{Iu_s + Jv_s + L + K'displ(u_s, v_s)}. \quad (3b)$$

The pre-warp associated with the ideal factorization is then obtained by solving the system above for u_i and v_i:

$$u_i = \frac{u_s + k_1 displ(u_s, v_s)}{1 + k_3 displ(u_s, v_s)} \quad (4a)$$

$$v_i = \frac{v_s + k_2 displ(u_s, v_s)}{1 + k_3 displ(u_s, v_s)} \quad (4b)$$

where k_1, k_2 and k_3 are constants for the given configuration of source and target cameras and, together with $displ(u_s, v_s)$, determine the amount of change $(\Delta u, \Delta v)$ in the coordinates of the source texels. A formal proof of the 1-D nature of the pre-warping equations can be found in [13].

Such a factorization proves to have many desirable properties. In particular, the coordinates of a pixel in the intermediate image can be computed independently from each other, *i.e.*, u_i does not depend on v_s and v_i does not depend on u_s. Also, when $displ_s(u_s, v_s) = 0$ no computation is required.

The evaluation of Equations (4a) and (4b) can be reduced to two additions, two multiplications and three lookup operations by quantizing the displacement values (in a pre-processing step) and storing the reciprocal of the denominator of Equation (4a) and the expressions $k_1 displ(u_s, v_s)$ and $k_2 displ(u_s, v_s)$ in lookup tables. We have used a uniform quantization scheme in which a quantized displacement is recovered as $displ' = \min + qi * qs$, where min is the minimum displacement value, $qs = (\max - \min)/254$ is the quantization step and $qi = \text{int}((displ(u_s, v_s) - \min)/qs)$ is the quantization index. The indices were stored in the alpha channel of the relief texture with one value reserved for transparency. In practice, this uniform quantization scheme works very well and the results are virtually indistinguishable from those obtained with the actual displacement values. Moreover, such a strategy reduces the storage requirements of relief textures to essentially the same as conventional textures (the values of *min* and qs need to be saved) and it also helps to improve cache coherence, since the displacement and color data associated with a texel are always used together. This scheme uses the alpha channel of source textures only and the alpha channel of the pre-warped textures can still be used for antialiasing and transparency.

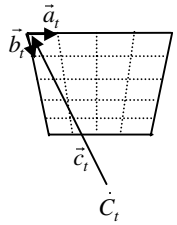

Figure 5. Pinhole camera model [11].

The Coefficients of the Pre-Warping Equations. The amount of shift $(\Delta u, \Delta v)$ to be applied to a source texel does not depend on the parameters of the target camera except for its COP [13]. Therefore, one can freely specify the parameters $\vec{a}_t$, b_t and $\vec{c}_t$ which define a temporary target camera used only for the purpose of the pre-warp and which usually differs from the virtual camera used for the visualization of the final scene. By appropriately choosing such parameters, it is possible to eliminate several of the coefficients in Equations (3a) and (3b) by forcing the corresponding scalar triple products to have the form $\vec{v} \cdot (\vec{v} \times \vec{w})$ or $\vec{w} \cdot (\vec{v} \times \vec{w})$. Such a procedure leads to a drastic simplification of the expressions used to compute coefficients k_1, k_2 and k_3. For instance, the condition $\vec{a}_t = \alpha \vec{a}_s$, $b_t = \beta b_s$ and $\vec{c}_t = \gamma(C_s - C_t)$, for nonzero $\alpha, \beta, \gamma \in \Re$, eliminates coefficients *B*, *D*, *E*, *H*, *I* and *J* and is trivially satisfied by letting source and target image planes coincide, including their origins and basis vectors (Figure 6). The subscripts of all vectors can then be dropped without risk of confusion and the coefficients of Equations (4a) and (4b) become

$$k_1 = \frac{\vec{f} \cdot (\vec{b} \times \vec{c})}{\vec{a} \cdot (\vec{b} \times \vec{c})},\ k_2 = \frac{\vec{f} \cdot (\vec{c} \times a)}{\vec{a} \cdot (\vec{b} \times \vec{c})} \text{ and } k_3 = \frac{1}{\vec{c} \cdot \vec{f}}.$$

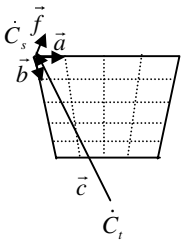

Figure 6. Parallel and perspective projection cameras sharing the same image plane (origin, $\vec{a}$ and $\vec{b}$ vectors).

