CREATING RASTER OMNIMAX IMAGES
FROM MULTIPLE PERSPECTIVE VIEWS
USING THE ELLIPTICAL WEIGHTED AVERAGE FILTER

Ned Greene
Paul S. Heckbert

Computer Graphics Lab
New York Institute of Technology
Old Westbury, NY 11568

Abstract

Creation of Omnimax animation by computer opens up fantastic new visual possibilities. Unfortunately, the fisheye distortion of Omnimax film images complicates synthesis by computer since most image synthesis programs are only capable of creating perspective views. As an alternative to modifying image synthesis programs to produce Omnimax projections directly, a method is presented for creating them from multiple perspective views. Four perspective views of the environment are created, each a projection onto a face of a cube centered at the camera, and then a mapping program creates an Omnimax projection from the four views. The mapping program convolves the cubeface images with a space variant filter to minimize aliasing during resampling. An efficient, high quality texture filter called the *elliptical weighted average filter* (EWA) was developed for this task. The EWA filter computes a weighted average over an arbitrarily oriented elliptical area. The use of lookup tables and incremental techniques make it more efficient than texture filters of similar quality and generality. EWA can be used as a filter for texture mapping onto 3-D surfaces in addition to image distortion. The methods described in the paper were used to make two animation sequences for *The Magic Egg*, a compilation of computer generated Omnimax animation sponsored by Siggraph '84.

CR Categories: I.3.3 **[Computer Graphics]**: Picture/Image Generation - *display algorithms;* I.3.7 **[Computer Graphics]**: Three-Dimensional Graphics and Realism - *Color, shading, shadowing, and texture*.

General Terms: algorithms.

Additional Key Words and Phrases: Omnimax, image distortion, texture mapping, space variant filter.

Appeared in IEEE Computer Graphics and Applications, June 1986, pp. 21-27. This version is missing figures.

1. RENDERING AND PROJECTIONS

Rendering programs usually perform a perspective transformation which simulates a pinhole camera: straight lines are transformed to straight lines and planes are transformed to planes. However, the Omnimax projection of a three dimensional environment is a 180 degree fisheye projection that does not preserve linearity or planarity, so rendering programs must be modified to perform an Omnimax projection rather than a perspective projection.

Rendering programs work in a variety of different ways and handle a variety of different geometric primitives. Because of their independence from the perspective projection, ray tracers are easily converted for Omnimax [1] (however their computational expense makes them impractical for many applications). Standard polygon renderers, on the other hand, rely heavily on the perspective projection to simplify and speed the process of scan conversion. Among other things, they take advantage of the fact that the perspective projection preserves the convexity of polygons and the linearity of their edges, properties not preserved by the Omnimax projection. In short, standard polygon renderers are much more difficult to convert for Omnimax projection than ray tracers and conversion reduces their computational efficiency.

Actually, the problem of software conversion should be examined in the context of a whole image generation system. With some systems the final frames of animation are generated by a single rendering program, but other systems create images piece by piece using a variety of programs. This latter approach is used at the New York Institute of Technology - one program renders polygons, another program renders quadric surfaces, and there are dozens of other programs for processing rendered images to produce final frames [2]. Converting all of the programs in this graphics system to handle Omnimax projection correctly would be prohibitively time consuming and it would complicate existing software and make maintenance more difficult. For these reasons a method of creating Omnimax projections from perspective views was sought.

We are interested in general techniques for creating Omnimax images which allow high resolution, high scene complexity and a variety of geometric modeling primitives. We are willing to sacrifice speed for quality and generality. Others have developed more efficient techniques for specific applications. Max has generated Omnimax animation of molecules using a local linear approximation to the distortion [3], and methods for real-time Omnimax distortion of low resolution images have also been developed [4].

2. THE OMNIMAX PROJECTION

Max has described the Omnimax projection lens and theatre geometry in detail [1] and the following description has been condensed from his account.

In an Omnimax theater, film is projected through a fisheye lens onto a hemispherical screen. The screen fills the audience's field of view and this extreme field of view necessitates using a large film format to provide sufficient resolution. Omnimax is the largest motion picture film format; frames are recorded ‘‘horizontally’’ on 70mm film stock.

A 180 degree fisheye lens (round type) projects half of the environment into a circular image. The Omnimax film image, shown in Figure 1, is a 180 degree fisheye projection with the bottom of the frame cropped to an approximately elliptical arc. The projected Omnimax image covers a full 180 degrees horizontally but due to the cropping of the frame it covers substantially less vertically, about 135 degrees. In a typical Omnimax theatre, the axis of projection (which pierces the film frame at the ‘‘projection center’’ in Figure 1) is elevated at 11 degrees from the horizontal.

