First Order Approximation for Texture Filtering

Zhouchen Lin
Microsoft Research, Asia
zhoulin@microsoft.com

Lifeng Wang
Microsoft Research, Asia
lfwang@microsoft.com

Liang Wan
The Chinese University of Hong Kong
lwan@cse.cuhk.edu.hk

Abstract

Texture mapping is a core technique in 3D graphics and is important for computer vision applications as well. The major challenge of texture mapping is high quality anti-aliasing at a reasonably high speed. Previous approaches do not address this problem adequately, since they do not precisely model the anisotropic and spatially variant nature in texture mapping.

In this paper, we handle this problem using signal processing and sampling theory. We first describe an analytic framework to deduce the ideal filter and the corresponding weight distribution for texture filtering. Since the ideal filter does not have a closed-form solution, we further propose a filter that is the first order, yet of high precision, approximation of the ideal filter and makes a closed-form solution possible. This first order approximating (FOA) filter has excellent anti-aliasing capability and a reasonably high rendering speed. The comparison with some well-known filters (box, cubic, EWA, and fast footprint MIPmapping, etc.) also testifies that our filter does have better anti-aliasing performance.

1. Introduction

Texture mapping is critical to all 3D graphics rendering systems by making rendered objects more realistic. This technology is also important for computer vision, especially analysis-by-synthesis techniques for extracting model parameters. Examples include image-based triangulation [10] and facial modeling fitting [7]. The major challenge of texture mapping is anti-aliasing. The ideal texture filter is impractical as it requires weighting among infinite number of samples. Therefore, finding a practical texture filter with high anti-aliasing capability and relatively high rendering speed is very useful.

Generally speaking, the process of texture mapping can be considered as resampling a source texture to create a destination texture according to a mapping function. Many well known kernels, such as the box, the tent, and the cubic interpolation kernels, are spatially-invariant and independent of the mapping function. They are known to produce aliasing when filtering textures with high frequencies.

In contrast, spatially-variant filters are better at reducing the aliasing. Among them, MIPmap [14] is widely used because of its low computational cost. It creates texture pyramids to manage texture level-of-detail (LOD) but uses square pixel approximation with poor weighting, hence resulting in overly-blurred and poor off-angle quality textures. Texture potential MIPmap [1] overcomes the problem of texture over-blurring by building a pyramid of potential maps, and finding the texture values by simple subtraction operations.

Several hardware anisotropic filtering algorithms can produce better results than MIPmap, such as Textram [11], Feline [8], fast footprint MIPmap [6], and Talisman [12]. Texram assumes a square pixel shape with a parallelogram footprint, and computes with equal weights. Feline uses several isotropic probes along a line to implement an anisotropic filter with Gaussian weights. Fast footprint MIPmap also adopts the square pixel assumption with quadrilateral footprint using MIPmap data. It computes the weights in the quadrilateral footprint with a Gaussian kernel. Talisman applies hardware acceleration, and assumes a square pixel shape with quadrangle footprint. The weights are achieved from pre-computed MIPmap.

Previous filters, however, typically treat texture mapping in a rather *ad hoc* fashion. The elliptical weight average (EWA) filter [4, 5, 2] is considered as a benchmark against which all texture mapping methods are compared. It uses circular pixel shape (with elliptical footprint) and Gaussian weights to calculate mapped textures. The EWA filter could be proven from the sampling theory to be close to, but still not, the ideal filter for either reconstructing or prefiltering continuous signals. Thus the EWA filter still produces aliasing effects when filtering textures with very high frequencies and artifacts when up-sampling low frequency signals. Effort has been made to approximate the ideal filter [13, 3], but only partial approximations are obtained.

In this paper, we start from basic signal processing and sampling theory to derive the ideal weight distribution for

texture mapping. Next, we derive a first order approximating (FOA) filter which is a good approximation to the ideal filter. We also analyze this filter thoroughly and show that it has a closed-form solution. Finally we compare our proposed filter with typical texture mapping methods in 2D/3D scenarios.

2. Theory of texture filtering

Texture mapping is a procedure that warps a source texture f onto a surface through a transform T, and prefilters the warped texture to produce the anti-aliased destination texture $\tilde{f}$. It is actually a resampling process. Ideally, to compute the ideal filter associated with T, we identify the following steps [15]:

- Reconstruct a continuous signal f_c from a discrete signal f using a sinc filter, i.e.,

$$f_c(\mathbf{x}) = \sum_{\mathbf{k}} f(\mathbf{k})\text{sinc}(\mathbf{x} - \mathbf{k}). \tag{1}$$

- Map the continuous signal f_c to the desired continuous signal $\tilde{f}_c$ using the transform T, i.e.,

$$\tilde{f}_c(\tilde{\mathbf{x}}) = f_c(T(\tilde{\mathbf{x}})). \tag{2}$$

- Filter $\tilde{f}_c$ using a sinc function so that it is band-limited, i.e.,

$$\tilde{f}_c^{bl}(\tilde{\mathbf{x}}) = \int \tilde{f}_c(\tilde{\mathbf{t}})\text{sinc}(\tilde{\mathbf{x}} - \tilde{\mathbf{t}})\text{d}\tilde{\mathbf{t}}. \tag{3}$$