Occlusion-Compatible Ordering. The COP of a parallel projection image is at infinity. Its epipole is the projection of the other camera's COP onto the plane of the parallel projection image. By similarity of triangles, whenever two samples fall along the same viewing ray, the one whose projection is closer to the epipole is also closer to the viewer (Figure 7). Thus, an occlusion-compatible order [11] (essentially a painter's algorithm) for parallel projection images with depth is obtained by warping pixels from the borders towards the epipole.

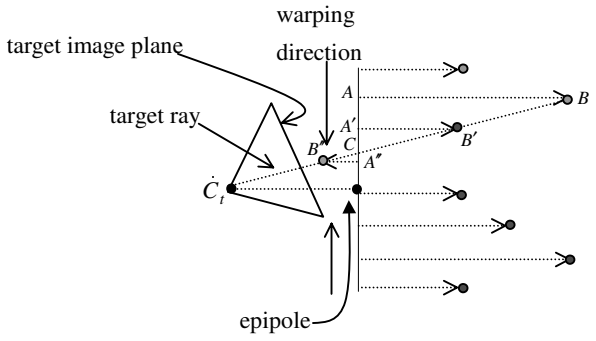

Figure 7. Triangles ABC, $A'B'C$ and $A''B''C$ are similar. Similarity of triangles guarantees that occlusion compatible order is achieved by warping from the borders towards the epipole.

3.2 Reconstructing Pre-warped Textures

Section 3.1 has shown how to determine the coordinates of infinitesimal points in the intermediate image from points in the source image. Determining these is only the beginning of the image-warping process. The more expensive step is reconstruction and resampling onto the pixel grid of the intermediate image. The simplest and most common approaches to reconstruction and resampling are splatting and meshing. Splatting requires spreading each input pixel over several output pixels to assure full coverage and proper interpolation. Meshing requires rasterizing a quadrilateral for each pixel in the NxN input texture.

The special structure of our pre-warp equations allows us to implement reconstruction and resampling as a two-pass process using 1-D transforms along rows and columns [2]. The reader should make a clear distinction between the two steps of our method: pre-warping followed by texture mapping, and the two phases used to implement the pre-warping step itself. Such phases consist of a horizontal pass and a vertical pass.

3.2.1 Two-pass Reconstruction

Assuming that the horizontal pass takes place first, the steps of a two-pass reconstruction algorithm are illustrated in Figure 8. Figure 8(a) shows two source texels *A* and *B* and their positions after the pre-warp (outlined circles). The first texel of each row is moved to its final column (Figure 8(b)) and, as the subsequent texels are warped, color and final row coordinates are interpolated during rasterization (Figure 8(c)). Fractional coordinate values (for both rows and columns) are used for filtering purposes in a similar way as described in [5]. Notice that adjacent texels are usually warped to adjacent positions and the situation shown in Figure 4-3(c) is used to stress the interpolation scheme. The warp may, however, map adjacent texels to relatively distant positions if such texels are at different sides of a depth discontinuity. Let texel *C* be right above texel *B* after all rows have been warped (Figure 8(d)). During the vertical pass, texels are moved to their final row coordinates (Figure 8(e)) and colors are interpolated (Figure 8(f)). Figure 9 illustrates the stages of the two-pass warp and reconstruction for the case of a building façade. Figure 9(a) shows a source relief texture. Figures 9(b) and 9(c) present the results of the horizontal and vertical passes, respectively. The final view of the texture-mapped polygon, whose borders are shown in red, is presented in Figure 9(d). A pseudocode for a two-pass reconstruction algorithm that disregards filtering issues for simplicity is presented in Figure 10.

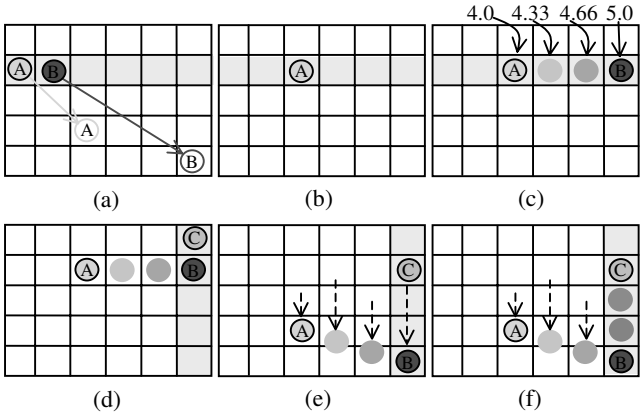

Figure 8. Warping of one texel. (a) Source texels *A* and B and their final positions after the warp. (b) The first texel of the current row is moved to its final column. (c) Next texel is moved to its final column and color and final row coordinates are interpolated during rasterization. (d) After all rows have been warped, texel *C* is adjacent to texel *B*. (e) Along each column, texels are moved to their final rows. (f) Color is interpolated during rasterization.

