Low-cost Subpixel Rendering for Diverse Displays

Thomas Engelhardt[1], Thorsten-Walther Schmidt[†1], Jan Kautz[2], and Carsten Dachsbacher[‡1]

[1]Karlsruhe Institute of Technology [2]University College London

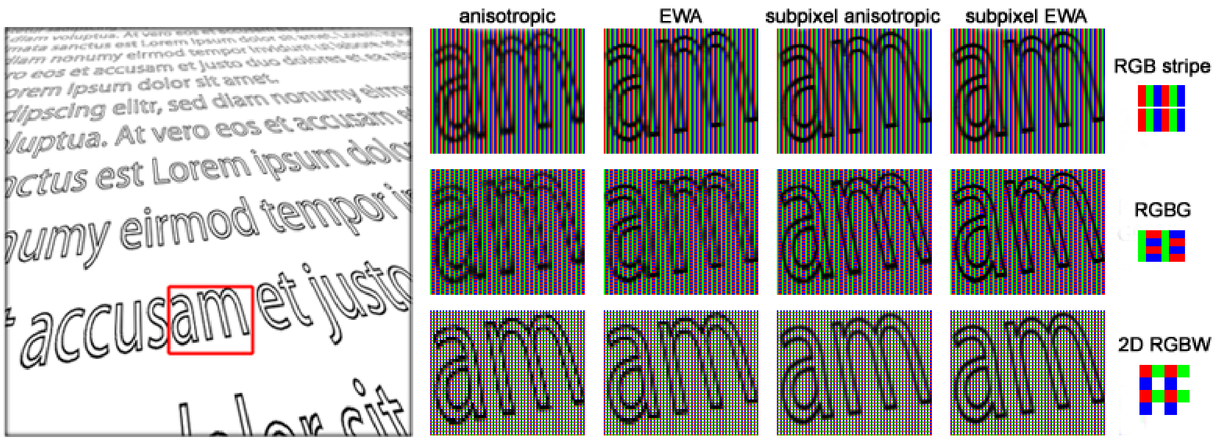

Figure 1: *Subpixel rendering of a plane with a structured texture for different subpixel layouts, similar to those found in diverse displays nowadays. This image showcases high-quality texture filtering, one of the applications of our proposed optimal filtering framework, comparing standard filtering (anisotropic and EWA) to our subpixel-aware filtering (again, using anisotropic and EWA). Note how our method significantly reduces aliasing. Please see the additional material for more results.*

Abstract

Subpixel rendering increases the apparent display resolution by taking into account the subpixel structure of a given display. In essence, each subpixel is addressed individually, allowing the underlying signal to be sampled more densely. Unfortunately, naïve subpixel sampling introduces color aliasing, as each subpixel only displays a specific color (usually R, G, and B subpixels are used). As previous work has shown, chromatic aliasing can be reduced significantly by taking the sensitivity of the human visual system into account. In this work, we find optimal filters for subpixel rendering for a diverse set of 1D and 2D subpixel layout patterns. We demonstrate that these optimal filters can be approximated well with analytical functions. We incorporate our filters into GPU-based multisample antialiasing to yield subpixel rendering at a very low cost (1-2ms filtering time at HD resolution). We also show that texture filtering can be adapted to perform efficient subpixel rendering. Finally, we analyze the findings of a user study we performed, which underpins the increased visual fidelity that can be achieved for diverse display layouts, by using our optimal filters.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Antialiasing, Bitmap and Framebuffer Operations

1. Introduction

Matrix display devices, such as cathode ray tubes, LCD monitors, and projectors, display color images with pixels consisting of three or four subpixels representing the primary colors. Besides dynamic range and gamut, the display resolution is a physical limitation restricting the reproducibility of real-world images. Recent work bypasses some of these limitations by exploiting the characteristics of the human visual system (HVS), e.g. by considering contrast sensitivity to enhance the perceived contrast, or accounting

† thorsten.schmidt@kit.edu
‡ dachsbacher@kit.edu

for retinal integration over time to increase the apparent resolution [DER*10].

In this work we focus on better utilizing the display resolution: normally, every pixel is assigned a color and the subpixels' brightness is adjusted such that their joint emission creates the respective stimulus. By treating each subpixel as an individual entity and taking their relative position into account, there is an opportunity to increase the perceived resolution of the display. Microsoft's ClearType functionality [Pla00, PKH*00] is a well-known example of subpixel-aware rendering that is targeted at font rasterization. Obviously, a naïve sampling of the image signal for each subpixel, i.e. ignoring the subpixels' respective colors, yields distracting color fringing artifacts. The reason why subpixel rendering can work is that the contrast sensitivity of human vision is different for luminance and chrominance. This allows one to derive an image filter that provides higher perceived resolution, while suppressing color fringing to an imperceptible degree, as shown by Platt [Pla00] and Klompenhouwer et al. [KdH03].

Many displays found in current commodity hardware provide high resolution where the benefit of subpixel rendering is mostly noticeable for high-contrast parts in images, such as black text on white background. However, there are also many cases where the display resolution (given in pixels-per-inch, PPI) is significantly lower and subpixel rendering is also beneficial for color images, e.g. when using large screens or projections for nearby observers, or simply because of the need for low display production cost.

In this paper, we make the following contributions:

- We generalize the approach of Platt [Pla00] to obtain filters for displays with diverse 1D and 2D subpixel layouts, including displays that provide a fourth primary color to improve luminance efficiency [ECH05].
- We derive analytical filters which are both easier to evaluate in practice and more suitable for image filtering.
- We show how to use our filters together with GPU-based multisample antialiasing to yield subpixel rendering at little additional cost compared to standard antialiasing.
- We demonstrate how texture sampling can be adapted to directly employ subpixel-aware rendering.
- We conducted user studies simulating different subpixel layouts and filters, confirming the optimality of our proposed filters.

1.1. Previous Work

Sampling and reconstruction of image signals is an integral part of computer graphics and image processing and has been widely studied for different tasks such as text rendering [KU81], polygon rasterization [Duf89] or image filtering [MN88]. Commonly, one color is determined for every pixel, ignoring the relative locations of the subpixels that emit the primary colors spanning the gamut of the display. Taking into account their arrangement allows one to increase the resolution of the luminance signal of the display, but care has to be taken not to produce chromaticity errors. Platt [Pla00] and Platt et al. [PKH*00] derive subpixel-aware filters for luminance and chrominance signals, assuming an RGB Stripe layout, by converting an RGB image into an opponent color space and using perceptual metrics. Messing and Daly's method [MD02] operates in a similar manner, but does not derive optimal filters. Messing et al. [MKD03] adapt Platt et al.'s work, proposing to use constrained optimization to solve for filters than can mask out defective subpixels; they demonstrate how their framework can be set up for one specific 2D subpixel arrangement. Later work by the same authors [MK06] extends this to arbitrary subpixel patterns. However, as in Platt et al.'s work, the resulting filters are given in discretized form, i.e., for a specific display resolution, while our optimal filters are analytical and can thus be employed to efficiently downsample input signals of arbitrarily high resolution. and do not need explicit storage of filter kernels, which makes them more GPU friendly. Furthermore, since our filters do not attempt to compensate for defective display hardware, we decided to directly build our method on the simpler approach by Platt et al.

