Texture Mapping with a Jacobian-Based Spatially-Variant Filter

Ke Deng∗ Jingdan Zhang∗ Lifeng Wang† Baining Guo†
† Microsoft Research Asia Tsinghua University
Beijing, P.R.China Beijing, P.R.China

Abstract

In this paper, we describe a new method to map a texture on a surface with a spatially-variant filter. Our filter takes into consideration the effects of anisotropy using a Jacobian approximation while computing the sampling rate, and the interpolation weights are computed with a sinc function. We also discuss how to do forward and backward mapping with the filter, and extend our algorithms to 3D meshes. Our experimental results verify our analysis.

Keywords: texture mapping, antialiasing, spatially-variant filtering

Introduction

Texture mapping is an established rendering technique used to enhance the realism of 3D models. It has been extensively analyzed in the field of image processing and computer graphics [4, 7]. Texture aliasing may occur during this process, as mentioned by Wolberg [7]. Spatially-invariant kernels produce aliasing when warping images or textures with high frequencies. Using spatially-variant filters can help to reduce the aliasing effect with a variant sampling rate (e.g. MIPMAP [6] and EWA [2, 3, 1]). Recently Wang et al. [5] analyzed the issues of sampling rate, and used an affine-approximated filter for reconstruction from multiple views. In this paper we propose a new texture mapping method which gives better results than existing ones.

Texture filter

Texture filtering involves the sampling rate and the kernel shape. Here we start from texture mapping with a homography. A mapping of a point \( (x', y') \) in the destination image to its correspondent point \( (x, y) \) in the source texture can be shown as:

\[
\begin{align*}
x &= h_{11}x' + h_{12}y' + h_{13} \\
y &= h_{21}x' + h_{22}y' + h_{23}
\end{align*}
\]

The sampling rate is illustrated by the Jacobian of the homography:

\[
\begin{bmatrix}
dx \\
dy
\end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\
J_{21} & J_{22} \end{bmatrix} \begin{bmatrix} dx' \\
dy'
\end{bmatrix},
\]

From the analysis in [5], we can find that a sinc kernel is an ideal sampling kernel to avoid frequency overlapping in the sampling process. With the Jacobian-based sampling rate and the sinc kernel, our filter is taken as:

\[
\begin{align*}
r(x, y) &= \text{sinc}((J_{11} + J_{21})x, (J_{12} + J_{22})y) \otimes \text{sinc}(x, y) \\
&= \text{sinc}(x \cdot \max(J_{11} + J_{21}, 1), y \cdot \max(J_{12} + J_{22}, 1))
\end{align*}
\]

where \( \otimes \text{sinc}(x, y) \) means a necessary prefilter to eliminate frequency that is higher than destination resolution.

Comparison with EWA

EWA is a well-known spatially-variant filter. Figure 1 shows the difference between a sinc kernel and a Gaussian kernel in 2D case. Our kernel in 2D is also a sinc kernel and takes the form of \( \text{sinc}(x, y) = \text{sinc}(x)\text{sinc}(y) \), and the Gaussian kernel takes the form of \( f(x, y) = e^{-2(x^2+y^2)} \). It can be observed that a Gaussian will not be a good approximation of a sinc kernel when extended beyond the area \( \delta (-1..1,-1..1) \) in 2D, which makes the result more blurred than ours in figure 2.
Forward mapping and backward mapping There are two approaches for mapping a texture to a surface: backward mapping and forward mapping. When doing backward mapping, we compute each pixel in the destination image by resampling the source texture. Destination pixels in scan-line order are inversely mapped and sampled on the source texture. We can also splat source pixels to destination [8], which is called forward mapping [1]. We apply our filter to both approaches, and find that backward mapping gives better result than forward mapping in that forward mapping tends to create artifacts at the boundary because of the weight leakage at boundaries.

Experimental result Figure 2 is an example of mapping a “Pacific Graphics 2002” texture to a complicated 3D bunny mesh. We apply shading after rendering each pixel on screen in a manner similar to [8]. (a) is the mapping result using a linear interpolation kernel by OpenGL with MIPMAP on, and it looks blurred. (b) is the mapping result using EWA. (c) is the result using our method.

Discussion and future work Our filter is based on a Jacobian sampling rate and a sinc kernel function, which gives better results than traditional filters. But our method consumes more computation time, because the filter is spatially-variant for each pixel. Accelerating is necessary for extending our method to real-time rendering. Our analysis is useful not only for texture mapping on surfaces, but also for applications such as analysis by synthesis, point-based rendering, super-resolution techniques, and VDTM in image-based rendering.

References