A New Look At Mipmap Level Estimation Techniques

Leon Shirman and Yakov Kamen
Epson Research and Development, Inc.

ABSTRACT

In this work, we study various methods of mipmap level estimation. We show that despite their differences, these methods depend on the interpolated inverse homogeneous coordinate. We introduce a new method based on homogeneous coordinates only that has functionality and efficiency advantages over traditional approaches.

Keywords: texture mapping, mipmapping, mipmap level, inverse homogeneous coordinates

1. INTRODUCTION

Texture mapping is a widely used technique in modern graphics applications. Aliasing artifacts that arise when mapping two-dimensional images onto three-dimensional objects are well-known and have been studied by many researchers [Watt93, Wolberg90]. Various filtering schemes are usually used to deal with this problem [Crow84, Greene86]. However, by far the most popular filtering scheme is mipmapping [Williams83]. In this method, the original texture is averaged down to successively lower resolutions, each image in the *mipmap* sequence being one half of the resolution of the previous image. Then, instead of always using the original texture map, an appropriately scaled image is used.

Choosing the proper image in the sequence, or, alternatively, the proper *mipmap level*, is extremely important to minimize aliasing. However, surprisingly few approaches to selecting this level are described in the literature. The most common ones are Heckbert's derivatives method [Heckbert83] and area-based estimation [Watt93]. Both of these techniques are computationally expensive, which is a serious disadvantage, because mipmap level estimation has to be done for each pixel of the rendered image.

In this work, we start by examining existing mipmap estimation methods. We show that despite their apparent formulation differences, all of them depend on the interpolated inverse homogeneous coordinate. Further, we introduce a new model that is a function of inverse homogeneous coordinate only. This model possesses various desirable properties and is very computationally efficient. We conclude by giving direction for future research.

2. OVERVIEW OF EXISTING METHODS

Traditionally, mipmap level estimation is based on the relationship between screen and texture coordinates, namely, on the mapping from screen to texture coordinates $u = u(x,y)$ and $v = v(x,y)$.

2.1. Derivative-based Methods

Consider an axis-aligned unit square (i.e. pixel) in screen space with lower left coordinates (x_0, y_0). According to Taylor expansion to the first order,

$$u(x,y) = u(x_0,y_0) + \frac{\partial u}{\partial x}(x-x_0) + \frac{\partial u}{\partial y}(y-y_0) + O(x-x_0, y-y_0)$$

and similarly for v. Therefore, the bottom and left sides of the unit square will map approximately to vectors $\left(\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}\right)$ and $\left(\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}\right)$ respectively.

Heckbert [Heckbert 83] suggests the following *maximum length* formula for mipmap level estimation (1):

$$d = \log\left(\max\left(\sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2}, \sqrt{\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2} \right) \right)$$

where log is taken to the base 2. In geometric terms, this simply means selecting the larger of the Euclidean lengths of the two vectors $\left(\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}\right)$ and $\left(\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}\right)$ for mipmap computation.

Clearly, the above formulation is somewhat arbitrary. Instead of taking Euclidean norm, it is possible to take Manhattan or max norms; instead of taking the larger length, we could consider average length, etc. For example, using average of Manhattan lengths yields a computationally more attractive formula:

$$d = \log\left(\frac{1}{2}\left(\left|\frac{\partial u}{\partial x}\right| + \left|\frac{\partial v}{\partial x}\right| + \left|\frac{\partial u}{\partial y}\right| + \left|\frac{\partial v}{\partial y}\right| \right) \right)$$

Other possible expressions for d can be readily derived.

In addition to the problem of choosing norm in texture space, derivative-based methods have two even more serious drawbacks. First, calculation of partial derivatives of u and v per pixel is required. This is expensive. Second, derivative-based methods are not invariant to rigid transformations in the screen space, such as rotations and mirror images. Figure 1 shows a pair of perspectively distorted, trilinearly mipmapped triangles in two different orientations. Mipmap level distribution according to the maximum length model is superimposed on the texture map. Each shaded region corresponds to an integer mipmap level (i.e. from 1 to 2, from 2 to 3, etc.). This dependence of mipmap levels on triangle orientation could create visible artifacts during animation.

