EG UK Theory and Practice of Computer Graphics (2010)
John Collomosse, Ian Grimstead (Editors)

Anisotropic Kuwahara Filtering with Polynomial Weighting Functions*

Jan Eric Kyprianidis [1] Amir Semmo [1] Henry Kang [2] Jürgen Döllner [1]

[1] Hasso-Plattner-Institut, Germany
[2] University of Missouri, St. Louis, USA

Abstract

In this work we present new weighting functions for the anisotropic Kuwahara filter. The anisotropic Kuwahara filter is an edge-preserving filter that is especially useful for creating stylized abstractions from images or videos. It is based on a generalization of the Kuwahara filter that is adapted to the local shape of features. For the smoothing process, the anisotropic Kuwahara filter uses weighting functions that use convolution in their definition. For an efficient implementation, these weighting functions are usually sampled into a texture map. By contrast, our new weighting functions do not require convolution and can be efficiently computed directly during the filtering in real-time. We show that our approach creates output of similar quality as the original anisotropic Kuwahara filter and present an evaluation scheme to compute the new weighting functions efficiently by using rotational symmetries.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Picture/Image Generation]: Display algorithms

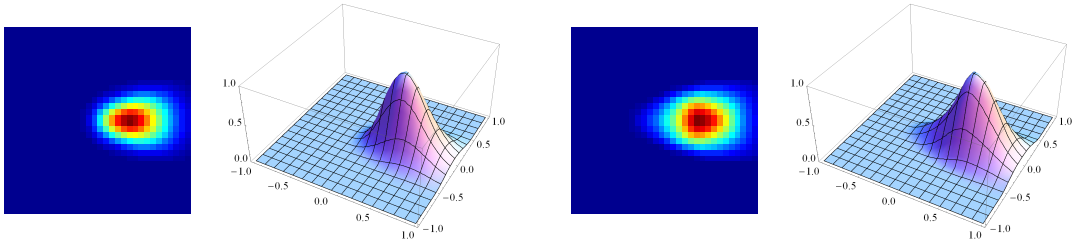


Figure 1: *Left: Weighting function* $(\chi_0 \star G_\rho) \cdot G_\sigma$ *based on convolution used in the definition of the anisotropic Kuwahara filter [KKD09, PPC07]. Right: Proposed weighting function based on the polynomial* $[(x+\zeta) - \eta y^2]^2$.

1. Introduction

Within the field of non-photorealistic rendering, a classical area of research is the stylization and abstraction of photographs using edge-preserving smoothing and enhancement filters. Prominent techniques in this area have in common that they remove detail in low-contrast regions without filtering across discontinuities and thus leave the overall structure of the input image unaffected. Popular examples are the bilateral filter [WOG06, KD08, KLC09] and mean shift [CM02, DS02].

Another popular filter in this field is the Kuwahara filter [KHEK76]. The general idea behind this filter is to divide the filter kernel into four rectangular subregions which overlap by one pixel. The filter response is then defined by the mean of a subregion with minimum variance. The Kuwahara filter produces clearly noticeable artifacts. These are due to the use of rectangular subregions. In addition, the subregion selection process is unstable if noise is present or subregions have the same variance. This results in randomly chosen subregions and corresponding artifacts. A more detailed discussion of limitations of the Kuwahara filter can be found in [PPC07].

* This is the author's version of the work. The definitive version is available at diglib.eg.org and www.blackwell-synergy.com.

(a) (b) (c)

Figure 2: *Examples created with the anisotropic Kuwahara filter using the proposed weighting functions.*

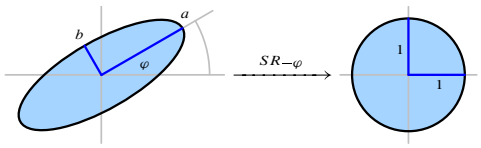
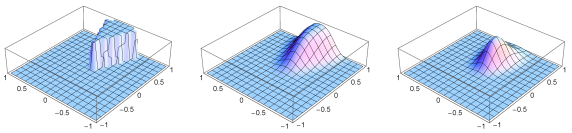


Figure 3: *The mapping $SR_{-\varphi}$ maps a rotated ellipse to the unit disc.*



(a) (b) (c)

Figure 4: *Construction of the weighting functions of the anisotropic Kuwahara filter: (a) Characteristic function χ_0. (b) $\chi_0 \star G_\rho$. (c) $K_0 = (\chi_0 \star G_\rho) \cdot G_\sigma$.*

Several attempts have been made to address the limitations of the Kuwahara filter. [PPC07] define a new criterion to overcome the limitations of the unstable subregion selection process. Instead of selecting a single subregion, the result is defined as the weighted sum of the means of the subregions. The weights are defined based on the variances of the subregions. This results in smoother region boundaries and fewer artifacts. To improve this further, the rectangular subregions are replaced by smooth weighting functions defined over sectors of a disc.