There are advantages in computing both coordinates of pre-warped texels in the first step of the algorithm. For instance, it avoids nonlinear distortions in the final image that would otherwise be introduced if row coordinates were computed during the second pass using interpolated displacement values [13].

We have compared the results produced by this algorithm with the results of rendering relief textures as meshes of micropolygons. The results are essentially the same in most cases. Improper color interpolation may happen across depth discontinuities, where no information about the surface is available, and are the major source of artifacts in images produced by two-pass 1-D

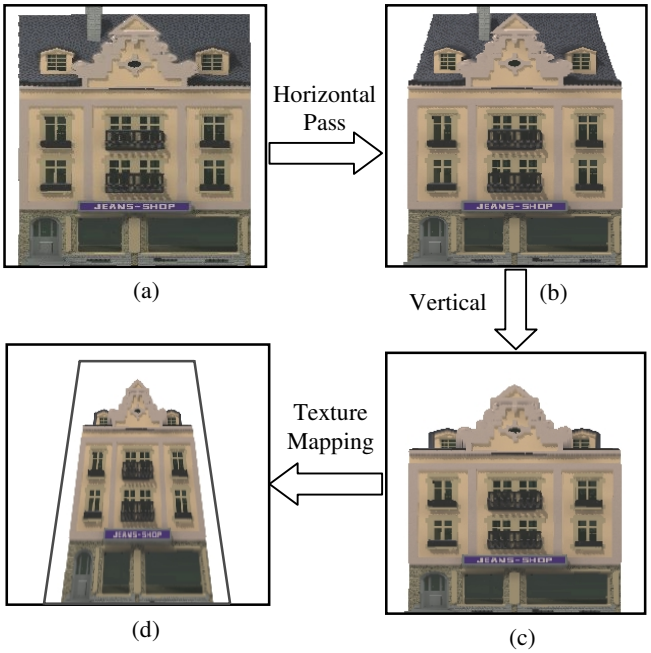

Figure 9. Stages of the relief texture-mapping algorithm. (a) Source relief texture. (b) Image produced by the horizontal pass. (c) Pre-warped texture obtained after the vertical pass. (d) Final view, showing the borders of the texture-mapped polygon.

reconstruction strategies. In practice, however, depth discontinuities are frequently associated with either smooth color changes or sharp color transitions matching the discontinuities. In both cases, the results produced by the 1-D approach are similar to the ones obtained with a 2-D warp followed by the rasterization of 2-D micropolygons. The examples shown in this paper and the accompanying animations were rendered with the two-pass reconstruction algorithm described.

```
get U_in, V_in, C_in, D_in
U_next = Equation_5a(U_in,D_in)
V_next = Equation_5b(V_in,D_in)
for (U_out = integer(U_prev+1); U_out ≤ U_next; U_out++)
        linearly interpolate C_out between C_prev and C_in
        linearly interpolate V_out between V_prev and V_in
        put C_out, V_out at U_out
U_prev=U_next; V_prev=V_next; C_prev=C_in
```

Figure 10. Pseudocode for a first-pass left-to-right horizontal warp and resampling of one texel with coordinates (U, V), color C and displacement D. No antialiasing computed for simplicity.

3.2.2 Pipelined Reconstruction

The overwriting of texels during the first pass may cause self-occlusions. Although bottlenecks [2] are not an issue during the pre-warp step and, in practice, self-occlusions seem not to introduce noticeable artifacts in the pre-warped textures, we present a solution that is capable of handling an arbitrary number of foldovers and that does not require depth comparison. It consists of interspersing the horizontal and vertical warps and is related to the work described in [10]. As before, assume the horizontal pass is completed first and the rows are processed in occlusion-compatible order. As the horizontal warp produces each intermediate texel, this is immediately rasterized into the appropriate column. Since each vertical warp receives and processes its texels in occlusion-compatible order, correct visibility is preserved in the output. Also, because each texel is processed immediately after its generation, no information is overwritten and self-occlusions are avoided. The steps of the algorithm are illustrated in Figure 11, where gray circles represent the texels previously warped to the corresponding columns.

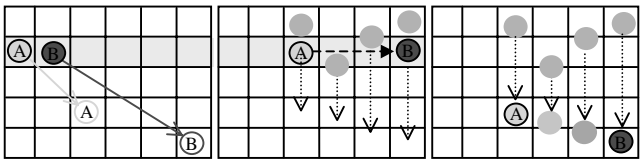

Figure 11. Pipelined reconstruction: (a) Two adjacent texels and their final positions. (b) and (c) Horizontal and vertical interpolation interspersed.

