QUASI-CONVOLUTION PYRAMIDAL BLURRING

Martin Kraus
Computer Graphics and Visualization Group, Technische Universität München, Boltzmannstr. 3, Garching, Germany

Keywords: Rendering, image processing, blurring, pyramid algorithm, GPU, real-time.

Abstract: Efficient image blurring techniques based on the pyramid algorithm can be implemented on modern graphics hardware; thus, image blurring with arbitrary blur width is possible in real time even for large images. However, pyramidal blurring methods do not achieve the image quality provided by convolution filters; in particular, the shape of the corresponding filter kernel varies locally, which potentially results in objectionable rendering artifacts. In this work, a new analysis filter is designed that significantly reduces this variation for a particular pyramidal blurring technique. Moreover, an efficient implementation for programmable graphics hardware is presented. The proposed method is named "quasi-convolution pyramidal blurring" since the resulting effect is very close to image blurring based on a convolution filter for many applications.

1 INTRODUCTION AND RELATED WORK

As the programmability of graphics processing units (GPUs) allows for the implementation of increasingly complex image processing techniques, many effects in real-time rendering are nowadays implemented as post-processing effects. Examples include tone mapping (Goodnight et al., 2003), glow (James and O'Rorke, 2004), and depth-of-field rendering (Demers, 2004; Hammon, 2007; Kraus and Strengert, 2007a). Many of these real-time effects require extremely efficient image blurring; for example, depth-of-field rendering is often based on multiple blurred versions of a pinhole image. Thus, full-screen images have to be blurred with potentially large blur filters multiple times per frame at real-time frame rates.

Unfortunately, convolution filters cannot provide the required performance for large blur filters and the Fast Fourier Transformation (FFT) is not efficient enough for large images. As shown by Burt (Burt, 1981), the pyramid algorithm provides a better complexity than the FFT for blurring; therefore, many real-time depth-of-field rendering techniques employ pyramid methods in one way or another. For example, Demers (Demers, 2004) uses a mip map (Williams, 1983) to generate multiple, downsampled, i.e., pre-filtered, versions of a pinhole image. Hammon (Hammon, 2007) computes only one downsampled level to accelerate the blurring with filters of medium size, while Kraus and Strengert (Kraus and Strengert, 2007a) employ a full pyramid algorithm for the blurring of sub-images, which are computed by a decomposition of a pinhole image according to the depth of its pixels.

The specific analysis filters and synthesis filters for the pyramid algorithm are often determined by trial-and-error, i.e., the filter size is increased at the cost of memory bandwidth until a sufficient image quality is achieved. A more thorough exploration of appropriate filter designs and their efficient implementation on GPUs has been provided by Kraus and Strengert (Kraus and Strengert, 2007b), which improved the pyramidal blurring on GPUs presented by Strengert et al. (Strengert et al., 2006). This improved method is summarized in Section 2.

The first contribution of this work is a quantitative analysis of the filters proposed by Kraus and Strengert by means of response functions in Section 3, which reveal strong local variations of the corresponding blur filter due to the grid structure of the image pyramid. This shortcoming can result in objectionable rendering artifacts; for example, it causes pulsating artifacts if a moving pixel (or a small group of consistently moving pixels) of high contrast is blurred in an animated sequence since the blur depends on the pixel's position within the image.

To overcome this deficiency of pyramidal blurring, a new analysis filter is designed in Section 4, which is the second contribution of this work. It reduces the variations of the corresponding blur filter considerably—in particular the variation of its maximum amplitude. Thus, the pyramidal blurring proposed in this work is significantly closer to blurring

Kraus M. (2008).
QUASI-CONVOLUTION PYRAMIDAL BLURRING.
In *Proceedings of the Third International Conference on Computer Graphics Theory and Applications*, pages 155-162
DOI: 10.5220/0001094401550162

by a convolution filter and is therefore called "quasi-convolution pyramidal blurring."

In addition to the two mentioned contributions, an efficient GPU implementation of the new analysis filter is described in Section 5, while some experiments demonstrating the benefits of the proposed method are presented in Section 6.

2 PYRAMIDAL BLURRING

Image blurring with the pyramid algorithm was first suggested by Burt (Burt, 1981). In the first part of the method, called analysis, an image pyramid of downsampled or reduced image levels is computed by applying a (usually small) analysis filter mask and subsampling the result by a factor of 2 in each dimension. In the second part of the method, called synthesis, one of the levels is chosen based on the specified blur width. The coarse image of the chosen level is iteratively upsampled to the original dimensions by applying a synthesis filter. Figure 1 illustrates this method for a one-dimensional image of 16 pixels.

