PLANAR 2-PASS TEXTURE MAPPING AND WARPING

Alvy Ray Smith
Pixar

ABSTRACT

The 2-pass transformation replaces a 2-D (2-dimensional) transformation with a sequence of orthogonal, simpler 1-D transformations. It may be used for the closely related processes of texture mapping and warping in computer graphics and image processing. First, texture maps onto planar quadric and superquadric surfaces and, second, planar bicubic and biquadratic warps of images are shown to be 2-pass transformable. A distinction between serial and parallel warps is introduced to solve a confusion in terms between computer graphics and image processing. It is shown that an n-th order serial polynomial warp is equivalent to an (n^2+n)-th order parallel polynomial warp. It is also shown that the serial equivalent to a parallel polynomial warp is generally not a polynomial warp, being more complicated than a polynomial. The unusual problem of bottlenecking and the usual one of antialiasing are discussed in the 2-pass context.

KEY WORDS AND PHRASES

2-pass, texture mapping, warping, serial warp, parallel warp, quadrics, superquadrics, bicubic warp, biquadratic warp, bottleneck, antialiasing

CR CATEGORY

8.2

INTRODUCTION

The term *warping* is often used in image processing to mean a 2-D resampling of an image. For example, when a photo of the Earth's surface is transmitted from a satellite to ground, it is typically warped to correct the surface patch for surface curvature, an oblique viewing angle, or lens aberrations. In general, warping is a continuous mapping of a 2-D planar region into another 2-D planar region. In image processing, it is the digital approximation of such a mapping which is of interest.

The term *texture mapping* is used in computer graphics to also mean 2-D resampling of an image, particularly when the target is the 2-D projection of a 3-D surface, viewed through a viewing transformation with perspective. It is the digital approximation of a presumably continuous mapping that is of interest in computer graphics also.

A useful distinction may be made between the use of warping in image processing and texture mapping in computer graphics. Image processing uses the mapping to correct or rectify images - to map a nonrectangular patch onto a rectangular one - while computer graphics proceeds the other direction to map a rectangular patch onto a nonrectangular one.

Another distinction - a confusion really - between the two historically separate disciplines is not helpful. The terms *biquadratic* and *bicubic* are used differently by the two groups. The terms *serial map (warp)* and *parallel map (warp)* are introduced to clarify this situation.

Examples from each of computer graphics and image processing are examined here and shown to be effectively simplified using the 2-pass technique [3]. First, the 2-pass texture mapping of a texture onto a planar superquadric surface is derived. For example, a rectangular framebuffer picture may be mapped onto a disk by the technique. Then the 2-pass warping of an image onto a planar bicubic or biquadratic patch is derived. The case where the edge of an image remains a rectangle but the interior points are warped - which is called a *frozen-edge warp* - is particulary suitable for image processing. Then the mathematical relationship between serial and parallel warps is derived using some of these results as examples.

Finally, two generic problems with the 2-pass technique are emphasized. The mathematical problem of *bottlenecking* is reexamined, and since it is the digital approximation of the 2-pass technique which is of primary interest, the important subject of antialiasing is investigated.

The work here builds on the original paper [3] and contains results derived in a series of internal technical memos [7-10].

2-PASS NOTATION AND REVIEW

If u_s and v_s are the coordinates of a source picture and u_t and v_t those of a target, then a general *parallel 2-D coordinate mapping* - or *parallel map* for short - may be defined by any two functions x and y such that

$$(u_t,v_t) = (x(u_s,v_s), y(u_s,v_s)) .$$

In this paper, u and v are understood to be parameterizations of the horizontal and vertical coordinates of a picture. Parameters u and v with no subscript are assumed to be u_s and v_s.

A *picture* of course is a mapping of each point of uv parameter space to a set of colors - e.g., the grays. The term *image* will be used here to mean picture - as in "image computing" or "image processing" - as well as for its usual mathematical meaning, with the context distinguishing which use is intended. The term *image computing* will be used to mean all of traditional computer graphics plus all of traditional image processing. This paper is an image computing paper because parallel maps include the texture maps of computer graphics and the warps of image processing. Thus parallel maps will also be called *parallel warps*.

The objects on which digital image computing focuses are digitized pictures - *sampled* versions of continuous images, where each sample is of course a pixel. Although we will talk of mappings of the set of continuous pictures to itself, it is really the sampled source and target pictures which are of ultimate interest.

Sampling and filtering theory is now a well-understood discipline for correctly representing continuous pictures by arrays of pixels, but it is computationally expensive. The *2-pass technique* is interesting because it suggests a cheaper solution for parallel maps on sampled pictures although it is a technique described in terms of continuous functions and pictures.

Perhaps the most general presentation of the 2-pass technique appears in [3], but approximately simultaneous work was occurring in industry [2] and in image processing [5]. All of these in turn refer to earlier works on the special case of image rotation in the plane.

The basic idea is to replace a parallel map with a sequence, or composition, of parallel maps, where each mapping in the sequence is computationally more interesting (e.g., cheaper) than the original map. The 2-pass technique replaces a given parallel map with a sequence of two parallel maps, where the first applied is called the *first pass,* or *horizontal pass,* and takes the form

$$(u_i,v_i) = (f_{v_s}(u_s),v_s)$$

where u_i and v_i are the coordinates of the *intermediate* image. The *second pass,* or *vertical pass* takes the form

$$(u_t,v_t) = (u_i,g_{u_i}(v_i)) .$$

The decomposition of a parallel map into a sequence of mappings results in an overall mapping from source to target called a *serial 2-D coordinate mapping* or a *serial map (serial warp)* for short. (It has been shown that three mappings in the sequence is of interest in the case of rotation in the plane [6].)

It is sometimes important to reverse the order of the decomposition, with the vertical pass first, the horizontal pass second. The functional expressions are generally different for the two orderings. When a distinction between the two orderings is required, "horizontal pass first" or the "vertical pass first" will be used. The horizontal pass first is assumed unless otherwise stated. See Figure 1.

Each of the two mappings, the first and second passes, is considered simpler than the given parallel map because f_{v_s} applies across a line of constant v_s and g_{u_i} down a line of constant u_i. In the digital approximation, lines of constant v_s are the horizontal scanlines and those of constant u_i are vertical scanlines. Resampling along a scanline is a 1-D problem. Thus, in a sense, the 2-pass technique is the replacement of a 2-D resampling problem with a sequence of two 1-D resamplings. A more careful analysis shows that a true 1-D resampling is sometimes inadequate, but a simplified 2-D resampling may suffice. This will be explored further below.

The 2-Pass Technique

The problem of replacing a given parallel map with a serial equivalent was solved in [3]. It is restated and solved again here to show the use of the current notation. Given parallel map

$$(x(u_s,v_s), y(u_s,v_s)) = (u_t,v_t) ,$$

it is desired to decompose it into two sequential mappings

$$(x'(u_s,v_s), v_s) = (u_i,v_i)$$

$$(u_i, y'(u_i,v_i)) = (u_t,v_t) .$$

The problem is to express x' and y', the two sequential *scanline functions,* as functions of the given mappings x and y and the source coordinates u_s and v_s for x' or the intermediate coordinates u_i and v_i for y'. The horizontal-pass scanline function is obvious:

$$x'(u_s,v_s) \equiv f_{v_s}(u_s) = x(u_s,v_s) .$$

We shall call this the *first step* of the 2-pass technique. For the vertical-pass scanline function, note that $u_i = x(u_s,v_s)$ and $v_s = v_i$. Let h_{u_i} be the solution of $u_i = x(u_s,v_s)$ for u_s. This solution is the *second step.* So h_{u_i} is a function of $v_s = v_i$ and

$$y'(u_i,v_i) \equiv g_{u_i}(v_i) = y(h_{u_i}(v_i),v_i)$$

for the *third step.* Vertical-pass-first scanline functions are derived in an analogous manner.