Each pixel in the raster grid of an Omnimax image corresponds to a ray through the Omnimax projection lens. Mapping pixels to rays is done by simulating the C430 Omnimax projection lens as follows:

The limiting circle of the 180 degree fisheye projection is assigned unit radius and its center is chosen as the image origin. The projection lens is rotationally symmetric about the axis of projection, so the effect of the lens can be described by a single function which relates radial displacement from the center of the circle to the angle of the projected ray from the axis of projection. Points in the image are assigned polar coordinates (r, θ) and converted to spherical coordinates with angular coordinates θ and ϕ where θ is longitude and ϕ is the angle from the axis of projection. The transformation from polar to spherical coordinates keeps θ the same and transforms r into ϕ. The expression for ϕ as a function of r, adapted from Max [1], is

$$\phi(r) = 1.411269r - .094389r^3 + .25674r^5$$

See Max [1] for a more detailed discussion of lens and theatre geometry and film frame layout.

3. OBTAINING AN OMNIMAX PROJECTION FROM A CUBE PROJECTION

To produce an Omnimax projection, standard image generation programs are used to obtain the projection of the environment onto four faces of a cube centered at the viewpoint, and this cube of texture is then processed to obtain an Omnimax projection. The orientation of the cube corresponds to a coordinate system with a horizontal left-right axis and another axis coinciding with the axis of projection. Each face of the cube of texture is obtained by pointing the camera down the appropriate axis of the coordinate system and rendering with a 90 degree view angle. The rendered views are the projections on the front, top, left, and right faces of the cube.

Figure 3 shows the projection of a three dimensional scene onto four faces of a cube. This cube of texture is shown unfolded in Figure 4 wherein the darkened areas correspond to regions of the scene that lie outside the Omnimax field of view. Figure 2 shows the region of the Omnimax frame that each of the cubefaces contributes to.

Creating the Omnimax projection from the cube of texture can be done in scanline order. Each pixel in the Omnimax projection corresponds to a ray through the projection lens, the direction of which can be found with some straightforward trigonometry using the above lens distortion formula. This ray is then intersected with the cube of texture to determine which texture pixel corresponds to the pixel in the Omnimax projection. Actually, this point sampling approach will cause aliasing, the most familiar symptom of which is jaggies; an efficient filtering technique which eliminates visible aliasing is described in the following sections.

Figure 5 is an Omnimax projection obtained from the cube of texture of Figure 3. Figure 5 has twice the horizontal resolution of the cubefaces of Figure 3, and this corresponds to a ratio of “source pixels” (the bright area of Figure 4) to “destination pixels” in the Omnimax projection of roughly 2 : 3. The exact ratio of source pixels to destination pixels isn’t important as long as there is rough parity (having far more source pixels than destination pixels wastes source texture resolution; having far fewer source pixels than destination pixels magnifies the source texture unnecessarily). Empirical tests have shown that a resolution of approximately 2000x1500 is sufficient for raster Omnimax frames [7].

A potential problem with this projection scheme is discontinuities in the Omnimax projection at the boundaries between cubeface regions shown in Figure 2. If, for example, a polygon in the Omnimax projection straddles a seam, the polygon will be clipped and rendered in two separate cubefaces. Whether or not the projected polygon will exhibit visible artifacts at the seam depends on the algorithms used for clipping, shading, etc., and in some cases it may be necessary to use more sophisticated algorithms. In practice, seams have never been a problem for us.

It should be mentioned that the framework for projection need not be a cube. An Omnimax projection can be obtained from any combination of perspective views with viewpoint at the camera which collectively cover the Omnimax field of view. For example, it's possible to use just two perspective views, one covering the left half of the Omnimax field of view, the other covering the right half. This scheme, however, requires using very large view angles which result in poor image resolution near image center. All things considered, the cube appears to be the best framework for projection.

4. FILTERING

In order to control aliasing, the cubeface images must be filtered, not point sampled [5] (see Figure 7). Instead of treating pixels in the Omnimax frame as points they are treated as squares or circular areas and the corresponding rays through the projection lens are considered to be cones which intersect cubefaces in quadrilaterals or ellipses [6,7]. The quadrilateral or elliptical areas are then filtered to obtain pixel values. Omnimax distortion is nonlinear, so the resampling grids on the cubefaces are nonuniform (see Figure 8). Consequently, a space variant filter is required and the more efficient Fourier convolution techniques cannot be applied. In response to the need for generality and speed a new filtering technique was developed.