2.2 Invariant Derivatives Method

In the previous section, we only considered partial derivatives of u and v along axis-aligned directions. Since these derivatives change when axes are rotated, so does mipmap level distribution. In order to make the distribution invariant under rigid transformations, we should consider partial derivatives along a specified direction, and then attempt to find a maximum or an average of these derivatives.

From elementary calculus, a partial derivative of u along direction $s = [\cos(\alpha), \sin(\alpha)]$ is given by

$$\frac{\partial u}{\partial s} = \frac{\partial u}{\partial x}\cos(\alpha) + \frac{\partial u}{\partial y}\sin(\alpha)$$

and similarly for v. Then, the square of Euclidean length l of the partial derivative in the texture space is given by

$$l^2 = \left(\frac{\partial u}{\partial s}\right)^2 + \left(\frac{\partial v}{\partial s}\right)^2 = \left(\frac{\partial u}{\partial x}\cos(\alpha) + \frac{\partial u}{\partial y}\sin(\alpha)\right)^2 + \left(\frac{\partial v}{\partial x}\cos(\alpha) + \frac{\partial v}{\partial y}\sin(\alpha)\right)^2$$

We tried to find the maximum of the above expression by differentiating it with respect to α and equating the result to zero. Using Mathematica, we were able to derive closed-form expressions for α; however, they were too complicated to be of practical use.

However, finding the average of partial derivatives turns out to be more tractable. The average can be computed by integrating the above equation for the full circle:

$$l^2_{ave} = \frac{1}{2\pi}\int_0^{2\pi}\left(\left(\frac{\partial u}{\partial s}\right)^2 + \left(\frac{\partial v}{\partial s}\right)^2\right)d\alpha$$

Substituting and carrying out integration yields

$$l^2_{ave} = \frac{1}{2}\left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right)$$

and therefore the mipmap level can be computed (2)

$$d = \frac{1}{2}\log\left(\frac{1}{2}\left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right)\right)$$

Figure 2 shows distribution of mipmap levels using this new method. As we can see, the distribution is invariant under rigid transformations.

We could also attempt to compute maximum and average lengths of partial derivatives using non-Euclidean norms. We have not investigated this approach.

2.3 Area Estimation

Together with Heckbert's method, the area estimation method is relatively well-known and described in literature [Watt93]. The idea of the method is very simple. Instead of computing partial derivatives of u and v and subsequently ignoring higher-order terms of the Taylor expansion, we compute the area a of the quadrilateral in the texture space that the current pixel maps to. Then the mipmap level is

$$d = \log\left(\sqrt{a}\right) = \frac{1}{2}\log(a) \qquad (3)$$

Figure 3 shows mipmap level distribution using this model. By construction, the distribution is clearly invariant under rigid transformations. Another interesting property of this method is that mipmap level is constant along q = const lines, where q is linearly interpolated among the inverses of the homogeneous coordinates at three triangle vertices.

This is because along these lines the mapping from screen to texture space is linear [Blinn92] and therefore pixels map into quadrilaterals with the same area (this is not exactly true, as pixels have non-zero size, but since that size is typically very small compared to the whole image, we can make the assumption that q = const for the whole pixel). Once again, efficiency is a drawback of this method: it is expensive to calculate the area of the image of each pixel in texture coordinates. This is even more costly than calculating partial derivatives for previously described schemes.

In the above approach, we assumed that a pixel is a square. We could change that assumption and consider a pixel to be a circle, and observe that the image of a circle in screen coordinates under perspective transformation is an ellipse in texture coordinates. This observation was used by Greene and Heckbert [Greene86] for construction of elliptical weighted average (EWA) filter. They showed that ellipse's coefficients can be expressed in terms of partial derivatives of $u(x,y)$ and $v(x,y)$ (section 2.1). Our attempts to derive a practically useful expression for the area of this ellipse in terms of the partial derivatives were not successful. However, since we assume that either square or circular pixels are small compared to the whole image, the results obtained with either method should be very similar.