A difficulty with the 2-pass technique is obtaining a closed form for the second-pass scanline function. Sometimes this is not possible and numerical techniques must be used. In other cases, such as when there are multiple solutions h_{u_i}, numerical techniques are preferred. The bicubic and biquadratic warping techniques in this paper are of this variety [8, 9].

Another difficulty is the so-called *bottleneck problem.* There are cases [3] where the two scanline functions exist in closed form but are useless. If the source picture is mapped to a point or to a line segment by the first pass, then it hardly matters that a mapping exists which takes this intermediate image to the desired target picture. Although the shape would be correct, the color information would have been lost in the vanishing area bottleneck of the intermediate image. Although experience has shown that a bottleneck can always be avoided - e.g., by reversing the order of horizontal and vertical passes - this has not yet been proved to be the case.

A final difficulty appears only in the digital approximation of the 2-pass technique. This is the antialiasing problem alluded to earlier. If only 1-D sampling and filtering is used along scanlines, then serious aliasing can occur. In many interesting cases studied in [2, 3, 5], this is not a problem, but in others - such as in this paper (and in [7]) - it is. Again, experience has shown that aliasing artifacts can always be avoided - e.g., by reversing the order of horizontal and vertical passes - but this has not yet been proved to be the case.

SUPERQUADRIC TEXTURE MAPPING

Following Al Barr [1], given a "horizontal" plane curve **h** and a "vertical" modulating plane curve **m**

$$\mathbf{h}(v) = [h_1(v) \;\; h_2(v)], \quad v_0 \le v \le v_1,$$

$$\mathbf{m}(u) = [m_1(u) \;\; m_2(u)], \quad u_0 \le u \le u_1,$$

a *spherical product surface* is defined to be

$$\mathbf{S}(u,v) = [m_1(u)h_1(v) \;\; m_1(u)h_2(v) \;\; m_2(u)]$$

where $u_0 \le u \le u_1$, $v_0 \le v \le v_1$. We have reversed the orientation of **h** and **m** from [1], so **h** is actually vertical here - i.e., it is a function of the vertical parameter v - and **m** is horizontal. A *projected* spherical product surface has $m_2(u) = 0$ - i.e., is the orthographic projection of a spherical product surface into the xy-plane.

Barr [1] has shown how the (super)quadrics - (super)ellipsoids, (super)hyperboloids of one and two sheets, and (super)toroids - can be represented as rescaled spherical product surfaces. A (super)toroid may be thought of as an extended (super)ellipsoid; it has the same form except the modulating function is offset by a constant α and u varies over 2π radians. Table 1 gives the details of the defining functions for the superquadric family. An n-hyperboloid is an hyperboloid with n sheets. In all cases, the superquadrics give the quadrics if the two *squareness parameters* ε and ε' are set to 1.

It will be shown below (see also [7]) that the projected (super)quadrics, under "standard computer graphics transformations", are 2-pass transformable. A parallel map is *2-pass transformable* if it can be converted into an equivalent serial map by the 2-pass technique described above.

The standard computer graphics transformations are the following: In computer graphics, objects are flown through 3-space and projected into 2-space with a perspective projection using a 4x4 matrix multiplication followed with division by the homogeneous coordinate. These classic transformations shall be called *CG transforms*. The 4x4 matrix will be represented here by

$$\mathbf{T} = \begin{bmatrix} \mathbf{a} & \mathbf{e} & \mathbf{i} & \mathbf{m} \\ \mathbf{b} & \mathbf{f} & \mathbf{j} & \mathbf{n} \\ \mathbf{c} & \mathbf{g} & \mathbf{k} & \mathbf{o} \\ \mathbf{d} & \mathbf{h} & \mathbf{l} & \mathbf{p} \end{bmatrix}.$$

The transformation of a point $[X(u,v)\; Y(u,v)\; Z(u,v)\; 1]$ by a CG transform is accomplished by

$$[X'\; Y'\; Z'\; W'] = [X\; Y\; Z\; 1]\mathbf{T}\,.$$

The homogeneous divide by W' after this transform gives

$$x(u,v) = \frac{X'}{W'} = \frac{\mathbf{a}X + \mathbf{b}Y + \mathbf{c}Z + \mathbf{d}}{\mathbf{m}X + \mathbf{n}Y + \mathbf{o}Z + \mathbf{p}}$$

$$y(u,v) = \frac{Y'}{W'} = \frac{\mathbf{e}X + \mathbf{f}Y + \mathbf{g}Z + \mathbf{h}}{\mathbf{m}X + \mathbf{n}Y + \mathbf{o}Z + \mathbf{p}}.$$

For example, if $X(u,v) = u$, $Y(u,v) = v$, and $Z(u,v) = 0$ and we apply the 2-pass technique, the simple rectangle under CG transform result of [2, 3] is obtained.

The 2-D Superquadric 2-Pass Functions

The 2-D superquadrics are just the projected 3-D superquadrics - i.e., with $m_2(u) = 0$. The horizontal and vertical scanline functions for the 2-D superquadrics under CG transforms are derived to show a typical application of the 2-pass technique.

The general class of 2-D superquadrics under 3-D CG transforms is given by

$$[\, m_1(u)h_1(v) \;\; m_1(u)h_2(v) \;\; 0 \;\; 1\,] \begin{bmatrix} \mathbf{a} & \mathbf{e} & \mathbf{i} & \mathbf{m} \\ \mathbf{b} & \mathbf{f} & \mathbf{j} & \mathbf{n} \\ \mathbf{c} & \mathbf{g} & \mathbf{k} & \mathbf{o} \\ \mathbf{d} & \mathbf{h} & \mathbf{l} & \mathbf{p} \end{bmatrix}.$$

Scaling factors for the two axes of the projected superquadric have been subsumed into the diagonal elements of the CG transform matrix. Thus

$$x(u,v) = \frac{\mathbf{a}m_1(u)h_1(v) + \mathbf{b}m_1(u)h_2(v) + \mathbf{d}}{\mathbf{m}m_1(u)h_1(v) + \mathbf{n}m_1(u)h_2(v) + \mathbf{p}}$$

$$y(u,v) = \frac{\mathbf{e}m_1(u)h_1(v) + \mathbf{f}m_1(u)h_2(v) + \mathbf{h}}{\mathbf{m}m_1(u)h_1(v) + \mathbf{n}m_1(u)h_2(v) + \mathbf{p}}.$$

Application of the first step of the 2-pass technique immediately yields, as the horizontal scanline function for scanline v_s:

$$f_{v_s}(u) = \frac{\mathbf{A}(v_s)m_1(u) + \mathbf{d}}{\mathbf{M}(v_s)m_1(u) + \mathbf{p}}$$

where

$$\mathbf{A}(v) = \mathbf{a}h_1(v) + \mathbf{b}h_2(v)$$

$$\mathbf{M}(v) = \mathbf{m}h_1(v) + \mathbf{n}h_2(v)\,.$$

The second step is to solve

$$u_i = \frac{\mathbf{A}(v)m_1(u) + \mathbf{d}}{\mathbf{M}(v)m_1(u) + \mathbf{p}}$$

for $u = h_{u_i}(v)$ for vertical scanline u_i. Expanding and rearranging gives

$$m_1(u) = \frac{\mathbf{d} - \mathbf{p}u_i}{\mathbf{M}(v)u_i - \mathbf{A}(v)}.$$

As will be seen immediately, it is unnecessary to complete the solution of this equation for u.