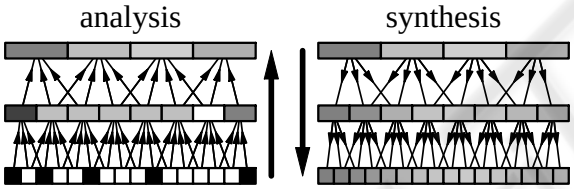


Figure 1: Illustration of pyramidal blurring in 1D.

An efficient GPU implementation of this algorithm was presented by Strengert et al. (Strengert et al., 2006) for a 2×2 box analysis filter and a synthesis filter that corresponds to filtering the coarse image by a biquadratic B-spline filter. The resulting image quality can be improved by applying a 4×4 box analysis filter or an analysis filter corresponding to a biquadratic B-spline filter as suggested by Kraus and Strengert (Kraus and Strengert, 2007b). While this improvement often results in an acceptable image quality when blurring static images, rendering artifacts become visible in animations since the proposed pyramidal blur deviates significantly from blurring by convolution filtering, i.e., the blur varies depending on the image position.

In this work, the deviation from convolution filters is quantitatively analyzed and a new analysis filter is designed that allows for an efficient GPU implementation while minimizing the deviation from a convolution filter. We employ the synthesis filter proposed by Strengert et al. since biquadratic B-spline filtering offers several interesting features such as compact support, C^1 continuity, similarity to a Gaussian distribution function and therefore almost radial symmetry, and the possibility of an efficient implementation based on bilinear interpolation (Strengert et al., 2006).

3 QUANTITATIVE ANALYSIS OF RESPONSE FUNCTIONS

In order to analyze the deviation of pyramidal blurring from convolution filtering, we consider the continuous limit case of infinitely many downsampling and upsampling steps; thus, the "pixels" of the input image are infinitely small. Without loss of generality, the size of a pixel of the coarsest image level, which is used as input for the synthesis, is set to 1 and the sampling positions of these pixels are positioned at integer coordinates. We discuss only one-dimensional grayscale images since the extension to two-dimensional color images is straightforward for separable filters and linear color spaces.

The limit of infinitely small input pixels allows us to define continuous response functions for a black input image with a single, infinitely small intensity peak at position $p \in \mathbb{R}$ in a coordinate system where the pixels of the coarsest image level are at integer coordinates. We distinguish between two kinds of response functions: the first is denoted by $\varphi_i(p)$ and specifies the intensity of a pixel of the coarsest image level at integer position $i \in \mathbb{Z}$ after downsampling the input image with a peak at position p.

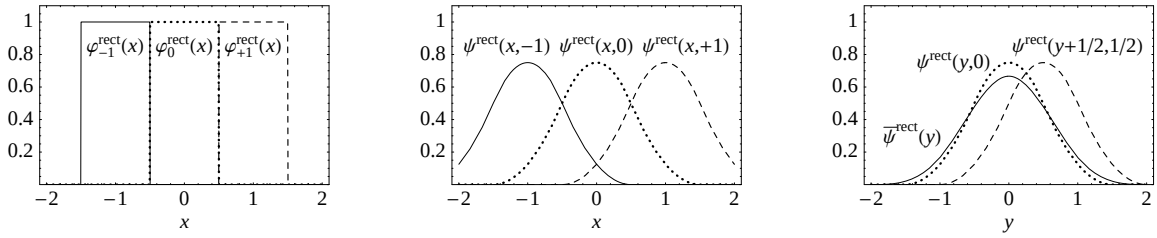
The second kind of response functions is denoted by $\psi(x,p)$ and specifies the intensity of the blurred image (of infinitely high resolution) at position $x \in \mathbb{R}$ for a peak at position $p \in \mathbb{R}$. In this work, the blurred image is always obtained by filtering the coarsest image level by a quadratic B-spline. We denote the quadratic B-spline function centered at i by $\varphi_i^{\text{quad}}(x)$ (see Equation 5 for its definition); thus, $\psi(x,p)$ is defined as the sum over all pixels of the product of the response function for the i-th pixel $\varphi_i(p)$ with the quadratic B-spline $\varphi_i^{\text{quad}}(x)$.

$$\psi(x,p) = \sum_i \varphi_i(p)\varphi_i^{\text{quad}}(x) \tag{1}$$

With the help of these definitions we compute $\varphi_i(p)$ and $\psi(x,p)$ for three analysis filters discussed by Kraus and Strengert (Kraus and Strengert, 2007b).