The most widespread application of subpixel rendering is Microsoft's ClearType [Pla00, PKH*00], which is targeted at font rendering. It has even been evaluated perceptually using S-CIELAB [XFMW08]. Klompenhouwer et al. [KdH03] describe a subpixel-aware scaling of images and conclude that considering subpixel arrangements is a crucial part of the signal processing chain. Fang et al. [FAY*09] address subpixel-aware image downscaling (fixed scale) using edge detection and empirically determined low-pass filters for suppressing chromaticity errors. Subpixel rendering is also closely related to image sensors: most cameras use a sensor overlaid with a Bayer filter to selectively sample different wavelengths at interleaved locations, and the full RGB image is reconstructed by interpolating the missing data. Atcheson [Atc05] describes a subpixel-aware reconstruction for directly displaying images from Bayer patterns.

Recently, Didyk et al. [DER*10, TDR*11] describe an interesting approach to increasing the apparent display resolution for moving images and animations by accounting for the integration of the signal in the HVS. This work is orthogonal to our subpixel rendering and we believe that both could be combined to enhance the display of images.

We also investigate subpixel-aware texture filtering that can be directly used in (GPU-based) rendering. Texture filtering is a crucial part in the rendering pipeline. In their seminal work, Greene and Heckbert [GH86] introduce the Elliptical Weighted Average (EWA) filter for high-quality texture filtering. McCormack et al. [MPFJ99] describe Feline, a fast approximation of the EWA using mip-mapping that lowers the filter cost. Shin et al. [SLK01] improve this work by in-

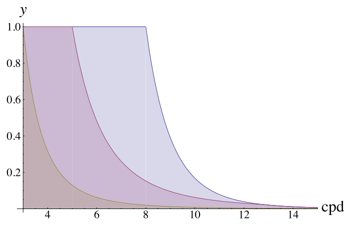

Figure 2: *The simplified chromatic contrast sensitivity function is shown (right to left) for Y', C_1 and C_2 [Pla00].*

troducing an additional weighting of the samples. Recently, Mavridis and Papaioannou [MP11] describe a filter for modern GPUs that closely matches the EWA filter.

2. Subpixel Rendering

In this section we briefly describe the approach for deriving optimal filters for subpixel rendering. For displaying an image on matrix displays, such as LCD monitors, the image signal is normally sampled at the centers of the pixels, or an average pixel color is determined. This image sample then controls the brightness of the individual subpixels to create the perceived color. Subpixel rendering in contrast takes the underlying spatial subpixel structure within a pixel into account and retrieves an image sample for *each* subpixel. This approach increases the perceived resolution of the display, however, it is likely to cause obvious color artifacts if the display colors of the subpixels are ignored (see Klompenhouwer et al. [KdH03] for a discussion). These artifacts are not surprising, as the chrominance signal is now undersampled and therefore prone to aliasing. Fortunately, color aliasing can be tolerated to a certain degree. This can be understood by converting the color displayed by the pixel to an *opponent color space* that describes color in a manner similar to the human visual system. Color is separated into a luminance channel, as well as an opponent red-green and yellow-blue chrominance channel. The HVS acts as low-pass filter in the opponent color space and diminishes the significance of the chrominance channels quickly for high frequency image details. This behavior can be modeled with the (chrominance) contrast sensitivity function ((C-)CSF, Fig. 2), and in turn be exploited to reduce the amount of perceptible aliasing.

2.1. The Contrast Sensitivity Function

The human eye's sensitivity to luminance differences depends on their spatial frequency [CR68]. This sensitivity is greatest at about 5–10 cpd (cycles per degree), and falls off for lower or higher cpd. The idea of contrast sensitivity can also be applied to color differences, and is usually given for opponent colors, i.e. red-green and yellow-blue. The sensitivity for opponent color gratings (chrominance contrast sensitivity) is highest at small cpd's and then falls off quickly [Mul85]. The actual shape of the function depends on the exact opponent color space. We follow Platt [Pla00] and approximate the actual contrast sensitivity functions with simple functions, as depicted in Fig. 2.

2.2. Optimal Filtering

The goal of an optimal filter for subpixel rendering is to suppress color aliasing to be unnoticeable while keeping a high spatial resolution of the luminance signal. We first review Platt's [Pla00] approach to derive an optimal filter for RGB-stripe matrix displays. This assumes that a display pixel has $k = 3$ subpixels, i.e., a red, green, and blue subpixel. It also assumes that the RGB image signal is sampled at subpixels, denoted as α_k. At each subpixel position, an RGB color value $\boldsymbol{\gamma}_k$ is sampled from the image signal. The design of an optimal filter is inspired by a perceptual error metric, that seeks to minimize the error that is introduced when displaying an RGB color value $\boldsymbol{\gamma}_k$ of the image signal with only the single color intensity α_k of the k-th subpixel in an arbitrary scanline of the display.

Computing this error in a way that exploits the characteristics of the HVS for optimal display requires a conversion to an opponent color space, where the error at the position of the k-th subpixel can be expressed as:

$$\mathbf{E}(k) = \underbrace{3\mathbf{m}_k\alpha_k}_{\text{displayed color}} - \underbrace{\sum_{d=1}^{3}\mathbf{m}_d\gamma_{k,d}}_{\text{image signal}}, \tag{1}$$

where d iterates over the RGB color channels and $\mathbf{m}_i$ is the $(i \bmod 3)$-th column vector of a 3×3 matrix that converts RGB to an opponent color space. We use the linear $Y'C_1C_2$ opponent color space which was designed to work best with linear transformations [JF05] and for which the conversion from RGB is given by:

$$M_{Y'C_1C_2} = \begin{pmatrix} 0.23 & 0.73 & 0.06 \\ 0.20 & -0.31 & 0.11 \\ 0.23 & 0.66 & -0.89 \end{pmatrix}. \tag{2}$$