3. INVERSE *W* METHOD

In this section, we show that all the methods discussed above, despite their apparent differences, can be described using inverse homogeneous coordinates $q = 1 / w$. Further, we introduce a new mipmap estimation method that depends on q only.

Many of the methods in the previous section relied on partial derivatives of $u(x,y)$ and $v(x,y)$. These derivatives are usually computed numerically in practical implementations; however, a closed-form solution can be easily derived. Consider a horizontal (y = const) scanline with endpoints x_0 and x_1, u values u_0 and u_1, and weights w_0 and w_1. Then

$$u(x) = \frac{u_0 / w_0 + t\left(u_1 / w_1 - u_0 / w_0\right)}{1/w_0 + t\left(1/w_1 - 1/w_0\right)}, \qquad t = \frac{x - x_0}{x_1 - x_0}$$

Differentiating, we obtain

$$u'(x) = \frac{u'(t)}{x_1 - x_0} = \frac{u_1 - u_0}{x_1 - x_0} \frac{w_0 w_1}{\left(w_1 + t\left(w_0 - w_1\right)\right)^2}$$

However, since $w(t) = w_0 + t(w_1 - w_0)$, then

$$u'(x) = \frac{u_1 - u_0}{x_1 - x_0} \frac{w_0 w_1}{\left(w_0 + w_1 - w\right)^2} \qquad (4)$$

Thus, within each scanline, the partial derivative along the scanline depends only on the homogeneous coordinate. Similar result obviously holds for vertical scanlines, and, in fact, for arbitrary directions.

The above expression can be rewritten in terms of inverse w as follows:

$$q(t) = \frac{1}{w_0} + t\left(\frac{1}{w_1} - \frac{1}{w_0}\right)$$

as follows:

$$u'(x) = \frac{u_1 - u_0}{x_1 - x_0} \frac{1}{q^2 w_0 w_1} = \frac{u_1 - u_0}{x_1 - x_0} \frac{q_0 q_1}{q^2} \qquad (5)$$

where $q_0 = 1/w_0$ and $q_1 = 1/w_1$. This formulation is more convenient for practical use, because it is the inverse homogeneous coordinate q, not w, that needs to be interpolated for the perspective division [Blinn92].

These expressions for partial derivatives can be used to derive closed-form expressions for mipmap level using derivative-based methods from section 2. For example, Heckbert's formula (1) simplifies to

$$d = \max\left(\log(A_x) - 2\log(q),\ \log(A_y) - 2\log(q)\right)$$

where

$$A_x = \sqrt{\frac{(u_1^x - u_0^x)^2 + (v_1^x - v_0^x)^2}{(x_1 - x_0)^2}}\, q_0^x q_1^x$$

and

$$A_y = \sqrt{\frac{(u_1^y - u_0^y)^2 + (v_1^y - v_0^y)^2}{(y_1 - y_0)^2}}\, q_0^y q_1^y$$

The superscript x or y denotes the fact that u, v, and q values are taken from endpoints of horizontal or vertical scanlines, respectively.

Similarly, invariant derivatives computation (2) can be expressed as

$$d = C - 2\log(q)$$

where C depends on two components, one constant along horizontal and the other along vertical scanlines:

$$C = \sqrt{\frac{1}{2}\left(A_x^2 + A_y^2\right)}$$

This result allows a faster practical implementation.

Even though both maximum length and invariant derivatives methods depend on q, strictly speaking, they are not functions of q only. In fact, evaluation of per scanline quantities A_x, A_y, or C accounts for most of computation cost. On the other hand, area estimation method (section 2.3) relies just on the inverse w. This leads us to our next model, where the mipmap level for the whole triangle is a function of q only. We derive the formulation of this model from the equation (5).