Image filtering for Omnimax distortion is very similar to the technique of texture mapping onto curved surfaces in 3-D since both require space variant filters, and the filter we present can be used for either application. Applying the terminology of texture mapping, the cubeface image is referred to as the *texture* with coordinates u and v and the distorted output picture as the *screen* with coordinates x and y. Some of the methods that have been proposed for texture filtering are classified according to kernel shape in Table 1. The most important features of a texture filter are the generality of its shape, which determines quality, and the number of samples which must be accessed and the cost of each access, which determine speed.

High quality filters such as those by Catmull, Blinn-Newell, Feibush-Levoy-Cook, and Gangnet-Perny-Coueignoux operate on arbitrary quadrilaterals or ellipses, but their computational cost is proportional to texture space area, and this cost can be very high when large texture areas map to small screen areas.

The fastest filters such as those by Dungan-Stenger-Sutty, Williams, Crow, and Perlin have constant cost per screen pixel, but they are limited to orthogonally oriented squares, rectangles, or ellipses, so they cannot filter elongated diagonal areas accurately. This limited shape control causes either aliasing or blurring of the texture. These "constant cost" filters also require extra memory and time for pre-integration or pre-filtering of the texture. When the texture changes from frame to frame, as it does for this method of Omnimax frame creation, the cost per screen pixel when setup is considered is not constant, but includes a term proportional to texture area. In general the constant cost filters are most desirable for static textures with mappings which compress large areas to small. Since the texture areas corresponding to each screen pixel are small for Omnimax distortion (typically 9-16 pixels in our program), an efficient high quality filter may be nearly as fast as a constant cost filter for this application.

5. ELLIPTICAL WEIGHTED AVERAGE FILTER (EWA)

The *elliptical weighted average filter* (EWA) was developed in order to attain more efficient high quality filtering. The area of the filter is an arbitrarily oriented ellipse, as in the filters by Feibush-Levoy-Cook [8] and Gangnet-Perny-Coueignoux [9]. These two methods require every texture sample point to be mapped to or from screen space for weighting by the circular kernel residing in screen space. To avoid this cost EWA distorts the circular screen space kernel into an ellipse in texture space so that texture weighting can be done more directly. This is similar to the method used by Blinn and Newell who distorted a screen square into a texture parallelogram [10]. The radial cross-section of the kernel is controlled by a weight lookup table, so it can be any function. Like other high quality filters, EWA's cost is proportional to the texture space area, but because it avoids mapping each texture pixel between screen space and texture space it is faster than filters of similar quality.

A circular screen space pixel will normally map to an elliptical texture space area under the perspective transformation, assuming local planarity of the surface (this approximation breaks down near vanishing points, in which case the circles can map to parabolas or hyperbolas). Point-in-ellipse testing can be done with one function evaluation (this is faster than point-in-quadrilateral testing, which requires substitution into four line equations). The function for this test is a quadratic in the texture coordinates u and v:

$$Q(u,v) \equiv Au^2 + Buv + Cv^2$$

where $u = 0$, $v = 0$ is the center of the ellipse. This function is an elliptical paraboloid, concave upward. Points inside the ellipse satisfy $Q(u,v) < F$ for some threshold F. In texture space, the contours of Q are concentric ellipses (Figure 9), but when mapped to screen space, they are nearly circular. Since Q is parabolic, Q is proportional to r^2, where r is distance from the center of a pixel in screen space. This radius r is just the parameter needed when indexing a kernel, so Q can serve two purposes: inclusion testing and kernel index.

The kernel $f(r)$ is stored in a weight lookup table *WTAB*. Rather than index *WTAB* by r, which would necessitate the calculation of $r = \sqrt{Q}$ at each pixel, we define

$$WTAB[Q] \equiv f(\sqrt{Q})$$

so that the array can be indexed directly by Q. Warping a lookup table for computational efficiency is a useful trick which has been applied elsewhere [3,11]. A good kernel to use is the gaussian $f(r) = e^{-\alpha r^2}$ shown in Figure 10, for which $WTAB[Q] = e^{-\alpha Q}$. The gaussian is preferred to the theoretically optimal *sinc* kernel because it decays much more quickly. By proper scaling of A, B, C, and F, the length of the *WTAB* array can be controlled to minimize quantization artifacts (several thousand entries have proven sufficient).

To evaluate Q efficiently, the method of *finite differences* is employed. Since Q is quadratic, two additions will suffice to update Q from one pixel to the next [3]. The following pseudocode implements the EWA filter for monochrome pictures (it is easily modified for color). Integer variables are in lower case and floating point variables are in upper case.

```
/* Let texture[u,v] be a 2-dimensional array holding texture image */
/* Compute texture space ellipse center (U0, V0) from screen coordinates (x, y) */
/* Compute (Ux,Vx) = (∂u/∂x, ∂v/∂x) and (Uy,Vy) = (∂u/∂y, ∂v/∂y) */
/* Now find ellipse coefficients: */
A = Vx*Vx+Vy*Vy
B = −2.*(Ux*Vx+Uy*Vy)
C = Ux*Ux+Uy*Uy
F = Ux*Vy−Uy*Vx
F = F*F
/* scale A, B, C, and F equally so that F = WTAB length */

/* Ellipse is AU^2 + BUV + CV^2 = F,   where U = u − U0, V = v − V0 */

EWA(U0,V0,A,B,C,F)
begin
  /* Find bounding box around ellipse: u1 ≤ u ≤ u2, v1 ≤ v ≤ v2 */
  NUM = 0.
  DEN = 0.
  DDQ = 2.*A
  U = u1−U0
  for v=v1 to v2 do begin                       /* scan the box */
    V = v−V0
    DQ = A*(2.*U+1.)+B*V                        /* Q(U+1,V)-Q(U,V) */
```

```
        Q = (C*V+B*U)*V+A*U*U
        for u=u1 to u2 do begin
          if Q<F then begin                           /* ignore pixel if Q out of range */
            WEIGHT = WTAB[floor(Q)]
            NUM = NUM+WEIGHT*texture[v,u]            /* read and weight texture pixel */
            DEN = DEN+WEIGHT                          /* denominator (for normalization) */
          end
          Q = Q+DQ
          DQ = DQ+DDQ
        end
      end
      return(NUM/DEN)
    end
```

This implementation can be optimized further by removing redundant calculations from the v loop and, with proper checking, by using integer arithmetic throughout.

The EWA filter computes weighted averages of elliptical areas incrementally, requiring 1 floating point multiply, 4 floating point adds, 1 integerization, and 1 table lookup per texture pixel. Blinn and Newell's method, which is the filter most similar to EWA, appears to have comparable speed but inferior quality, based on the very brief description in their paper [10]. The texture filters by Feibush-Levoy-Cook [8] and Gangnet-Perny-Coueignoux [9] have the same quality as EWA, but they require each sample point to be mapped between screen space and texture space, an operation requiring at least one division and two multiplications even when done incrementally [12]. Based on this analysis the EWA filter appears to be about twice as fast as the other methods of similar quality.

6. AREAS FOR FUTURE WORK

The mapping program for creating an Omnimax projection from a cube projection can be easily modified to produce other nonlinear projections such as cylindrical projections (e.g. the Mercator projection) or a picture of the environment as it would appear reflected in a chrome ball. The projection formulas for these applications are easily derived. The cubefaces required depend on the projection; all six are required, for example, to produce a chrome ball reflection.

Since the computational cost of EWA filtering is proportional to texture space area, texture mapping with EWA is very slow compared with constant cost filtering when large texture areas map to small screen areas. An untested method of dealing with this problem is to use EWA (or some other high quality filter) on a prefiltered image pyramid [6,13]. EWA filtering would be performed on small elliptical areas at the appropriate level in the image pyramid. Actually, performing such filtering at two adjacent levels in the image pyramid and interpolating the results as in Williams [13] would be preferable. In theory, this hybrid approach offers the best of both worlds: high quality filtering and constant worst case cost per pixel.

An unexplored approach to digital Omnimax distortion is decomposition of the 2-D mapping and filtering into two passes of 1-D mapping and filtering [14] using least squares approximations [15].

7. CONCLUSIONS

This method of producing Omnimax projections is completely general in the sense that an Omnimax projection can be produced of any scene that can be rendered in perspective. Only a modest programming effort is required since existing rendering programs need not be modified and the only new software is the mapping program. All rendering can be done with existing, efficient programs which exploit the linearity of the perspective transformation.

These techniques were used at the New York Institute of Technology to create two sequences of animation for *The Magic Egg*, a compilation of Omnimax animation sponsored by Siggraph '84 and produced by Garrick Films [16,17]. Figure 6 is a frame from one of these sequences, a cycle of Omnimax animation in which the viewpoint moves through a labyrinth of vines with flowers and leaves which trace the edges of a diamond lattice. Cubefaces were rendered at a resolution of 1024x960 and from these, 1966x1436x24 bit Omnimax frames were made. The high resolution requirements of Omnimax format result in large data volumes (up to 8.5 megabytes per frame) and slow frame generation. On a VAX 11/780 creation of the cubeface images took between one and six hours per frame followed by mapping and filtering which took one hour.

