High-quality multi-pass image resampling

Richard Szeliski, Simon Winder, and Matt Uyttendaele

February 2010

Technical Report

MSR-TR-2010-10

This paper develops a family of multi-pass image resampling algorithms that use one-dimensional filtering stages to achieve high-quality results at low computational cost. Our key insight is to perform a frequency-domain analysis to ensure that very little aliasing occurs at each stage in the multi-pass transform and to insert additional stages where necessary to ensure this. Using one-dimensional resampling enables the use of small resampling kernels, thus producing highly efficient algorithms. We compare our results with other state of the art software and hardware resampling algorithms.

Microsoft Research

Microsoft Corporation

One Microsoft Way

Redmond, WA 98052

http://www.research.microsoft.com

1 Introduction

Current texture mapping hardware normally uses MIP-mapping (Williams 1983), sometimes combined with multiple-sample anisotropic filtering (Barkans 1997). Unfortunately, these algorithms sometimes produce considerable aliasing in areas of high-frequency content. Better filters, such as Elliptical Weighted Average (EWA) filter (Greene and Heckbert 1986)), have been developed, but even these produce visible artifacts or excessive blurring when textures are animated. While the theory of high-quality image image filtering is well known (Mitchell and Netravali 1988, Heckbert 1989, Wolberg 1990), it is usually applied only to (separable) image rescaling, with sub-optimal texture-mapping algorithms being used in other cases.

Today's CPU multi-media accelerators and GPUs have more than enough power to support better resampling algorithms, especially for applications where visual quality is important, such as photo manipulation, animated slideshows, panoramic image and map viewing, and visual effects. What is missing are algorithms that can efficiently filter away high-frequency content that might cause aliasing while simultaneously preserving important texture details.

In this paper, we develop a family of high-quality multi-pass texture mapping algorithms, which use a series of one-dimensional filtering stages to achieve good efficiency while maintaining high visual fidelity. The key to our approach is to use Fourier analysis to ensure that none of the stages performs excessive blurring or aliasing, so that the resulting resampled signal contains as much high-frequency detail as possible while avoiding aliasing. Figures 4 and 8 show the basic idea. The image is first upsampled to prevent aliasing during subsequent shearing steps, and then downsampled to its final size with high-quality low-pass filtering. While this paper focuses on the case of affine transforms, the basic approach can be extended to full perspective, as discussed in Section 8.

2 Previous work

Image resampling (a.k.a. image warping or texture mapping) algorithms have been under active development for almost three decades, and several good surveys and textooks on these subjects

can be found (Heckbert 1989, Wolberg 1990, Dodgson 1992, Akenine-Möller and Haines 2002). These algorithms fall roughly into four distinct categories: image filtering, multi-pass transforms, pyramidal MIP-mapping, and elliptical weighted averaging.

Image filtering algorithms focus on optimizing the shape of interpolation and/or low-pass filters to minimize a number of competing visual artifacts, such as aliasing, ringing, and blurring. Mitchell and Netravali (1988) introduce a taxonomy for these artifacts, and also design a cubic reconstruction filter that heuristically optimizes some of these parameters. (Dodgson(1992) has a more extended discussion of visual criteria.) Monographs and surveys on image resampling and warping such as (Heckbert 1989, Wolberg 1990, Dodgson 1992, Akenine-Möller and Haines 2002) have nice tutorial sections on image filtering and reconstruction, as do classic image processing books (Oppenheim *et al.* 1999) and research papers in image processing and computer vision (Unser 1999, Triggs 2001). The first four books also cover the topic of geometric transforms which underlie affine (and more general) image warping.

Multi-pass or scanline algorithms use multiple one-dimensional image re-scaling and/or shearing passes, combined with filtering of varying quality, to implement image rotations and other affine or non-linear image transforms. Heckbert (1989) and Wolberg (1990) both have nice reviews of these algorithms, including the seminal two-pass transform developed by Catmull and Smith (1980). Unfortunately, none of these techniques use high-quality (multi-tap) image filters inside their one-dimensional resampling stages, nor do they account for aliasing in the orthogonal dimension during shearing (see Section 5). It is somewhat surprising that no one has so far merged high quality image filtering with scanline algorithms, which is what this paper aims to do.

MIP-mapping algorithms (Williams 1983) construct an image pyramid ahead of time, which make subsequent downsampling operations more efficient by avoiding the need for large low-pass filter kernels. In its usual form, tri-linear filtering is used. First, the two nearest pyramid levels are found, using a heuristic rule based on the local affine warp being performed (Ewins *et al.* 1998). The results of bi-linearly interpolating each of these two images is then linearly blended. Unfortunately, the bi-linear resampling introduces aliasing, while blending imagery from the coarser level introduces additional blur. Using (4×4) bi-cubic interpolation has been proposed, but according

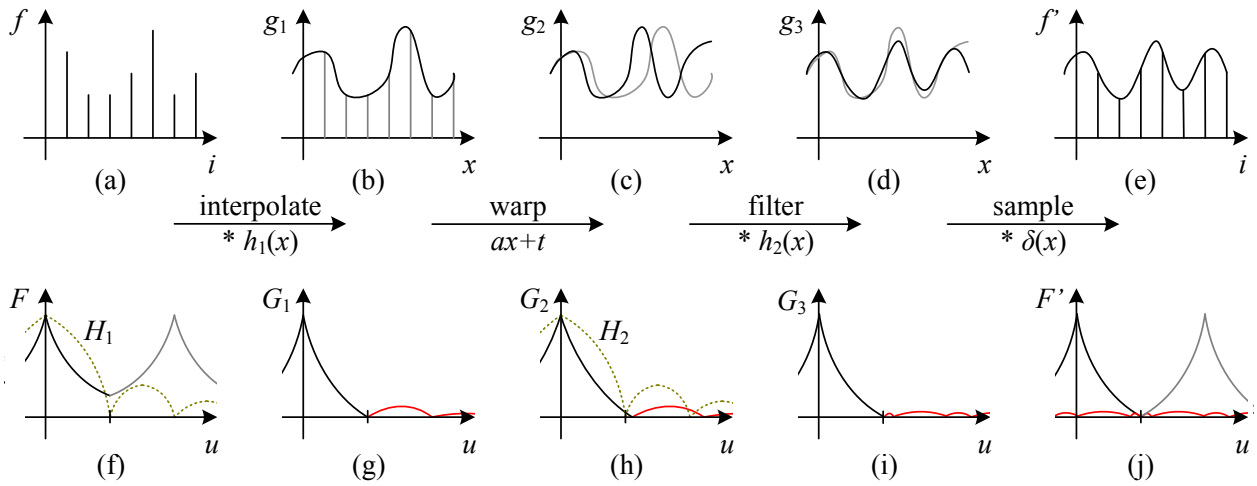


Figure 1: *One-dimensional signal resampling: (a) original sampled signal $f(i)$; (b) interpolated signal $g_1(x)$; (c) warped signal $g_2(x)$; (d) filtered signal $g_3(x)$; (e) sampled signal $f'(i)$. The corresponding spectra are shown below the signals in figures (f–j), with the aliased portions shown in red.*

.

to Akenine-Möller and Haines (2002), this option is not widely available in hardware.