To measure the significance of the error on the perceived image it has to be weighted with the contrast sensitivity function (CSF). To this end, the error must be represented in frequency space:

$$\hat{\mathbf{e}}_f = \sum_{k=0}^{N-1}\mathbf{E}(k)\exp\left(-\frac{2i\pi kf}{N}\right), \tag{3}$$

where $\hat{\mathbf{e}}_f$ denotes one of the N (complex) Fourier coefficients which are computed as a weighted sum over the error at each of the N subpixels in a single scanline. The contrast sensitivity function acts as a low-pass filter on the error which quickly filters out errors that occur at high frequencies in the image signal. As mentioned before, we use the C-CSF suggested by Platt [Pla00] (Fig. 2), which models it as a set of simple low-pass filters. We also examined other C-CSF models that fit experimental data (on a small sample of subjects) more accurately [Mul85]; however, these functions

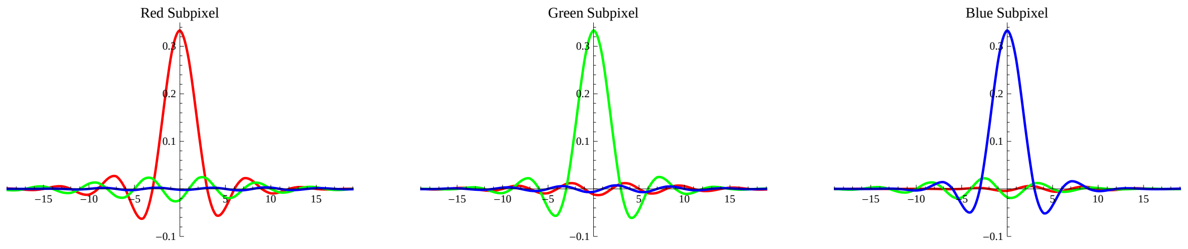

Figure 3: *Optimal filters for the RGB stripe layout. The curves are colored according to the color component that influences the subpixel denoted. The x-axis is given in subpixel units, i.e. every pixel is three units wide.*

tend to penalize luminance for low frequencies, which leads to band-pass filters. These have high support in the spatial domain and are also harder to model analytically. To support our original goal of efficient, low-cost subpixel rendering, we decided to utilize the simpler C-CSF by Platt.

The total squared error over all frequencies, luminance, and opponent color channels can then be computed as a sum over dot products:

$$\mathcal{E} = \sum_{c\in\{Y',C_1,C_2\}} \sum_{n=0}^{N-1} \mathbf{W}_c(f_n)\left\langle \hat{\mathbf{e}}_n, \hat{\mathbf{e}}_n^* \right\rangle, \tag{4}$$

where $\mathbf{W}_c(.)$ is the CSF for opponent channel c, and n iterates over all N Fourier coefficients $\hat{\mathbf{e}}_n$. f_n is the frequency (in cycles per degree) the n-th Fourier coefficient corresponds to. Note that the relation between Fourier coefficients and spatial frequency depends on the distance of the viewer to the display and the display's resolution. In the rest of the paper, we assume that 300 subpixels are viewed at roughly 16 cpd. With a pixel density of 100 pixels per inch (PPI) this corresponds to a viewer distance of roughly 25 cm. These numbers can easily be adjusted for different viewing configurations.

To derive the subpixel intensity values that minimize the error, one has to compute the gradient

$$\nabla\mathcal{E} = \left(\frac{\partial E}{\partial\alpha_1}, \cdots, \frac{\partial E}{\partial\alpha_N}\right) \tag{5}$$

and solve for $\nabla\mathcal{E} = 0$. The i-th component of $\nabla\mathcal{E}$ can be derived as:

$$\frac{\partial E}{\partial\alpha_i} = \sum_{c,n,k} W_c(f_n) m_{c,i}\left(m_{c,k}\alpha_k - \sum_{d=1}^{3} m_{c,d}\gamma_{k,d}\right)\cos\left(\frac{2n(k-i)}{\pi^{-1}N}\right), \tag{6}$$

where c runs over all three opponent color channels, and n and k run over N subpixels and N Fourier coefficients, respectively.

Using Eq. 6, one can restructure $\nabla\mathcal{E} = 0$ into a system of linear equations, whose solution yields a direct mapping of the RGB color values to subpixel intensity values:

$$\mathbf{A}\mathbf{x} = \sum_{d=1}^{3} \mathbf{B}_d\mathbf{y}_d \;\rightarrow\; \mathbf{x} = \mathbf{A}^{-1}\sum_{d=1}^{3} \mathbf{B}_d\mathbf{y}_d,$$

with $\mathbf{x} = (\alpha_1, \cdots, \alpha_N)^T$ and $\mathbf{y}_d = (\gamma_{1,d}, \cdots, \gamma_{N,d})^T$. The matrices $\mathbf{C}_d = \mathbf{A}^{-1}\mathbf{B}_d$ form a direct mapping of the d-th color channel of the RGB values $\boldsymbol{\gamma}_k$ to the subpixel intensities α_k of the k-th subpixel. Platt [Pla00] observed that those mapping matrices are block Toeplitz matrices and thus, instead of solving the linear system for each scanline, nine discrete filter kernels can be extracted from those matrices. To control the intensity of each subpixel, one obtains three discrete filter kernels for every subpixel color, which combine all three RGB values into a single intensity α_k of the k-th subpixel. All 9 filter kernels are shown in Fig. 3. These filters can be stored and applied directly to obtain a subpixel-filtered image. However, due to their large spatial extent, the direct application of these filters is costly and we thus strive for simplifications without significant loss in filtering quality. Furthermore, we seek to model these filters analytically in order to simplify their description and evaluation. This is discussed in the following section.

2.3. Frequency Analysis and Analytical Formulation

To gain further insight into the optimal filters, we investigated the impulse response spectrum of each filter kernel illustrated in Fig. 4. We note that for each subpixel we obtain a low-pass filter and two band-pass filters that peak at about 16 cpd. We also see that the impact that band-pass filters have on the intensity of the subpixel is restricted to a narrow frequency band. Furthermore, due to their low amplitude, they hardly contribute to the intensity at all. Due to their limited support in frequency space and their marginal contribution, it is safe to omit those filters without introducing distracting artifacts. We have also verified this observation in our experiments (see Sect. 4.1).