If we were considering mipmap level estimation in one dimension, then we could do so in terms of the derivative $u'(x)$ as follows:

$$d = \log(|u'(x)|)$$

$$= \log\left(\left|\frac{u_1 - u_0}{x_1 - x_0}\right|\right) + 2\log\left(\sqrt{q_0 q_1}\right) - 2\log(q)$$

$$= d_{ave} + 2\log(q_{ave}) - 2\log(q)$$

where d_{ave} is the average mipmap level for the whole line segment, and q_{ave} is the geometric average of the inverse w's at the endpoints. Generalizing into two dimensions, we have (6):

$$d = d_{ave} + 2\log(q_{ave}) - 2\log(q), \quad q_{ave} = \sqrt[3]{q_0 q_1 q_2}$$

where d_{ave} is the average mipmap level for the whole triangle (discussed below), q_{ave} is the geometric mean of inverse w's at the triangle vertices, and q is the interpolated inverse w. This *inverse w* method has a clear geometric interpretation. First, we calculate the average mipmap level for the whole triangle. Then, we assume that this level is reached at some interior point within the triangle. Since we are calculating $\log(q)$, it is natural to assume that at this point, $\log(q)$ is the average of logs at triangle vertices:

$$\log(q_{ave}) = \frac{1}{3}\left(\log(q_0) + \log(q_1) + \log(q_2)\right)$$

In other words, q_{ave} is the geometric mean of the inverse w's: $q_{ave} = \sqrt[3]{q_0 q_1 q_2}$.

The average mipmap level for the whole triangle can be computed using a variety of ways, for example by calculating the ratio of areas in texture and screen spaces

$$d_{ave} = \frac{1}{2}\log\left(\frac{A_T}{A_S}\right)$$

or by computing the ratio of triangle perimeter lengths in the same spaces:

$$d_{ave} = \log\left(\frac{P_T}{P_S}\right)$$

Figure 4 shows distribution of mipmap levels using inverse w method. By construction, mipmap level is constant along q = const lines within each triangle. Note, however, that since

$$C = d_{ave} + 2\log(q_{ave})$$

is determined per triangle, mipmap levels could be shifted across triangle boundaries. This property of the method is discussed in the next section.

4. DISCUSSION

The inverse w method, introduced in the previous section, possesses various desirable properties. Similar to area estimation method, it is invariant under rigid transformations. Also, mipmap levels are constant along q = const lines. In geometric terms, this means that mipmap level is a function of the distance to the eye point.

Most importantly, however, and unlike the other approaches discussed above, this method is very efficient. The only computations per pixel required are a subtraction, a shift (multiplication by 2) and calculation of $\log(q)$. The logarithm function can be efficiently implemented with a lookup table. Further, we found that for vast majority of triangles in practical applications, it is acceptable to simply compute $C - 2\log(q)$ at the triangle vertices, and then perform linear interpolation for interior pixels. That way, mipmap level estimation simply becomes another triangle interpolant, such as screen and texture coordinates, color, etc. In case when interpolation is not accurate enough, i.e. when the triangle has high w ratio among its vertices and therefore spans several mipmap levels, it is always possible to subdivide the triangle so that w ratios for subtriangles are appropriately reduced [Kamen98].

A possible area for future research is determination of the constant C. In the previous section, we defined it as

$$C = d_{ave} + 2\log(q_{ave})$$

While this formulation has a geometric interpretation, it is still somewhat artificial. For example, since C is calculated per triangle, there could be a discontinuity of mipmap levels across triangle boundaries (Fig. 4). This may or may not be desirable.

In all of the described methods, the final mipmap level depended in one way or another on some relationship between screen and texture coordinates. What if we base our estimation on the world coordinates instead of screen coordinates? It is highly likely that in the world coordinates, texture coordinates are assigned to vertices in such a manner as to maintain the constant "stretching" of the texture map on the object surface. This way, texture appears to be naturally painted on the surface without distortions. If that is the case, the constant C will be the same for the whole object, and therefore the continuity of the mipmap level across the whole surface can be maintained. Otherwise, C should clearly be determined on a per-triangle basis.