The third step yields the vertical scanline function by substituting the expression just obtained for $m_1(u)$ into the expression above for $y(u,v)$:

$$g_{u_i}(v) = \frac{(\mathbf{d} - \mathbf{p}u_i)(\mathbf{e}h_1(v) + \mathbf{f}h_2(v)) + \mathbf{h}(\mathbf{M}(v)u_i - \mathbf{A}(v))}{(\mathbf{d} - \mathbf{p}u_i)(\mathbf{m}h_1(v) + \mathbf{n}h_2(v)) + \mathbf{p}(\mathbf{M}(v)u_i - \mathbf{A}(v))}.$$

Since $v_i = v_s$, we will drop the subscript from v_i also. The scanline function may be rearranged to give

$$g_{u_i}(v) = \frac{\mathbf{E}(u_i)h_1(v) + \mathbf{F}(u_i)h_2(v)}{\mathbf{G}h_1(v) + \mathbf{H}h_2(v)}.$$

$\mathbf{E}(u)$, $\mathbf{F}(u)$, $\mathbf{G}$, and $\mathbf{H}$ are defined in Table 3, which, with

Table 2, summarizes the application of the formulas just derived to the superquadric family. All quantities are as defined above if not otherwise specified.

The vertical-pass-first scanline functions may be derived similarly by exchanging x and y and also u and v in the expression for the parallel mapping and then applying the 2-pass technique. To get an analytic solution, $\varepsilon' = 1$ must be assumed. This will be called a *semi-superquadric* case.

Suppose it is desired to map the contents of a framebuffer onto a disk with $\mathbf{T}$ the identity transformation. The parameterization used thus far is awkward for this case since it generates a polar view. A more natural mapping is that used for Figure 2. It corresponds to an ellipsoid expressed as the spherical product surface

$$\mathbf{S}(u,v) = [\ m_2(u)h_1(v)\ \ h_2(v)\ \ m_1(u)h_1(v)\ ]$$

with $m_1(u), m_2(u), h_1(v), h_2(v)$ as before but with $-\pi \le u \le \pi,\ -\frac{\pi}{2} \le v \le \frac{\pi}{2}$. This is nothing more than a permutation of the 3-D coordinates used before and a swapping of the parameter space axes. For the example of mapping a framebuffer to a disk, the horizontal-pass-first formulas hold for the superquadric case in the alternative parameterization, just as before, and the vertical-pass-first functions are similarly good only up to the semi-superquadrics. Figure 2 also illustrates exercise of the two squareness parameters.

BICUBIC AND BIQUADRATIC WARPING

A bicubic patch may be described parametrically by two parameters u and v, each varying over the interval [0.,1.], and the two equations below:

$$x(u,v) = [\ u^3\ u^2\ u\ 1\ ] \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \equiv \mathbf{u}\mathbf{A}\mathbf{v}^T$$

$$y(u,v) = [\ u^3\ u^2\ u\ 1\ ] \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 & b_7 \\ b_8 & b_9 & b_{10} & b_{11} \\ b_{12} & b_{13} & b_{14} & b_{15} \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \equiv \mathbf{u}\mathbf{B}\mathbf{v}^T$$

The addition of a third equation of similar form, $z(u,v) = \mathbf{u}\mathbf{C}\mathbf{v}^T$, defines a full 3-D bicubic patch, and a fourth, $w(u,v) = \mathbf{u}\mathbf{D}\mathbf{v}^T$, may be added for the homogeneous coordinate convenient for perspective transformations. However, attention shall be restricted here to the planar case. Moreover, it will be computationally advantageous to consider only those planar patches which are *nonfolded,* or single valued. That is, no point (x,y) is the image of more than one point (u,v).

The equations for x and y expand into

$$x(u,v) = a_0u^3v^3+a_1u^3v^2+a_2u^3v+a_3u^3+a_4u^2v^3+a_5u^2v^2+a_6u^2v+a_7u^2+$$
$$a_8uv^3+a_9uv^2+a_{10}uv+a_{11}u+a_{12}v^3+a_{13}v^2+a_{14}v+a_{15}$$
$$y(u,v) = b_0u^3v^3+b_1u^3v^2+b_2u^3v+b_3u^3+b_4u^2v^3+b_5u^2v^2+b_6u^2v+b_7u^2+$$
$$b_8uv^3+b_9uv^2+b_{10}uv+b_{11}u+b_{12}v^3+b_{13}v^2+b_{14}v+b_{15}$$

which may be cast into expressions of the known coordinates x_i and y_i by solving for the boundary conditions. For example, $v = 0$ corresponds to one edge of the bicubic patch and $v = 1$ to the opposite edge, and similarly for $u = 0$ and $u = 1$. Define matrix $\mathbf{X}$ to be the following array of sixteen specific x coordinates (see Figure 3):

$$\mathbf{X} = \begin{bmatrix} x(0,0) & x(u_0,0) & x(u_1,0) & x(1,0) \\ x(0,v_0) & x(u_0,v_0) & x(u_1,v_0) & x(1,v_0) \\ x(0,v_1) & x(u_0,v_1) & x(u_1,v_1) & x(1,v_1) \\ x(0,1) & x(u_0,1) & x(u_1,1) & x(1,1) \end{bmatrix} \equiv \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

Then it can be shown that the x_i are related to the a_i by a compact matrix equation:

$$\mathbf{X} = \mathbf{V}\mathbf{A}^T\mathbf{U}^T$$

where

$$\mathbf{U} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ u_0^3 & u_0^2 & u_0 & 1 \\ u_1^3 & u_1^2 & u_1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

and $\mathbf{V}$ is defined similarly in terms of v_0 and v_1.

This equation can be solved for the a_i in terms of the x_i :

$$\mathbf{A} = \mathbf{U}^{-1}\mathbf{X}^T\mathbf{V}^{-T}$$

where $\mathbf{V}^{-T}$ represents the inverse of the transpose of $\mathbf{V}$. Similarly, the b_i in terms of the y_i are given by

$$\mathbf{B} = \mathbf{U}^{-1}\mathbf{Y}^T\mathbf{V}^{-T}$$

where $\mathbf{Y}$ is defined similarly to $\mathbf{X}$, but for specific y coordinates. It can be shown that $\mathbf{U}^{-1}$ may be represented as follows, where for notational convenience, we let $\overline{u}_0 = 1-u_0$, $\overline{u}_1 = 1-u_1$, $\underline{u}_0 = 1+u_0$, $\underline{u}_1 = 1+u_1$, $u_{10} = u_1-u_0$, $u_{01} = u_0+u_1$, $\underline{u}_{01} = u_0+u_1+1$, and $\overline{u}_{01} = u_0u_1+u_0+u_1$, and similarly for v_0 and v_1 :

$$\mathbf{U}^{-1} = \begin{bmatrix} -1 & 1 & -1 & 1 \\ \underline{u}_{01} & -\underline{u}_1 & \underline{u}_0 & -u_{01} \\ -\overline{u}_{01} & u_1 & -u_0 & u_0u_1 \\ u_0u_1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{u_0u_1} & 0 & 0 & 0 \\ 0 & \frac{1}{u_0\overline{u}_0u_{10}} & 0 & 0 \\ 0 & 0 & \frac{1}{u_1\overline{u}_1u_{10}} & 0 \\ 0 & 0 & 0 & \frac{1}{\overline{u}_0\overline{u}_1} \end{bmatrix}$$

and similarly for $\mathbf{V}^{-1}$ in terms of the v_i.