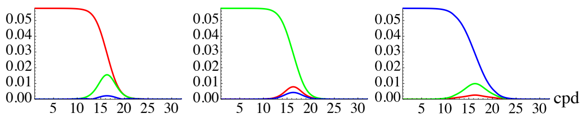

Figure 4: *Filter response spectra for all three subpixels of an RGB display. Colored curves show the filter response of the corresponding color channels in the RGB signal that influence the red (left), green (middle) and blue (right) subpixels.*

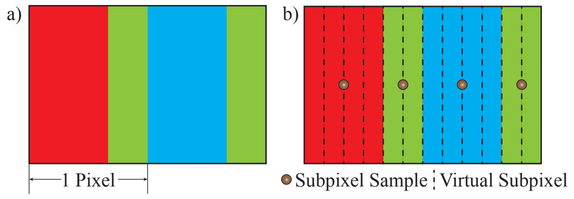

Figure 5: *a) The PenTile RGBG subpixel layout reduces the number of red/blue subpixels, while increasing their size; a single pixel consists of only two subpixels. b) We use virtual subpixels to incorporate such patterns into our derivation.*

The filter kernels based on the derivation from Sect. 2.2 are discrete and only defined at subpixel positions. This is highly impractical for image filtering applications, as the filter usually needs to be evaluated for multiple pixel samples that do not align with the subpixel positions. This implies that the optimal filter would have to be recomputed and stored for each sample layout used. Hence, we strive for an analytical description of the filter kernels. Empirically we have found that the low-pass filters can be modeled well with a Gaussian-windowed sinc function:

$$f(x) = \frac{1}{3}\frac{\sin ax}{ax}\exp\left(-\frac{x^2}{b}\right). \tag{7}$$

Using a non-linear least square fit, we determine the coefficients a and b as presented in Table 1. Our proposed analytical model fits the data exceptionally well, with a root mean squared error of less than 0.02 for each color channel. We also considered fitting well-known filters like the Mitchell filter [MN88], but deemed it unsuitable since there are only two negative lobes.

Table 1: *Parameters for optimal filtering for displays with different subpixel patterns. For RGB stripe and 2D RGBW parameters are given in subpixel units. RGBG parameters are shown in units of virtual subpixels (cf. Fig. 5).*

	RGB Stripe			
	R	G	B	–
a	$\pi/3$	$\pi/3$	$\pi/3$	–
b	92.65	76.58	46.62	–
	PenTile RGBG			
	R	G	B	G
a	$\pi/12$	$\pi/6$	$\pi/12$	$\pi/6$
b	457.37	476.62	457.37	479.06
	2D RGBW			
	R	G	B	W
a	2.06	1.75	1.89	3.04
b	7.12	1.40	2.66	16.36

2.4. Arbitrary 1D Subpixel Patterns

We now extend the basic analysis to arbitrary 1D subpixel patterns. We exemplify this with the PenTile RGBG subpixel layout, which is commonly found on mobile devices such as smartphones. Exploiting the different sensitivity characteristics of the HVS, these displays offer less resolution for the red and blue color channels. To that end, each pixel of the PenTile RGBG display is built of a green plus either a red or blue subpixel, where the latter have twice the extent of the green subpixel (Fig. 5a). To apply optimal filtering, we have to take the layout of the subpixel pattern into account. For irregular sampling patterns we use zero padding by introducing virtual subpixels that are laid out in a regular pattern and then align the centers of each physical subpixel of the irregular layout to the virtual subpixels (Fig. 5b). This leaves virtual subpixels that cannot be assigned to physical subpixels and need to be ignored. To this end, we modify $\mathbf{m}_k$, which was used to select columns of the color conversion matrix (Eq. 2), so that if k does not align with a subpixel center, we set it to $\mathbf{0}$, effectively "deactivating" irrelevant virtual subpixels (similar to [MK06]) . Fig. 6 shows the optimal filters we obtain for PenTile subpixels. As can be seen, the filters for red and blue subpixels adapt to the lower resolution and thus the filters have larger support than the filters for the two green subpixels.

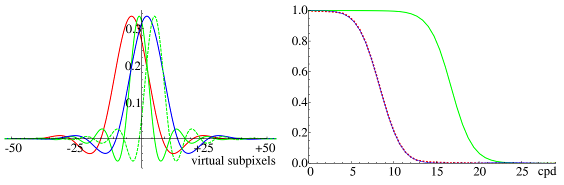

Figure 6: *Optimal filters for the PenTile RGBG layout. Band-pass filters have been omitted, as their contribution is similarly low as for RGB displays. There are two green filters for the two different green subpixels involved in order to display an RGB color signal (cf. Fig. 5). The frequency plot shows the characteristics of the filters (note that the red and blue curves are very similar).*

2.5. 2D Subpixel Patterns

Similar to Messing et al. [MKD03], we extend the mathematical framework to two-dimensional subpixel patterns. However, we use the original formulation by Platt instead of constrained optimization, since most subpixel geometries exhibit rectangular structure, which is quite amenable to a straight-forward extension of the 1D approach. In situations where subpixel shape is not negligible, we can introduce virtual subpixels, as in the previous subsection. In the 2D case the error metric is a function of both the horizontal subpixel position s and the vertical subpixel position t:

$$\mathbf{E}(s,t) = 3\mathbf{m}_{s,t}\alpha_{s,t} - \sum_{d=1}^{3}\mathbf{m}_d\gamma_{s,t,d}.$$

In frequency space the error then becomes:

$$\hat{\mathbf{e}}_{n_x,n_y} = \sum_{s=1,t=1}^{N} \mathbf{E}(s,t) \exp\left(-\frac{2i\pi s n_x}{N}\right) \exp\left(-\frac{2i\pi t n_y}{N}\right).$$

In the same manner as in Sect. 2.2, we can derive a gradient vector of the total error and set it to zero. In the 2D case the gradient is given as:

$$\nabla\mathcal{E} = \left(\frac{\partial\mathcal{E}}{\partial\alpha_{1,1}}, \cdots, \frac{\partial\mathcal{E}}{\partial\alpha_{1,N}}, \cdots, \frac{\partial\mathcal{E}}{\partial\alpha_{N,N}}\right)^T,$$

and contains N^2 components. The analytical expression for the partial derivative of the error is:

$$\frac{\partial\mathcal{E}}{\partial\alpha_{s,t}} = \sum_{c,k,l,u,v} W_c(f_k, f_l) m_{c,s,t} \left[m_{c,u,v}\alpha_{u,v} - \sum m_{c,d}\gamma_{d,u,v}\right] \Phi,$$

with $\Phi = \cos\left(\frac{2\pi}{N}(k(u-s)+l(v-t))\right)$. In the same manner as in the 1D case, we can construct and solve a system of linear equations, where the solution of this system yields matrices which transform an entire 2D block of RGB color values into subpixel intensities. Also as in the 1D case, we can extract discrete filter kernels that transform color values into subpixel intensities. The system of linear equations grows quadratically with the number of subpixels and is inefficient to solve if many subpixels (i.e., hundreds) are taken into account. Fortunately, we only need to solve the equations once for each subpixel pattern.