- Sample $\tilde{f}_c^{bl}$ to produce the discrete output $\tilde{f}$, i.e.,

$$\tilde{f}(\tilde{\mathbf{l}}) = \tilde{f}_c^{bl}(\tilde{\mathbf{l}}). \tag{4}$$

These four steps are illustrated in Figure 1, where we use the embellishment $\tilde{}$ to indicate those variables in the destination domain. By plugging (1), (2), and (3) into (4) recursively, we have:

$$\tilde{f}(\tilde{\mathbf{l}}) = \sum_{\mathbf{k}} f(\mathbf{k}) \int \text{sinc}(T(\tilde{\mathbf{t}}) - \mathbf{k})\text{sinc}(\tilde{\mathbf{l}} - \tilde{\mathbf{t}})\text{d}\tilde{\mathbf{t}}.$$

We refer to

$$h(\tilde{\mathbf{x}}, \mathbf{k}) \equiv \int \text{sinc}(T(\tilde{\mathbf{t}}) - \mathbf{k})\text{sinc}(\tilde{\mathbf{x}} - \tilde{\mathbf{t}})\text{d}\tilde{\mathbf{t}} \tag{5}$$

as the ideal filter, and its discrete version:

$$W(\tilde{\mathbf{l}}, \mathbf{k}) = \int \text{sinc}(T(\tilde{\mathbf{t}}) - \mathbf{k})\text{sinc}(\tilde{\mathbf{l}} - \tilde{\mathbf{t}})\text{d}\tilde{\mathbf{t}} \tag{6}$$

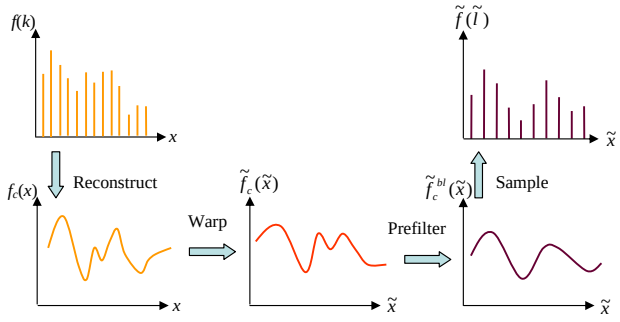


Figure 1. Steps involved in resampling. *Adapted from [15].*

as the ideal weighting function. It determines the interpolation weights from a source pixel $f(\mathbf{k})$ to a destination pixel $\tilde{f}(\tilde{\mathbf{l}})$.

Unfortunately, equations (5) and (6) have no closed-form solutions unless T is special. The numerical evaluation is impractical, considering the intensive computational cost due to the infinite support of the sinc function. As a result, approximations should be performed so that a closed-form solution is possible. In our approach, we take the first order approximation of T to produce a closed-form solution. The resultant filter is very close to the ideal one and produces excellent visual results compared to the previous ones.

3. First order approximating (FOA) filter

We use the first order Taylor series expansion to locally linearize the transform T in (6), which produces a local affine transform [4, 5, 8, 9]. Based on our analysis and experimental results, this first order approximation is good enough. More importantly, this makes possible a closed-form solution of (6).

The first order approximation of $T(\tilde{\mathbf{t}})$ is:

$$T(\tilde{\mathbf{t}}) = T(\tilde{\mathbf{l}} + (\tilde{\mathbf{t}} - \tilde{\mathbf{l}})) \approx T(\tilde{\mathbf{l}}) + \mathbf{J}(\tilde{\mathbf{l}})(\tilde{\mathbf{t}} - \tilde{\mathbf{l}}),$$

where $\mathbf{J}$ is the Jacobian matrix of T. Such an approximation has a reasonably high accuracy. Even $\tilde{\mathbf{t}}$ is far from $\tilde{\mathbf{l}}$, the deviation from the exact value is still quite small, since the modulation of $\text{sinc}(\tilde{\mathbf{l}} - \tilde{\mathbf{t}})$ in (6) decays quite fast. So we have:

$$\begin{aligned} & W(\tilde{\mathbf{l}}, \mathbf{k}) \\ \approx & \int \text{sinc}(T(\tilde{\mathbf{l}}) + \mathbf{J}(\tilde{\mathbf{l}})(\tilde{\mathbf{t}} - \tilde{\mathbf{l}}) - \mathbf{k})\text{sinc}(\tilde{\mathbf{l}} - \tilde{\mathbf{t}})\text{d}\tilde{\mathbf{t}} \\ = & \int \text{sinc}(\mathbf{J}(\tilde{\mathbf{l}})\{[\mathbf{J}(\tilde{\mathbf{l}})]^{-1}[T(\tilde{\mathbf{l}}) - \mathbf{k}] + (\tilde{\mathbf{t}} - \tilde{\mathbf{l}})\})\text{sinc}(\tilde{\mathbf{l}} - \tilde{\mathbf{t}})\text{d}\tilde{\mathbf{t}}. \end{aligned}$$