A special case is the *cubic patch,* a bicubic patch with terms which have exponents summing to three or less. This is equivalent to matrices $\mathbf{A}$ and $\mathbf{B}$ being all zeros above the bend-sinister diagonal. Several other special cases of interest are discussed below.

A Special Case: The Bicubic Frozen Edge

A simple special case (Figure 4) requires that the boundary of the bicubic patch be a rectangle. If the uv parameter space is thought of as a rectangular source picture and its image under $x(u,v)$ and $y(u,v)$ as a bicubic patch target picture, then the special case requires that the rectangle around the rectangular patch map to itself - hence the Frozen Edge. Only the four internal control points (x_5,y_5), (x_6,y_6), (x_9,y_9), and (x_{10},y_{10}) move from source to target. Thus

$$\mathbf{X}_s = \begin{bmatrix} 0 & u_0 & u_1 & 1 \\ 0 & x_5 & x_6 & 1 \\ 0 & x_9 & x_{10} & 1 \\ 0 & u_0 & u_1 & 1 \end{bmatrix}$$

$$\mathbf{Y}_s = \begin{bmatrix} 0 & 0 & 0 & 0 \\ v_0 & y_5 & y_6 & v_0 \\ v_1 & y_9 & y_{10} & v_1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

And these collapse the expressions for **A** to

$$\mathbf{A}_s = \begin{bmatrix} a_0 & -(a_0+a_2) & a_2 & 0 \\ -(a_0+a_8) & a_0+a_2+a_8+a_{10} & -(a_2+a_{10}) & 0 \\ a_8 & -(a_8+a_{10}) & a_{10} & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \mathbf{U}^{-1}\mathbf{X}_s^T\mathbf{V}^{-T}$$

and similarly for **B**, both of which become particularly simple for the special case of the cubic frozen edge.

A Special Case: The Biquadratic Patch

By a derivation analogous to that for the bicubic patch above and using the notation of Figure 5,

$$\mathbf{A}_2 = \mathbf{U}_2^{-1}\mathbf{X}_2^T\mathbf{V}_2^{-T}$$

where the subscript 2 denotes the biquadratic case (3×3 matrices) and

$$\mathbf{U}_2^{-1} = \begin{bmatrix} 1 & -1 & 1 \\ -\underline{u}_1 & 1 & -u_1 \\ u_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{u_1} & 0 & 0 \\ 0 & \frac{1}{u_1\overline{u}_1} & 0 \\ 0 & 0 & \frac{1}{\overline{u}_1} \end{bmatrix}$$

and similarly for $\mathbf{V}_2^{-1}$ and also $\mathbf{B}_2$.

Common examples of biquadratic warps are the pincushion and barrel distortions of video - see Figure 7. So biquadratic warps may be used to correct for these distortions.

Analogous to the cubic patch, a *quadratic patch* (or *quadric patch)* is defined to have only those terms with exponents summing to two or less, so all elements above the bend-sinister diagonals of the corresponding 3×3 $\mathbf{A}_2$ and $\mathbf{B}_2$ matrices are zeros.

A Special Case: The Biquadratic Frozen Edge

There is a Frozen Edge special case of the general biquadratic patch analogous to that for the bicubic patch discussed above. See Figure 6. Using the techniques above, it can be readily shown that

$$\mathbf{A}_{2s} = \begin{bmatrix} a_0 & -a_0 & 0 \\ -a_0 & a_0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{U}_2^{-1}\mathbf{X}_{2s}^T\mathbf{V}_2^{-T}$$

$$\mathbf{B}_{2s} = \begin{bmatrix} b_0 & -b_0 & 0 \\ -b_0 & b_0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \mathbf{U}_2^{-1}\mathbf{Y}_{2s}^T\mathbf{V}_2^{-T}$$

where

$$a_0 = \frac{x_4-u_1}{u_1\overline{u}_1v_1\overline{v}_1}$$

$$b_0 = \frac{y_4-v_1}{u_1\overline{u}_1v_1\overline{v}_1}$$

and point (x_4, y_4) is the only control point that moves.

The quadratic frozen edge is a particularly simple special case of the biquadratic frozen edge. The bilinear patch was fully treated in [3].

The Horizontal Function $f_{v_s}(u)$

For subsequent convenience, let the rows of **A** be denoted by $\mathbf{a}_0$, $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$. Also let

$$\mathbf{f}(v) = \mathbf{A}\mathbf{v}^T = [\, f_0(v)\ f_1(v)\ f_2(v)\ f_3(v)\,]^T = [\, \mathbf{a}_0\mathbf{v}^T\ \mathbf{a}_1\mathbf{v}^T\ \mathbf{a}_2\mathbf{v}^T\ \mathbf{a}_3\mathbf{v}^T\,]^T$$

Direct application of the 2-pass technique first step yields

$$f_{v_s}(u) = \mathbf{u}\mathbf{A}\mathbf{v}_s^T = \mathbf{u}\mathbf{f}(v_s)$$

for horizontal scanline v_s and

$$\mathbf{v}_s^T = [\, v_s^3\ v_s^2\ v_s\ 1\,]\,.$$

The biquadratic case may also be represented in an analogous fashion.

The Auxiliary Function $h_{u_i}(v)$

The 2-pass technique second step requires that

$$x(u,v)-u_i = 0$$

be solved for $u = h(v)$. (The subscript u_i is dropped from h and g for convenience in this and the following section.) In words, the set of u's is desired which map into u_i under the horizontal scanline functions. Since $x(u,v) = \mathbf{u}\mathbf{f}(v)$ is cubic in u, the equation may be written as a general cubic equation, the *auxiliary cubic* equation,

$$\alpha u^3+\beta u^2+\gamma u+\delta = 0$$

where the coefficients are the following functions of v :

$$\alpha = f_0(v) = \mathbf{a}_0\mathbf{v}^T$$

$$\beta = f_1(v) = \mathbf{a}_1\mathbf{v}^T$$

$$\gamma = f_2(v) = \mathbf{a}_2\mathbf{v}^T$$

$$\delta = f_3(v)-u_i = \mathbf{a}_3\mathbf{v}^T-u_i\ .$$

In general, the auxiliary cubic has three (non-polynomial) solutions, $h_0(v)$, $h_1(v)$, and $h_2(v)$, which may be obtained using classic cubic equation solution techniques. It is difficult to determine which of the three are the valid solutions, and two of them may be complex. The solution method suggested below capitalizes on the restriction to planar nonfolded patches to avoid these difficulties.

A Special Case: The Planar Nonfolded Bicubic Patch

Nonfolded patches can have only one valid solution; only one u on each horizontal scanline can map to the current vertical scanline u_i. This set of u's is a smooth function of v, $u = h(v)$. Hence, if we find a solution $u = h(0)$ for the first horizontal scanline, it can be assumed to be in the vicinity of the solution for the next adjacent scanline and hence be used as a first approximation in a Newton iteration to the solution for the second scanline. Then this solution can be used as a starting value for an iteration to a solution for the next scanline, and so forth, exploiting the coherence.

So the problem of solving for h reduces to that of finding good starting values for the Newton iteration. A crude value would be u_i itself. A more refined value comes from the fact that the horizontal scanline functions have already been derived in the first pass. For a given v, the scanline function $f_v(u)$ can be computed for a small number, say n, of values of u equally spaced along [0, 1] to find two images which bracket u_i. Either of the corresponding values of u could be used as a starting value for the iteration or a linear interpolation between them. A set of starting values could be generated for all vertical scanlines by such a procedure applied along just one horizontal scanline between passes. Figure 1 illustrates the use of Newton iteration to solve a planar nonfolded bicubic patch with the 2-pass technique.

The Vertical Function $g_{u_i}(v)$

Direct application of the 2-pass technique third step yields three functions g_i, depending on which of the three auxiliary functions h_i, is used. Assuming only the planar nonfolded case, this reduces to one function $g(v)$. Thus, for vertical scanline u_i,

$$g(v) = \mathbf{u_h B v}^T$$

where

$$\mathbf{u_h} = [\ h^3(v)\ \ h^2(v)\ \ h(v)\ \ 1\ ]\ .$$

In general, this vector changes from (vertical) scanline to scanline u_i. The biquadratic case is analogously handled.

A nonfolded patch would have $g(v)$ one-to-one on the domain [0, 1] *for those points which are images of [0, 1] under some horizontal scanline function.* The fourth, or alpha, channel of a framebuffer could be used to determine if a point fit this condition; its alpha channel would be empty if it were the image of a point outside [0, 1]. So nonfolded means that all scanline functions, horizontal and vertical, are one-to-one mappings on the domain [0, 1], subject to this condition. This can be shown to be equivalent to the definition of nonfolded given in the Bicubic and Biquadratic Warping section. Ordinary B-spline or Bezier patch techniques may be used to ensure a patch is nonfolded.

SERIAL VS PARALLEL WARPS

Summarizing, polynomial warps may defined by

$$x(u_s,v_s) = \mathbf{u}_s \mathbf{A} \mathbf{v}_s^T$$
$$y(u_s,v_s) = \mathbf{u}_s \mathbf{B} \mathbf{v}_s^T$$

where **A** and **B** are matrices of polynomial coefficients and $\mathbf{u}_s$ and $\mathbf{v}_s$ are vectors of powers of u_s and v_s. A third-order parallel polynomial warp would have **A** and **B** be 4×4 matrices of constants and

$$\mathbf{u}_s = [\ u_s^3\ u_s^2\ u_s\ \ 1\ ]$$
$$\mathbf{v}_s = [\ v_s^3\ v_s^2\ v_s\ \ 1\ ]$$

Since x and y are third-order polynomials in two variables, the resulting warp is called a *bicubic* warp. The *biquadratic* warp is defined similarly but for **A** and **B** both 3×3 matrices and

$$\mathbf{u}_s = [\ u_s^2\ u_s\ \ 1\ ]$$
$$\mathbf{v}_s = [\ v_s^2\ v_s\ \ 1\ ]$$

A second- or third-order serial polynomial warp would have

$$f_{v_s}(u_s) = x(u_s,v_s) = \mathbf{u}_s \mathbf{A} \mathbf{v}_s^T$$
$$g_{u_i}(v_i) = y(u_i,v_i) = \mathbf{u}_i \mathbf{B} \mathbf{v}_i^T$$

which is quite similar in form to the parallel polynomial warp described above. The parallel equivalent of this serial polynomial warp is derived below where it is seen to be a higher-order polynomial mapping and hence quite different in form from the parallel polynomial warp.

The Parallel Equivalent of a Serial Warp

Given a general serial warp

$$(x(u_s,v_s),v_s) = (u_i,v_i)$$
$$(u_i,y(u_i,v_i)) = (u_t,v_t)$$

the problem is to find $x'(u_s,v_s) = u_t$ and $y'(u_s,v_s) = v_t$ in terms of u_s, v_s, x, and y.

The solution is straightforward using the notation:

$$u_t = u_i = x(u_s,v_s)$$

is already the desired solution for x'. That is,

$$x'(u_s,v_s) = x(u_s,v_s)\ .$$

Since $v_i = v_s$ and $u_i = x(u_s,v_s)$,

$$y'(u_s,v_s) = y(u_i,v_i) = y(x(u_s,v_s),v_s)\ .$$

Example

The parallel equivalent of a serial biquadratic polynomial warp is given immediately by

$$x'(u_s,v_s) = \mathbf{u}_s \mathbf{A} \mathbf{v}_s^T$$
$$y'(u_s,v_s) = [\ (\mathbf{u}_s \mathbf{A} \mathbf{v}_s^T)^2\ \ \mathbf{u}_s \mathbf{A} \mathbf{v}_s^T\ \ 1\ ]\mathbf{B} \mathbf{v}_s^T\ .$$

Thus the parallel equivalent of a second-order serial polynomial warp is fourth order in u_s and sixth order in v_s. Similarly, a third-order serial polynomial warp is equivalent to a parallel polynomial warp which is ninth order in u_s and twelfth order in v_s. In general, the parallel equivalent of an n-th order serial polynomial warp is n^2-th order in u_s and (n^2+n)-th order in v_s.

In particular, for the biquadratic Frozen Edge case discussed above and implemented serially by Thomas Porter as part of the Pixar Image Computer demonstration at the National Computer Graphics Association (NCGA) convention in Dallas, April, 1985,

$$x(u_s,v_s) = a_0 u_s(1-u_s)v_s(1-v_s)+u_s \equiv a_0 u_s \overline{u}_s v_s \overline{v}_s + u_s$$
$$y(u_i,v_i) = b_0 u_i(1-u_i)v_i(1-v_i)+v_i \equiv b_0 u_i \overline{u}_i v_i \overline{v}_i + u_i$$

The parallel equivalent of this is easily shown to be

$$x'(u_s,v_s) = a_0 u_s \overline{u}_s v_s \overline{v}_s + u_s$$

$$y'(u_s,v_s) = a_0 b_0 u_s \overline{u}_s v_s^2 \overline{v}_s^2 (1 - u_s \overline{u}_s v_s \overline{v}_s) + v_s$$

which is a sixth-order parallel polynomial warp. Clearly, it is important to know if a "biquadratic" warp is serial or parallel.

Similarly the "bicubic" warp of the scarab beetle in [5] is a serial third-order warp equivalent to a twelfth-order parallel polynomial warp. The mappings in [5] are in reverse order to those given here with $u_s = \mathbf{u}_i \mathbf{A} \mathbf{v}_i^T$ and $v_i = \mathbf{u}_t \mathbf{B} \mathbf{v}_t^T$, but the argument still holds, in the reverse direction. The problem of converting between the order presented here and the reverse order is another whole problem not considered further here.