The anisotropic Kuwahara filter [KKD09] builds upon the generalized Kuwahara filtering concept by [PPC07] and replaces the weighting functions defined over sectors of a disc by weighting functions defined over ellipses. By adapting shape, scale and orientation of these ellipses to the local structure of the input, artifacts are avoided. Due to this adaption, directional image features are better preserved and emphasized. This results in overall sharper edges and the enhancement of directional image features. For a more detailed discussion of related work and comparisons with other techniques, we refer to [KKD09].

In this work, we present a modification of the anisotropic Kuwahara filter. We introduce new weighting functions that are not based on convolution. Consequently, they are applicable for calculation on the fly and can be computed at real-time rates. In addition, the proposed weighting functions are parameterizable. The eccentricity and expansion can be adjusted, which allows to control the overlapping areas to adjacent sectors.

2. Anisotropic Kuwahara Filtering

In order to adapt the filter shape to the local structure of the image, the anisotropic Kuwahara filter requires information about the local orientation and a measure for the anisotropy. Both can be derived from the eigenvalues and eigenvectors of the smoothed structure tensor [BBL*06]. The local orientation is given by the argument of the minor eigenvector and the anisotropy can be derived from the major and minor eigenvalues [YBFU96]:

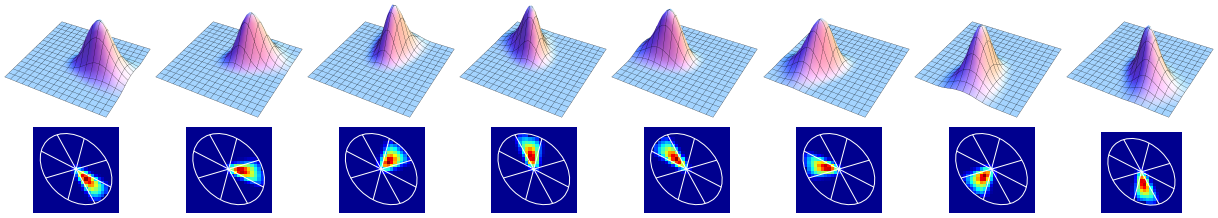
$$A = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}$$

The anisotropy A ranges from 0 to 1, where 0 corresponds to isotropic and 1 corresponds to entirely anisotropic regions.

Let $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$ denote the input image, let $(x_0, y_0) \in \mathbb{R}^2$ be a point and let φ be the local orientation and A be the anisotropy at (x_0, y_0). To adjust the eccentricity of the filter shape depending on the amount of anisotropy we set:

$$a = \frac{\alpha + A}{\alpha} r \quad \text{and} \quad b = \frac{\alpha}{\alpha + A} r \, .$$

Here r denotes the desired radius of the filter and $\alpha > 0$ is a tuning parameter that is typically set to $\alpha = 1$. In this case,



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5: *The weighting functions $K_i = K_0 \circ R_{2\pi i/8}$ (top) and their pullback $w_i = K_i \circ SR_{-\varphi}$ (bottom) for $i = 1, \ldots 8$.*

since A is in $[0, 1]$, it follows that we have $r \leq a \leq 2r$ and $r/2 \leq b \leq r$. Now let

$$S = \begin{pmatrix} a^{-1} & 0 \\ 0 & b^{-1} \end{pmatrix}$$

and let $R_{-\varphi}$ be the matrix defining a rotation by $-\varphi$. The mapping $SR_{-\varphi}$ then defines a linear coordinate transform that maps a rotated ellipse to the unit disc (Figure 3). The anisotropic Kuwahara filter now partitions this ellipse into different sectors similar to the rectangular areas of the original Kuwahara filter. Let N denote the number of sectors, with typical values $N = 4$ or $N = 8$. The different sectors must overlap. To achieve this, the anisotropic Kuwahara uses weighting functions that define how much influence a pixel has on a sector. To define these weighting functions over the ellipse, the general idea is to define corresponding weighting functions over the unit disc and then pull these back to the ellipse.