The performance of MIP-mapping degrades even further when the resampling becomes anisotropic. Ripmapping (Akenine-Möller and Haines 2002) extends the idea of the pyramidal MIP-map by creating rectangular smaller images as well. While this requires a 300% memory overhead (as opposed to only 30% for MIP-maps), it produces better-quality results when images are zoomed anisotropically (changes in aspect ratios). For general skewed anisotropy, a variety of multi-sample anisotropic filters have been proposed (Schilling *et al.* 1996, Barkans 1997). While these offer a noticeable improvement over regular MIP-mapping in heavily foreshortened regions, they still suffer from the aliasing introduced by low-quality tri-linear filters (see our Experimental Results section).

Finally, Elliptic Weighted Average (EWA) filters convolve the image directly with a non-separable oriented (skewed) Gaussian filter (Greene and Heckbert 1986). While this has the reputation in some quarters of producing high quality results (Akenine-Möller and Haines 2002), Gaussian filtering is known to simultaneously produce both aliasing and blurring. Since the filter

is isotropic in the warped coordinates, it incorrectly filters out corner frequencies in the spectrum and, being non-separable, the naïve implementation of EWA is also quite slow, although faster algortihms based on MIP-mapping have recently been proposed (McCormack *et al.* 1999, Hüttner and Straßer 1999).

The remainder of the paper is structured as follows. Sections 3 and 4 review the basics of one-dimensional and (separable) two-dimensional image resampling. Section 5 presents our novel three-pass optimal resampling algorithm for performing one-dimensional image shears, while Section 6 builds on these results to develop an efficient four-pass general affine resampling algorithm. Section 7 uses a variety of test images and motions to compare our algorithm to previously developed state of the art resampling algortihms. We close with a discussion of future directions for research that this work suggests.

3 One-dimensional resampling

Before we describe our new algorithms, we first briefly review the theory of optimal one-dimensional signal resampling. We use the framework shown in Figure 1, which Heckbert (1989) calls *ideal resampling* and Dodgson (1992) calls the *four part decomposition* (and both attribute to (Smith 1981)).

The original source image (or *texture map*) is a sampled signal $f(i)$, as shown in Figure 1a. Because the signal is sampled, its Fourier transform is infinitely replicated along the frequency axis, as shown in Figure 1f.

To resample the signal, we first (conceptually) convert it into a continuous signal $g(x)$ by convolving it with an *interpolation filter*, $h_1(x)$,

$$g_1(x) = \sum_i f(i) h_1(x - i), \tag{1}$$

as shown in Figure 1b. In the frequency domain, this corresponds to multiplying the original signal spectrum $F(u) = \mathcal{F}\{f(x)\}$, with the spectrum of the interpolation filter $H_1(u)$ to obtain

$$G_1(u) = F(u) H_1(u), \tag{2}$$

as shown in Figure 1f–g.

If the filter is of insufficient quality, phantom replicas of the original spectrum persist in higher frequencies, as shown in red in Figure 1g. These replicas correspond to the *aliasing* introduced during the interpolation process, and are often visible as unpleasant discontinuities (*jaggies*) or motion artifacts (*crawl*).

Examples of interpolation filters include *linear interpolation*, *cubic interpolation* (Mitchell and Netravali 1988), and *windowed sinc interpolation* (Oppenheim *et al.* 1999). A complete discussion of the merits of various one-dimensional interpolation filters is beyond the scope of this paper, since they have been widely studied in the fields of signal processing (Oppenheim *et al.* 1999, Wolberg 1990, Dodgson 1992), image processing (Unser 1999, Triggs 2001) and graphics (Heckbert 1986). In this paper, we use a raised cosine-weighted sinc filter with 4 cycles (9 taps when interpolating).

The next step is to apply a spatial transformation to the original signal domain, e.g.,

$$x = ax' + t, \tag{3}$$

which is an *affine* spatial warp. (Other transformation, such as perspective or arbitrary warps are also possible (Heckbert 1989, Wolberg 1990).) Note how we always specify the *inverse warp*, i.e., the mapping from final pixel coordinates x' to original coordinates x.

The *warped* or *transformed* continuous signal and its Fourier transform (in the affine case) are

$$g_2(x') = g_1(ax' + t) \quad \Leftrightarrow \quad G_2(u) = \frac{1}{a} G_1(u/a) e^{jut/a}. \tag{4}$$

If the original signal is being compressed (Figure 1c), the Fourier transform becomes dilated (stretched) along the frequency axis (Figure 1h).

Before resampling the warped signal, we *pre-filter* (low-pass filter) it by convolving it with another kernel,

$$g_3(x) = g_2(x) * h_2(x) \quad \Leftrightarrow \quad G_3(u) = G_2(u) H_2(u) \tag{5}$$

(Figure 1d/i). This is particularly necessary if the signal is being *minified* or *decimated*, i.e., if $a > 1$ in (3). If this filtering is not performed carefully, some additional aliasing may be introduced into the final sampled signal (Figure 1j).

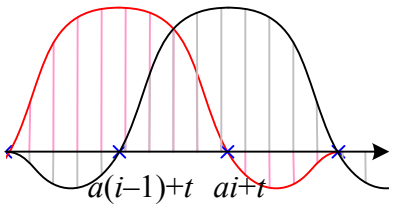


Figure 2: *Polyphase filtering. The coefficients used from* $h(x)$ *for the black sample point at* $ai + t$ *and the red sample point at* $a(i - 1) + t$ *are different, and would be stored in different* phases *of the two-dimensional polyphase lookup table* $h_{\mathrm{P}}(k; \phi)$.

Fortunately, because of the linearity of convolution operators, the three stages of filtering and warping can be combined into a single composite filter

$$H_3(u) = H_1(u/a)H_2(u), \tag{6}$$

which is often just a *scaled* version of the original interpolation filter $h_1(x)$.[1] The final *discrete* convolution can be written as

$$f'(i) = g_3(i) = \sum_j h_3(ai + t - j)f(j) = \sum_j h([ai + t - j]/s)f(j), \tag{7}$$

where $s = \max(1, |a|)$.

