# Interactive Geometric Image Transformation Using Texture Mapping

Carl Korobkin
Silicon Graphics Computer Systems*

## Abstract

General-purpose computer graphics workstation technology is rapidly displacing more traditional "black box" image processing solutions. One core computer graphics technology that is driving this trend is *texture mapping*[1][4].

In recent years, the technique of texture mapping has moved from the domain of software rendering systems to that of high performance general-purpose graphics workstation hardware. A graphics pipeline incorporating a texture mapping (image resampling) engine is a state-of-the-art geometric image transformation engine.

Shown here is how an advanced form of texture mapping, called *projective texture*[10], can be used to interactively perform a full range of traditional geometric image processing tasks as well as enabling a wide range of new techniques. These include arbitrary projection of two-dimensional images onto geometry, realistic lighting/transmission effects, and generation of shadows using shadow maps[12]. These effects are obtained in real time using hardware that performs correct projective texture mapping.

**CR Categories and Subject Descriptors:** I.2.0 **[Image Processing]:** Geometric Image Transformation; I.3.3 **[Computer Graphics]:** Picture/Image Generation; I.3.7 **[Computer Graphics]:** Three-Dimensional Graphics and Realism - *color, shading, shadowing, and texture*

**Additional Key Words and Phrases:** warping, texture mapping

## 1 Introduction

In the domain of image processing, one of the most generic and common categories of operations is *geometric transformation* of imagery. Given a discrete input image, a geometric transformation operator produces a discrete output image that is the input image spatially translated, rotated, scaled, nonlinearly warped or viewed from an alternate perspective.

From a computational standpoint, the process of geometrically transforming an image is essentially one of calculating linear and non-linear input image pixel addresses for integer output image pixel addresses. Such computations are easily described under the framework of a Cartesian coordinate system and vector-space representation[9]. Since this mapping process can produce non-integer input image addresses for integer output image addresses, *geometric resampling* of the stored input image array is required.

*2011 N. Shoreline Blvd., Mountain View, CA 94043

What the image processing community refers to as geometric image transformation is equivalent to what the computer graphics community calls *texture mapping*. Texture mapping might alternately be referred to as *image mapping* – in its most basic form, a two-dimensional image (the texture) is draped onto a two- or three-dimensional polygon in a scene. The color of each pixel on the surface of the output polygon is modified by some corresponding value(s) in the input image; this correspondence is established by a series of steps including a resampling of the input image[5]. More complex geometry in a scene may be approximated by surfaces composed of facets of polygons – these may conveniently described in some higher-order surface representation such as non-uniform rational B-splines (NURBS).

In the context of texture mapping, geometric image transformation amounts to a mapping between the underlying geometry (representing the output image space) and a stored image array (texture) (representing the input image space). With a general-purpose graphics engine, this output geometry is three-dimensional, viewed in perspective, and interactively manipulated.

## 2 Perspective-Correct Texture Mapping

With standard texture mapping, an image is applied to a polygon by assigning *texture coordinates* to the polygon's vertices. These coordinates are interpolated across the surface of the polygon, serving as an index into the texture image array for each of the polygon's pixels.

Producing an image of a three-dimensional scene requires finding the projection of that scene onto a two-dimensional screen. If the image of the three-dimensional scene is to appear realistic, then the projection from three to two dimensions must be a perspective projection.

For a scene comprised of polygons, the projected vertices of these polygons determine boundary edges of the geometry. Scan conversion uses iteration to enumerate pixels on the screen that are covered by each polygon. This iteration in the plane of projection introduces a homogeneous variation in the texture coordinates. If the homogeneous variation is ignored in favor of a simpler linear iteration, incorrect images are produced that can lead to objectionable effects such as texture "swimming" during scene animation[5]. Correct interpolation of texture coordinates requires each to be divided by a common denominator for each pixel of a projected texture mapped polygon[6].

### 2.1 Projective Textures

Projective texture mapping is a generalization of the standard technique. Here, texture coordinates are assigned to polygon vertices as a result of a projection rather than being assigned fixed values. A texture is mapped onto a surface via a projection, after which the surface is projected onto a two dimensional viewing screen. This provides for the notion of the texture (image) being associated with

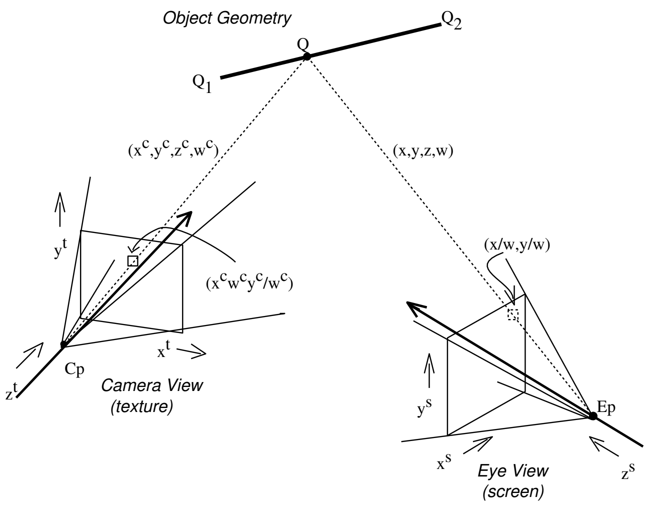

Figure 1. Object geometry in the camera and clip coordinate systems.

some "virtual camera" position within the scene; imagery may be projected (reprojected) onto an arbitrarily oriented surface, which is then viewed from some viewpoint.

Since two-dimensional images are typically views of three-dimensional scenes from the physical perspective of a camera imaging the scene, this cabability has huge implications in many application areas. It turns out that handling this situation during texture coordinate iteration is essentially no different from the more usual case in which a texture is mapped linearly onto a polygon.

## 2.2 Mathematics of Projective Textures

To aid in describing the iteration process, four coordinate systems are introduced. The *clip* coordinate system is a homogeneous representation of three-dimensional space, with $x$, $y$, $z$, and $w$ coordinates. The origin of this coordinate system is the viewpoint. The term clip coordinate system is used because it is this system in which clipping is often carried out. The *screen* coordinate system represents the two-dimensional screen of the viewer (by default, assumed to be attached to the physical display) with two coordinates. These are obtained from clip coordinates by dividing $x$ and $y$ by $w$, so that screen coordinates are given by $x^s = x/w$ and $y^s = y/w$ (the $s$ superscript indicates screen coordinates). The *camera* coordinate system is a second homogeneous coordinate system with coordinates $x^c$, $y^c$, $z^c$, and $w^c$; the origin of this system is at the camera projection source. Finally, the *texture* coordinate system corresponds to a screen of the texture associated with the camera. Texture coordinates are given by $x^t = x^c/w^c$ and $y^t = y^c/w^c$ Given $(x^s, y^s)$, a point on a scan-converted polygon, our goal is to find its corresponding texture coordinates, $(x^t, y^t)$.

Figure 1 shows a line segment in the clip coordinate system and its projection onto the two-dimensional screen. This line segment represents a span between two edges of a polygon. In clip coordinates, the endpoints of the line segment are given by

$$Q_1 = (x_1, y_1, z_1, w_1) \qquad \text{and} \qquad Q_2 = (x_2, y_2, z_2, w_2).$$

A point $Q$ along the line segment can be written in clip coordinates as

$$Q = (1-t)Q_1 + tQ_2 \tag{1}$$

for some $t \in [0,1]$. In screen coordinates, we write the corresponding projected point as

$$Q^s = (1-t^s)Q_1^s + t^s Q_2^s \tag{2}$$

where $Q_1^s = Q_1/w_1$ and $Q_2^s = Q_2/w_2$.