Let χ_0 be the characteristic function that is 1 for all points of $\mathbb{R}^2$ with argument in $(-\pi/N, \pi/N]$ and 0 otherwise. Then

$$K_0 = (\chi_0 \star G_\rho) \cdot G_\sigma$$

defines smooth weighting function over the unit circle. Here G_ρ and G_σ denote Gaussian functions and $\star$ denotes convolution. The convolution smoothes the characteristic function such that pixels from neighboring sectors are also considered in the weighting process. The multiplication achieves a decay with increasing radius (Figure 4). Reasonable values for σ and ρ are $\sigma = 0.4$ and $\rho = \sigma/3$. Weighting functions for the other sectors can be defined by smoothing the corresponding characteristic function or simply by rotating K_0:

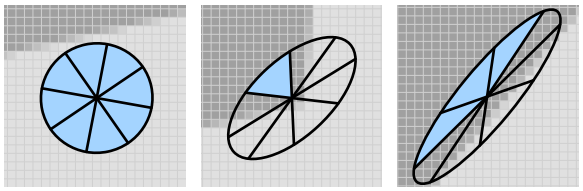
$$K_i = K_0 \circ R_{-2\pi i/N}, \quad i = 0, \ldots, N-1$$

Here, $\circ$ denotes composition of functions. By pulling back K_i, we finally get weighting functions w_i defined over the ellipse (Figure 5):

$$w_i = K_i \circ SR_{-\varphi} = K_0 \circ R_{-2\pi i/N} SR_{-\varphi}$$

Now let

$$m_i = \frac{1}{k} \int f(x)\, w_i(x - x_0)\, \mathrm{d}x \qquad (1)$$



(a) Homogeneous (b) Corner (c) Edge

Figure 6: *The anisotropic Kuwahara filter uses weighting functions defined over an ellipse, whose shape is based on the local orientation and anisotropy. The filter response is defined as a weighed sum of the local averages, where more weight is given to averages with low standard deviation.*

be the weighted local averages and

$$s_i^2 = \frac{1}{k} \int f^2(x)\, w_i(x - x_0)\, \mathrm{d}x - m_i^2 \qquad (2)$$

be the squared standard deviations, where k denotes the corresponding normalization factor. The output of the anisotropic Kuwahara filter is then defined by:

$$\frac{\sum_i \alpha_i m_i}{\sum_i \alpha_i}, \qquad \alpha_i = \frac{1}{1 + \|s_i\|^q}.$$

This definition of the weighting factors α_i ensures that more weight is given to sectors with low standard deviation, i.e. those that are more homogeneous (Figure 6). The parameter q is typically set to $q = 8$.

3. Alternative Weighting Functions

Since the weighting functions w_i are adapted to the local image structure, they are generally different per pixel. It is thus not possible to perform the convolutions in equation (1) and (2) in frequency space. The calculation therefore has to be carried out in the spatial domain. An efficient implementation hence depends on the possibility to quickly evaluate K_i. The direct computation of K_i would be absurd, since the computation requires convolution. In [KKD10] K_0, K_1, K_2 and K_3 are therefore sampled into a RGBA texture map. By using this texture map, the weights can be calculated using a single texture lookup for $N = 4$ and two texture lookups for $N = 8$.

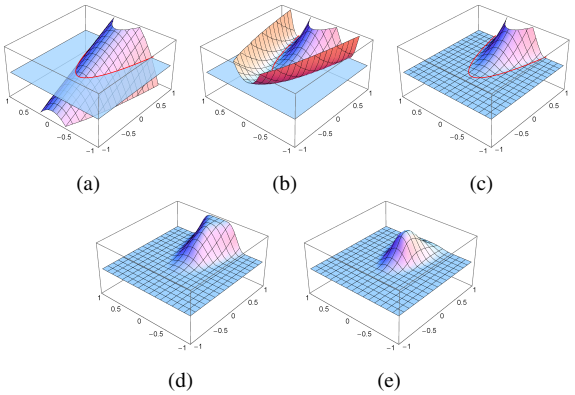


Figure 7: *Construction of the proposed weighting functions: (a) Polynomial* $(x+\zeta)-\eta y^2$. *(b) Squared polynomial* $[(x+\zeta)-\eta y^2]^2$. *(c)* $\tilde{k}_0$. *(d) Normalized* $\tilde{k}_0$. *(e) Weighting function* $\tilde{K}_0$.

In this work we use a different approach. Instead of discretizing or approximating the K_i, we aim for replacing them with other functions, that are simpler to compute. By construction, the sum $\sum_i K_i$ is equal to the Gaussian function G_σ. The K_i therefore define a smooth partition of the Gaussian function G_σ. The proposed weighting functions also have this property and their construction is illustrated in Figure 7. Basis for the construction is the polynomial $(x+\zeta)-\eta y^2$ that is shown in Figure 7(a). The red parabola shows the zero-crossing of this polynomial. By clamping negative values to zero, we get a function that is non-zero inside and zero outside the parabola. By taking the square, we ensure that the transition at the zero-crossing is smooth (Figure 7(c)):

$$\tilde{k}_0(x,y) = \begin{cases} \left[(x+\zeta)-\eta y^2\right]^2 & x \geq \eta y^2 - \zeta \\ 0 & \text{otherwise} \end{cases}$$

The functions for the other sectors are defined as corresponding rotations of $\tilde{k}_0$:

$$\tilde{k}_i = \tilde{k}_0 \circ R_{-2\pi i/N}$$

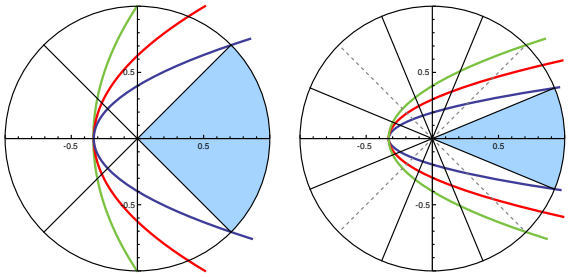
By normalizing $\tilde{k}_i$ and multiplying with G_σ, we get weighting functions defined over the unit disc:

$$\widetilde{K}_i(x,y) = \frac{\tilde{k}_i(x,y)}{\sum_{j=0}^{N-1} \tilde{k}_j(x,y)} \cdot G_\sigma(x,y)$$

As desired, the sum $\sum_j \widetilde{K}_i$ is equal to G_σ by construction, and the new weighting functions $\widetilde{w}_i$ are now defined as in the previous section by:

$$\widetilde{w}_i = \widetilde{K}_i \circ SR_{-\varphi}$$

The definition of $\tilde{k}_0$ includes two parameters ζ and η. The parameter ζ controls how much the different weighting functions overlap at the filter origin. For the anisotropic Kuwahara filter to work as expected, it is required that all weighting



(a) $N = 4$ (b) $N = 8$

Figure 8: *Zero-crossings of the polynomial* $(x+\zeta)-\eta y^2$ *for* $\zeta = \frac{1}{3}$ *and* $\eta = \eta\left(\zeta, \frac{\pi}{N}\right)$ *(green),* $\eta = \eta\left(\zeta, \frac{3\pi}{4N}\right)$ *(red),* $\eta = \eta\left(\zeta, \frac{2\pi}{N}\right)$ *(blue).*

functions overlap at their boundaries, especially at the filter origin. In the previous section, we explained that for $\alpha = 1$ the minor radius of the ellipse will not be smaller than half of the radius. Hence

$$\zeta = \frac{2}{r}$$

is a reasonable definition that ensures that all sectors overlap at the filter origin. The parameter η controls how much the sectors overlap at their sides. Given ζ and η, we can look at the zero-crossing of the polynomial $(x+\zeta)-\eta y^2$ (Figure 8), and vice versa we can define η dependent on ζ and the intersection of the zero-crossing with the unit circle:

$$\eta(\zeta,\gamma) = \frac{\zeta + \cos\gamma}{\sin^2\gamma}$$

In order for $\widetilde{K}_i$ to be well-defined on the unit disc, it is necessary that the sum $\sum_j \widetilde{K}_i$ is non-zero for every point of the unit disc. This can be achieved by requiring $\gamma \geq \frac{\pi}{N}$ (blue plot of Figure 8). Conversely, it is undesirable that more than two sectors overlap on one side. An approximate bound for this is $\gamma \leq \frac{2\pi}{N}$ (green plot of Figure 8). Hence, reasonable choices for η are:

$$\eta_{\min}(\zeta) = \eta\left(\zeta, \tfrac{2\pi}{N}\right) \leq \eta \leq \eta\left(\zeta, \tfrac{\pi}{N}\right) = \eta_{\max}(\zeta)$$