2.5.1. 2D RGBW Pattern

As an example, we have applied the analysis to the 2D PenTile RGBW pattern, where 2×2 subpixels containing red, green, blue and white primaries form a single pixel. To be able to consider such a pattern for optimal filtering we need to treat white as a primary color and thus employ a matrix that can convert from RGBW into the $Y'C_1C_2$ opponent color space [RKAJ08]:

$$M_{rgbw} = \begin{pmatrix} 0.15 & 0.47 & 0.04 & 0.37 \\ 0.13 & -0.20 & 0.07 & 0 \\ 0.14 & 0.43 & -0.57 & 0 \end{pmatrix}.$$

As most images and textures are given in RGB color space, we need to convert them first to RGBW. We do this by computing the minimum of all three RGB color channels, then subtracting that value from each component, treating it as pure white [ECH05]. Using M_{rgbw} and the above derivation, we can compute the optimal 2D filters which are shown in Fig. 7. Again, we ignore the band-pass filters and approximate the filters with low-pass characteristics using the following model:

$$f(x,y) = \frac{1}{4}\operatorname{sinc}\left(a\sqrt{x^2+y^2}\right)\exp\left(-\frac{(x^2+y^2)}{b}\right).$$

3. Subpixel Rendering in Image Synthesis

In the following we propose to include subpixel rendering in GPU-based image synthesis, which as we will see, induces only very small overhead. In particular, we describe how multisampled antialiasing (MSAA) can incorporate subpixel rendering, as well as how texture filtering can take advantage of subpixel rendering. For brevity we restrict the discussion in the following section to the RGB stripe layout. Similar arguments can be made for other subpixel layouts.

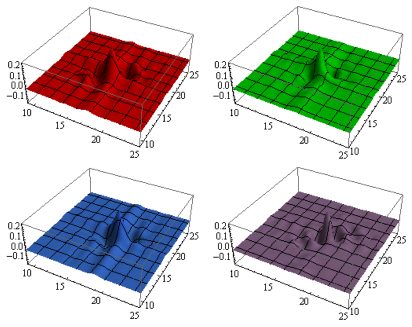

Figure 7: *Four 2D filters for red, green, blue, and white subpixels. The overall shape of the filters is similar to the ones in Fig. 3, but extended to the 2D RGBW pattern.*

3.1. MSAA Resolve

Multisampled antialiasing is a common technique used in interactive rendering to improve the quality of the displayed image. This technique computes images at a much higher resolution than required for display, using multiple color samples per pixel. Afterwards all color samples within a pixel are resolved into a single color value for display, applying an antialiasing filter that spans the entire pixel.

This technique can be easily extended to subpixel-awareness without increasing computation time of the resolve significantly. In order to apply subpixel filters during the MSAA resolve, we use the analytic formulation from Sect. 2.3 to compute the filter weights. We compute the distance d_x (in subpixel units) in the horizontal direction (for 1D patterns) between the subpixel centers and the sample positions and compute the filter weights $w(d_x)$, exemplarily shown for the RGB stripe layout, as follows:

$$w(d_x) = \begin{cases} \frac{1}{3}\operatorname{sinc}(ad_x)\exp\left(-bd_x^2\right) & \leftrightarrow \quad |d_x| \le 6 \\ 0 & \leftrightarrow \quad |d_x| > 6 \end{cases} \tag{8}$$

Since the optimal filter has infinite support, we truncate it after the first negative lobe, as the filter weights quickly diminish beyond this point. In case of the RGB stripe layout this is the case for distances d_x larger than 6 subpixels. For other layouts, this clamping distance has to be adjusted appropriately. This MSAA resolve can be executed very efficiently on the GPU, as will be shown in the results section. For 2D patterns, we compute distances d_x and d_y and evaluate the appropriate (clamped) 2D filter.

3.2. Texture Filtering

In rendering algorithms, texture filtering is an integral component. A single (sub-)pixel may cover a large texture region which must be filtered in order to avoid distracting artifacts due to undersampling of the texture. We propose subpixel texture filtering in order to increase the perceived display resolution, yielding textures that appear sharper.

3.2.1. Elliptical Weighted Average Subpixel Filtering

For high quality texture filtering, the elliptical weighted average (EWA) [GH86] is often employed. This filter determines the elliptical footprint of a pixel in texture space and computes the pixel's color from all texels that fall within the elliptical region. The size of the ellipse is determined using the partial derivatives of the texture coordinates. To account for all texels that fall within the elliptical region in texture space, all texels in the bounding box of the ellipse are enumerated and tested for overlap with the ellipse.

For subpixel rendering we can directly fold subpixel filtering into the texture filtering algorithm. The optimal filter is a low-pass filter with a support of multiple subpixels on the screen. To account for this extended filter support in texture space, we scale the partial derivatives of the texture coordinates to span the filter's support and compute the ellipse from the scaled derivatives. For simplicity we assume that all subpixels share the same derivatives, hence we need to compute the ellipse only once per pixel. To compute the subpixel-filtered color value, we then need to displace the ellipse in texture space. Afterwards we traverse the common bounding box of all ellipses and compute a texel's contribution to all subpixels simultaneously. We use a Gaussian with hand-tuned standard deviation to obtain best filtering results, although any other filter kernel can be used for EWA filtering.

3.2.2. GPU-Based Subpixel Texture Filtering

While EWA filtering produces superior results, it is expensive to evaluate. It is also possible to modify GPU-based anisotropic texture filtering to become subpixel-aware. While it is not as accurate as EWA filtering, its performance is substantially higher. To this end, we apply the subpixel filter directly in screen space. We compute the filter taps by displacing a pixel's interpolated texture coordinates along the directions of its partial derivatives and then sample the texture using a trilinear or anisotropic texture lookup. For texture sampling we require samples from multiple subpixels, which are then convolved with the subpixel filters. Consequently, we need to adjust the mip-map level from which we retrieve a texture sample. For this, we scale the partial derivatives to the extent of a subpixel and compute the appropriate mip-map level as described by Schilling et al. [SKS96].

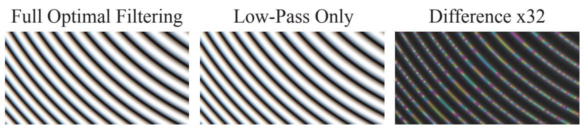

Figure 8: *The influence of the band-pass filters obtained by optimal filtering is visually insignificant as demonstrated on this test pattern.*

4. Results

In the following we discuss our results. All filters were implemented and executed on the GPU using Direct3D 11 shaders. Timings were taken on a PC with an Intel Core i7 860 processor with 2.80 GHz, 8 GB of main memory, and an AMD Radeon 5870 GPU. First we discuss our experimental evaluation of the band-pass filters derived for optimal filtering. Then we present our results for MSAA rendering for different subpixel layouts, which we simulated directly in the pixel shader, followed by a discussion of our subpixel texture filtering results. We also investigated the temporal properties of our filters, showing that our optimal filter is stable. Please see the supplementary material for an example animation. Finally, we present the findings of a user study we conducted in order to assess the visual quality of our proposed optimal subpixel filter, simulating different subpixel layouts. For all our results, we applied filtering in linear RGB color space, i.e. before gamma correction. This is the natural space for doing the MSAA resolve, but assumes linear input images for texture filtering (or requires inverse gamma correction beforehand).

4.1. Influence of Band-Pass Filters

We briefly discuss the influence of the omitted band-pass filters. As shown in Sect. 2.3 these filters only contribute to the filtered image for frequencies close to the cutoff frequency of the low-pass filters. We evaluate their influence on a high frequency pattern in Fig. 8, indicating that it is minimal. As can be seen in the difference image the contributions are negligible, and become only visible when scaled 32-fold.

4.2. MSAA Resolve

In order to measure the performance of our custom MSAA resolve, we render an image with 8×MSAA in full HD resolution (1920 × 1080) using a render target with 8-bit precision per color channel. Fig. 9 shows several examples of this (please also see the supplemental material). We enabled execution of the pixel shader per image sample and thus we effectively perform super-sampling when rendering into an MSAA texture. To perform the subpixel resolve, the shader has to process 40 samples for the optimal filter, compared to 8 samples for a common per-pixel resolve pass (which corresponds to a non-subpixel-aware box filter). For efficiency we precompute the coefficients for the optimal filter, and then the performance of the resolve pass scales linearly with the

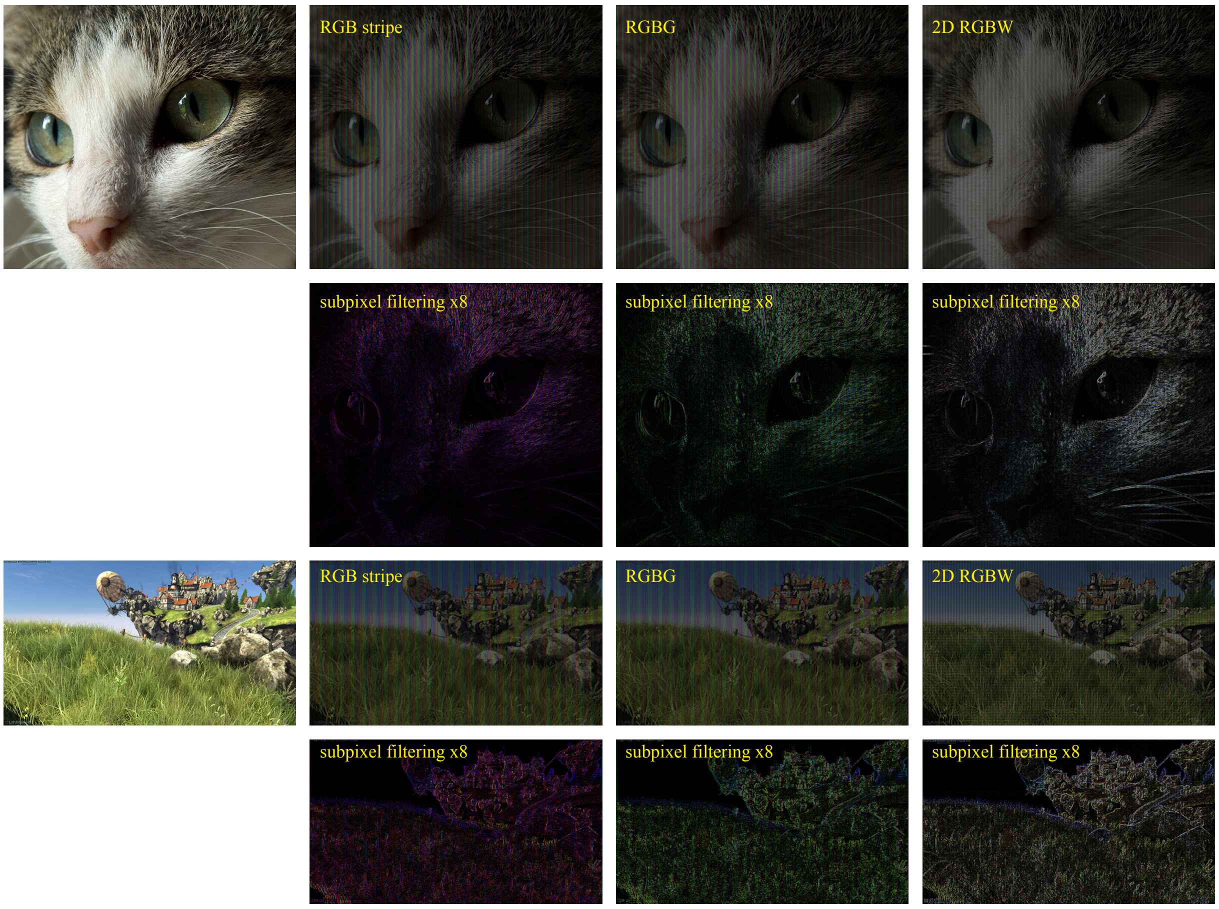

Figure 9: *These images show subpixel renderings of a photograph and an image from the UNIGINE game engine benchmark; both images are high resolution and have been injected into an MSAA render target. We show different subpixel arrangements with filtering obtained from our subpixel-aware MSAA resolve (please look at the images on the screen and zoom in). The second and fourth row show the absolute difference (×8) to images resolved with standard antialiasing. The colorful appearance is due to the relative positions of subpixels: the difference image for the RGB stripe pattern is mostly purple as the green subpixels reside on the RGB pixel centers, while red and blue subpixels are offset and thus most affected by the filtering. For the RGBG pattern the green subpixels are shifted, while all subpixels are roughly equally affected with the 2D RGBW pattern.* (Image captured from UNIGINE Heaven DX11 Benchmark. UNIGINE Corp. 2012. All rights reserved.)

number of samples in the MSAA texture and the color precision. For the standard per-pixel box filter the 8×MSAA resolve at 1920×1080 takes 0.53ms, compared to 2.29ms for our optimal filter.