To date, about twenty minutes of computer generated Omnimax animation has been produced in raster format, much of which was rendered with ray tracing on supercomputers. Our method provides an alternative to this brute force approach.

8. ACKNOWLEDGEMENTS

We would like to thank Eddie Garrick for his infectious enthusiasm for computer generated Omnimax films and Nelson Max for detailed comments on an early draft of this paper. The cube projection of Figures 3 and 4 was generated by Dick Lundin using a tree model created by Jules Bloomenthal and a terrain model created by Paul P. Xander.

9. REFERENCES

1. Nelson L. Max, "Computer Graphics Distortion for IMAX and OMNIMAX Projection", *Nicograph '83 Proc.*, Dec. 1983, pp. 137-159.

2. Tom Duff, "Tools for 3-D Rendering", *Computer Graphics* (SIGGRAPH '85 Proceedings), Vol. 19, No. 3, July 1985.

3. Nelson L. Max, "ATOMLLL - ATOMS with Shading and Highlights", *Computer Graphics* (SIGGRAPH '79 Proceedings), Vol. 13, No. 2, Aug. 1979, pp. 165-173.

4. "News Update", *Electronic Imaging*, Feb. 1985, p. 18.

5. Franklin C. Crow, "The Aliasing Problem in Computer-Generated Shaded Images", *C.A.C.M.*, Vol. 20, Nov. 1977, pp. 799-805.

6. William Dungan, Jr., Anthony Stenger, and George Sutty, "Texture Tile Considerations for Raster Graphics", *Computer Graphics* (SIGGRAPH '78 Proceedings), Vol. 12, No. 3, Aug. 1978, pp. 130-134.

7. Charles E. Henderson and Agnis Kaugars, "Satellite Imagery in the Production of an Omnimax Animated Film Sequence", *Nicograph '83*, Dec. 1983, pp. 127-136.

8. Eliot A. Feibush, Marc Levoy, and Robert L. Cook, "Synthetic Texturing Using Digital Filters", *Computer Graphics* (SIGGRAPH '80 Proceedings), Vol. 14, No. 3, July 1980, pp. 294-301.

9. Michel Gangnet, Didier Perny, and Philippe Coueignoux, "Perspective Mapping of Planar Textures" *Eurographics '82*, 1982, pp. 57-71 (slightly superior to the version which appeared in *Computer Graphics*, Vol. 16, No. 1, May 1982, pp. 70-100).

10. James F. Blinn and Martin E. Newell, "Texture and Reflection in Computer Generated Images", *C.A.C.M.*, Vol. 19, No. 10, Oct. 1976, pp. 542-547.

11. Kenneth Turkowski, ''Anti-Aliasing through the Use of Coordinate Transformations'', *ACM Transactions on Graphics*, Vol. 1, No. 3, July 1982, pp. 215-234.

12. Alvy Ray Smith, ''Incremental Rendering of Textures in Perspective'', *SIGGRAPH '80 Animation Graphics seminar notes*, July 1980.

13. Lance Williams, ''Pyramidal Parametrics'', *Computer Graphics* (SIGGRAPH '83 Proceedings), Vol. 17, No. 3, July 1983, pp. 1-11.

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

15. Donald Fraser, Robert A. Schowengerdt, and Ian Briggs, ''Rectification of Multichannel Images in Mass Storage Using Image Transposition'', *Computer Vision, Graphics, and Image Processing*, Vol. 29, No. 1, Jan. 1985, pp. 23-36.

16. Patrice M. Wagner, ''Omnimax: Perfecting a Medium for 3-D Digital Animation'', *Computer Graphics World*, Oct. 1984, pp. 10-16.

17. Cal Kirchhof, ''The Making of *The Magic Egg*'', *The Big Frame*, Spring 1985.

18. Edwin E. Catmull, *A Subdivision Algorithm for Computer Display of Curved Surfaces*, PhD thesis, Dept. of CS, U. of Utah, Dec. 1974.

19. Franklin C. Crow, ''Summed-Area Tables for Texture Mapping'', *Computer Graphics* (SIGGRAPH '84 Proceedings), Vol. 18, No. 3, July 1984, pp. 207-212.

20. Kenneth Perlin, ''Course Notes'', *SIGGRAPH '85 State of the Art in Image Synthesis seminar notes*, July 1985.