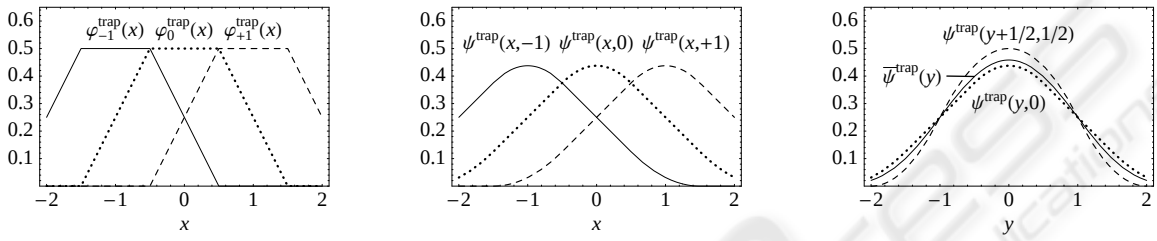
The analysis filter mask for the 2-tap box filter is $\frac{1}{2}(1\,1)$; thus, the corresponding response function for the i-th pixel of the coarsest image level is a simple rectangle function denoted by $\varphi_i^{\text{rect}}(p)$ and depicted in Figure 2a.

$$\varphi_i^{\text{rect}}(p) = \begin{cases} 1 & \text{if } \; i-\frac{1}{2} < p < i+\frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$



(a) (b) (c)

Figure 2: Response functions for the 2-tap box analysis filter $\frac{1}{2}(1\,1)$.



(a) (b) (c)

Figure 3: Response functions for the 4-tap box analysis filter $\frac{1}{4}(1\,1\,1\,1)$.

The corresponding response function $\psi^{\text{rect}}(x,p)$ for the blurred image is a quadratic B-spline centered at the integer coordinate closest to the peak position p.

$$\psi^{\text{rect}}(x,p) = \varphi^{\text{quad}}_{\lfloor p+\frac{1}{2}\rfloor}(x) \tag{3}$$

This function is illustrated in Figure 2b.

In the case of the 4-tap box analysis filter with the filter mask $\frac{1}{4}(1\,1\,1\,1)$, the shape of the response function for the i-th pixel is a trapezoid as illustrated in Figure 3a; therefore, the response function is denoted by $\varphi_i^{\text{trap}}(p)$.

$$\varphi_i^{\text{trap}}(p) = \begin{cases} \frac{1}{2}(p-i+\frac{3}{2}) & \text{if} \quad i-\frac{3}{2} < p < i-\frac{1}{2} \\ \frac{1}{2} & \text{if} \quad i-\frac{1}{2} \le p \le i+\frac{1}{2} \\ \frac{1}{2}(i+\frac{3}{2}-p) & \text{if} \quad i+\frac{1}{2} < p < i+\frac{3}{2} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The corresponding response function for the blurred image is denoted by $\psi^{\text{trap}}(x,p)$ and illustrated in Figure 3b for integer values of p. It should be noted that non-integer values of p result in different shapes as illustrated in Figure 3c for $p = 1/2$.

For the 4-tap analysis filter mask $\frac{1}{8}(1\,3\,3\,1)$, the response function for the i-th pixel is a quadratic B-spline, which is denoted by φ_i^{quad} and illustrated in Figure 4a.

$$\varphi_i^{\text{quad}}(p) = \begin{cases} \frac{1}{2}\left(p-i+\frac{3}{2}\right)^2 & \text{if} \quad i-\frac{3}{2} < p < i-\frac{1}{2} \\ \frac{3}{4}-(p-i)^2 & \text{if} \quad i-\frac{1}{2} \le p \le i+\frac{1}{2} \\ \frac{1}{2}\left(i+\frac{3}{2}-p\right)^2 & \text{if} \quad i+\frac{1}{2} < p < i+\frac{3}{2} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Correspondingly, the response function for the blurred image is denoted by $\psi^{\text{quad}}(x,p)$. An illustration for integer values of p is given in Figure 4b.