To find the camera coordinates of $Q$ given $Q^s$, we must find the value of $t$ corresponding to $t^s$ (in general $t \neq t^s$). This is accomplished by noting that

$$Q^s = (1-t^s)Q_1/w_1 + t^s Q_2/w_2 = \frac{(1-t)Q_1 + tQ_2}{(1-t)w_1 + tw_2} \tag{3}$$

and solving for $t$. This is most easily achieved by choosing $a$ and $b$ such that $1 - t^s = a/(a+b)$ and $t^s = b/(a+b)$; we also choose $A$ and $B$ such that $(1-t) = A/(A+B)$ and $t = B/(A+B)$. Equation 3 becomes

$$Q^s = \frac{aQ_1/w_1 + bQ_2/w_2}{(a+b)} = \frac{AQ_1 + BQ_2}{Aw_1 + Bw_2}. \tag{4}$$

It is easily verified that $A = aw_2$ and $B = bw_1$ satisfy this equation, allowing us to obtain $t$ and thus $Q$.

Because the relationship between camera coordinates and clip coordinates is affine (linear plus translation), there is a homogeneous matrix $M$ that relates them:

$$Q^l = MQ = \frac{A}{A+B}Q_1^l + \frac{B}{A+B}Q_2^l \tag{5}$$

where $Q_1^l = (x_1^c, y_1^c, z_1^c, w_1^c)$ and $Q_2^l = (x_2^c, y_2^c, z_2^c, w_2^c)$ are the camera coordinates of the points given by $Q_1$ and $Q_2$ in clip coordinates.

We finally obtain

$$\begin{aligned} Q^t &= Q^l/w^c \\ &= \frac{AQ_1^l + BQ_2^l}{Aw_1^c + Bw_2^c} \\ &= \frac{aQ_1^l/w_1 + bQ_2^l/w_2}{a(w_1^c/w_1) + b(w_2^c/w_2)}. \end{aligned} \tag{6}$$

Equation 6 gives the texture coordinates corresponding to a linearly interpolated point along a line segment in screen coordinates. To obtain these coordinates at a pixel, we must linearly interpolate $x^c/w$, $y^c/w$, and $w^c/w$, and divide at each pixel to obtain

$$x^c/w^c = \frac{x^c/w}{w^c/w} \qquad \text{and} \qquad y^c/w^c = \frac{y^c/w}{w^c/w}. \tag{7}$$

(For an alternate derivation of this result, see [6].)

If $w^c$ is constant across a polygon, then Equation 7 becomes

$$s = \frac{s/w}{1/w} \qquad \text{and} \qquad t = \frac{t/w}{1/w}, \tag{8}$$

where we have set $s = x^c/w^c$ and $t = y^c/w^c$. Equation 8 governs the iteration of texture coordinates that have simply been assigned to polygon vertices. It still implies a division for each pixel contained in a polygon. The more general situation of a projected texture implied by Equation 7 requires only that the divisor be $w^c/w$ instead of $1/w$.

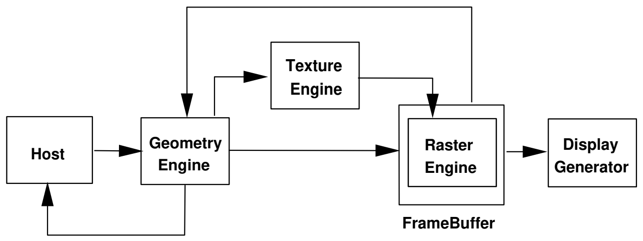

Figure 2. RealityEngine Graphics Pipeline

### 2.3 Compound Perspective Warping

A wide range of interactive geometric image transformations is realizable with graphics pipeline capable of

- fast transformation of 3-D polygonal surface geometry, and
- high-quality projective texture applied to these surfaces.

*Compound perspective warping* is the super-set of linear transformation, spatial warping, and perspective and is comprised of

- perspective from texture to geometry ($w^c$ term),
- spatial warping approximated by polygonal meshing, and
- perspective from geometry to the viewer ($w$ term).

Affine transformations are descibed when $w^c$ and $w$ are unity and the texture (image) is mapped to a first-order surface. For example, for a rectangular image to be transformed, the output image can be represented as a single flat 4-sided polygon The input image is mapped to this polygon by assigning texture coordinates to its vertices as an orthogonal projection. Subsequent rotates, translates, and scales of the output polygon with the input image "pinned" to it transform the image.