One possible approach to defining C for the whole object proceeds in two steps. The first step is computing the average mipmap level d_{wc} for the triangle in world coordinates. This can be done by computing the ratio of the area (perimeter) of the triangle in texture space to the triangle area (perimeter) in world coordinates. Assuming the

constant texture map stretching, d_{wc} will be the same for all triangles.

Since the rendering takes place in screen coordinates, d_{wc} needs to be adjusted by the scale of the viewing transformation. This scale s can be defined, for example, using factorization of viewing transformation. Abi-Ezzi [AbiEzzi90] showed that a perspective transformation can be factored into three parts: a rigid transformation (rotation and translation), a simplified projective transformation (essentially shear in homogeneous coordinate), and a scaling transform. In this formulation, s is the norm of the above scaling transform, so that

$$C = d_{wc} + \log(s)$$

Thus, C is constant for the whole object and needs to be recalculated only when a viewing transformation changes. Figure 5 shows mipmap level distribution when C is computed using this model. In this case, mipmap level depends on q, or, alternatively, to the distance from the eye point only.

As we can see from Figures 1-4, the methods discussed in section 2 preserve mipmap level continuity for a pair of coplanar triangles. However, this is not the case for complex objects, represented by many triangular facets. Therefore, these methods are similar to the inverse w method in this respect. On the other hand, the approach described above truly preserves the continuity. Figure 6 shows mipmap level distribution for a tessellated spline using maximum length model. Figure 7 shows continuous distribution for the same object.

5. CONCLUSION

In this work, we have studied various mipmap level computation methods. We have investigated current techniques, discussed their drawbacks, and introduced a new Invariant Derivatives method. While it belongs to the family of derivative-based schemes, it shows an improvement over existing solutions.

Further, we have shown that partial derivatives along any given direction depend on the interpolated inverse of homogeneous coordinates. This allowed us to derive simple closed-form expressions for derivative-based methods. More importantly, that lead us to a new conceptual model of computing the mipmap level based on the inverse homogeneous coordinates only. This approach is very efficient and possesses various desirable properties.

Finally, we have shown that under certain conditions, it is possible to extend this model to preserve the continuity of mipmap levels for the whole object. In this approach, mipmap level at a certain point depends on the distance to the eye point only, making this method more predictable and even more efficient. Various mathematical formulations of this model is an open research issue.

Another wide area for research is general description of mipmap level functions. So far, we have considered functions of the type $d = C - 2\log(q)$. However, more general formulations are possible, such as $d = C - \log(f(q))$ or $d = C + \log(g(w))$, where f and g are some monotone functions.

6. REFERENCES

[Abi-Ezzi90] S. Abi-Ezzi and M. Wozny, "Factoring a Homogeneous Transformation for a More Efficient Graphics Pipeline", *Proc. Eurographics 90*, pp. 245-255, Sept. 1990.

[Blinn92] J. Blinn, "Hyperbolic Interpolation", *Computer Graphics and Applications*, pp. 89-94, July 1992.

[Crow84] F. Crow, "Summed Area Tables for Texture Mapping", *Computer Graphics*, 18(3), pp. 207-212, 1984.

[Heckbert83] P. Heckbert, "Texture Mapping Polygons in Perspective", Tech. Memo No. 13, NYIT Computer Graphics Lab, April 1983.

[Greene86] N. Greene and P. Heckbert, "Creating Raster Omnimax Images Using the Elliptically Weighted Average Filter", *Computer Graphics and Applications*, pp. 21-27, June 1986.

[Kamen98] Y. Kamen and L. Shirman, "Triangle Rendering Using Adaptive Subdivision", to appear in *Computer Graphics and Applications*, 1998.

[Watt93] A. Watt, "3D Computer Graphics", 2nd edition, Addison-Wesley, 1993.

[Williams83] L. Williams, "Pyramidal Parametrics", *Computer Graphics*, 17(3), pp. 1-11, 1983.

[Wolberg90] G. Wolberg, "Digital Image Warping", IEEE Computer Society Press, Los Alamitos, California, 1990.