4.3. Texture-Filtering

We evaluated our proposed subpixel texture filtering in terms of quality and speed. Fig. 1 shows a comparison of different texture filtering techniques. For this example a square texture with a resolution of 6144^2 pixels was used, while the final rendering was 1024^2 pixels in size. We have implemented all filtering methods in pixel shaders. Our implementation of the EWA algorithm is based on the original formulation by Greene [GH86]. Furthermore, we simulated the different subpixel layouts in a pixel shader for display on a generic LCD display. As can be seen in the right part of Fig. 1, subpixel texture filtering reduces blurriness and increases the sharpness of the texture. In terms of performance, filtering cost mostly scales linearly with the number of required texture lookups. Since the performance of the filters depends strongly on the resolution of the input texture, we only report relative performance values. For the RGBG layout, we have found that the filtering costs when using our optimal subpixel filtering algorithm increase by a factor of about 1.1 to 1.2. This only moderate gain in rendering time is due to the reduced pixel resolution that the PenTile RGBG display offers. For the other subpixel layouts, filtering is 1.8 to 2.2 times more costly than standard filtering.

4.4. User Studies

We conducted a user study to compare the quality of our proposed optimal filter with other filters. To this end, we

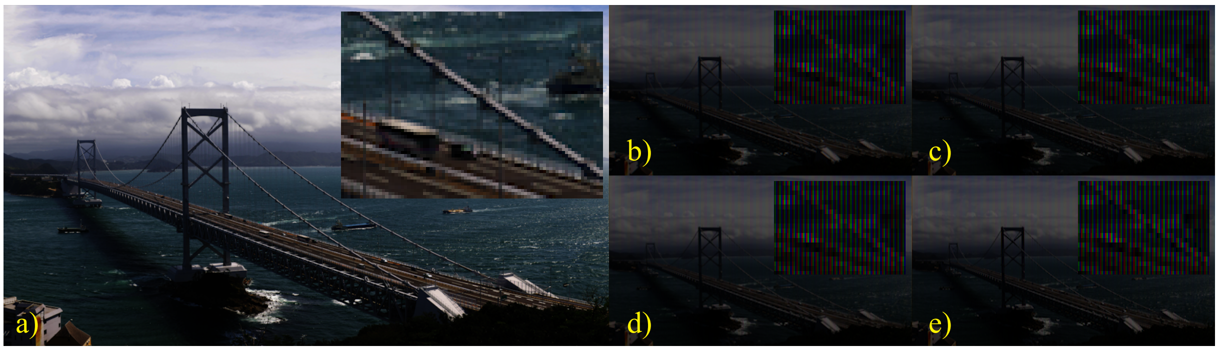

Figure 10: *Different (subpixel) filters employed in the user study (RGB stripe layout). a) Reference image. b) Mitchell filter, no supbixel rendering. c) Box filter, subpixel rendering. d) Mitchell filter, supbixel rendering. e) Our optimal filter, subpixel rendering. Filtered images are dimmer due to simulating subpixels with real pixels. Although the differences seem negligible in the zoom-ins, they are noticeable when viewing the resulting images on the screen (as supported by our user study).*

employed a 56" (142.24 cm) Quad-HD monitor with a resolution of 3840×2160 pixels, i.e., a pixel pitch of 0.32mm. Participants sat 1.8m away from the display, which corresponds to a distance of about 50cm on a regular 24" Full-HD display (Fig. 11 shows our set-up). The high-resolution display was used to show each of six different reference images (see Fig. 10 for an example; the supplemental material contains all images used) at the full resolution of the display. We then took grids of 3×3 monitor pixels to simulate a low-resolution display, e.g. for RGB stripe layout, the left column of the 3×3 pixels shows red only, the middle column green, and the right column blue. We used our optimal subpixel filter, a box-shaped subpixel filter (roughly corresponding to ClearType), a subpixel Mitchell filter, and a standard Mitchell filter (no subpixel processing) to create four downsampled versions (by 1/3 along each axis) of the original images. Both Mitchell filters use parameters $B = C = 1/3$, following the recommendation $B + 2C = 1$ [MN88]. Participants of our study performed a pairwise comparison experiment on these images: each stimulus consisted of two pairs of images (enumerating all possible combinations of filters in random order) and participants chose the image that (subjectively) best replicated the high-resolution image. Subjects were allowed to look at the reference, high-resolution image in order to better assess the image quality of filtered results. To this end, they could selectively display the reference, which was displayed in place of the filtered images after a brief (1 sec) pause. Twenty-two participants took part in the experiment. Fig. 12 summarizes the findings (more detail in supplemental material). Overall, our optimal subpixel filter was significantly better than any other filtering method ($p < 0.01$, t-test). There is a clear benefit to using our optimal filter, even for displaying photographs or game-like content.

In a second experiment, we compared different subpixel patterns. The set-up was identical to the first experiment, i.e., using a Quad-HD monitor to simulate three different lower-resolution displays with an RGB, an RGBG, and an RGBW pattern, see Fig. 1 for the exact layouts. Please note that for the user study we used slightly different RGBG and RGBW layouts: to ensure a fair comparison of all sub-pixel patterns we simulated all of them with the same number of pixels (6×6) on the physical high-resolution screen. For all three cases, we used our optimal subpixel filter to drive the rendering. Participants again performed a pairwise comparison, where each stimulus compared an image rendered with two different subpixel patterns (enumerating all possible pairs of patterns). This was done for the same six images as above. The same 22 participants took part and the findings are summarized in Fig. 13. Overall, RGB and RGBW were not significantly different. However, both were significantly better ($p < 0.01$) than RGBG.