The filter in (7) is a *polyphase* filter, since the filter coefficients being multiplied with the input signal $f(j)$ are potentially different for every value of i (Figure 2). To see this, we can re-write (7) as

$$f'(i) = \sum_j h([ai + t - j]/s)f(j) = \sum_j h_{\mathrm{P}}(j^* - j; \phi)f(j), \tag{8}$$

where

$$j^* = \lfloor ai + t \rfloor, \tag{9}$$

$$\phi = ai + t - j^*, \text{ and} \tag{10}$$

$$h_{\mathrm{P}}(k; \phi) = h([k + \phi]/s). \tag{11}$$

[1] For *ideal* (sinc) reconstruction and low-pass filtering, the Fourier transform is a box filter of the smaller width, and hence the combined filter is itself a sinc (of larger width).

The values in $h(k;\phi)$ can be precomputed for a given value of s and stored in a two-dimensional look-up table.[2] The number of discrete fractional values of ϕ that need to be stored is related to the desired precision of the convolution, and is typically 2^b, where b is the number of bits of desired precision in the output (say 10-bits for 8-bit RGB images to avoid error accumulation in multi-pass transforms).

We can write the above formula (7) in a functional form

$$f' = \mathcal{R}(f; h, s, a, t). \tag{12}$$

In other words, $\mathcal{R}$ is an algorithm, parameterized by a continuous filter kernel $h(x)$, scale factors s and a, and a translation t, which takes an input signal $f(i)$ and produces an output signal $f'(i)$. This operator is generalized in the next section to a pair of horizontal and vertical scale/shear operators, which are the basic building blocks for all subsequent algorithms.

4 Two-dimensional zooming

In this section, we review two-pass *separable* transforms, which can be accomplished by first resampling the image horizontally and then resampling the resulting image vertically (or vice-versa). We can perform these operations by extending our one-dimensional resampling operator (12) to a pair of horizontal and vertical image resampling operators,

$$f' = \mathcal{R}_{\mathrm{h}}(f, h, s, a_0, a_1, t) \quad \Leftrightarrow \tag{13}$$

$$f'(i,j) = \sum_k h(s[a_0 i + a_1 j + t - k]) f(k,j) \quad \text{and}$$

$$f' = \mathcal{R}_{\mathrm{v}}(f, h, s, a_0, a_1, t) \quad \Leftrightarrow \tag{14}$$

$$f'(i,j) = \sum_k h(s[a_1 i + a_0 j + t - k]) f(i,k).$$

Note that these operators not only support directional scaling and translation, but also support shearing (using a different translation for each row or column), which is used for more complex transformations.

[2] The values of $h(k;\phi)$ should be re-normalized so that $\sum_k h(k;\phi) = 1$.

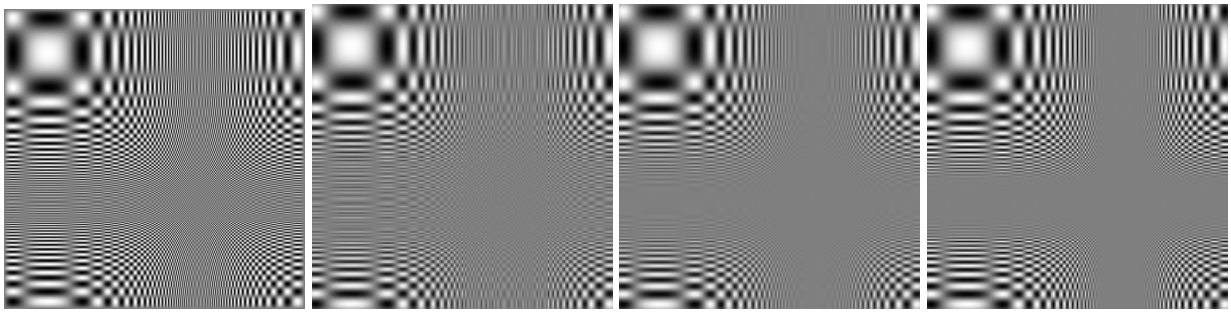


Figure 3: *Image magnification results: (a) original chirp image; (b) tri-linear MIP-mapping; (c) EWA filtering; (d) windowed sinc function. Notice how the other techniques produce either excessive blur or aliasing. (Please look at these results by magnifying your document viewer.)*

Image minification (zooming out) can be made more efficient using MIP-maps (Williams 1983) or ripmaps (Akenine-Möller and Haines 2002), as described in Section 2.

Figure 3 shows some examples of image magnification using tri-linear MIP-mapping, EWA filtering, and windowed sinc low-pass filtering. Note how the windowed sinc function produces the least aliasing and blur.

5 Shear

In order to better explain our general affine multi-pass resampling algorithm, we start with the simpler case of a pure horizontal shear,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \left[\begin{array}{cc|c} a_0 & a_1 & t \\ 0 & 1 & 0 \end{array}\right] \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{15}$$

This corresponds to a general invocation of $\mathcal{R}_{\mathrm{h}}$ with $a_1 \neq 0$.

In the frequency domain (ignoring the effect of the translation parameter t, since it does not affect aliasing), this corresponds to a transformation of

$$\boldsymbol{u}' = \boldsymbol{A}^T \boldsymbol{u} \quad \text{where} \quad \boldsymbol{A}^T = \begin{bmatrix} a_0 & 0 \\ a_1 & 1 \end{bmatrix}. \tag{16}$$