In order to compare the response functions $\psi(x,p)$, which depend on x and p, with convolution filters that only depend on the difference $y \stackrel{\text{def}}{=} x-p$, we define an averaged response function $\overline{\psi}(y)$ by integration over p.

$$\overline{\psi}(y) = \int_0^1 \mathrm{d}p\, \psi(y+p,p) \tag{6}$$

The corresponding functions $\overline{\psi}^{\text{rect}}(y)$, $\overline{\psi}^{\text{trap}}(y)$, and $\overline{\psi}^{\text{quad}}(y)$ are illustrated in Figures 2c, 3c, and 4c.

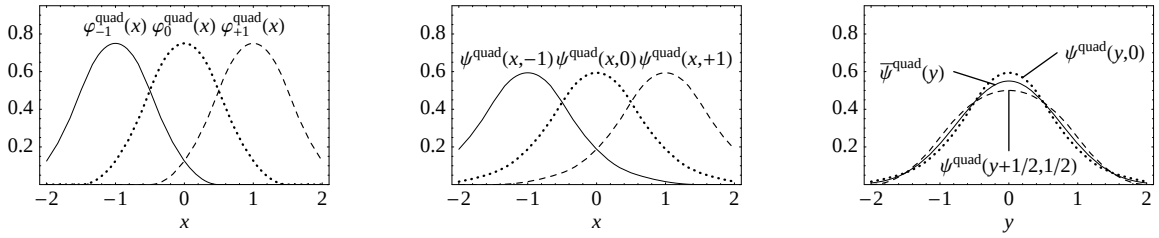
With the help of $\overline{\psi}(y)$ the deviation of a particular pyramidal blurring method from convolution blurring can be quantified by computing the root mean square deviation (RMSD), denoted by ε, between the response function $\psi(x,p)$ and $\overline{\psi}(x-p)$.

$$\varepsilon = \sqrt{\int_0^1 \mathrm{d}p \int_{-\infty}^{+\infty} \mathrm{d}x\, \left(\psi(x,p)-\overline{\psi}(x-p)\right)^2} \tag{7}$$

Additionally, we consider the RMSD between $\psi(p,p)$ and $\overline{\psi}(0)$, denoted by ε_0, since a variation of the maximum amplitude of a blur filter is more easily perceived than a variation at other positions and all averaged response functions $\overline{\psi}(y)$ considered in this work achieve their maxium for $y = 0$.

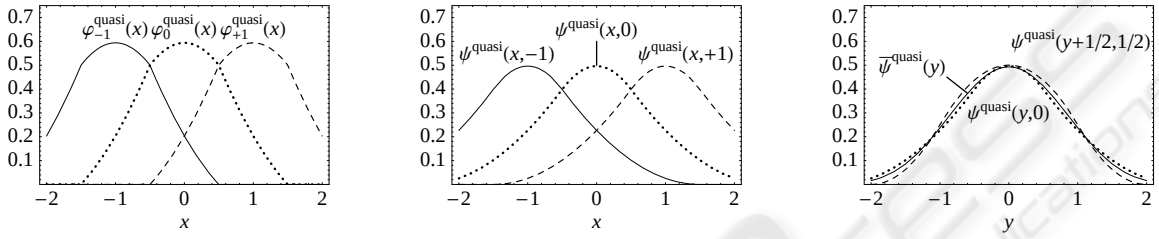
$$\varepsilon_0 = \sqrt{\int_0^1 \mathrm{d}p\, \left(\psi(p,p)-\overline{\psi}(0)\right)^2} \tag{8}$$

Actual values of ε and ε_0 for $\psi^{\text{rect}}(x,p)$, $\psi^{\text{trap}}(x,p)$, and $\psi^{\text{quad}}(x,p)$ are given in Table 1. Due



(a) (b) (c)

Figure 4: Response functions for the quadratic analysis filter $\frac{1}{8}(1\,3\,3\,1)$.



(a) (b) (c)

Figure 5: Response functions for the proposed quasi-convolution analysis filter $1/64\ (13\ 19\ 19\ 13)$.

to the strong deviation of $\psi^{rect}(x,p)$ from a convolution filter, it is rather unsuited for pyramidal blurring as already observed by Kraus and Strengert. Interestingly, $\psi^{quad}(x,p)$ provides no improvement compared to $\psi^{trap}(x,p)$ although $\varphi_i^{quad}(p)$ is C^1 continuous while $\varphi_i^{trap}(p)$ is only C^0 continuous.