5. Conclusion

We have presented methods for subpixel-aware MSAA and texture filtering. To this end, we have analyzed different 1D and 2D subpixel patterns, and derived (perceptually) optimal, analytic filters. Our results show that the perceived display resolution increases without noticeable artifacts. The additional cost for subpixel-aware MSAA and texture filtering is small compared to the benefit of using it. An interesting direction for future work is combining our low-cost subpixel rendering with temporal integration in the spirit of

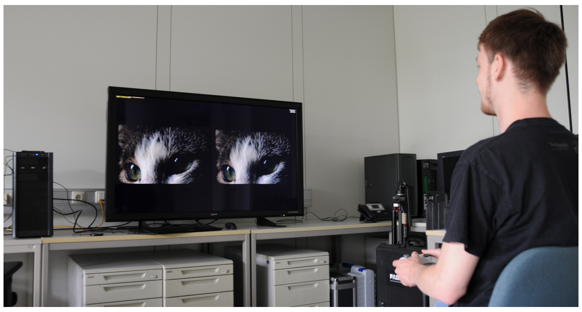

Figure 11: *Test set-up for the user study. Conditions in the image do not match actual testing conditions, where the environment illumination was completely darkened.*

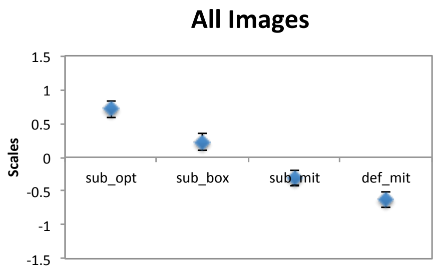

Figure 12: *User study results for comparing different subpixel filtering methods for six different images (bridge, cat, fairy, hairball, text, and tree). Participants were asked to perform a pairwise comparison between the different methods and choose the best reproduction of a given image. Thurstonian scaling was applied and our optimal filter was significantly better than any other method ($p < 0.01$, t-test).*

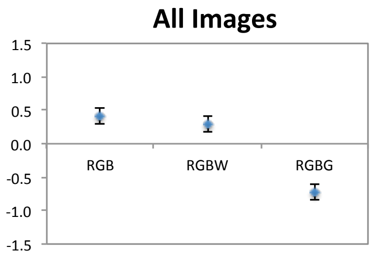

Figure 13: *User study results for comparing different subpixel patterns for six different images (bridge, cat, fairy, hairball, text, and tree). Participants were asked to perform a pairwise comparison between the different pattern layouts and choose the best reproduction of a given image. There was no statistical significant difference between RGB and RGBW, but GRGB was significantly worse than RGB and RGBW ($p < 0.01$, t-test).*

Didyk et al. and Templin et al. [DER*10, TDR*11], which works best for static images, in order to enhance the display of both static and dynamic images.

References

[Atc05] Atcheson B.: Subpixel rendering of Bayer-patterned images. http://www.cs.ubc.ca/~atcheson/projects.html, 2005. 2

[CR68] Campbell F. W., Robson J. G.: Application of Fourier analysis to the visibility of gratings. *Journal of Physiology 197*, 3 (1968), 551–566. 3

[DER*10] Didyk P., Eisemann E., Ritschel T., Myszkowski K., Seidel H.-P.: Apparent display resolution enhancement for moving images. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2010) 29*, 4 (2010). 2, 10

[Duf89] Duff T.: Polygon scan conversion by exact convolution. *Proceedings International Conference on Raster Imaging and Digital Typography* (1989), 151–168. 2

[ECH05] Elliott C., Credelle T., Higgins M.: Adding a white subpixel. *Information Display 21*, 5 (2005). 2, 6

[FAY*09] Fang L., Au O., Yang Y., Tang W., Wen X.: A new adaptive subpixel-based downsampling scheme using edge detection. In *Circuits and Systems (Proceedings of ISCAS)* (2009), pp. 3194 –3197. 2

[GH86] Greene N., Heckbert P.: Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications 6*, 6 (1986), 21–27. 2, 7, 8

[JF05] Johnson G. M., Fairchild M. D.: The effect of opponent noise on image quality. In *Proceedings of SPIE* (2005), vol. 5668, pp. 82–89. 3

[KdH03] Klompenhouwer M. A., de Haan G.: Subpixel image scaling for color matrix displays. *Journal of the Society for Information Display 11*, 1 (2003), 99–108. 2, 3

[KU81] Kajiya J., Ullner M.: Filtering high quality text for display on raster scan devices. In *Proceedings SIGGRAPH 1981* (1981), pp. 7–15. 2

[MD02] Messing D., Daly S.: Improved display resolution of subsampled colour images using subpixel addressing. In *International Conference on Image Processing* (2002), vol. 1, pp. I–625 – I–628. 2

[MK06] Messing D. S., Kerofsky L. J.: Using optimal rendering to visually mask defective subpixels. In *Proceedings of SPIE* (2006), vol. 6057, pp. 60570O–1–60570O–12. 2, 5

[MKD03] Messing D., Kerofsky L., Daly S.: Subpixel rendering on non-striped colour matrix displays. In *Image Processing (Proceedings of ICIP)* (2003), vol. 2, pp. 949–952. 2, 5

[MN88] Mitchell D. P., Netravali A. N.: Reconstruction filters in computer-graphics. *Computer Graphics (Proceedings of SIGGRAPH'88) 22* (1988), 221–228. 2, 5, 9

[MP11] Mavridis P., Papaioannou G.: High quality elliptical texture filtering on gpu. In *Symposium on Interactive 3D Graphics and Games* (2011), pp. 23–30. 3

[MPFJ99] McCormack J., Perry R., Farkas K. I., Jouppi N. P.: Feline: fast elliptical lines for anisotropic texture mapping. In *SIGGRAPH '99* (1999), pp. 243–250. 2

[Mul85] Mullen K. T.: The contrast sensitivity of human colour vision to red-green and blue-yellow chromatic gratings. *Journal of Physiology 359* (1985), 381–400. 3

[PKH*00] Platt J. C., Keely B., Hill B., Dresevic B., Betrisey C., Mitchell D. P., Hitchcock G., Blinn J. J., Whitted T.: Displaced filtering for patterned displays. In *Proceedings Society for Information Display Symposium* (May 2000), pp. 296–299. 2

[Pla00] Platt J. C.: Optimal filtering for patterned displays. *IEEE Signal Processing Letters 7*, 7 (2000), 179–180. 2, 3, 4

[RKAJ08] Reinhard E., Khan E. A., Akyüz A. O., Johnson G. M.: *Color Imaging: Fundamentals and Applications*. A. K. Peters, Ltd., 2008. 6

[SKS96] Schilling A., Knittel G., Strasser W.: Texram: A smart memory for texturing. *IEEE Computer Graphics and Applications 16* (1996), 32–41. 7

[SLK01] Shin H.-C., Lee J.-A., Kim L.-S.: SPAF: sub-texel precision anisotropic filtering. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001), pp. 99–108. 2

[TDR*11] Templin K., Didyk P., Ritschel T., Eisemann E., Myszkowski K., Seidel H.-P.: Apparent resolution enhancement for animations. In *Proceedings of the 27th Spring Conference on Computer Graphics* (2011), pp. 85–92. 2, 10

[XFMW08] Xu J., Farrell J., Matskewich T., Wandell B.: Prediction of preferred ClearType filters using the S-CIELAB metric. In *Image Processing (Proceedings of ICIP)* (2008), pp. 361 –364. 2