For all examples we use $r = 6$, $\zeta = \frac{1}{3}$ and

$$\eta\left(\tfrac{1}{3}, \tfrac{3\pi}{2N}\right) \approx \begin{cases} 0.84 & \text{if } N = 4 \\ 3.77 & \text{if } N = 8 \end{cases}.$$

Listing 1 shows how the proposed weighting functions can be implemented efficiently for $N = 8$. Here, it is important to use the `max` function. On GPUs this function is generally available as intrinsic function and much faster, since no branching is performed. For $N = 8$ the vector v has to be rotated by $\pi/4$. Since rotation by multiple of $\pi/2$ can be performed by swapping and negating coordinates, we perform the computation in two stages. We first calculate the weights

```
vec2 v = SR * vec2(x-x0, y-y0);
vec3 c = texture2D(src, vec2(x,y)).rgb;

float sum = 0;
float w[8];
float z, vxx, vyy;

vxx = zeta - eta * v.x * v.x;
vyy = zeta - eta * v.y * v.y;
z = max(0,  v.y + vxx);  sum += w[0] = z * z;
z = max(0, -v.x + vyy);  sum += w[2] = z * z;
z = max(0, -v.y + vxx);  sum += w[4] = z * z;
z = max(0,  v.x + vyy);  sum += w[6] = z * z;

v = sqrt(2)/2 * vec2( v.x - v.y, v.x + v.y );

vxx = zeta - eta * v.x * v.x;
vyy = zeta - eta * v.y * v.y;
z = max(0,  v.y + vxx);  sum += w[1] = z * z;
z = max(0, -v.x + vyy);  sum += w[3] = z * z;
z = max(0, -v.y + vxx);  sum += w[5] = z * z;
z = max(0,  v.x + vyy);  sum += w[7] = z * z;

float g = exp(-3.125 * dot(v,v)) / sum;

for (int k = 0; k < 8; ++k) {
    float wk = w[k] * g;
    m[k] += vec4(c * wk, wk);
    s[k] += c * c * wk;
}
```

Listing 1: *Pseudocode that shows how the proposed weighting functions can be efficiently evaluated inside the inner loop of the anisotropic Kuwahara filter for $N = 8$.*

for 0, $\pi/2$, π and $3/2\pi$. Then we rotate by $\pi/4$ and calculate the weights for $\pi/4$, $3/4\pi$, $5/4\pi$ and $7/4\pi$.

4. Discussion

In Figure 2, images are shown that have been processed with the anisotropic Kuwahara filter using the proposed weighting functions. The results are visually indistinguishable from the output of the anisotropic Kuwahara filter using the original weighting functions. As can be seen in Figure 9(b), the proposed approach creates the same feature-preserving direction-enhancing look as the original texture-based implementation. Minor differences (Figure 9(c)) appear in high-contrast areas. This is not surprising, since both filters do not match exactly. Because the primary aim of the filter is abstraction, these minor differences are completely irrelevant. However, this indicates that the proposed approach shares important properties of the original implementation as discussed in [KKD09]. These are for example prevention of overblurring in low-contrast areas (Figure 10(b)) and robustness against high-contrast noise (Figure 10(d)). We also tested application of the proposed approach to video. The proposed approach achieves the same outstanding temporal coherence with per-frame filtering.

Our implementation is based on the GLSL reference implementation of the anisotropic Kuwahara filter [KKD10].

Table 1: *Comparison with the GLSL reference implementation of the anisotropic Kuwahara filter [KKD10]. Image size: 512x512 pixels.*

GPU	Texture		Proposed			
	$N = 4$	$N = 8$	$N = 4$		$N = 8$	
NVidia FX570M	119.5 ms	557.9 ms	136.6 ms	-12.5%	579.6 ms	-3.7%
NVidia 9800 GT	28.3 ms	98.8 ms	31.5 ms	-10.2%	104.1 ms	-5.1%
NVidia 8800 GTX	27.5 ms	76.3 ms	30.6 ms	-10.1%	81.1 ms	-5.9%
NVidia GTX 285	10.8 ms	36.0 ms	11.9 ms	-9.2%	40.0 ms	-10.0%
NVidia GTX 480	5.6 ms	28.3 ms	6.1 ms	-8.0%	23.1 ms	**+22.5%**
ATI Radeon 4850	44.9 ms	74.7 ms	48.3 ms	-7.0%	90.8 ms	-17.7%
ATI Radeon 5850	17.9 ms	41.7 ms	18.5 ms	-3.2%	35.3 ms	**+18.1%**