Table 1: RMS deviation ε of response functions from an averaged filter and the RMSD ε_0 at the center of the filter.

analysis mask	response function	ε	ε_0
$\frac{1}{2}(0\,1\,1\,0)$	$\psi^{rect}(x,p)$	0.2658	0.0745
$\frac{1}{4}(1\,1\,1\,1)$	$\psi^{trap}(x,p)$	0.0376	0.0186
$\frac{1}{8}(1\,3\,3\,1)$	$\psi^{quad}(x,p)$	0.0510	0.0327
$\frac{1}{64}(13\,19\,19\,13)$	$\psi^{quasi}(x,p)$	0.0276	0.0027

4 QUASI-CONVOLUTION PYRAMIDAL BLURRING

In order to design a pyramidal blurring method that produces a blur that is visually similar to convolution filtering, we try to minimize ε and ε_0 defined in Equations 7 and 8 under several constraints; in particular, we will employ the synthesis filter corresponding to quadratic B-spline filtering. Moreover, we consider only symmetric 4-tap analysis filter masks; i.e., filter masks of the form $(a\ (1/2-a)\ (1/2-a)\ a)$.

By numeric methods we determined the minimum of ε under these constraints for a approximately equal to $13/64$; i.e., for the analysis filter mask $1/64\ (13\ 19\ 19\ 13)$. The minimum of ε_0 is achieved for a slightly larger value of a; however, the potential improvement is less than 5%; thus, we will neglect it in this work. We call the corresponding blurring method "quasi-convolution pyramidal blurring" since this analysis filter reduces ε and ε_0 significantly as shown in Table 1. Of particular interest is the strong decrease of ε_0, which is almost an order of magnitude smaller than for previously suggested pyramidal blurring methods.

It is an interesting feature of the analysis filter mask for quasi-convolution pyramidal blurring that it can be constructed by a linear combination of the 4-tap box filter and the analysis filter mask for quadratic B-splines:

$$\frac{1}{64}(13\,19\,19\,13) = \frac{5}{8}\times\frac{1}{4}(1\,1\,1\,1)+\frac{3}{8}\times\frac{1}{8}(1\,3\,3\,1). \quad (9)$$

Therefore, the response function $\psi^{quasi}(x,p)$ can be computed as the same linear combination of the corresponding response functions due to the linearity of the pyramid method:

$$\psi^{quasi}(x,p) = \frac{5}{8}\times\psi^{trap}(x,p)+\frac{3}{8}\times\psi^{quad}(x,p). \quad (10)$$

The response functions $\varphi_i^{quasi}(x)$ and $\psi^{quasi}(x,p)$ are illustrated in Figures 5a and 5b while $\overline{\psi}^{quasi}(y)$ is depicted in Figure 5c. In comparison to Figures 3c and

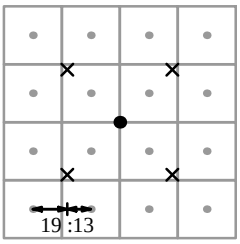


Figure 6: Illustration of the positions (crosses) of four bilinear texture lookups for the quasi-convolution analysis filter. The centers of texels of the finer level are indicated by grey dots, while the black dot indicates the center of the processed texel of the coarser image level.

4c, a strong reduction of the deviation of $\psi^{\text{quasi}}(y,0)$ and $\psi^{\text{quasi}}(y+1/2,1/2)$ from $\overline{\psi}^{\text{quasi}}(y)$ is obvious.

It should be noted that our results depend on several constraints, in particular the width of the analysis filter and the particular synthesis filter, which were both chosen to allow for an efficient GPU implementation as discussed in the next section. Wider analysis and synthesis filters are likely to allow for even smaller values of ε and ε_0, however, at higher computational costs at run time.

5 GPU IMPLEMENTATION

Sigg and Hadwiger (Sigg and Hadwiger, 2005) have proposed an efficient way of exploiting GPU-supported bilinear interpolation for cubic B-spline filtering. We can employ an analogous technique to compute the analysis filter mask $1/64\,(\,13\;19\;19\;13\,)$ by two linear interpolations. To this end the position of the first linear filtered texture lookup has to be placed between the first and second pixels at distances in the ratio 19:13 and the second lookup between the third and fourth at distances in the ratio 13:19. The mean of the two texture lookups is the correctly filtered result in the one-dimensional case.

For two-dimensional images, the analysis filter mask for quasi-convolution blurring is constructed by a tensor product of the one-dimensional filter mask:

$$\frac{1}{4096}\begin{pmatrix} 169 & 247 & 247 & 169 \\ 247 & 361 & 361 & 247 \\ 247 & 361 & 361 & 247 \\ 169 & 247 & 247 & 169 \end{pmatrix} \tag{11}$$

In this case, four bilinear texture lookups are necessary. Similarly to the one-dimensional case, the positions are placed at distances in the ratio 19:13 in horizontal and vertical direction, where the pixels closer to the center of the filter mask are also closer to the positions of the texture lookups. These positions (according to the OpenGL conventions for texture coordinates) are illustrated in Figure 6. The filtered result is computed by the mean of the four texture lookups.

Table 2: Timings for blurring a 1024×1024 4×16-bit RGBA image on a GeForce 7900 GTX.

pyramid	analysis filter		
levels	2×2 box	4×4 box	quasi-conv.
1	0.40 ms	0.57 ms	0.85 ms
2	0.65 ms	0.82 ms	1.27 ms
3	0.72 ms	0.89 ms	1.39 ms
4	0.76 ms	0.92 ms	1.44 ms
5	0.77 ms	0.94 ms	1.47 ms
6	0.78 ms	0.95 ms	1.49 ms
7	0.80 ms	0.96 ms	1.51 ms

For comparison, we also discuss implementations of the 2×2 box analysis filter, the 4×4 box analysis filter, and the biquadratic analysis filter. The 2×2 box filter mask can be implemented very efficiently by a single bilinear texture lookup positioned equidistantly between the centers of the four relevant texels. The most efficient way to implement the 4×4 box filter mask is a two-pass method with only two bilinear texture lookups (Kraus and Strengert, 2007b). For the biquadratic analysis filter, a variant of the presented implementation of the quasi-convolution analysis filter with adapted positions appears to provide the best performance. Thus, the biquadratic analysis filter and the quasi-convolution analysis filter achieve the same performance.

Measured timings for these implementations are summarized in Table 2 for blurring a 1024×1024 image. The number of pyramid levels determines the width of the blur; it corresponds to the number of performed analysis steps, which is equal to the number of synthesis steps. The employed synthesis filter corresponds to biquadratic B-spline filtering and can be implemented with only one bilinear lookup (Strengert et al., 2006).

6 RESULTS

Figures 7 to 10 illustrate the pyramidal blurring of an antialiased line by two pyramid levels. Analogously to Figure 1, the two downsampling steps of the analysis are depicted on the left-hand-side (bottom up) while the two upsampling steps of the synthesis are shown on the right-hand-side (top down). Thus, the blurred result is shown in the lower, right image

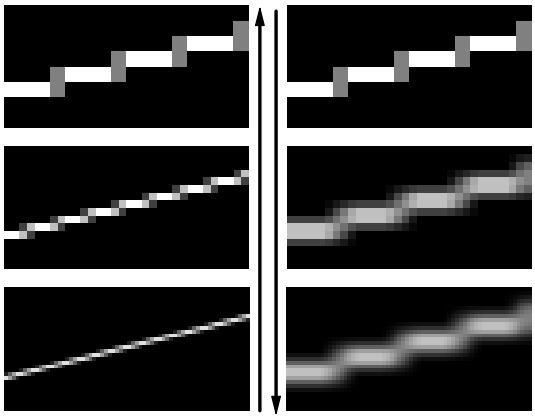


Figure 7: Pyramidal blurring of an antialiased line with the 2×2 box analysis filter.

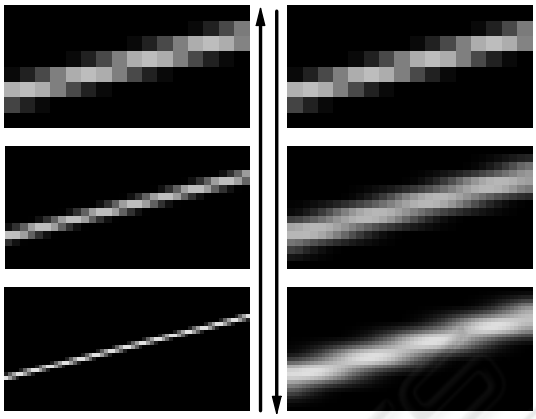


Figure 9: Pyramidal blurring of an antialiased line with the biquadratic analysis filter.