With a *global polynomial transformation*, spatial warping is implemented by defining a correspondence between a uniform polygonal mesh (representing the original image) and a warped mesh (representing the warped image)[8]. A *piecewise linear polynomial transformation* may be employed to account for local image geometric distortions[14].

All other geometric transformations are described by all possible combinations of the above with unity and non-unity $w^c$ and $w$ terms.

## 3 A High-Performance Graphics Workstation Architecture for Image Processing

The Silicon Graphics Onyx RealityEngine is representative of a general-purpose computer graphics workstations equipped with advanced geometric transformation and texture mapping hardware. The fundamental architecture of the RealityEngine graphics pipeline is illustrated in Figure 2. There are three principle processing subsystems.

The *Geometry Engine* subsystem is responsible for all affine and non-affine transformations of object geometry represented as polygon, vector, and point vertices. Each vertex is described in terms of its Cartesian coordinates (x,y,z,w), texture coordinates ($x^c$,$y^c$,$z^c$,$w^c$), color (r,g,b), transparency (alpha), and orientation (normal). Separate homogeneous (4x4) matrix transformation stacks are maintained for Cartesian and texture coordinates, providing for a fully flexible and intuitive camera imaging model supporting compound projective texture mapping. The Geometry Engine encapsulates 1.5 gigaflops of single-precision floating capacity and can sustain transformation rates of up to 975K meshed and textured polygons per second.

The *Texture Engine* subsystem is a high-speed resampling unit that supports both *mipmapped*[13] and non-mipmapped types of texture filtering. In non-mipmapped mode, the resampling method is point sampling, bilinear interpolation, or bicubic interpolation. In mipmapped mode, the resampling trilinear or quadlinear. The Texture Engine is capable of a sustained fill rate of up to a 320 million pixels per second for anti-aliased, trilinear mipmapped, z-buffered, lit, smooth shaded polygons. The Texture Engine interpolates texture coordinates as given by Equation 7, thus fully supporting projective texture mapping.

The *Raster Engine* subsystem is the scan conversion engine. It contains the framebuffer memory, texture storage memory (up to 16 megabytes) and all the hardware responsible for color blending, subpixel anti-aliasing, fogging, lighting, and hidden surface removal. Color computation is maintained at a full 48 bits per pixel (12 bits each for red, green, blue, and alpha). For hidden surface removal, a z-buffer with a full 32 bits of precision is maintained.

## 4 Applications

Real-time projective texture mapping is an extremely powerful technology that is widely applicable. Presented here are some example applications developed on the RealityEngine. All images are screen captures from fully interactive programs (i.e. camera projectors can be moved interactively, scenes can be flown through at 15 to 30 Hz frame rates).

### 4.1 3-D Terrain Reconstruction and Fly-Over

Digital terrain elevation data is used to describe a polygon mesh. Orthorectified and registered satellite imagery is draped over the terrain geometry using non-projective texture mapping. ($w^c = 1$). Figure 3 shows a flyover of Yellowstone National Park; Performer is used to create a polygon mesh from a DTEM (digtal terrain elevation model) and apply the corresponding LandSat image (as texture).

In 2-D, this application reduces to a more conventional roam. Dynamic image look-ahead and texture resource tiling allows for imagery of any size to be roamed on at frames rate of up to 60Hz. Roaming here is any combination of rotates, translates, and scales (zooms). In this scenario, the underlying geometry can be a simple 4-sided polygon and the texture and viewer projections are typically both orthographic ($w^c = 1$ and $w = 1$).

### 4.2 Real-Scene Reconstruction and Fly-Through

Projective texture mapping may be used to reproject imagery into a scene in order to reconstruct and walk (or fly) through the original scene. This requires a knowledge of the camera positions which acquired the imagery and the underlying geometry of the scene.

Consider, for instance, a photograph of a building's facade taken from some position. The effect of viewing the facade from arbitrary positions can be achieved by projecting the photograph back onto the building's facade and then viewing the scene from a different vantage point[2][7].