4. MODELING

Relief textures can be used as modeling primitives by simply instantiating them in a scene in such a way that the reprojected surfaces match the surfaces of the objects to be modeled. During the pre-warp, however, samples may have their coordinates mapped beyond the limits of the original texture. This corresponds, in the final image, to have samples projecting outside the limits of the polygon to be texture-mapped (Figure 12 (left)). The occurrence of such situations depends on the viewpoint and on the size of the displacements. This is similar to what happens when a light field [9] consisting of a single light slab is viewed from oblique angles.

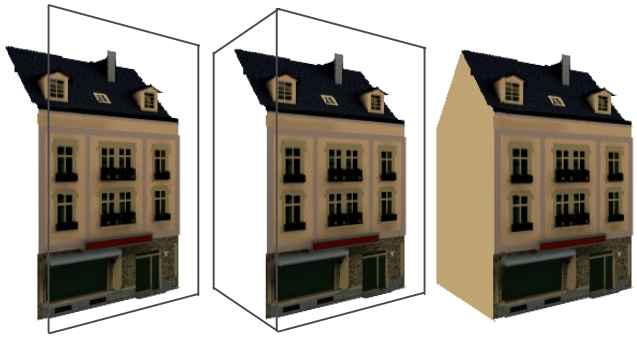

Figure 12. An extra quadrilateral is used to texture map outliers (center). Final view rendered with additional sidewall (right).

The problem of incomplete views can be overcome if extra perpendicular polygons are texture-mapped with the outliers. This situation is illustrated in Figure 12 (center). The final result, rendered with an additional sidewall (pentagon), is shown in Figure 12 (right). The details of the technique will be explained next, in the context of the more general problem of rendering three-dimensional objects from arbitrary viewpoints.

4.1 Object Representation

Several researchers have used image-based techniques to represent objects [9] [7] [8] [16] [12]. Relief texture mapping can also be used to render complex three-dimensional shapes. Figure 13 shows a relief texture representation of an object originally modeled with 35,280 polygons. It consists of six relief textures acquired from the faces of the object's bounding box. New views of the object can be obtained by pre-warping these textures and mapping the resulting images onto the faces of the box. But just warping each relief texture to its original face of the box is not enough to produce the desired result. Some samples may project onto other faces, depending on the viewpoint (Figure 14).

One solution to this problem is to pre-warp adjacent faces to the desired ones. The perpendicular orientation between faces allows such mappings to be performed using the same pre-warping equations (Equations (4a) and (4b)). The concept will be explained in 2-D. Its generalization to 3-D is straightforward. Figure 14 shows a division of the object space into numbered regions. If the viewer is in an odd region, the three closest faces are classified as *front*, *left*, and *right* with respect to the viewpoint.

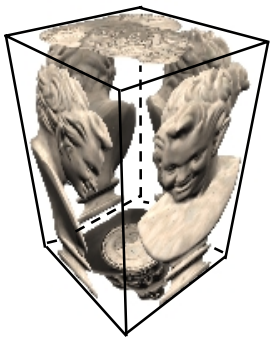

Figure 13. Object represented by six relief textures associated with the faces of a bounding box.

Thus, for instance, if the viewer is in region (1), face *a* is *front*, face *d* is *left*, and face *b* is *right*. In this case, faces *left* and *right* are pre-warped to the image plane of *front*. Then *front* is pre-warped to its own image plane, overwriting all samples except the ones intended to fill holes. If, however, the viewer is in an even region, the two closest faces are classified as *left* and *right*. For instance, if the viewer is in region (6), face *c* is *left* and face *d* is *right*. *left* is pre-warped to the image plane of *right*, then *right* is pre-warped to its own image plane. Likewise, *right* is pre-warped to the image plane of *left*, and then *left* is pre-warped to its own image plane. Notice that at most three polygons (in the full 3-D version of the algorithm) need to be displayed.

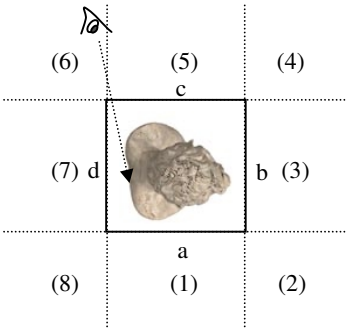

Figure 14. Samples from one face can project onto another. Letters identify the faces, and numbers identify regions used to define the faces that should be pre-warped from each region.

The perpendicular orientation between adjacent faces can be exploited to pre-warp a face to its adjacent image plane as if it were the adjacent face itself. When the viewer is in an odd region, the displacement values associated with *left* and *right* are converted to column indices for *front*, while their column indices can be used as displacement for *front* (Figure 15). Thus, *left* and *right* can be pre-warped to *front* as if they were *front* themselves. The even region is similar.

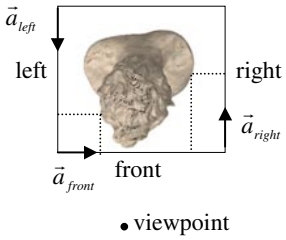

Figure 15. Height values from *left* and *right* become columns for *front*. Columns from *left* and *right* become height for *front*.

Figure 16 shows the statue rendered as two texture-mapped quadrilaterals (*left* and *right*), whose boundaries are shown to the right. The corresponding pre-warped textures are shown in Figure 17 and provide a clear illustration of the factorization of the planar perspective, which is compensated by the texture map stage of the warp.

4.1.1 Handling Surface Discontinuities

Treating relief textures as continuous surfaces may not be desirable in some situations. Improper reconstruction of originally non-connected surfaces may lead to the occurrence of "skins". The assumption about surface continuity can be relaxed if surfaces that would otherwise be rendered as "skins" had been appropriately sampled by adjacent relief textures. In this case, texels belonging to non-connected surfaces should not be interpolated during the pre-warp. A simple way to achieve this is to use depth thresholds to identify and mark such discontinuities during a pre-processing step. Figure 18 shows the rendering of a rat before and after skin removal. In the accompanying videotape, the skins between the façade and the roof of the *jeans shop store* were removed and the resulting hole was seamlessly filled by adding an extra conventionally texture-mapped polygon.

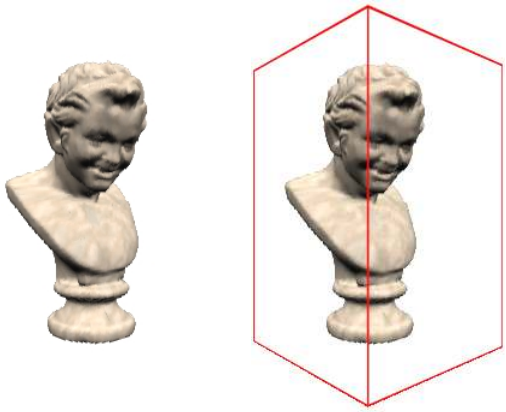

Figure 16. View of the statue (left) obtained by texture mapping two quads, whose boundaries are shown to the right.

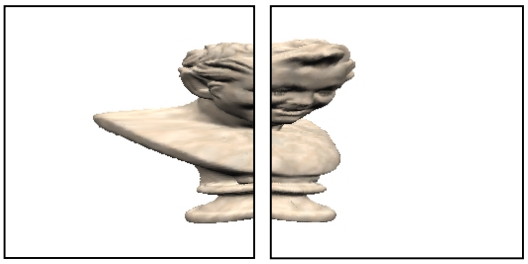

Figure 17. Pre-warped textures used to produce Figure 16.

4.2 Correct Occlusion

The relief texture-mapping algorithm, as described so far, does not handle interpenetrating polygons appropriately. Thus, for example, consider intersecting a planar polygon with the bounding box used to represent the statue shown in Figure 13. Since the intersection between two polygons defines a straight line, the resulting occlusion pattern will not match the perceived depth of the associated relief textures. In order to solve this problem, corrected depth values accounting for the perceived off-the-plane displacements must be computed (Figure 19). Thus, let $\dot{x}$ be a point in 3-space associated with texel *t,* whose coordinates in the source texture are (u_s, v_s). The Z coordinate of $\dot{x}$ in camera space when observed from a virtual COP $\dot{C}$ is given by

$$Z_{\dot{x}} = c_1 + u_s c_2 + v_s c_3 + displ(u_s, v_s) c_4$$

where $c_1 = \vec{c} \cdot \vec{n}$, $c_2 = \vec{a}_s \cdot \vec{n}$, $c_3 = \vec{b}_s \cdot \vec{n}$ and $c_4 = \vec{f}_s \cdot \vec{n}$ are

Figure 18. Renderings of a rat before (a) and after (b) surface discontinuity identification.