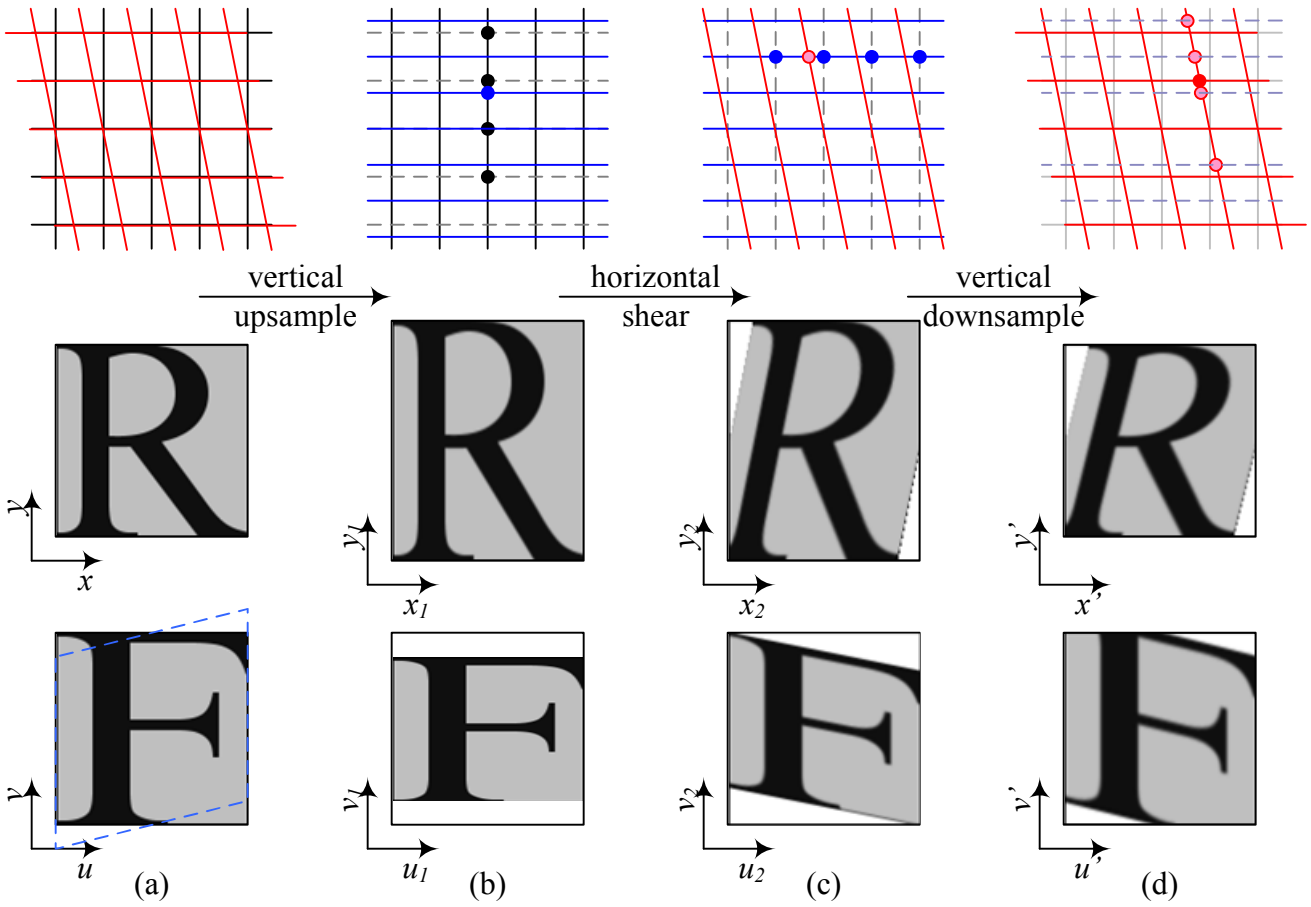


Figure 4: *Horizontal 3-pass shear: (a) original pixel grid, image, and its Fourier transform; (b) vertical upsampling onto the blue lines; (c) horizontal shear onto the diagonal red lines; (d) final vertical downsampling. The first row shows the sampling grids, the second row shows the images being resampled, and the third row shows their corresponding spectra. The frequency spectra in the third row are scaled to the unit square with maximum frequencies* $(\pm 1, \pm 1)$ *in order to make the upsampling and downsampling operations more intuitive.*

Thus, a horizontal shear in the spatial domain induces a vertical shear in the frequency domain, which can lead to aliasing if we do not first upsample the signal vertically. (Notice how the original frequency $(\pm 1, \pm 1)$ gets mapped to $(\pm a_0, \pm 1 \pm a_1)$, which can be beyond the vertical Nyquist frequency.)[3]

In order to avoid aliasing, we propose a three-pass algorithm, which consists of the following steps:

1. upsample vertically by the factor $r \geq 1 + |a_1|$;
2. shear and scale horizontally, with filtering to avoid aliasing;
3. low-pass filter and downsample vertically.

In terms of geometric transformation, this corresponds to factoring

$$\boldsymbol{A} = \begin{bmatrix} a_0 & a_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1/r \end{bmatrix} \begin{bmatrix} a_0 & a_1/r \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & r \end{bmatrix} = \boldsymbol{A}_1 \boldsymbol{A}_2 \boldsymbol{A}_3, \tag{17}$$

and applying the sequence of transformations

$$\boldsymbol{x} = \boldsymbol{A}_1 \boldsymbol{x}_1, \quad \boldsymbol{x}_1 = \boldsymbol{A}_2 \boldsymbol{x}_2, \quad \boldsymbol{x}_2 = \boldsymbol{A}_3 \boldsymbol{x}', \tag{18}$$

as shown in Figure 4. The transpose of the middle matrix $\boldsymbol{A}_2$ is

$$\boldsymbol{A}_2^T = \begin{bmatrix} a_0 & 0 \\ a_1/r & 1 \end{bmatrix}, \tag{19}$$

which, when multiplied by the maximum frequency present in the upsampled signal, $(\pm 1, \pm 1/r)$ still lies inside the Nyquist range $u' \in [-1, +1]^2$ (after horizontal filtering, which is applied during the scale/shear).

In our operational notation, this can be written as

$$f_1 = \mathcal{R}_{\mathrm{v}}(f, h, 1, 1/r, 0, 0); \tag{20}$$

$$f_2 = \mathcal{R}_{\mathrm{h}}(f_1, h, \max(1, |a_0|), a0, a_1/r, t); \tag{21}$$

$$f_3 = \mathcal{R}_{\mathrm{v}}(f_2, h, r, r, 0, 0). \tag{22}$$

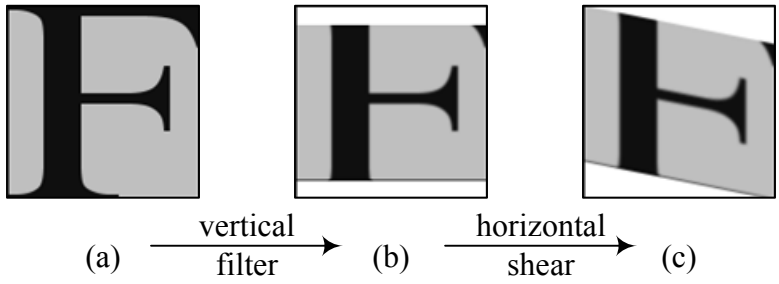


Figure 5: *Horizontal 2-pass shear: (a) original Fourier transform; (b) vertical low-pass filtering; (c) horizontal shear. Notice how some high vertical frequencies are lost.*

An alternative to this three-step process is a sub-optimal two-pass algorithm:

1. vertically low-pass filter the image with a bandwidth $1/r$;

2. shear, scale, (and pre-filter) the image horizontally.