The Serial Equivalent of a Parallel Warp

The problem here is to find a serial decomposition of a parallel mapping. But, of course, this is exactly the 2-pass problem. In particular, it is shown above that parallel polynomial warps are equivalent to serial warps with

$$x'(u_s,v_s) = \mathbf{u}_s \mathbf{A} \mathbf{v}_s^T$$

and

$$y'(u_i,v_i) = [\, h^3 \;\; h^2 \;\; h \;\; 1 \,] \mathbf{B} \mathbf{v}_i^T$$

for the bicubic case and similarly for the biquadratic case, where h is the solution of $\mathbf{u}_s \mathbf{A} \mathbf{v}_s^T - u_i = 0$ for u_s. There are three solutions h for the bicubic case and two for the biquadratic case. The solutions are not polynomials, which proves that, in general, the serial equivalent of a parallel polynomial warp is not a serial polynomial warp.

SHADING, DEPTH, MATTING, AND NORMALS

Normals and z information at each point may be carried along through the 2-pass technique just as are colors (where alpha, or opacity, is assumed to be a fourth component of each color). Then shading information is computed from the normal information and color information, and depth from the z information, as traditionally done. The alpha channel is correctly transformed by the 2-pass technique. Therefore it serves as a matte channel so that a 2-pass transformed object may be correctly composited with other images.

BOTTLENECK PROBLEM

The area of the intermediate picture can be much less than that of either the source or the target pictures. This is called [3] the bottleneck problem. In fact, the intermediate area can be zero. This happens, for example, for a rotate about the picture plane normal by 90 degrees.

In general there are several paths from source to target using the 2-pass technique. First, there is the horizontal pass followed by the vertical pass - the method usually assumed in this paper. Second, there is the vertical pass followed by the horizontal pass. Then there are variations on these two which incorporate a "preprocessing" step for which there is subsequent compensation. For example, in the case of rotation by almost 90 degrees - say 87 degrees - the bottleneck can be avoided by doing a simple transpose of rows and columns to effect a 90-degree rotation then a normal 2-pass transformation to get the remaining -3 degrees.

The method which has been used successfully for CG transforms of a simple rectangle is to compute the intermediate areas which would obtain via a set of four paths and select the path with the largest intermediate area. The four paths are (1) horizontal pass first, (2) vertical pass first, (3) transpose rows and columns then horizontal pass first, and (4) transpose then vertical pass first. The intermediate areas can be readily computed - the formulas are given in [3]. As mentioned in The 2-Pass Technique section, this has always worked but has not been proved to do so.

The bottlenecking solution for the projected superquadrics used here is simply that above. The justification is that a CG transform of a projected superquadric is the same as a mapping onto a projected superquadric in canonical position followed by a CG transform of the resulting 2-D image. The first step is non-degenerate - see Figure 2. Since this image falls within a rectangle, the second step is exactly that solved with the technique above.

The bottlenecking problem for the bicubic and biquadratic warps is ignored here because warping is generally not used for large geometric distortions - such as scalings and rotations - but rather for relocation of points within the vicinity of the original positions - i.e., for scale factors of about 1. and rotations of about 0.

ANTIALIASING

Most of this paper may be read independently of a digital realization of the technique presented. In this section we address those artifacts which arise strictly because digital approximations of the scanline functions are implemented. The 2-pass technique is attractive because it promises to cheapen the antialiasing computations required by the equivalent serial mapping. How true is it that the 2-pass technique replaces a full 2-D antialiasing problem with a sequence of two simpler 1-D antialiasing problems - in the case of 2-D quadrics under CG transform?

Suppose the 2-D antialiasing method we would have chosen in a full 2-D antialiasing application would integrate all the pixels in neighborhood $N(p)$ of source pixel p to obtain the target pixel p'. Presumably $N(p)$ would be the pixel support of some antialiasing filter. Then a necessary condition that a serial map perform the same filtering would be that all these same pixels be used in the computation of p'. This implies that all pixels in $N(p)$ map into the same vertical scanline during the first pass, so that the vertical pass can then map all of them - their intermediate images actually - into p'.

This condition is rarely met, so it is surprising how often the use of two serial 1-D filtering steps actually works. Intuitively, it works whenever the horizontal pass does not skew neighboring horizontal scanlines in $N(p)$ very far with respect to one another. Another way to say this is that the 1-D filtering trick works whenever there is not a high frequency change, in the vertical direction, of the object being transformed. "High frequency" in this case means near or greater than the spatial frequency of the horizontal scanlines. Figure 8 illustrates the point, and also the fact that changing the order sometimes corrects the problem. As pointed out earlier, there is no proof that this is always the case.

It should be no surprise that 1-D filtering in the discrete 2-pass should occasionally result in aliasing artifacts - one whole dimension cannot be thrown away without seeing some effect. The *continuous* 2-pass technique, of course, isn't plagued by this problem.

A higher-order antialiasing technique to use when the 1-D filtering trick fails is easily described. Consider a horizontal scanline of pixels, modeled as a row of abutting squares. The input domain image of an output pixel which intersects this scanline is an area bounded above and below by parallel line segments and left and right by curves (straight lines for several interesting transformations). The average intensity over this area is a more accurate average than the strictly 1-D average over say the midline of this area. Notice that this "scanline area" technique takes the 1-D passes back into 2-D computations. It is essentially a box filtering technique. Higher order filters would give better results as usual.

All 1-D sampling and filtering used for the figures in this paper is derived from the standard theories as described, for example, in [11, 12].

ACKNOWLEDGEMENTS

Colleagues Ed Catmull, Charlie Gunn, Pat Hanrahan, and Tom Porter have all provided important ideas for this 2-pass work. Ed and I worked on the original idea together. Charlie championed Newton iteration. Tom implemented the serial biquadratic frozen-edge and Pat the full planar serial biquadratic on the Pixar Image Computer. Tom's implementation - and subsequent arguments about what a biquadratic warp really is - inspired the resolution, presented herein, of the long-standing confusion between serial and parallel maps.

REFERENCES

[1] Barr, Alan H. *Superquadrics and Angle-Preserving Transformations,* **IEEE Computer Graphics and Applications,** January, 1981, pp. 11-23.

[2] Bennett, Philip P., and Steven A. Gabriel, *System for Spatially Transforming Images,* **United States Patent 4,472,732,** September 18, 1984. (Assigned to Ampex Corporation. This is one of the patents for the very successful ADO (Ampex Digital Optics) product. See also U. S. Patents 4,463,372 and 4,468,688.)

[3] Catmull, Edwin, and Alvy Ray Smith. *3-D Transformations of Images in Scanline Order,* **Computer Graphics,** Vol. 14, No. 3, pp. 279-285, July, 1980. (SIGGRAPH 80 Conference Proceedings).

[4] Fant, Karl M. *A Nonaliasing, Real-Time Spatial Transform Technique,* **IEEE Computer Graphics and Applications,** January, 1986, pp. 71-80. (See also the *Letters to the Editor* section of **IEEE Computer Graphics and Applications,** March, 1986, pp. 66-67, and July, 1986, pp. 3 and 8.)

[5] Fraser, Donald, Robert A. Schowengerdt, and Ian Briggs. *Rectification of Multichannel Images in Mass Storage Using Image Transposition,* **Computer Vision, Graphics, and Image Processing,** Volume 29, Number 1, January, 1985, pp. 23-36. (See also *Corrigendum,* Volume 31, Number 3, September, 1985, p. 395.)