constants for a given viewing configuration, $\vec{n}$ is the unit vector normal to the image plane of the virtual camera, $\vec{c} = \dot{C}_s - \dot{C}$, and $\dot{C}_s$, $\vec{a}_s$, $\vec{b}_s$ and $\vec{f}_s$ are the camera parameters associated with the relief texture. Let (u_j, v_j) be the coordinates of texel t' obtained after pre-warping t. Notice that the perceived depth at t' is $Z_{\dot{x}}$ and such value can be interpolated along rows and columns in the same way as described for color in section 3.2. Alternatively, one can compute and interpolate only the difference Δz between the actual polygon depth at t' and its perceived depth, which can be encoded using a smaller number of bits. Since t' is on the polygon to be texture mapped, its Z coordinate in the virtual camera space can be expressed as $Z_{t'} = c_1 + u_j c_2 + v_j c_3$. During the pre-warp, Δz values can be linearly interpolated along rows and columns. The interpolated values can be used to compute the amount by which the depth buffer must be changed to produce correct visibility. Figure 19 shows the statue rendered with an interpenetrating polygon seen from different distances. In this example, Δz was interpolated and the resulting values were quantized using the same strategy described in section 3.1 before being used to modulate the depth buffer.

Figure 19. Depth-correction using 8-bit quantized Δz values.

4.3 Multiresolution

Image pyramids have been long used in computer graphics for antialiasing [19]. Representing relief textures using fixed resolution causes a constant amount of work to be carried out during the pre-warp, independently of the number of pixels covered on the screen. The use of *relief texture pyramids* can be used not only to reduce aliasing but also to keep the warping cost proportional to the texture contribution to the final image.

Level i of a relief texture pyramid is constructed by averaging color and depth data associated to groups of $2^i x 2^i$ adjacent texels from the highest resolution relief texture (level zero). The lengths of vectors $\vec{a}$ and $\vec{b}$ are doubled from level i to level $i+1$ in order to compensate for the halving of the number of texels in each dimension, so that the spatial coverage of the relief texture remains unchanged. Figure 20 shows a statue rendered using the first four levels of a texture pyramid.

Although mip-mapping is frequently used in computer graphics to reduce aliasing artifacts introduced by texture minification, bilinear interpolation is the preferred image resampling strategy during the texture mapping stage of the relief texture-mapping algorithm. It produces sharper images, is less prone to undesirable blurring due to polygon orientation and is computationally less expensive than trilinear interpolation. An in-depth discussion of this subject can be found in [13].

Figure 20. Textured LODs obtained by relief texture mapping the first four levels of a relief texture map pyramid. Relief texture resolution: 256x256 texels (left) down to 32x32 texels (right).

4.4 Modeling Immersive Environments

Relief texture mapping can be used not just to represent objects, but complete environments. This is useful for the many applications of computer graphics that require immersive virtual environments. Moreover, the relief textures used to represent a whole environment can be generated nearly automatically, making this an elegant method of representing acquired real scenes.

In our experiments, a laser rangefinder and a digital camera were used to scan the environment. The resulting data were then projected onto relief textures instantiated at user-defined locations (Figure 22 (left)). Registration is naturally enforced by the depth information associated with the samples (Figure 22 (right)).

Figure 23 shows two renderings of a partial model of Sitterson Hall's reading room obtained using the three relief textures depicted in Figure 22 and an extra polygon representing the ceiling. Notice the parallax effect that allows the plaque on the wall to become visible in the image to the right.

5. RESULTS

For a typical 256x256-texel relief texture mapped onto a single quadrilateral (*e.g.*, Figure 9) using the two-pass approach described in Section 3.2.1, the current software prototype, written in C++, achieves an average frame rate of 9.42 frames per second. Such measurements were performed on a Pentium II PC running at 400MHz with an Intergraph graphics accelerator (Intense 3D RealiZm II VX113A-T) with 16 MB of texture memory and 16MB of frame buffer memory. The final view of the surface was displayed on a 512x512-pixel window. The percentage of the rendering time spent with pre-warping and resampling, loading pre-warped textures into texture memory, and the actual texture mapping operation are shown in Table 1. Notice that, since the pre-warping and resampling operations dominated the rendering time, one can expect a considerable speedup from a hardware implementation of the algorithm. Also notice that pre-warping cost is independent of the dimensions of the output window.

When a large number of texels have zero displacement, such as the case of the brick texture shown in Figure 21, speed-ups of over 100% were verified in the current software prototype by just skipping the unnecessary transformations.

Table 1: Percentage of the average rendering time associated with the steps of the relief texture-mapping algorithm (one relief texture mapped onto one quadrilateral).

Pre-warping and resampling	Loading warped textures into texture memory	Actual texture mapping operation	Others
94.10%	2.65%	0.066%	3.18%

6. LIMITATIONS

A relief texture is a single-layer image representation. If multiple layers of surfaces are required (*e.g.*, in the case of objects containing holes), alternative representations such as LDI's [17] or image-based objects [12] should probably be preferred. Although parallel projection LDIs can be rendered in occlusion compatible order, the existence of multiple samples along each ray introduces ambiguity about which samples should be connected, making 1-D interpolation and mesh-based reconstruction impractical. In many cases, the rendering of several layers each consisting of individual relief textures can be used to achieve similar results. Objects, on the other hand, consist of six perpendicular relief textures and such a representation is equivalent to a multi-layer representation [13].