Let $\hat{\mathbf{t}} = \tilde{\mathbf{l}} - \tilde{\mathbf{t}}$. We can rewrite the interpolation kernel as

$$\begin{aligned} W(\tilde{\mathbf{l}}, \mathbf{k}) &= \int \text{sinc}(\mathbf{J}(\tilde{\mathbf{l}})\{[\mathbf{J}(\tilde{\mathbf{l}})]^{-1}[T(\tilde{\mathbf{l}}) - \mathbf{k}] - \hat{\mathbf{t}}\})\text{sinc}(\hat{\mathbf{t}})\text{d}\hat{\mathbf{t}} \\ &= [\text{sinc}(\mathbf{J}(\tilde{\mathbf{l}})\tilde{\mathbf{x}}) \otimes \text{sinc}(\tilde{\mathbf{x}})]\Big|_{\tilde{\mathbf{x}}=[\mathbf{J}(\tilde{\mathbf{l}})]^{-1}[T(\tilde{\mathbf{l}})-\mathbf{k}]}, \end{aligned}$$

where $\otimes$ is the convolution. Denote

$$h(\tilde{\mathbf{x}}) = \text{sinc}(\mathbf{J}(\tilde{\mathbf{l}})\tilde{\mathbf{x}}) \otimes \text{sinc}(\tilde{\mathbf{x}}). \quad (7)$$

It is the first order approximating filter. Then

$$W(\tilde{\mathbf{l}}, \mathbf{k}) = h([\mathbf{J}(\tilde{\mathbf{l}})]^{-1}[T(\tilde{\mathbf{l}}) - \mathbf{k}]), \quad (8)$$

which is the first order approximating weight distribution.

To find the closed-form solution for (8), we apply the Fourier transform. It is well known that with the Fourier transform, convolution becomes product and a sinc function corresponds to a BOX function over a unit cube C. Let $\mathcal{F}$ denote the Fourier transform, then

$$\begin{aligned} & \mathcal{F}[\text{sinc}(\mathbf{J}(\tilde{\mathbf{l}})\tilde{\mathbf{x}}) \otimes \text{sinc}(\tilde{\mathbf{x}})](\boldsymbol{\omega}) \\ = \; & \mathcal{F}[\text{sinc}(\mathbf{J}(\tilde{\mathbf{l}})\tilde{\mathbf{x}})](\boldsymbol{\omega}) \cdot \mathcal{F}[\text{sinc}(\tilde{\mathbf{x}})](\boldsymbol{\omega}) \\ = \; & |[\mathbf{J}(\tilde{\mathbf{l}})]^{-1}|\text{BOX}_{\text{C}}([\mathbf{J}(\tilde{\mathbf{l}})]^{-1}\boldsymbol{\omega})\text{BOX}_{\text{C}}(\boldsymbol{\omega}) \\ = \; & |\mathbf{J}(\tilde{\mathbf{l}})|^{-1}\text{BOX}_{\tilde{\text{C}}}(\boldsymbol{\omega})\text{BOX}_{\text{C}}(\boldsymbol{\omega}) \\ = \; & |\mathbf{J}(\tilde{\mathbf{l}})|^{-1}\text{BOX}_{\tilde{\text{C}}\cap\text{C}}(\boldsymbol{\omega}), \end{aligned}$$

where $\tilde{\text{C}} = [\mathbf{J}(\tilde{\mathbf{l}})]\text{C}$ is the parallelogram transformed from C via the Jacobian $\mathbf{J}$. Therefore,

$$\begin{aligned} h(\tilde{\mathbf{x}}) &= \mathcal{F}^{-1}[|\mathbf{J}(\tilde{\mathbf{l}})|^{-1}\text{BOX}_{\tilde{\text{C}}\cap\text{C}}(\boldsymbol{\omega})](\tilde{\mathbf{x}}) \\ &= |\mathbf{J}(\tilde{\mathbf{l}})|^{-1}\mathcal{F}^{-1}[\text{BOX}_{\tilde{\text{C}}\cap\text{C}}(\boldsymbol{\omega})](\tilde{\mathbf{x}}). \end{aligned} \quad (9)$$

In the following, we present the closed-form solutions in the 1D and the 2D cases, respectively. For higher dimensions, it is theoretically possible to find the closed-form solutions to the FOA filters, but the solutions may be much more complex, as can seen from the 2D case. However, for our goal, i.e. texture mapping, the investigation on the 2D case is already sufficient[1].

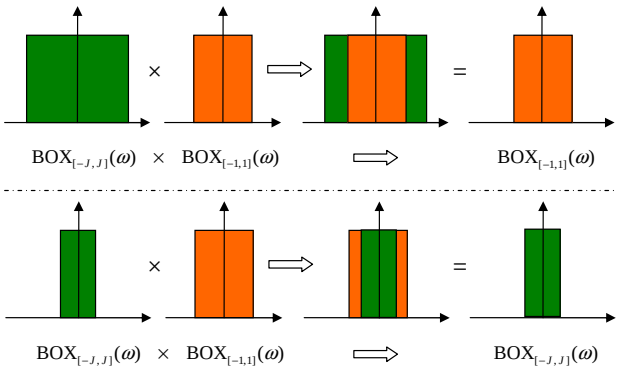


Figure 2. Explanation of prefiltering with two 1D sinc functions.