[6] Paeth, Alan. *A Fast Algorithm for General Raster Rotation,* **Graphics Interface '86 Proceedings,** May, 1986, pp. 77-81.

[7] Smith, Alvy Ray. *Projected Superquadrics are 2-Pass Transformable,* **Technical Memo 54,** Computer Graphics Department, Computer Division, Lucasfilm Ltd., August, 1982 (now Technical Memo 54, Pixar).

[8] Smith, Alvy Ray. *A 2-Pass Solution to the Planar Biquadratic Patch,* **Technical Memo 128,** Computer Graphics Department, Computer Division, Lucasfilm Ltd., May, 1985 (now Technical Memo 128, Pixar).

[9] Smith, Alvy Ray. *A 2-Pass Solution to the Planar Bicubic Patch,* **Technical Memo 132,** Computer Graphics Department, Computer Division, Lucasfilm Ltd., June, 1985 (now Technical Memo 132, Pixar).

[10] Smith, Alvy Ray. *Serial vs Parallel Warps,* **Technical Memo 134,** Computer Graphics Department, Computer Division, Lucasfilm Ltd., June, 1985 (now Technical Memo 134, Pixar).

[11] Smith, Alvy Ray. *Digital Filtering Tutorial for Computer Graphics* **Technical Memo 27,** Computer Graphics Department, Computer Division, Lucasfilm Ltd., March, 1983 (now Technical Memo 27, Pixar). Also in Tutorial Notes for Siggraph 1983 and Siggraph 1984.

[12] Smith, Alvy Ray. *Digital Filtering Tutorial, Part II* **Technical Memo 44,** Computer Graphics Department, Computer Division, Lucasfilm Ltd., May, 1983 (now Technical Memo 44, Pixar). Also in Tutorial Notes for Siggraph 1983 and Siggraph 1984.

(Figures 1, 2, 7, and 8 are on next page.)

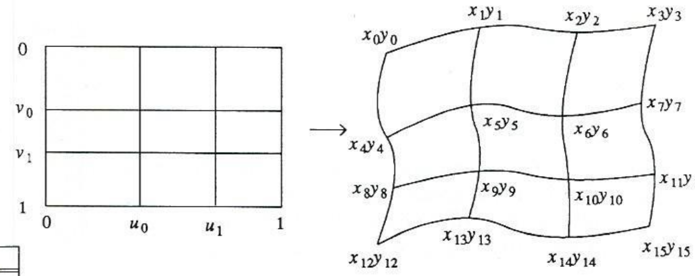


Figure 3. Bicubic patch transformation.

Table 1. Superquadric Defining Functions						
Type	$m_1(u)$	$m_2(u)$	u domain	$h_1(v)$	$h_2(v)$	v domain
Ellipsoid	$\cos^{\varepsilon} u$	$\sin^{\varepsilon} u$	$-\frac{\pi}{2} \le u \le \frac{\pi}{2}$	$\cos^{\varepsilon'} v$	$\sin^{\varepsilon'} v$	$-\pi \le v < \pi$
1-Hyperboloid	$\sec^{\varepsilon} u$	$\tan^{\varepsilon} u$	$-\frac{\pi}{2} < u < \frac{\pi}{2}$	$\cos^{\varepsilon'} v$	$\sin^{\varepsilon'} v$	$-\pi \le v < \pi$
2-Hyperboloid	$\sec^{\varepsilon} u$	$\tan^{\varepsilon} u$	$-\frac{\pi}{2} < u < \frac{\pi}{2}$	$\sec^{\varepsilon'} v$	$\tan^{\varepsilon'} v$	$-\frac{\pi}{2} < v < \frac{\pi}{2}$ * $\frac{\pi}{2} < v < \frac{3\pi}{2}$ **
Toroid	$\alpha + \cos^{\varepsilon} u$	$\sin^{\varepsilon} u$	$-\pi \le u < \pi$	$\cos^{\varepsilon'} v$	$\sin^{\varepsilon'} v$	$-\pi \le v < \pi$

* For the first sheet.

** For the second sheet.

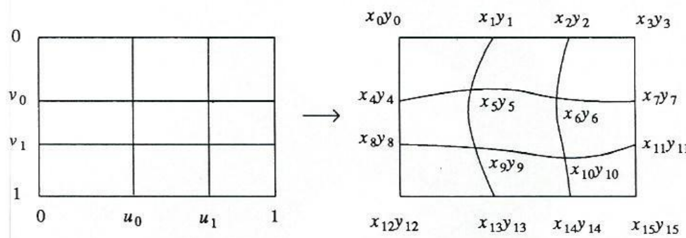


Figure 4. Bicubic frozen edge.

Table 2. 2-D Superquadric 2-Pass Horizontal Scanline Functions			
Type	$f_{v_s}(u)$	$A(v)$	$M(v)$
Ellipsoid	$\frac{A(v_s)\cos^{\varepsilon} u + d}{M(v_s)\cos^{\varepsilon} u + p}$	$a\cos^{\varepsilon'} v + b\sin^{\varepsilon'} v$	$m\cos^{\varepsilon'} v + n\sin^{\varepsilon'} v$
1-Hyperboloid	$\frac{A(v_s)\sec^{\varepsilon} u + d}{M(v_s)\sec^{\varepsilon} u + p}$	$a\cos^{\varepsilon'} v + b\sin^{\varepsilon'} v$	$m\cos^{\varepsilon'} v + n\sin^{\varepsilon'} v$
2-Hyperboloid	$\frac{A(v_s)\sec^{\varepsilon} u + d}{M(v_s)\sec^{\varepsilon} u + p}$	$a\sec^{\varepsilon'} v + b\tan^{\varepsilon'} v$	$m\sec^{\varepsilon'} v + n\tan^{\varepsilon'} v$
Toroid	$\frac{A(v_s)(\alpha + \cos^{\varepsilon} u) + d}{M(v_s)(\alpha + \cos^{\varepsilon} u) + p}$	$a\cos^{\varepsilon'} v + b\sin^{\varepsilon'} v$	$m\cos^{\varepsilon'} v + n\sin^{\varepsilon'} v$

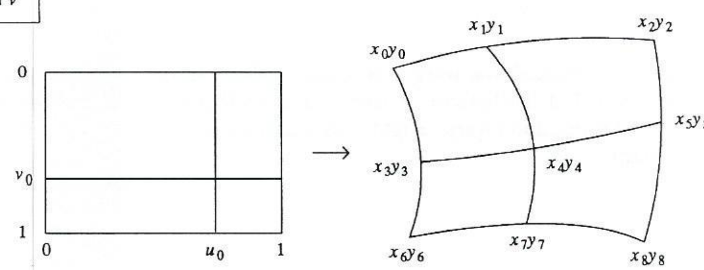


Figure 5. Biquadratic patch transformation.

Table 3. 2-D Superquadric 2-Pass Vertical Scanline Functions		
Type	$g_{u_i}(v)$	$E(u)$, $F(u)$, G, H
Ellipsoid	$\frac{E(u_i)\cos^{\varepsilon'} v + F(u_i)\sin^{\varepsilon'} v}{G\cos^{\varepsilon'} v + H\sin^{\varepsilon'} v}$	$E(u) = \begin{vmatrix} m & p \\ e & h \end{vmatrix} u - \begin{vmatrix} a & d \\ e & h \end{vmatrix}$
1-Hyperboloid	$\frac{E(u_i)\cos^{\varepsilon'} v + F(u_i)\sin^{\varepsilon'} v}{G\cos^{\varepsilon'} v + H\sin^{\varepsilon'} v}$	$F(u) = \begin{vmatrix} n & p \\ f & h \end{vmatrix} u - \begin{vmatrix} b & d \\ f & h \end{vmatrix}$
2-Hyperboloid	$\frac{E(u_i)\sec^{\varepsilon'} v + F(u_i)\tan^{\varepsilon'} v}{G\sec^{\varepsilon'} v + H\tan^{\varepsilon'} v}$	$G = \begin{vmatrix} m & p \\ a & d \end{vmatrix}$
Toroid	$\frac{E(u_i)\cos^{\varepsilon'} v + F(u_i)\sin^{\varepsilon'} v}{G\cos^{\varepsilon'} v + H\sin^{\varepsilon'} v}$	$H = \begin{vmatrix} n & p \\ b & d \end{vmatrix}$

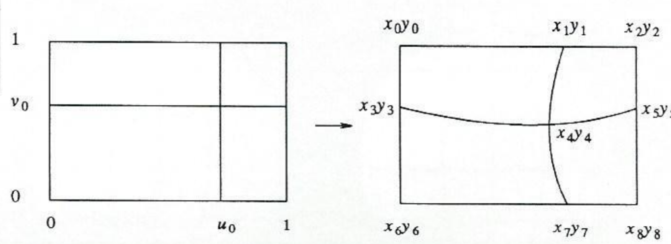
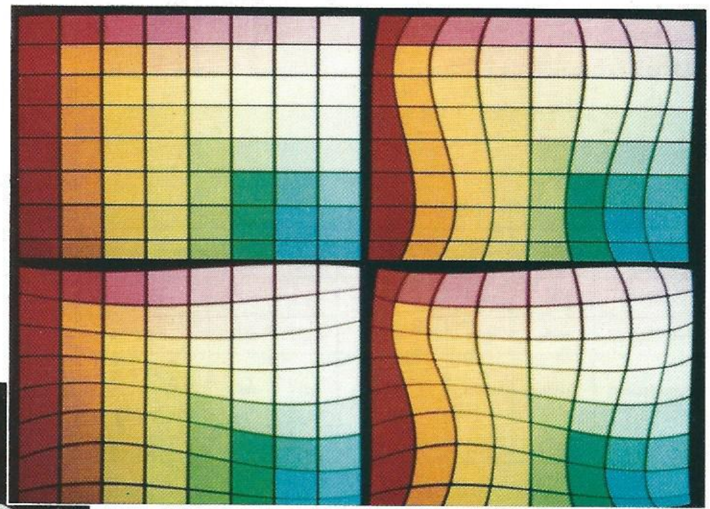
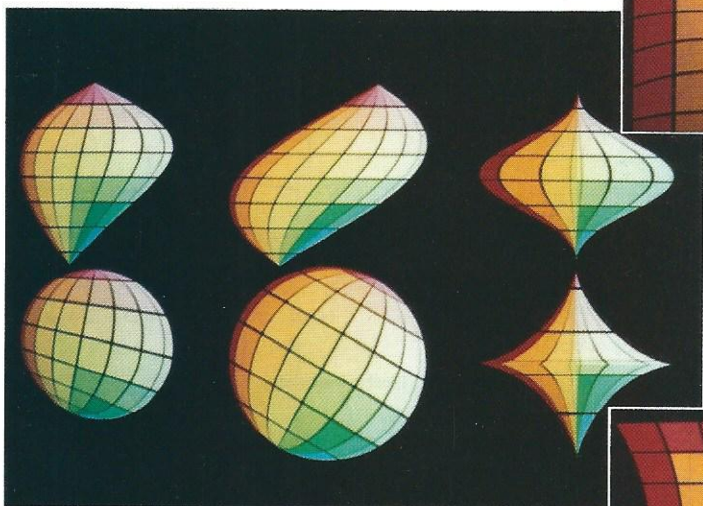


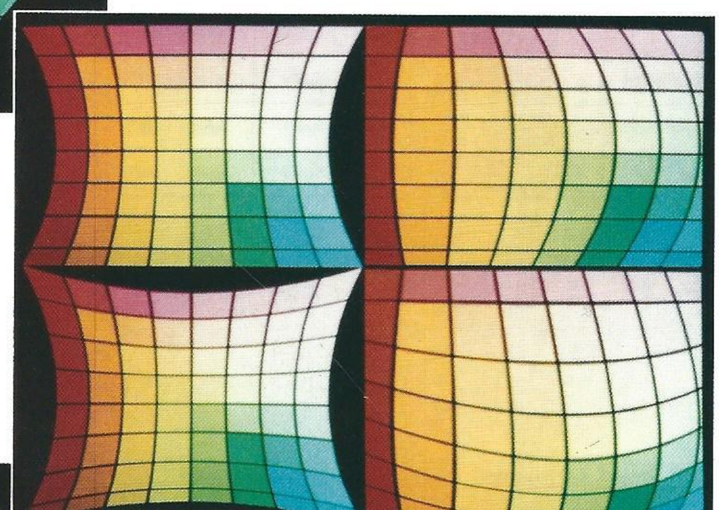
Figure 6. Biquadratic frozen edge.



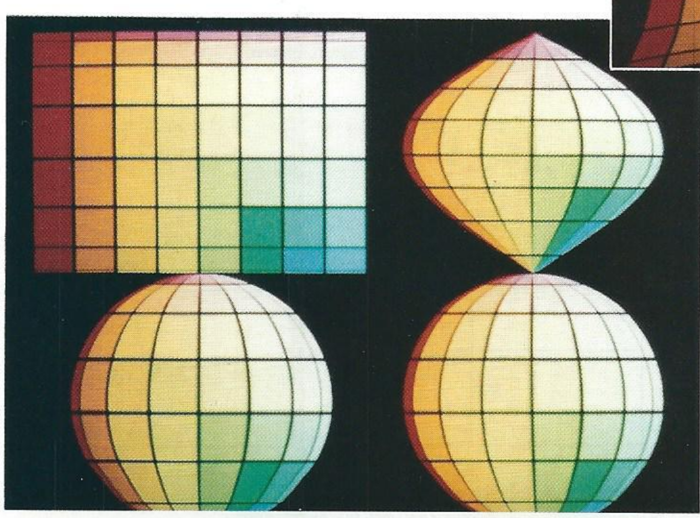
→
Figure 1. Planar nonfolded bicubic warp. Source at upper left. Target at lower right. Intermediate image at upper right is the horizontal pass for the horizontal pass first. Lower left is vertical pass for the vertical pass first.



←
Figure 2. Superquadric texture mapping. Horizontal passes at the top, vertical at the bottom. Source is same as in Figure 1. Left mapping is onto a disk in perspective. Middle is a simple rotation. Right is superdisk with $\varepsilon = \varepsilon' = 3$.



→
Figure 7. Biquadratic warps. Horizontal passes at the top, vertical at the bottom. Source is that of Figure 1. Left is a pincushion warp. Right is an assymmetric barrel warp.



←
Figure 8. Antialiasing failure. Source is that of Figure 1. Left are the vertical and horizontal passes with the vertical pass first. Right are the horizontal and vertical passes with the horizontal pass first. The left disk is seriously aliased at the poles. Reversing order of the passes solves the problem.