In some applications, it may not be possible to constrain the viewpoint from crossing the plane of a relief texture. In such a case, the relief texture-mapped polygon will not be rendered even if the represented surface may still be visible. In these cases, relief textures should to be rendered as meshes of micro-polygons.

Ideally, only texels that effectively contribute to some screen fragments should be pre-warped. While such a strategy would make optimal use of the pre-warping, in practice this is not feasible, since it would require an inverse mapping. Alternatively, one can consider selecting the most appropriate level of a relief texture pyramid by using the projected area (in screen space) of the quadrilateral to be texture mapped.

7. SUMMARY AND FUTURE WORK

We have presented an extension to texture mapping that supports the representation of 3-D surface details and view motion parallax. It results from an exact factorization of the 3-D image warping equation [11] into a pre-warp followed by conventional texture mapping. We have shown that, from a conventional rendering perspective, such a new factorization presents several advantages over the conventional plane-plus-parallax factorization. The simple pre-warping functions allow surface reconstruction (color and depth) to be performed in 1-D. This property should allow a simple and efficient hardware implementation.

One important area for investigation is the design of efficient hardware implementations for relief texture mapping using our pre-warping functions. Adding this pre-warping capability to the texture memory of a graphics accelerator may allow this approach to become as commonly used as conventional texture mapping. Automatic acquisition of relief textures from 3-D environments is another important area for exploration. Other avenues for exploration involve the use of normal maps [6] [3] for view-dependent lighting and the use of relief textures for geometry simplification.

Acknowledgements

We would like to thank Chris Dwyer, Anselmo Lastra, Steve Molnar, Lars Nyland, Jason Smith and Mary Whitton for their assistance and suggestions, and the anonymous reviewers for their insightful comments. Special thanks go to Frederick P. Brooks, Jr. for his detailed critique of an earlier draft of this paper. Cássio Ribeiro designed *Relief Town*. The UNC IBR group provided the reading room data set. De Espona Infográfica provided the other models.

This work was sponsored by CNPq/Brazil under Process # 200054/95, DARPA under order # E278 and NFS under grant # MIP-9612643.

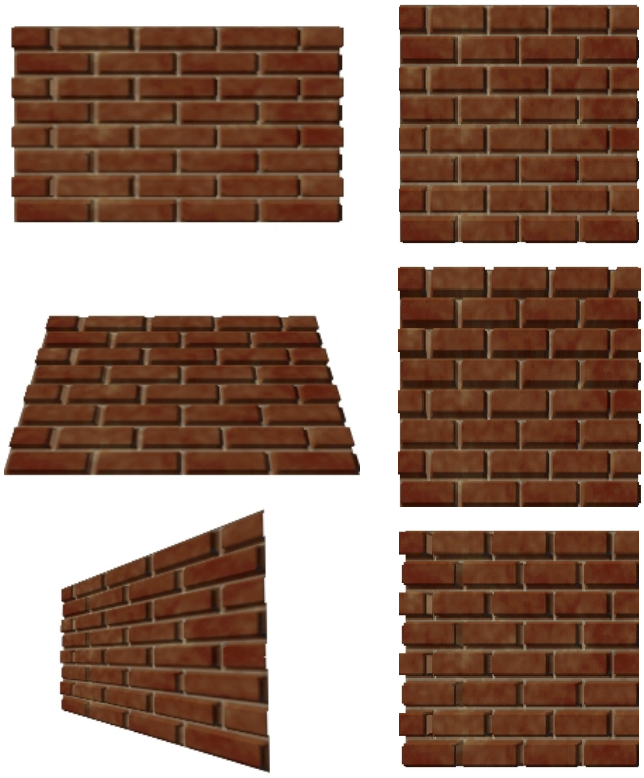

Figure 21. Three views of a relief texture-mapped brick wall. The images on the left show one quadrilateral texture-mapped with the corresponding pre-warped images shown to the right. Brick texels have zero displacement and, therefore, do not move.

References

[1] Catmull, E. A Subdivision Algorithm for Computer Display of Curved Surfaces. Ph.D. Dissertation, Department of Computer Science, University of Utah, December 1974.

[2] Catmull, E., Smith, A. 3D Transformations of Images in Scanline Order. *Proc. SIGGRAPH 80* (Seattle, Washington, July 14-18, 1980), pp. 279-285.

