Efficient Magnification of Bi-Level Textures

Jörn Loviscach*
Hochschule Bremen

1 Introduction

Standard texture magnification through bilinear interpolation blurs road signs and similar items excessively. We use a small, single-channel MIP map and a pixel shader of twelve instructions to strongly improve the look of bi-level (aka binary) textures. The main idea is to threshold the most detailed level of the MIP texture into a bi-level image. For an optimum result, the texture does not contain the original image, but a processed version.

The focus on bi-level textures and on strict efficiency (one texture read, no branching) sets this method apart from recent work: Ramanarayanan et al. [2004] as well as Tumblin and Choudhury [2004] introduce data structures for textures containing smooth gradients as well as hard edges. Sen [2004] additionally addresses GPU-based rendering using a pixel shader of 45 instructions.

2 Thresholding and Preprocessing

Instead of pure thresholding, a variable scaling a is used to determine the grayscale pixel color $c \in [0,1]$ by $c = \mathrm{clamp}(\frac{1}{2} + (t - \frac{1}{2})a)$, where $t \in [0,1]$ is the result of a MIP texture request. The clamping to $[0,1]$ is built into graphics hardware. Using the screen space derivatives of the texture coordinates we estimate the local number p of texels per length of a screen pixel and then set $a = \max(3/p, 1)$.

On strong minification ($p > \frac{1}{3}$) we have the same output as for classical MIP mapping, i.e. we see grayscale averages. For increasing magnification, $p \to 0$ and $a \to \infty$, thus producing a thresholded image. This smooth transition from grayscale to bi-level hides the switching between MIP map level 0 and 1. Furthermore, on magnification it introduces anti-aliasing through a grayscale transition region between $c = 0$ and $c = 1$. Due to the term $3/p$, this region is approximately one third of a pixel wide.

The upper MIP levels are filled as usual; level 0 is specially optimized. Starting from a vector graphics file, we create a high-resolution bitmap of the image. Here, we find where and at which angle the outline intersects the edges between the centers of level-0 texels. Our objective is to fill level 0 of the MIP map in such a way that the thresholded bilinearly interpolated image behaves similarly. For this image, locations and angles of the intersections can easily be found from the grayscale values at the adjacent texels.

An outline that intersects n edges between texels leads to $2n$ conditions: n locations and n angles. The shape of the contour is affected by at most $2n$ texels, often only a few less than that. Thus, when trying to solve what grayscale values should be assigned to these texels, we have slightly too few unknowns to satisfy all conditions precisely. We even lose one unknown because the thresholded image does not change if we scale all grayscale values about $\frac{1}{2}$.

Since the system is overdetermined, we employ an optimization process: MIP map level 0 is filled with the values 0 and 255 according to the high-resolution image at the texel centers. Then we restrict our view to the small number of texels along contours. We find their values using an approach related to simulated annealing.

*e-mail: jlovisca@informatik.hs-bremen.de

This optimization minimizes the deviation from the high-resolution image. To create a distance metric, we sum all quadratic differences between the intended and the actual locations of the intersections. For the angle data we add a similar expression, thus enforcing a smooth outline. To keep the stem width of typeface characters constant, the weight of the location-error terms is increased the more the intended outline runs in perfect horizontal or vertical direction.

During the optimization we keep grayscale values away from the middle range 112 to 143. So after optimization the outline generated by thresholding still intersects the same edges between texels. Furthermore, this gap in the range ensures that no grayscale values around 128 are introduced. Otherwise, the small scaling factor a in the transition to the first level of the MIP map would lead to dust-like artifacts around the outline.

3 Results

Lettering such as that in Fig. 1 can be created by textures of a few thousand texels using a pixel shader with a fillrate of 850 Mpixel/s on an Nvidia GeForce 6800 GT. The transition from MIP map level 0 to level 1 does not introduce sudden blur. On top of that, the use of scaling instead of pure thresholding suppresses jaggies.

Visually robust shapes such as found on road signs are reproduced well. Detailed shapes such as serifs may suffer, however, from softening. The method cannot resolve two outline curves within a single texel. However, even sub-texel features may be approximated if the user adjusts the location and angle values. An authoring software to create and edit the optimized texture has been developed.

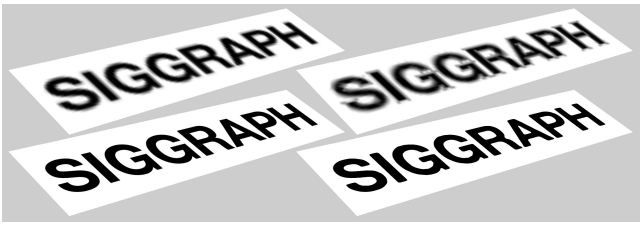

Figure 1: Bilinear interpolation (upper left), optimized texture before and after thresholding (upper/lower right), all at 128×32 texels; a texture of 2048×512 texels for comparison (lower left).

References

RAMANARAYANAN, G., BALA, K., AND WALTER, B. 2004. Feature-based textures. In *Eurographics Workshop on Rendering*, A. K. H. W. Jensen, Ed., 265–274.

SEN, P. 2004. Silhouette maps for improved texture magnification. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, Eurographics Association, 65–73.

TUMBLIN, J., AND CHOUDHURY, P. 2004. Bixels: Picture samples with sharp embedded boundaries. In *Eurographics Workshop on Rendering*, A. K. H. W. Jensen, Ed., 186–196.