As we can see in Figure 5c, this results in a loss of some high-frequency information, compared to Figure 4d. (See the Experiments section for some examples.)

Unfortunately, the above derivation suggests that the upsampling rate can get arbitrarily high as $|a_1| \gg 1$. In fact, the maximum upsampling rate need never exceed $r = 3$ (Figure 6). This is because the pink portions of the spectrum along the left and right edge (high horizontal frequencies) do not appear in the final image, and can therefore be filtered away during the horizontal shear.

To compute a better value for r, we first compute the maximum values of the original frequencies that will appear in the final image, $(u_{\max}, v_{\max})$, as shown in Figure 6e. The value of $v_{\max}$ can be less than 1 if we are considering the general form of a shear matrix with vertical scaling included,

$$\boldsymbol{A} = \begin{bmatrix} a_{00} & a_{01} \\ 0 & a_{11} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & a_{11}/r \end{bmatrix} \begin{bmatrix} a_{00} & a_{01}/r \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & r \end{bmatrix}, \tag{23}$$

where we have combined the vertical scaling with the initial vertical resampling stage. To avoid

[3] From here on we use the convention that the frequencies range over $[-1, +1]$, since this simplifies our notation.

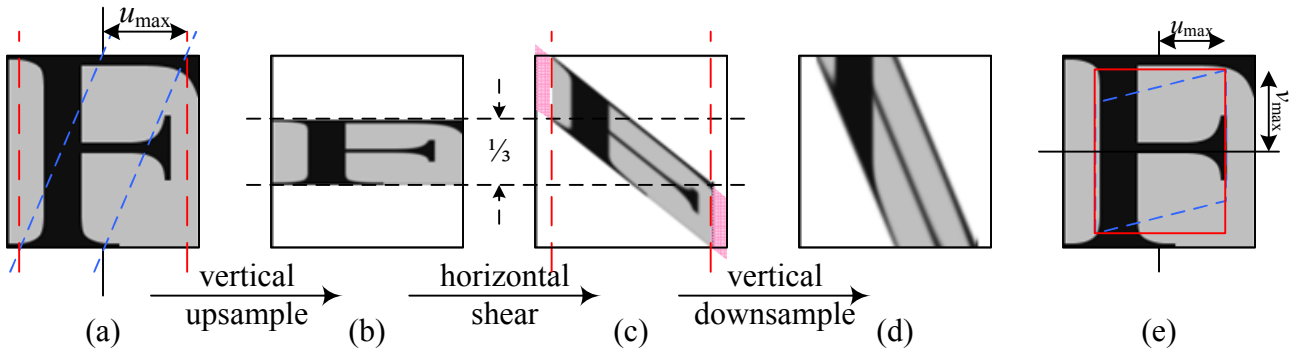


Figure 6: *Maximum vertical resampling rate: (a) original Fourier transform; (b) vertical upsampling by a factor of* 3*; (c) horizontal shear and low-pass filtering horizontally; (d) final vertical downsampling; (e) general case for computing* $u_{\max}$ *and* $v_{\max}$*. Because horizontal frequencies start being suppressed (moved to beyond the Nyquist frequency), it is not necessary to upsample by more than a factor of 3.*

aliasing, we must then ensure that

$$\boldsymbol{A}_2^T \begin{bmatrix} \pm u_{\max} \\ \pm a_{11} v_{\max}/r \end{bmatrix} = \begin{bmatrix} \pm a_{00} u_{\max} \\ \pm a_{01} u_{\max}/r \pm a_{11} v_{\max}/r \end{bmatrix} \tag{24}$$

lies within the bounding box $[-1,+1]^2$, i.e., that $|a_{01}|u_{\max}/r + |a_{11}|v_{\max}/r \leq 1$ or $r \geq |a_{01}|u_{\max} + |a_{11}|v_{\max}$. Whenever $|a_{11}|v_{\max} > 1$, we can clamp this value to 1, since there is no need to further upsample the signal. When the vertical scaling a_{11} is sufficiently small (magnification), $r < 1$. Since there is no risk of aliasing during the horizontal shear, we set $r = 1$ and drop the final vertical downsampling stage. The formula for r thus becomes

$$r \geq \max(1, |a_{01}|u_{\max} + \min(1, |a_{11}|v_{\max})). \tag{25}$$

The final three (or two) stage resampling algorithm is therefore:

$$f_1 = \mathcal{R}_{\mathrm{v}}(f, h, 1/v_{\max}, a_{11}/r, 0, 0); \tag{26}$$

$$f_2 = \mathcal{R}_{\mathrm{h}}(f_1, h, \max(1, |a_{00}|), a_{00}, a_{01}/r, t); \tag{27}$$

$$f_3 = \mathcal{R}_{\mathrm{v}}(f_2, h, r, r, 0, 0), \tag{28}$$

where the last stage is skipped if $r = 1$.

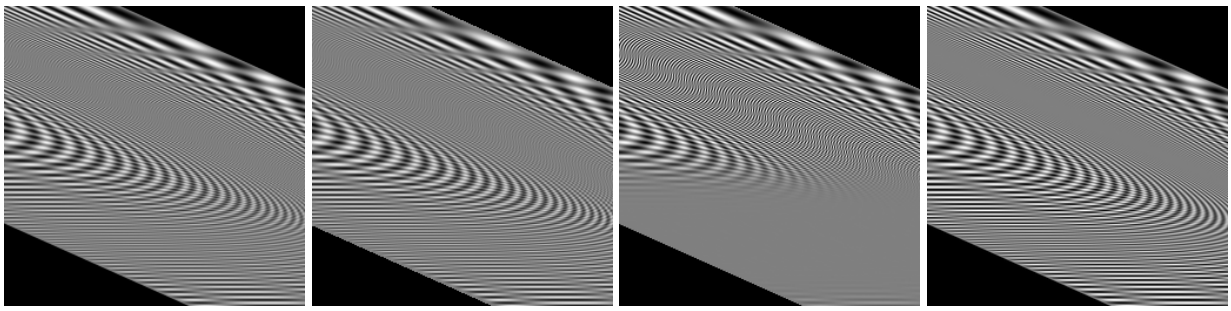


Figure 7: *Shearing algorithm results: (a) tri-linear MIP-mapping with anisotropic filtering; (b) EWA filtering, (c) sub-optimal vertical blur only algorithm; (d) optimal three-stage algorithm. Please see our on-line web page http://research.microsoft.com/en-us/um/redmond/groups/ivm/HQMPIR/ for more results as well as animations that better show the aliasing artifacts.*

Figure 7 shows some examples of shears rendered using our optimal three-pass algorithm, the sub-optimal (blurred) 2-pass shear algorithm, as well as a variety of previously developed resampling algorithms.