[3] Cohen, J., Olano, M., Manocha, D. Appearance-Preserving Simplification. *Proc. SIGGRAPH 98* (Orlando, FL, July 19-24, 1998), pp. 115-122.

[4] Debevec, P., Taylor, C., Malik, J. Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. *Proc. SIGGRAPH 96* (New Orleans, LA, August 4-9, 1996), pp. 11-20.

[5] Fant, Karl. A Nonaliasing, Real-Time Spatial Transform Technique. IEEE CG&A, Vol. 6, No 1, January 1986, pp. 71-80.

[6] Fournier, A. Normal Distribution Functions and Multiple Surfaces. *Graphics Interface '92 Workshop on Local Illumination*. pp. 45-52.

[7] Gortler, S., et al.. The Lumigraph. *Proc. SIGGRAPH 96* (New Orleans, LA, August 4-9, 1996), pp. 43-54.

[8] Grossman, J., Dally, W. Point Sample Rendering. *Proceedings of the 9th Eurographics Workshop on Rendering*. Vienna, Austria, June 1998. *Rendering Techniques '98*, Springer-Verlag, pp. 181-192.

[9] Levoy, M., Hanrahan, P. Light Field Rendering *Proc. SIGGRAPH 96* (New Orleans, LA, August 4-9, 1996), pp. 31-42.

[10] Max, N. A One-Pass Version of Two-Pass Image Resampling. Journal of Graphics Tools, Vol. 3, No. 1, pp. 33-41.

[11] McMillan, L. An Image-Based Approach to Three-Dimensional Computer Graphics. Ph.D. Dissertation. UNC Computer Science Technical Report TR97-013, April 1997.

[12] Oliveira, M., Bishop, G. Image-Based Objects. Proceedings of 1999 ACM Symposium on Interactive 3D Graphics. pp. 191-198.

[13] Oliveira, M. Relief Texture Mapping. Ph.D. Dissertation. UNC Computer Science Technical Report TR00-009. March 2000. http://www.cs.unc.edu/~ibr/pubs/oliveira-diss/TR00-009.pdf.

[14] Robertson, P. Fast Perspective Views of Images Using One-Dimensional Operations. IEEE CG&A, vol. 7, pp. 47-56, Feb. 1987.

[15] Sawhney, H. 3D Geometry from Planar Parallax. In IEEE CVPR'94, pages 929-934. IEEE Computer Society, Seattle, Washington, June 1994.

[16] Schaufler, G. Per-Object Image Warping with Layered Impostors. *Proceedings of the 9th Eurographics Workshop on Rendering*. Vienna, Austria, June 1998. *Rendering Techniques '98*, Springer-Verlag, pp. 145-156.

[17] Shade, J., et al. Layered Depth Images. *Proc. SIGGRAPH 98* (Orlando, FL, July 19-24, 1998), pp. 231-242.

[18] Smith, Alvy Ray. Planar 2-Pass Texture Mapping and Warping. *Proc. SIGGRAPH 87* (Anaheim, CA, July 27-31, 1987), pp. 263-272.

[19] Williams, L. Pyramidal Parametrics. *Proc. SIGGRAPH 83* (Detroit, MI, July 25-29, 1983), pp. 1-11.

[20] Wolberg, George. Separable Image Warping with Spatial Lookup Tables. *Proc. SIGGRAPH 89* (Boston, MA, July 31-4 August, 1989), pp. 369-378.

[21] Woo, M., et al. OpenGL Programming Guide. 2nd edition. Addison Wesley, 1997.

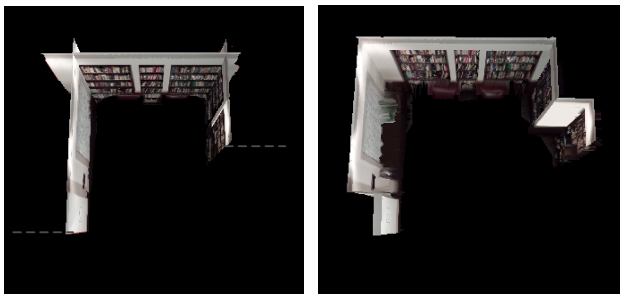

Figure 22. Modeling of an immersive environment using three relief textures (left). The dashed lines represent two extra polygons used to capture outliers. Registration is enforced by the depth information (right).

Figure 23. Sitterson Hall's reading room rendered using relief texture mapping. Notice the parallax effect that allows the plaque on the wall to become visible in the image to the right. The partial model of the reading room used to render these images consists of three 256x256 relief textures and six quadrilaterals.