Table 1 compares the execution times of our approach with the texture-based implementation. Using textures as lookup tables to speed up computations is a common technique used in real-time graphics. Furthermore, modern computer games heavily rely on texture mapping. GPU hardware vendors have optimized their hardware to support this. On modern GPUs memory access is, however, a critical bottleneck. This is clearly observable for next generation DirectX 11 hardware. Here the proposed approach outperforms the texture-based approach for $N = 8$, as two texture lookups are required per kernel element and the proposed approach therefore performs much better in that case. On the contrary, the texture-based approach is slightly faster on DirectX 10 generation hardware.

Our approach also offers new opportunities for optimized implementations on GPUs. GPU computing APIs such as CUDA and OpenCL offer a much finer control over the GPU and allow usage of advanced features such as shared memory. For Gaussian smoothing, it has been shown that a shared memory implementation outperforms a texture-based implementation [Pod07]. We expect a similar result for a shared memory implementation of our approach.

5. Conclusions

We have presented alternative weighting functions for the anisotropic Kuwahara filter. The proposed weighting functions do not require convolution and can be efficiently computed directly during the filtering process. Our approach creates output of similar quality as the texture-based implementation of the anisotropic Kuwahara filter. Even though the weighting functions are explicitly computed during the filtering process, our approach performs better on next generation GPUs when using eight sectors. The proposed weighting functions also contribute to a further generalization of the Kuwahara filter. Since the weights are calculated on the fly, this offers new opportunities to control the smoothing process interactively or automatically at real-time.

References

[BBL*06] Brox T., Boomgaard R., Lauze F., Weijer J., Weickert J., Mrázek P., Kornprobst P.: Adaptive

(a) Original (b) Proposed method (c) Difference to [KKD09]

Figure 9: *Directional image features are preserved and enhanced [KKD09, Figure 7].*

(a) Original (b) Proposed method (c) Original (d) Proposed method

Figure 10: *Left: Overblurring is avoided in low-contrast regions [KKD09, Figure 10]. Right: Application to an image with 2% Gaussian noise and 5% impulse noise [KKD09, Figure 11].*

structure tensors and their applications. *Visualization and Processing of Tensor Fields* (2006), 17–47. 2

[CM02] Comaniciu D., Meer P.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 5 (2002), 603–619. 1

[DS02] DeCarlo D., Santella A.: Stylization and abstraction of photographs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 769–776. 1

[KD08] Kyprianidis J. E., Döllner J.: Image abstraction by structure adaptive filtering. In *Proc. EG UK Theory and Practice of Computer Graphics* (2008), Eurographics Association, pp. 51–58. 1

[KHEK76] Kuwahara M., Hachimura K., Eiho S., Kinoshita M.: *Digital processing of biomedical images*. Plenum Press, 1976, pp. 187–203. 1

[KKD09] Kyprianidis J. E., Kang H., Döllner J.: Image and video abstraction by anisotropic kuwahara filtering. *Computer Graphics Forum 28*, 7 (2009), 1955–1963. Special issue on Pacific Graphics 2009. 1, 2, 5, 6

[KKD10] Kyprianidis J. E., Kang H., Döllner J.: Anisotropic kuwahara filtering on the GPU. In *GPU Pro - Advanced Rendering Techniques*, Engel W., (Ed.). AK Peters, 2010. 3, 5

[KLC09] Kang H., Lee S., Chui C. K.: Flow-based image abstraction. *IEEE Transactions on Visualization and Computer Graphics 15*, 1 (2009), 62–76. 1

[Pod07] Podlozhnyuk V.: *Image Convolution with CUDA*. Tech. rep., NVIDIA, 2007. 5

[PPC07] Papari G., Petkov N., Campisi P.: Artistic edge and corner enhancing smoothing. *IEEE Transactions on Image Processing 16*, 10 (2007), 2449–2462. 1, 2

[WOG06] Winnemöller H., Olsen S. C., Gooch B.: Real-time video abstraction. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (2006), pp. 1221–1226. 1

[YBFU96] Yang G. Z., Burger P., Firmin D. N., Underwood S. R.: Structure adaptive anisotropic image filtering. *Image and Vision Computing 14*, 2 (1996), 135 – 145. 2