6 General affine

It is well known that any 2D affine transform can be decomposed into two shear operations (Heckbert 1989, Wolberg 1990). For example, if we perform the horizontal shear first, we have

$$\boldsymbol{A} = \left[\begin{array}{cc|c} a_{00} & a_{01} & t_0 \\ a_{10} & a_{11} & t_1 \\ 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{cc|c} b_0 & b_1 & t_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \left[\begin{array}{cc|c} 1 & 0 & 0 \\ a_{10} & a_{11} & t_1 \\ 0 & 0 & 1 \end{array}\right], \tag{29}$$

with

$$b_0 = a_{00} - a_{01}a_{10}/a_{11}, \quad b_1 = a_{01}/a_{11}, \quad \text{and} \quad t_2 = t_0 - a_{01}t_1/a_{11}. \tag{30}$$

Notice that the above algorithm becomes degenerate as $a_{11} \to 0$, which is a symptom of the *bottleneck problem* (Wolberg 1990). Fortunately, we can transpose the input (or output) image and adjust the transform matrix accordingly.

To determine whether to transpose the image, we first re-scale the first two rows of $\boldsymbol{A}$ into unit vectors,

$$\hat{\boldsymbol{A}} = \begin{bmatrix} \hat{a}_{00} & \hat{a}_{01} \\ \hat{a}_{10} & \hat{a}_{11} \end{bmatrix} = \begin{bmatrix} a_{00}/l_0 & a_{01}/l_0 \\ a_{10}/l_1 & a_{11}/l_1 \end{bmatrix}, \tag{31}$$

where $l_i = \sqrt{a_{i0}^2 + a_{i1}^2}$. We then compute the absolute cosines of these vectors with the x and y axes, $|\hat{a}_{00}|$ and $|\hat{a}_{11}|$, and compare these to the absolute cosines with the transposed axes, i.e., $|\hat{a}_{01}|$ and $|\hat{a}_{10}|$. Whenever $|\hat{a}_{00}| + |\hat{a}_{11}| < |\hat{a}_{01}| + |\hat{a}_{10}|$, we transpose the image.

Having developed a three-pass transform for each of the two shears, we could concatenate these to obtain a six-pass separable general affine transform. However, it turns out that we can collapse some of the shears and subsequent upsampling or downsampling operations to obtain a four-pass transform, as shown in Figure 8.

The trick is to perform the horizontal upsampling needed for later vertical shearing at the same time as the original horizontal shear. In a similar vein, the vertical downsampling can be performed in the same pass as the vertical shear and scale.

In terms of geometric transformations, this corresponds to a factorization of the form

$$\begin{aligned} \boldsymbol{A} &= \left[\begin{array}{cc} 1 & 0 \\ 0 & a_{11}/r_{\mathrm{v}} \end{array}\right] \left[\begin{array}{cc|c} b_0 & a_{01}/r_{\mathrm{v}} & t_2 \\ 0 & 1 & 0 \end{array}\right] \left[\begin{array}{cc} 1/r_{\mathrm{h}} & 0 \\ 0 & 1 \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & r_{\mathrm{v}} \end{array}\right] \left[\begin{array}{cc|c} 1 & 0 & 0 \\ a_{10}/(a_{11}r_{\mathrm{h}}) & 1 & t_1/a_{11} \end{array}\right] \left[\begin{array}{cc} r_{\mathrm{h}} & 0 \\ 0 & 1 \end{array}\right] \\ &= \left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & a_{11}/r_{\mathrm{v}} & 0 \\ 0 & 0 & 1 \end{array}\right] \left[\begin{array}{cc|c} b_0/r_{\mathrm{h}} & a_{01}/r_{\mathrm{v}} & t_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \\ &\quad \left[\begin{array}{cc|c} 1 & 0 & 0 \\ a_{10}r_{\mathrm{v}}/(a_{11}r_{\mathrm{h}}) & r_{\mathrm{v}} & t_1 r_{\mathrm{v}}/a_{11} \\ 0 & 0 & 1 \end{array}\right] \left[\begin{array}{cc|c} r_{\mathrm{h}} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]. \end{aligned} \tag{32}$$

In order to compute the appropriate values for r_{v} and r_{h}, we must first determine which frequency in the original image needs to be preserved in the final image, as shown in Figure 6e. Frequencies that get mapped completely outside the final spectrum can be pre-filtered away during the upsampling stages, thereby reducing the total number of samples generated. We compute the