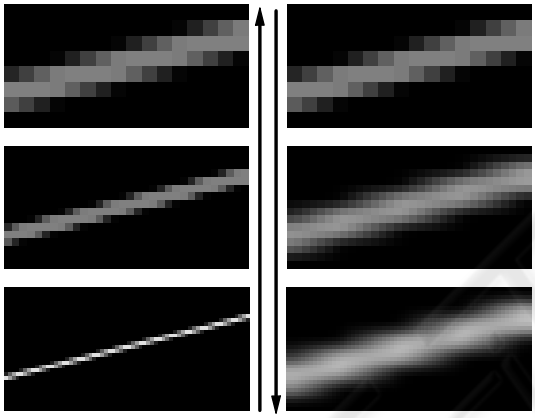


Figure 8: Pyramidal blurring of an antialiased line with the 4×4 box analysis filter.

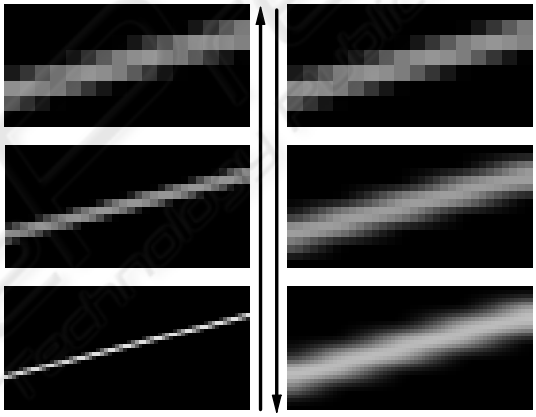


Figure 10: Pyramidal blurring of an antialiased line with the quasi-convolution analysis filter.

of each figure. Linear intensity scaling was employed to enhance the images; however, the same scaling of intensities was employed in corresponding images of Figures 7, 8, 9, and 10.

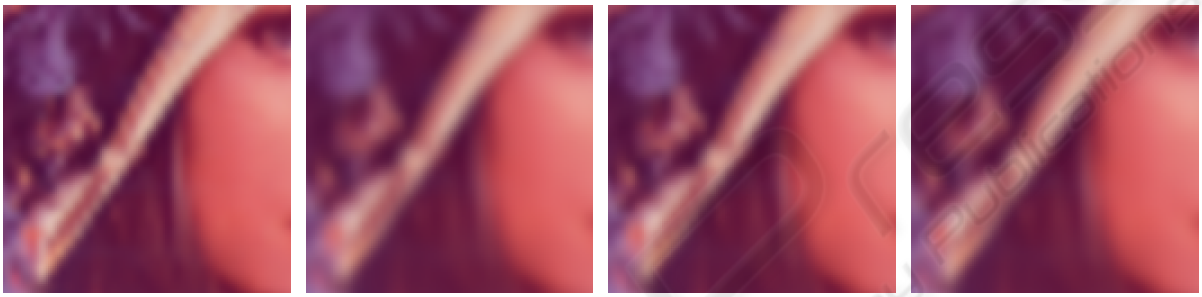
Blurring with the 2×2 box analysis filter in Figure 7 results in strong staircasing artifacts in the lower, right image. The biquadratic analysis filter employed in Figure 9 also results in a clearly visible oscillation of the blurred line's intensity. Similar oscillations also occur in animations, where they are often more objectionable since their position is aligned with the pyramidal grid, i.e, they often result in fixed-pattern distortions of the processed images.

The 4×4 box filter employed in Figure 8 and the quasi-convolution filter used in Figure 10 produce significantly better results than the biquadratic analysis filter. Unfortunately, the employed linear intensity scaling cannot reveal the differences between Figures 8 and 10. Therefore, additional nonlinear contrast enhancement was employed in Figure 13 to compare the resulting images of the blurred line. The left image in Figure 13 reveals an oscillation of intensity for the 4×4 box filter, while the line blurred with quasi-convolution in the right image of Figure 13 shows almost no such oscillation for the same image enhancement settings. Although nonlinear image enhancement is necessary to show these differences, their relevance should not be underestimated since several image post-processing techniques (e.g., for tone mapping) use blurred intermediate images in nonlinear computations; thus, even small-scale artifacts can become objectionable.

To compare quasi-convolution blurring with actual convolution filtering, Figure 14 shows the two-dimensional convolution of the image of an antialiased line with the averaged filter $\bar{\psi}^{\text{quasi}}$ of quasi-

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

Figure 11: Detail of a Manga image (http://commons.wikimedia.org/wiki/Image:Manga.png) blurred with (a) the 2×2 box analysis filter, (b) the 4×4 box analysis filter, (c) the biquadratic analysis filter, and (d) the quasi-convolution analysis filter.



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

Figure 12: Detail of the 512×512 Lenna image blurred with (a) the 2×2 box analysis filter, (b) the 4×4 box analysis filter, (c) the biquadratic analysis filter, and (d) the quasi-convolution analysis filter.

convolution blurring.

Figures 11 and 12 show details of actual images, which were blurred by the presented methods. While the staircasing artifacts generated by the 2×2 box analysis filter are clearly visible in Figures 11a and 12a, the artifacts generated by the quadratic analysis filter in Figures 11c and 12c are less obvious. Applying the 4×4 box analysis filter or the quasi-convolution filter results in even less artifacts, which are usually not perceivable in static images such as Figures 11b, 11d, 12b, and 12d. However, they are more easily perceived if the blurred image is translated with respect to the pyramidal grid in an animation.

This comparison approves the results of our quantitative analysis in Section 3; in particular, the biquadratic analysis filter appears to provide no advantages in comparison to the 4×4 box filter or the quasi-convolution filter. The improved image quality provided by the quasi-convolution filter compared to the 4×4 box filter appears to be less relevant unless the differences are amplified by non-linear effects; for example, by tone mapping techniques for high-dynamic range images.

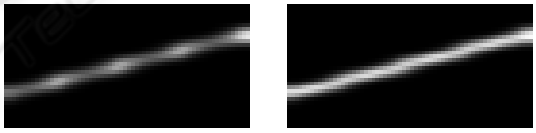


Figure 13: Contrast-enhanced blurred images for the 4×4 box filter (left) and the quasi-convolution filter (right).

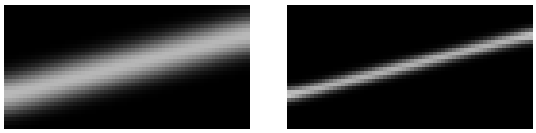


Figure 14: Convolution filtering corresponding to the average quasi-convolution blur depicted in Figure 10. The right image has been contrast-enhanced in the same way as the images in Figure 13.

7 CONCLUSIONS AND FUTURE WORK

This work introduces quasi-convolution pyramidal blurring; in particular, a new analysis filter is proposed and quantitatively compared to existing filters. This comparison shows that the proposed filter significantly reduces deviations of pyramidal blurring from

the corresponding convolution filter. Furthermore, an efficient implementation on GPUs has been demonstrated. The proposed pyramidal blurring method can be employed in several image post-processing effects in real-time rendering to improve the performance, image quality, and/or permissible blur widths. Therefore, more and better cinematographic effects can be implemented by means of real-time rendering.

In the future, the quantitative analysis should be extended to other synthesis filters, in particular C^2-continuous cubic B-splines, which might allow for even smaller deviations from convolution filters. Moreover, a generalization of the proposed pyramidal blurring technique to approximate arbitrary convolution filters would allow us to automatically replace convolution filters in existing image processing techniques.

REFERENCES

Burt, P. J. (1981). Fast Filter Transforms for Image Processing. *Computer Graphics and Image Processing*, 16:20–51.

Demers, J. (2004). Depth of Field: A Survey of Techniques. In Fernando, R., editor, *GPU Gems*, pages 375–390.

Goodnight, N., Wang, R., Woolley, C., and Humphreys, G. (2003). Interactive Time-Dependent Tone Mapping Using Programmable Graphics Hardware. In *Proceedings Rendering Techniques 2003*, pages 26–37.

Hammon, E. (2007). Practical Post-Process Depth of Field. In Nguyen, H., editor, *GPU Gems 3*, pages 583–605.

James, G. and O'Rorke, J. (2004). Real-Time Glow. In Fernando, R., editor, *GPU Gems*.

Kraus, M. and Strengert, M. (2007a). Depth-of-Field Rendering by Pyramidal Image Processing. *Computer Graphics Forum (Conference Issue)*, 26(3):645–654.

Kraus, M. and Strengert, M. (2007b). Pyramid Filters Based on Bilinear Interpolation. In *Proceedings GRAPP 2007 (Volume GM/R)*, pages 21–28.

Sigg, C. and Hadwiger, M. (2005). Fast Third-Order Texture Filtering. In Pharr, M., editor, *GPU Gems 2*, pages 313–329.

Strengert, M., Kraus, M., and Ertl, T. (2006). Pyramid Methods in GPU-Based Image Processing. In *Proceedings Vision, Modeling, and Visualization 2006*, pages 169–176.

Williams, L. (1983). Pyramidal Parametrics. In *Proceedings ACM SIGGRAPH '83*, pages 1–11.