Figure 4 shows a reconstructed scene of downtown San Jose. An airborne camera acquired the imagery from two perspectives. The geometry of the buildings was subsequently extracted from these images using photogrammtric techniques. The scene is reconstructed on-the-fly (and flown over) on the RealityEngine by drawing the buildings as polygons described by the extracted geometries and using projected texture mapping to directly re-project the imagery onto the buildings, as seen by the original camera locations. (Data and imagery supplied by ESL Inc., Sunnyvale CA.)

### 4.3 Satellite Imagery Reprojection

Interactive *ortho-rectification* of satellite (or any oblique imagery) is readily accomplished using projected texture. In Figure 5, the left image shows the satellite camera frustum with the acquired imagery (texture) on its screen, the center image shows the imagery reprojected onto the tesselated digital elevation data, and the right image shows the final rectified view with an additional orthographic projection applied.

### 4.4 Lighting/Transmission/Shadow Effects

A similar technique can be used to simulate the effects of some illumination source on a scene.

Figure 6 shows a texture representing an intensity map of a cross-section of spotlight's beam illuminating a section of the Yellowstone Park data. The Reality engine is capable of computing shadows on the fly (note shadows cast on mountains). This is an example of how projective texture mapping with shadows can be applied to line-of-sight applications.

Another application of projective texture mapping consists of viewing the projection of a slide or movie on an arbitrary surface[11][3]. One may also project live video onto geometry in a scene. Figure 7 shows a slide projected onto a scene (note shadows).

## 5 Conclusions

Advanced projective texture mapping is a technology that has huge implications to change the way things are done in multiple imaging application areas such photogrammetry, SAR, non-destructive evaluation, medical, seismic and sonar interpretation, film and video, and terrain analysis and scene generation.

High-performance general-purpose graphics workstations which incorporate this technology are uniquely positioned to replace existing image processing solutions (array processors, parallel processing, custom "black box" hardware and software) with new levels of performance and capabilities.

## Acknowledgements

Many thanks to Brian Cabral, Nancy Cam, Jim Foran, Chris Tanner, Bob Drebin, Keith Seto, and Bill James for their numerous contributions, comments, and suggestions.

## References

[1] Ed Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, University of Utah, 1974.

[2] Robert N. Devich and Frederick M. Weinhaus. Image perspective transformations. *SPIE*, 238, 1980.

[3] Julie O'B. Dorsey, Francois X. Sillion, and Donald P. Greenberg. Design and simulation of opera lighting and projection effects. In *Proceedings of SIGGRAPH '91*, pages 41–50, 1991.

[4] Paul S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.

[5] Paul S. Heckbert. Fundamentals of texture mapping and image warping. M.sc. thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, June 1989.

[6] Paul S. Heckbert and Henry P. Moreton. Interpolation for polygon texture mapping and shading. In David F. Rogers and Rae A. Earnshaw, editors, *State of the Art in Computer Graphics: Visualization and Modeling*, pages 101–111. Springer-Verlag, 1991.

[7] Kazufumi Kaneda, Eihachiro Nakamae, Tomoyuki Nishita, Hideo Tanaka, and Takao Noguchi. Three dimensional terrain modeling and display for environmental assessment. In *Proceedings of SIGGRAPH '89*, pages 207–214, 1989.

[8] Masaaki Oka, Kyoya Tsutsui, Akio Ohba, and Yoshitaka Kurauchi. Real-time manipulation of texture-mapped surfaces. *Computer Graphics (Proceedings of SIGGRAPH '87 )*, July 1987.

[9] W.K. Pratt. *Digital Image Processing*. John Wiley, 1990.

[10] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, and Paul Haeberli. Fast shadows and lighting effects using texture mapping. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):249–252, July 1992.

[11] Steve Upstill. *The RenderMan Companion*, pages 371–374. Addison Wesley, 1990.

[12] Lance Williams. Casting curved shadows on curved surfaces. In *Proceedings of SIGGRAPH '78*, pages 270–274, 1978.

[13] Lance Williams. Pyramidal parametrics. *Computer Graphics (SIGGRAPH '83 Proceedings)*, 17(3):1–11, July 1983.

[14] George Wolberg. *Digital Image Warping*. IEEE, 1990.