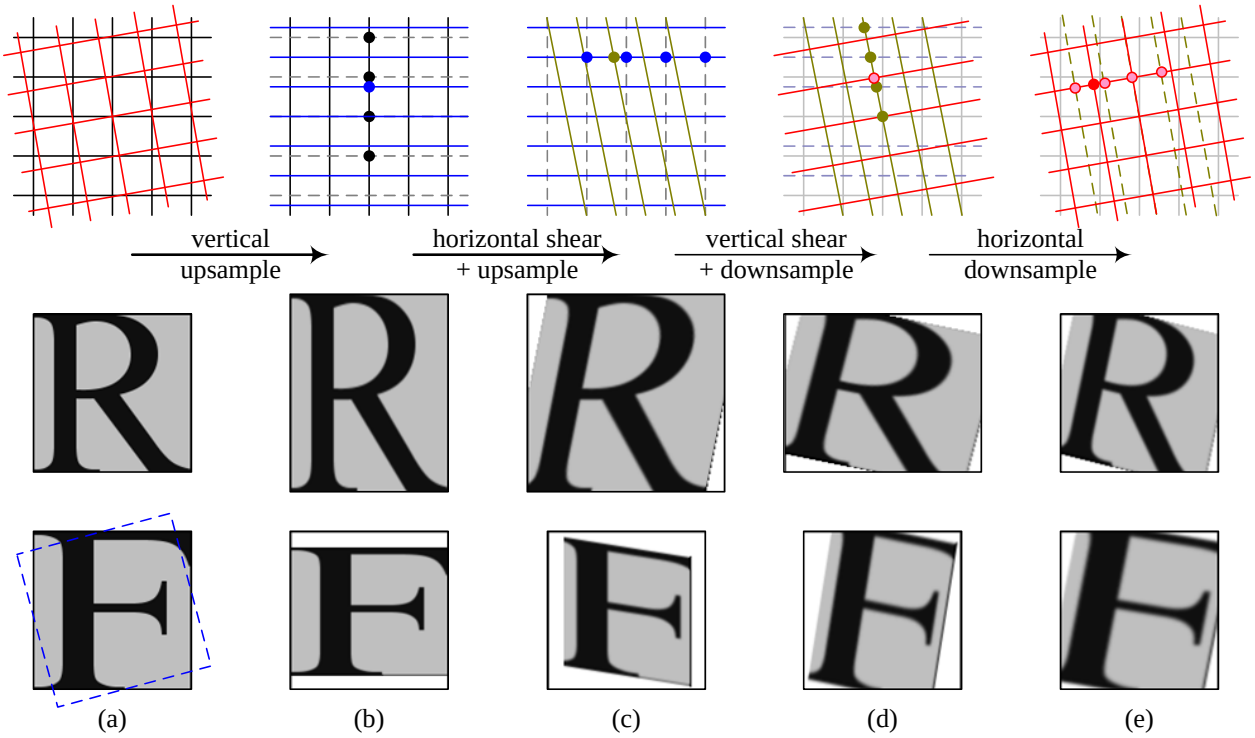
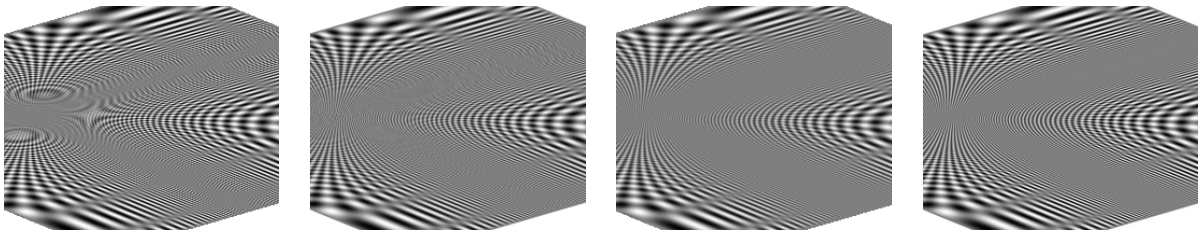


Figure 8: *4-pass rotation: (a) original pixel grid, image, and its Fourier transform; (b) vertical upsampling; (c) horizontal shear and upsampling; (d) vertical shear and downsampling; (e) horizontal downsampling. The general affine case looks similar except that the first two stages perform general resampling.*



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

Figure 9: *Two-dimensional chirp pattern affinely resampled using: (a) high-quality bicubic; (b) trilinear MIP-map with anisotropic filtering; (c) EWA filter; (d) high quality four-pass rendering (this paper). Please zoom in on the images to see more details.*

values of $(u_{\max}, v_{\max})$ by intersecting the $[-1,+1]^2$ square with the projection of the final spectrum onto the original spectrum through $\boldsymbol{A}^{-T}$ (the dashed blue lines in Figureřeffig:rotation).

Once we know the $(u_{\max}, v_{\max})$ extents, we can compute the upsampling rates using

$$r_{\mathrm{v}} \geq \max(1, |a_{01}|u_{\max} + \min(1, |a_{11}|v_{\max})) \quad \text{and} \tag{33}$$

$$r_{\mathrm{h}} \geq \max(1, |a_{10}/a_{11}|r_{\mathrm{v}}v_{\max} + \min(1, |b_0|u_{\max})). \tag{34}$$

The final four-step algorithm therefore consists of:

$$f_1 = \mathcal{R}_{\mathrm{v}}(f, h, 1/v_{\max}, a_{11}/r_{\mathrm{v}}, 0, t_1); \tag{35}$$

$$f_2 = \mathcal{R}_{\mathrm{h}}(f_1, h, 1/u_{\max}, b_0/r_{\mathrm{h}}, a_{01}/r_v, t_2); \tag{36}$$

$$f_3 = \mathcal{R}_{\mathrm{h}}(f_2, h, r_{\mathrm{v}}, r_{\mathrm{v}}, a_{10}r_{\mathrm{v}}/(a_{11}r_{\mathrm{h}}), 0); \tag{37}$$

$$f_4 = \mathcal{R}_{\mathrm{v}}(f_3, h, r_{\mathrm{h}}, r_{\mathrm{h}}, 0, 0). \tag{38}$$

If mip-maps or ripmaps are being used (Section 4), the amount of initial downsampling can be reduced by finding the smallest pyramid level that contains the frequency content needed to reconstruct the warped signal and adjusting the matrix entries in $\boldsymbol{A}$ appropriately.

7 Experimental results

We have evaluated our algorithm on both synthetic and natural images. In our on-line web page http://research.microsoft.com/en-us/um/redmond/groups/ivm/HQMPIR/, results are shown for pure

zoom (Figure 3), the three pass 1d shear (Figure 4), and two four pass cases - simultaneous rotation plus zoom and tilted orthographic rotation (Figure 9). In each case we also show results for the highest quality Windows GDI+ warp, EWA, and the NVidia 7800 GPU performing tri-linear anisotropic texture filtering. Filtering artifacts are sometimes more apparent under animation, so we provide animated examples of each of these transforms.

The synthetic image we use for most of our evaluation is a 2D chirp (Figure 9), since it contains a broad spectrum of frequencies up to the Nyquist rate in both u and v. Using this pattern, excessive low-pass filtering appears as a dimming of the chirp and aliasing appears as a Moiré pattern. The equation we used to generate the chirp is:

$$f(x, y) = \cos(2\pi k x 2^x) \cdot \cos(2\pi k y 2^y) \tag{39}$$

In our evaluation we show the transformed images as well as the result of applying the transform and then applying the inverse transform to warp back to the original pixel grid. For animated results, this round-trip transform is useful because the eye is not distracted by the motion of the image and can focus on the blur and aliasing artifacts (Dodgson 1992).

We have also evaluated the results on the natural images shown in Figure 10. For these, results we refer users to the supplementary web page, where the ability to compare images and the animations show the benefits of our algorithm. The greatest aliasing effects can be seen for diagonal frequencies in the GDI+ and GPU anisotropic filters, and this can be seen as Moiré patterns in areas of fine detail. The balance between aliasing and blurring can be set subjectively for EWA by altering the α parameter, but one or both are always present. When viewing our results, notice that our algorithm has far less aliasing, evident by the lack of Moiré in both the chirp results and the picket fence of the lighthouse image. Other algorithms also display more aliasing in the wheel areas of the bikes image. Fortunately, the lack of aliasing in our algorithm does not come at the expense of sharpness, since we maintain high frequency details where other techniques have blurred them out.