[1]Note that texture mapping onto a mesh in a 3D space is still a 2D mapping if considered locally.

For a 1D signal (Figure 2), $\tilde{\text{C}}$ is simply a scaled version of C, then

$$\begin{aligned} h(\tilde{x}) &= \frac{1}{J(\tilde{l})}\mathcal{F}^{-1}[\text{BOX}_{\min\{J(\tilde{l}),1\}}(\omega)](\tilde{x}) \\ &= \frac{\min\{J(\tilde{l}),1\}}{J(\tilde{l})}\text{sinc}(\min\{J(\tilde{l}),1\}\tilde{x}). \end{aligned}$$

Therefore,

$$\begin{aligned} W(\tilde{l},k) &= \frac{\min\{J(\tilde{l}),1\}}{J(\tilde{l})}\text{sinc}\left(\frac{\min\{J(\tilde{l}),1\}}{J(\tilde{l})}[T(\tilde{l})-k]\right) \\ &= \begin{cases} \text{sinc}(T(\tilde{l})-k) & \text{if } J(\tilde{l}) \le 1 \\ \frac{1}{J(\tilde{l})}\text{sinc}\left(\frac{1}{J(\tilde{l})}(T(\tilde{l})-k)\right) & \text{otherwise} \end{cases}. \end{aligned}$$

For the 2D case, $\tilde{\text{C}}$ is a parallelogram. Figure 3 shows the common part of $\tilde{\text{C}}$ and C after clipping against each other. We find that there is a closed-form solution to the inverse Fourier transform of the clipped box function.

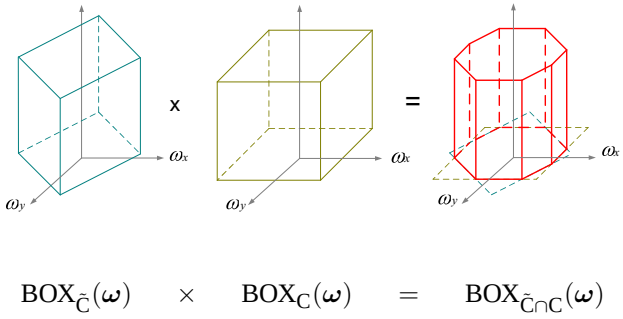


Figure 3. Explanation of prefiltering with two 2D sinc functions. The spectrum of the ideal kernel is the common part of two box functions over C and $\tilde{\text{C}} = \mathbf{J}\text{C}$, respectively. $\tilde{\text{C}}$ is associated with the warping of the input signal while C is associated with the pre-filtering of the output signal.

Suppose that C has N vertices outside $\tilde{\text{C}}$, and $\tilde{\text{C}}$ has $\tilde{N}$ vertices outside C. Then there are 6 cases:

$$\begin{pmatrix} N \\ \tilde{N} \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \begin{pmatrix} 4 \\ 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 4 \\ 2 \end{pmatrix}.$$

Case 1 (Figure 4(a)): $\tilde{\text{C}}$ is completely inside C. This happens when $|J_{11}| + |J_{12}| \le 1$ and $|J_{21}| + |J_{22}| \le 1$, where J_{ij} are the entries of the Jacobian:

$$\mathbf{J} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}.$$

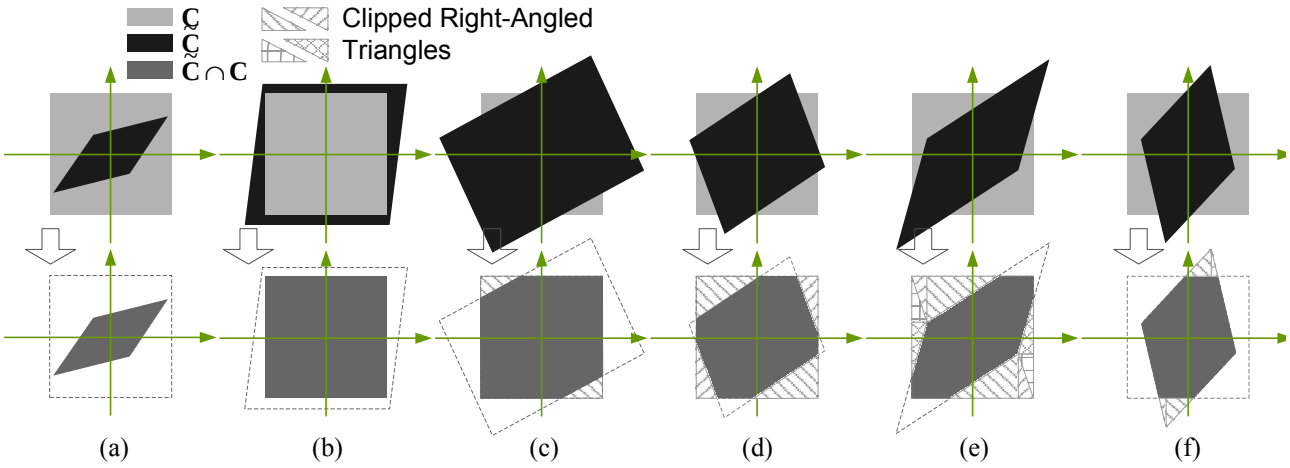


Figure 4. Six cases of the common part of C and $\tilde{C}$ in the 2D case. (a) $\tilde{C}$ is completely inside C. (b) C is completely inside $\tilde{C}$. (c) All vertices of $\tilde{C}$ are outside C but C has only two vertices outside $\tilde{C}$. (d) All vertices of $\tilde{C}$ are outside C, so does C. (e) Both $\tilde{C}$ and C have two vertices outside each other. (f) All vertices of C are outside $\tilde{C}$ but $\tilde{C}$ has only two vertices outside C.

In this case,

$$\begin{aligned} h(\tilde{\mathbf{x}}) &= |\mathbf{J}(\tilde{\mathbf{l}})|^{-1}\mathcal{F}^{-1}[\mathrm{BOX}_{\tilde{\mathrm{C}}}(\boldsymbol{\omega})](\tilde{\mathbf{x}}) \\ &= |\mathbf{J}(\tilde{\mathbf{l}})|^{-1}\mathcal{F}^{-1}[\mathrm{BOX}_{\mathrm{C}}([\mathbf{J}(\tilde{\mathbf{l}})]^{-1}\boldsymbol{\omega})](\tilde{\mathbf{x}}) \\ &= \mathcal{F}^{-1}[\mathrm{BOX}_{\mathrm{C}}(\boldsymbol{\omega})]([\mathbf{J}(\tilde{\mathbf{l}})]\tilde{\mathbf{x}}) \\ &= \mathrm{sinc}([\mathbf{J}(\tilde{\mathbf{l}})]\tilde{\mathbf{x}}). \end{aligned}$$

This means that the pre-filtering by $\mathrm{sinc}(\tilde{\mathbf{x}})$ has no effect at all and hence can be waived. A simple example is image zoom-in. As a result,

$$W(\tilde{\mathbf{l}}, \mathbf{k}) = \mathrm{sinc}(T(\tilde{\mathbf{l}}) - \mathbf{k}).$$

Case 2 (Figure 4(b)): C is completely inside $\tilde{C}$. Since a linear transform keeps the relationship of inclusion, by applying $\mathbf{J}^{-1}$ to both C and $\tilde{C}$, this case is equivalent to $\hat{C} = \mathbf{J}^{-1}C$ being completely inside $C = \mathbf{J}^{-1}\tilde{C}$. This happens when $|J_{11}| + |J_{21}| \leq |\mathbf{J}|$ and $|J_{12}| + |J_{22}| \leq |\mathbf{J}|$, where $|\mathbf{J}|$ is the determinant of $\mathbf{J}$. Thus,

$$\begin{aligned} h(\tilde{\mathbf{x}}) &= |\mathbf{J}(\tilde{\mathbf{l}})|^{-1}\mathcal{F}^{-1}[\mathrm{BOX}_{\mathrm{C}}(\boldsymbol{\omega})](\tilde{\mathbf{x}}) \\ &= |\mathbf{J}(\tilde{\mathbf{l}})|^{-1}\mathrm{sinc}(\tilde{\mathbf{x}}). \end{aligned}$$

This means only prefiltering is necessary. A simple example is image zoom-out. Accordingly,

$$W(\tilde{\mathbf{l}}, \mathbf{k}) = |\mathbf{J}(\tilde{\mathbf{l}})|^{-1}\mathrm{sinc}([\mathbf{J}(\tilde{\mathbf{l}})]^{-1}[T(\tilde{\mathbf{l}}) - \mathbf{k}]).$$

Cases 3 and 4 (Figures 4(c) and (d)): All vertices of $\tilde{C}$ are outside C but C is not completely inside $\tilde{C}$. In these cases, it is more convenient to clip C against $\tilde{C}$, i.e.,

$$\begin{aligned} & \mathcal{F}^{-1}[\mathrm{BOX}_{\tilde{\mathrm{C}}\cap\mathrm{C}}(\boldsymbol{\omega})](\tilde{\mathbf{x}}) \\ = \; & \mathcal{F}^{-1}[\mathrm{BOX}_{\mathrm{C}}(\boldsymbol{\omega})](\tilde{\mathbf{x}}) - \mathcal{F}^{-1}[\mathrm{BOX}_{\text{clipped parts}}(\boldsymbol{\omega})](\tilde{\mathbf{x}}). \end{aligned}$$

The clipped parts are all *regular* right-angled triangles, whose sides that form the right angle are either horizontal or vertical. The formula of the inverse Fourier transform over a regular right-angled triangle can be found in Appendix A. Due to the symmetry nature of the clipped parts with respect to the origin, we should compute the triangles in pairs by utilizing the following property of Fourier transform:

$$\begin{aligned} & \mathcal{F}^{-1}[\mathrm{BOX}_{\mathrm{A}\cup(-\mathrm{A})}(\boldsymbol{\omega})](\tilde{\mathbf{x}}) \\ = \; & 2\mathbf{REAL}\left(\mathcal{F}^{-1}[\mathrm{BOX}_{\mathrm{A}}(\boldsymbol{\omega})](\tilde{\mathbf{x}})\right), \end{aligned}$$

where A is an arbitrary region, $-$A is symmetric to A with respect to the origin and $\mathrm{A} \cap (-\mathrm{A}) = \emptyset$, and $\mathbf{REAL}(\mathbf{x})$ is the real part of $\mathbf{x}$.

Cases 5 (Figure 4(e)): $\tilde{C}$ and C both have two vertices outside each other. In this case, it is also a little more convenient to clip C against $\tilde{C}$. The clipped parts are two symmetric irregular concave quadrilaterals. However, each of them can be decomposed into *four* regular right-angled triangles as shown in the lower part of Figure 4(e). Applying the inverse Fourier transform over regular right-angled triangles, the closed-form solution of $\mathcal{F}^{-1}[\mathrm{BOX}_{\tilde{\mathrm{C}}\cap\mathrm{C}}(\boldsymbol{\omega})](\tilde{\mathbf{x}})$ is possible.

Case 6 (Figures 4(f)): Four vertices of C are all outside $\tilde{C}$, but $\tilde{C}$ has only two vertices outside C. In this case, it is more convenient to clip $\tilde{C}$ against C. The clipped parts are two triangles that can be decomposed into two regular right-angled triangles as shown in the lower part of Figure 4(f). Similar to the previous three cases, we can find the closed-form solution for this case.

Note that in our analysis, if we use a Gaussian function instead of the sinc function in (7)[2], we get the EWA filter (since the convolution of two Gaussians is also a Gaussian). However, the Gaussian function is not as good as the sinc function in removing aliasing. Our experiments will testify to this point. Readers may find that our analysis seems similar to that in [13, 3]. However, both the work therein simply approximated the ideal filter with *separable* 2D sinc functions, while our analysis is much more refined and complete. Our analysis shows that the optimal first order approximation has six cases, and for the latter 4 four cases shown in Figure 4 the optimal FOA filter is *not* a separable 2D sinc function.

4. Experimental results

In this section, we compare results of our FOA filter (9) with the box, cubic, EWA, and fast footprint MIPmap filters, etc., in different scenarios. Since the length of the filter is infinite, we truncate the corresponding FOA filter so that its radius is 2 pixels. Moreover, we also compute the ideal weighting function (6) with numerical evaluation. It is referred to the numerically evaluated ideal (NEI) filter below.

Down sampling images We show the downsampled results in Figure 5. The rendering task is downsampling an image from 1000×1000 to 200×200. Note that the NEI and the FOA filters (Figures 5(e) and (f)) have better quality than other filters. The EWA filter (Figure 5(d)) has comparable result with our approach, but the quality of other filters (Figures 5(a)$\sim$(c)) is apparently inferior to that of ours.

Table 1. Speed (fps) comparison of different methods.

box	cubic	MIPmap	EWA	NEI	FOA
80.4	69.7	375.1	12.0	0.06	4.2

The experiments were run on a 1.4GHz PC with a Geforce III graphics card. Table 1 compares the speeds of various filters. It is notable that the NEI weighting function (6) is highly computation-intensive. Using our first order approximation improves the speed by about 70 times, while the RMSE error is less than 0.02 greylevel, which is much less than the quantization error. The EWA filter is faster than our FOA filter but its speed is at the same order of magnitude as ours. MIPmap is by far the fastest, since it is a hardware implementation.

[2]In [5], the covariance matrix is $\mathbf{\Sigma} = \sigma\mathbf{I}$, where σ is chosen as 2 and $\mathbf{I}$ is the identity matrix.

Rendering 2D planes Figure 6 compares the results of filtering a real 604×784 image in a perspective view. Our FOA filter (Figure 6(d)) has very similar results with the NEI filter (Figure 6(c), the RMSE error is less than 0.15 greylevel) but at significantly faster speed. As can be seen, both our FOA filter and the NEI filter retain the high frequency components without resulting in aliasing. The cubic kernel (Figure 6(a)) results in a highly aliased image, while the EWA filter (Figure 6(b)) overly smoothes the image. Furthermore, when the image plane moves, for example, rotates, our FOA filter also has very similar results with the NEI filter, and both suffer less from temporal aliasing than other methods.

Rendering 3D meshes Figure 7 shows the results of rendering a texture-mapped 3D teapot. The isotropic filter (with MIPmap off, Figure 7(a)) produces serious aliasing. MIPmap (Figure 7(b)) produces a better result, but it is overly smooth in some areas or aliased in others. The EWA filter (Figure 7(c)) generates a much better result than Figures 7(a) and (b), but it is still a little blurred in some areas (for example, the area shown in the inset). Our FOA filter (Figure 7(d)) performs slightly better than the EWA filter.

5. Conclusion

In this paper, by using the signal processing and sampling theory, we derive the ideal weighting function for texture filtering. As there is no closed-form solution for the ideal weighting function, we further propose the first order approximating filter by approximating the texture transform with its first order Taylor expansion. Such an approximation makes the closed-form solution possible. Our FOA filter is very close to the ideal filter, and is a generalization of the EWA, a well-known filter. Our analysis and experiments both testify that our FOA filter produces better rendering results than traditional ones. In terms of rendering speed, our FOA filter is much faster than numerically evaluation of the ideal filter, and at the same magnitude as the EWA filter.

6. Appendix A: Inverse Fourier Transform over a Regular Right-Angled Triangle

Suppose the vertices of a regular right-angled triangle Δ are: $(\omega_{x0}, \omega_{y0})$, $(\omega_{x0}, \omega_{y1})$ and $(\omega_{x1}, \omega_{y0})$. Then the in-

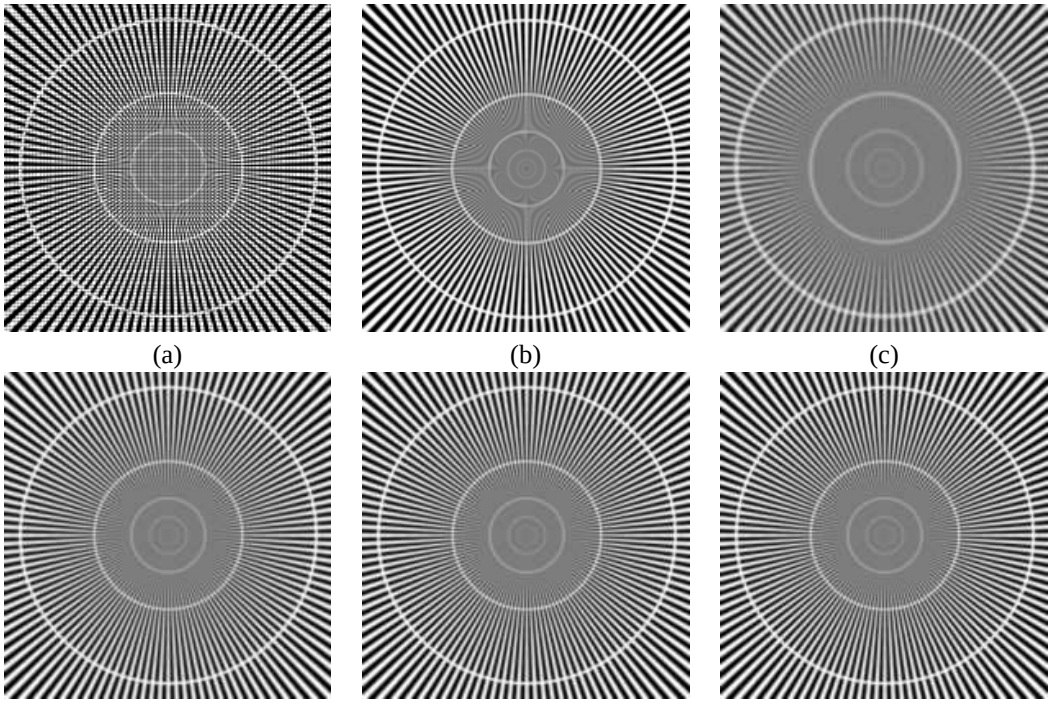


Figure 5. Comparison of downsampling results using different filters. (a)∼(f) are results of using box, cubic, MIPmap, EWA, numerically evaluated ideal (NEI), and first order approximating (FOA) filters, respectively.

verse Fourier transform over this triangle is:

$$\begin{aligned} & \mathcal{F}^{-1}[\mathrm{BOX}_{\Delta}(\omega_x,\omega_y)](x,y) \\ = \; & \frac{1}{4\pi^2}\mathrm{sgn}_{xy}\int_{\omega_{x1}}^{\omega_{x0}} e^{ix\omega_x}\mathrm{d}\omega_x \int_{A\omega_x+B}^{\omega_{y0}} e^{iy\omega_y}\mathrm{d}\omega_y \\ = \; & \frac{1}{4\pi^2}\mathrm{sgn}_{xy}\left\{\frac{e^{iy\omega_{y0}}}{xy}\left(e^{ix\omega_{x0}}-e^{ix\omega_{x1}}\right)\right. \\ & \left.-\frac{e^{iBy}}{y\,(x+Ay)}\left[e^{i(x+Ay)\omega_{x0}}-e^{i(x+Ay)\omega_{x1}}\right]\right\} \end{aligned}$$

where

$$\mathrm{sgn}_{xy} = \mathrm{sgn}(\omega_{x0}-\omega_{x1})\mathrm{sgn}(\omega_{y0}-\omega_{y1}),$$

$$A = -\frac{\omega_{y1}-\omega_{y0}}{\omega_{x1}-\omega_{x0}}, \text{ and } B = \frac{\omega_{x1}\omega_{y1}-\omega_{x0}\omega_{y0}}{\omega_{x1}-\omega_{x0}}.$$

References

[1] R. J. Cant and P. A. Shrubsole. Texture potential mip mapping, a new high-quality texture antialiasing algorithm. *ACM Trans. Graph.*, 19(3):164–184, 2000.

[2] B. Chen, F. Dachille, and A. Kaufman. Forward image warping. *IEEE Visualization*, pages 89–96, 1999.

[3] K. Deng, L. Wang, J. Zhang, and B. Guo. Texture mapping with a jacobian-based spatially-variant filter. In *Proceedings of Pacific Graphics 2002*, pages 460–461, 2002.

[4] N. Greene and P. Heckbert. Creating raster ominimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6):21–27, June 1986.

[5] P. Heckbert. Fundamentals of texture mapping and image warping. *Master's thesis, CS Division, UCB/CSD 89/516, U.C. Berkeley*, 1989.

[6] T. Hüttner and W. Straßer. Fast footprint mipmapping. *Proceeding of the 1999 EUROGRAPHICS/SIGGRAPH Workshop on Graphics Hardware*, pages 35–44, 1999.

[7] S. Kang and M. Jones. Appearance-based structure from motion using linear classes of 3-D models. *Int'l J. of Computer Vision*, 49(1):5–22, August 2002.

[8] J. McCormack, R. Perry, K. I. Farkas, and N. P. Jouppi. Feline: Fast elliptical lines for anisotropic texture mapping. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 243–250, 1999.

[9] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. Evaluation and design of filters using a taylor series expansion. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):184–199, 1997.

[10] D. Morris and T. Kanade. Image-consistant surface triangulation. *CVPR*, 1:332–338, June 2000.

[11] A. Schilling, G. Knittel, and W. Straer. Texram - a smart memory for texturing. *IEEE Computer Graphics and Applications 16(3)*, pages 32–41, 1996.

[12] J. Torborg and J. Kajiya. Talisman: Commodity realtime 3d graphics for the pc. *Computer Graphics proceeding, Annual Conf., ACM SIGGraph'96*, pages 353–363, 1996.

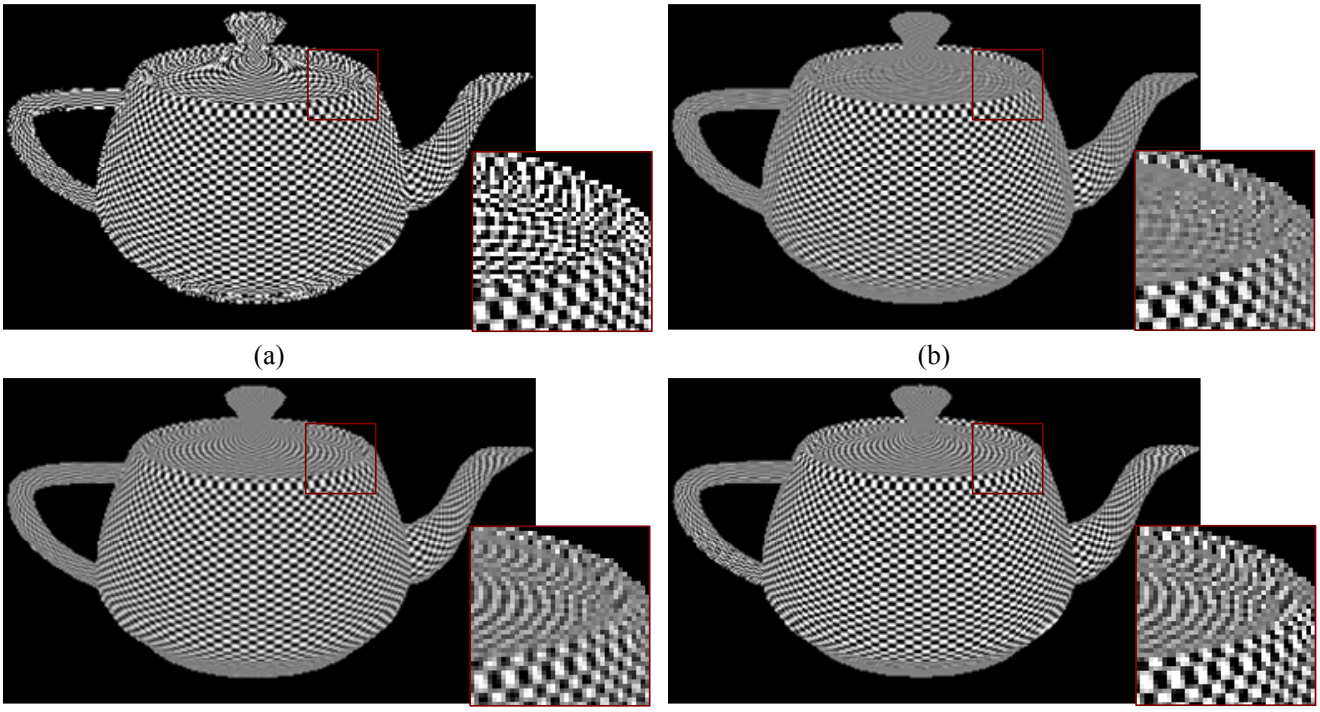
(a) (b) (c) (d)

Figure 6. Texture filtering on a perspective plane with a real picture. *(a) Using cubic kernel. (b) Using the EWA filter. (c) Using our NEI filter. (d) Using our FOA filter.*

[13] L. Wang, S. Kang, H.-Y. Shum, and R. Szeliski. Optimal texture map reconstruction from multiple views. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 347–354, 2001.

[14] L. Williams. Pyramidal parametrics. *Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques*, pages 1–11, 1983.

[15] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.



(a) (b)

(c) (d)

Figure 7. "Checker board" texture filtering on a 3D teapot. *(a) Using OpenGL with MIPmap off. (b) Using OpenGL with MIPmap on. (c) Using the EWA filter. (d) Using our FOA filter.*