Figure 10: *(a) Lighthouse test image; (b) Bikes test image.*

8 Extensions and future work

While this paper has developed the basic theory for multi-pass non-aliasing affine transforms, it also suggests a number of promising directions for future research. These include optimizing the order of shears and scalings, rendering triangle meshes, memory optimizations, and full perspective warping.

Order of shears In this paper, we have implemented the general affine warp as a horizontal shear followed by a vertical shear (with additional transpose and/or resampling stages, as needed). In theory, the order of the shears should not matter if perfect filtering is used. In practice, non-ideal filters may induce a preferred ordering, as might computational considerations. For example, we may want to defer large amounts of magnification until later stages in the pipeline.

Filter selection and optimization In our current implementation, we have used a 4 cycle (9-tap) windowed sinc filter, since it provides a reasonable tradeoff between quality and efficiency. Using more taps (up to 6 or so) results in slight reductions in aliasing, while using fewer results in significant degradation. It would be worthwhile to investigate alternative filters, especially those

designed to reduce ringing while preserving high frequency content. Dodgson (1992) contains a nice discussion of *non-linear* filters that might be appropriate.

Triangle mesh rendering In this paper, we haven't said anything about the sizes of the intermediate images required during the intermediate stages. If we generalize the final image to a triangle or polygon, we can warp its shape back through successive stages and also add required additional pixels around the boundaries to ensure that each filter has sufficient support. Only the pixels inside each support region would then have to be resampled.

A more interesting question is whether triangle meshes could be rendered by re-using warped and filtered samples from adjacent triangles. This is a subtle issue, since if there are large discontinuities in the local affine transforms across triangle edges, visible errors might be induced.

Tiling and pipelining While GPUs are relatively insensitive to the order in which texture memory pixels are fetched (so long as pixels are re-used several times during computation), the same is not true for regular CPU memory. Optimizing memory accesses by breaking up the image into smaller 2D tiles and potentially pipelining the computation may results in significant speedups.

Perspective The family of multipass scanline algorithms such as (Catmull and Smith 1980) on which our work is based includes transforms such as perspective. We have not yet fully developed the theory of optimal multi-pass perspective algorithms because achieving full computational efficiency is tricky.

Perspective resampling is usually implemented by locally computing an affine approximation to the full transform and using its parameters to control the amount of filtering (Greene and Heckbert 1986, Wolberg 1990, McCormack *et al.* 1999). We could easily take Catmull and Smith's original 2-pass perspective transform and replace each stage with an optimal per-pixel polyphase filter. (The filter bandwidth would vary spatially.) The difficulty lies in computing the amount of upsampling that needs to be applied before each one-dimensional (perspective) shearing stage. Since this quantity is non-linear in the affine parameters because of the absolute value, there is no rational linear formula that would locally determine the amount of upsampling required, which

leads to a more complex algorithm. We could always just upsample each image/stage by the theoretical maximum of $r = 3$; instead we leave the development of the full perspective case to future work.

9 Conclusions

In this paper, we have developed a 4-stage scanline algorithm for affine image warping and resampling. Our algorithm uses optimal one-dimensional filtering at each stage to ensure that the image is neither excessively blurred nor aliased, which is not the case for previously developed algorithms. Because each stage only uses one-dimensional filters, the overall computation efficiency is very good, being amenable to GPU implementation using pixel shaders. While our algorithm may not be suitable for some applications such as high polygon complexity scenes, we believe that it forms the basis of a new family of higher quality resampling and texture mapping algorithms with wide applicability to scenes that require high visual fidelity.

References

Akenine-Möller, T. and Haines, E. (2002). *Real-Time Rendering*. A K Peters, Wellesley, Massachusetts, second edition.

Barkans, A. C. (1997). High quality rendering using the Talisman architecture. In *Proceedings of the Eurographics Workshop on Graphics Hardware*.

Betrisey, C. *et al.*. (2000). Displaced filtering for patterned displays. In *Society for Information Display Symposium,*, pages 296–299.

Catmull, E. and Smith, A. R. (1980). 3-d transformations of images in scanline order. *Computer Graphics (SIGGRAPH'80)*, *14(3)*, 279–285.

Dodgson, N. A. (1992). *Image Resampling*. Technical Report TR261, Wolfson College and Computer Laboratory, University of Cambridge.

Ewins, J. *et al.*. (1998). Mip-map level selection for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, *4(4)*, 317–329.

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

Heckbert, P. (1986). Survey of texture mapping. *IEEE Computer Graphics and Applications*, *6(11)*, 56–67.

Heckbert, P. (1989). *Fundamentals of Texture Mapping and Image Warping*. Master's thesis, The University of California at Berkeley.

Hüttner, T. and Straßer, W. (1999). Fast footprint MIPmapping. In *1999 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, pages 35–44.

McCormack, J., Perry, R., Farkas, K. I., and Jouppi, N. P. (1999). Feline: Fast elliptical lines for anisotropic texture mapping. In *Proceedings of SIGGRAPH 99*, pages 243–250.

Mitchell, D. P. and Netravali, A. N. (1988). Reconstruction filters in computer graphics. *Computer Graphics (Proceedings of SIGGRAPH 88)*, *22(4)*, 221–228.

Oppenheim, A. V., Schafer, R. W., and Buck, J. R. (1999). *Discrete-Time Signal Processing*. Prentice Hall, Englewood Cliffs, New Jersey, 2nd edition.

Schilling, A., Knittel, G., and Straßer, W. (1996). Texram: A smart memory for texturing. *IEEE Computer Graphics & Applications*, *16(3)*, 32–41.

Smith, A. R. (1981). *Digital Filtering Tutorial for Computer Graphics*. Technical Memo 27, Computer Graphics Project, Lucasfilm Ltd. Revised Mar 1983, available on http://alvyray.com/Memos/MemosPixar.htm.

Triggs, B. (2001). Empirical filter estimation for subpixel interpolation and matching. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pages 550–557, Vancouver, Canada.

Unser, M. (1999). Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, *16(6)*, 22–38.

Williams, L. (1983). Pyramidal parametrics. *Computer Graphics*, *17(3)*, 1–11.

Wolberg, G. (1990). *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos.