Pyramid of Filters — Fast Image Filtering without FFT*

Sergey Ershov[0000−0002−5493−1076], Ildar Valiev[0000−0003−2937−8480], and Alexey Voloboy[0000−0003−1252−8294]

Keldysh Institute of Applied Mathematics RAS, Miusskaya sq. 4, 125047 Moscow, Russia
{ersh,valiev,voloboy}@gin.keldysh.ru

Abstract. Methods of computer graphics which had already become common in design of new optical systems and materials find currently new applications such as stomatology and ophthalmology. Some modern imaging systems are now designed in conjunction with the human vision system which is at their end. As a result simulation of the effects of human vision becomes necessary. These include partial defocusing and resulting "blur" of image, scattering and halo/corona and so on. Such effects are usually simulated convolving the original, "ideal" image with the pixel spread function. The latter frequently has size about that of the source image, so straightforward calculation of convolution would take a giant number of operations. Therefore in case of high resolution a decent speed is usually achieved by using the Fast Fourier Transform (FFT) for convolution, but since FFT operates periodic functions on a lattice with resolution being an integer power of a prime numbers, the required working resolution may considerably increase that of the original image and required memory becomes inadmissible. This paper presents an alternative method that allows calculations in much smaller memory avoiding overheads introduced by FFT requirements.

Keywords: Optical Simulation · Imaging System Design · Human Vision Effects · Image Processing · Filters · Convolution.

1 Introduction

Methods of computer graphics which had already become common in design of new optical systems and materials, are now applied in stomathology and ophthalmology. Some imaging systems are now designed in conjunction with the human vision system which is at their end. Simulation of many effects of human vision, as well as defocusing and aberration of lenses etc use 2D convolution of the original, "ideal" image with a kernel for the pixel spread function [1]:

$$\bar{f}(x,y) = \sum_{x',y'} K(x'-x, y'-y) f(x',y') \tag{1}$$

where (x,y) are the pixel coordinates, f is the source image, and $K(x,y)$ is the filter's kernel. Summation goes over the neighborhood of the "target" pixel (x,y) where the kernel is not 0.

* Supported by RFBR, grants 18-01-00569, 18-31-20032 and 19-01-00435.

In simulation of such human vision effects as halo and glare [2–6] the size of filter (read: of its support) can be about the size of the source image. For HD photos and other high-precision means this the image size can be as huge as even 40000x25000 pixels. Obviously, if for each of about a *billion* of target pixels calculate a sum of about a *billion* terms, the speed of such direct calculation will be below any expectation.

Since the kernel depends only on the *difference* between the source and target coordinates, efficiency can be drastically improved using Fast Fourier Transform (FFT) [7], with which the number of operations is about the number of pixels.

However FFT is not good regarding memory. First, FFT operates periodic functions (both input, kernel and the result) so that the boundaries of the image are *glued*. As a result, the FFT convolution to the black source image with the only white pixel at (0,0) produces filtered image which is gray not only near (0,0) but also near the opposite end $(x = N_x, y = N_y)$. To avoid this *artifact* one has to add "margins" of size at least twice filter radius and pad the source image to 0 there.

Meanwhile, as said above, in human vision simulation that radius is about the image size, thus we apply FFT convolution to the *thrice* larger image, 120000x75000 pixels in the above example. And this is not the end. The usual FFT is radix-2, i.e. the resolution must be an integer power of *two*. Although there do exist radix-3, 5 and other variants [7], this exotic is less efficient. The nearest *larger* power of two for 120000x75000 is already 131072x131072 pixels.

Since the number of FFT operations is nearly linear in the total number of pixels, it still remains more or less admissible, and modern PCs can do that in half an hour or hour for 3 color components. The really critical thing is though *memory* because 131072x131072 pixels, times 3 color channels is already 192 Gb. One must keep both the source image, the filtered image and the kernel at a time, *all* of that resolution. Plus several auxiliary arrays etc. So for our example this totals to more that 512 Gb which is *rarely* available nowadays.

Meanwhile, the source image is just about 12 Gb so all this huge memory is merely the FFT *overhead*. Happily, for symmetric kernels it is possible to design an alternative method that works in much less memory while still has decent speed. This paper describes two such algorithms (that can be also combined).

2 Pyramid of resolutions

This is closely related to the well-known *Level Of Detail* or *mipmap* [1] or *multigrids* in finite-difference methods. Decomposition into a series of images with decreasing resolution (based on Laplace pyramid) was even used in simulation of human vision [8]. Other alternatives to FFT such as wavelets etc are discussed in [9]. We shall proceed from the multigrid idea, though differently, and obtain a pyramid of *filters* (and filtered images). The idea of changing the resolution is much similar to that in the well-known image pyramid [1].

The kernel for human vision is rather sophisticated [4, 6] but is mainly a weighted sum of isotropic components like

$$K(r) = \frac{1}{(1 + (r/a)^2)^\beta} \tag{2}$$

where $r \equiv \sqrt{x^2 + y^2}$ is the *distance* between the source and target pixels. The spread a can be both small (less than a pixel) for some components, and large for other components. As a result, for small r the kernel changes very steeply, while at large r it inevitably changes more and more slowly.

2.1 Intentions

To utilize that let us decompose our filter into the sum of the "central" and "peripheral" parts, so that the former be distinct from zero for only, say, $r \leq 50$ while the "periphery", which can be not 0 *everywhere*, be *smooth* (slowly changing) enough.

The convolution with the original filter is then the sum of convolutions with the "central" and "peripheral" sub-kernels. The former, because of the small filter radius, can be calculated even *directly*, without any FFT and thus in its original resolution. It requires about 100^2 operations per pixel which is a lot but still admissible for modern PCs.

As to the *smooth* peripheral part, it is insensitive to the fine-scale detail of the source image. And the filtered image is smooth. So we can calculate it in *decreased resolution* (say tenfold) and then interpolate the result to the original resolution. And in this highly reduced resolution we can already apply FFT because the overhead is admissible.

Alternatively, we can repeat the procedure, now decomposing the peripheral component into again the "central part of periphery" and "periphery of periphery". Again the central part will be limited to say 50 pixels and the peripheral will be so smooth that admits already a 10-fold reduction of resolution (as compared to level 1, so already $10^2 = 100$ as compared to the *original* resolution!). In our example this is merely 250x400 so the filter, even if it covers the whole image area, can be applied *directly* (without FFT at all). So in this case we proceed without FFT at all levels.

The first central component is termed the *first level of pyramid*. The "central part of periphery" is the *second level of pyramid*. And "periphery of periphery" constitutes the *third level* (which in principle can be subdivided that sample manner leading to the *fourth level* and further). How this pyramid looks in reality is shown in Figure 1.

While convolving with the 1st level, we work in the original resolution and keep the source and the processed images which totals to about 24 Gb. The memory to keep the filter of 50 pixels radius is negligible.

At the second level, all is the same only for the tenfold reduced resolution, so it needs merely about $24/10^2$ Gb $= 240$ Mb that is negligible. The third level takes even less.

So the total memory and speed are determined by the first level and are admissible.

2.2 Real reduction of resolution

In the above reasoning we spoke about *tenfold* reduction of resolution which is surely imaginary. In reality reduction ratio is determined by the desired *accuracy* (which is in turn determined by the kernel and requirements to the "smoothness" of periphery), and is usually just 3. That is, resolution of the image processed at the next level is 3 times lower than that at the current level, and each pixel of the image at the next level

"comprises" a group of 3x3=9 current level pixels. In the simplest case, the value stored in that pixel is just the *sum* over that group (i.e. 9 pixels of the current level), though later we shall also use another relations). The first 3 levels of pyramid for one of the filters used in simulation of human vision acuity [4] is shown in Figure 1.

For the lower reduction ratio we shall need more levels to reach finally so low resolution that can be processed directly (without FFT). It can be already 7.

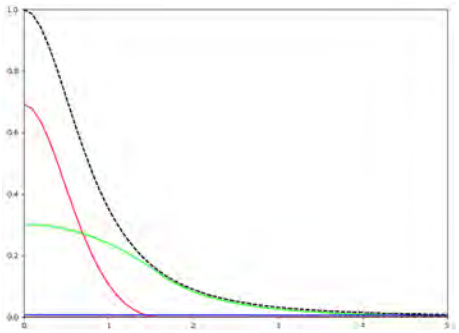

Fig. 1. Decomposition of the kernel $K(r) = (1+(10r)^2)^{-3/2}$. Dashed line is the original kernel. Red is the first level (central part in the original resolution), green is the second level (thrice lower resolution), blue and further (indistinguishable) — the third level and further. The sum over all levels gives *exactly* the original kernel.

The radius of the central part which in our idea was said to be 50 pixels is, surely, not that round. It is *calculated* from the criterion: *outside* that area the function changes so gradually that the inaccuracy of the linear interpolation from the 3fold reduced resolution is below allowed threshold. For human vision kernel, it is usually somewhere between 7 thru 25 pixels depending on other parameters.

All that and other details are explained below.

2.3 Calculation of levels. "Second order" pyramid (with gradient)

How the second level (of the kernel and image) is constructed? Let the reduction ratio be $2M + 1$ where M is a positive integer. Than each pixel of the next level *substitutes somehow* the group $(2M + 1) \times (2M + 1)$ pixels of current level "around" it.

It is convenient to take $M = 1$ so that reduction of resolution is 3fold, and in our calculations we used that value. Though $M = 2$ and larger is also possible.

Obviously, the convolution of the 1st level (i.e. the original) image with the 1st level kernel can be written as

$$\bar{f}(x, y) = \sum_{x',y'} K((x' - x)\Delta, (y' - y)\Delta) f(x', y') \tag{3}$$

where x, y, x', y' are the *pixel* coordinates i.e. the integer indices, K is the kernel as a function of the physical (continuous!) coordinates (meters, degrees, ...), while Δ is the pixel step i.e. the change of that physical coordinate from one pixel to another.

Grouping the pixels of the 1st level into "clusters" of $(2M+1) \times (2M+1)$ pixels, so that on the original resolution grid their centers are

$$((2M+1)X' + 1, (2M+1)Y' + 1) \tag{4}$$

where (X', Y') are the *pixels at the reduced resolution*, we can write result of convolution at the point $x = (2M+1)X+1, y = (2M+1)Y+1$ (present in both resolutions!) as

$$\begin{aligned} \bar{f}(x,y) = \sum_{X',Y'} \sum_{\xi=-M}^{M} \sum_{\eta=-M}^{M} & K((X'-X)\Delta' + \xi\Delta,\ (Y'-Y)\Delta' + \eta\Delta) \\ & \times f(X'+\xi, Y'+\eta) \\ \Delta' \equiv (2M+1)\Delta & \end{aligned} \tag{5}$$

Near the origin the kernel usually changes steeply while far from it, for $r \geq r_0$ (below we shall calculate that r_0), the kernel becomes so smooth that for it the original high resolution becomes *superfluous*. So according to our idea the kernel (which we shall assume isotropic just for convenience) is decomposed into the sum of the "central part" (vanishing for $r > r_0$) and "periphery":

$$K(r) = K_c(r) + K_p(r) \tag{6}$$

The peripheral part $K_p(r)$ is known for $r \geq r_0$ where by construction it equals $K(r)$, while for $r \leq r_0$ it is arbitrary. It is natural to require that there $K_p(r)$ be maximally simple and smooth, e.g. a parabola seamlessly going into $K(r)$ at $r = r_0$:

$$K_p(r) = \begin{cases} \alpha + \beta r^2, & r \leq r_0 \\ K(r), & r \geq r_0 \end{cases} \tag{7}$$

where

$$\alpha = K(r_0) - \frac{r_0 K'(r_0)}{2} \tag{8}$$

$$\beta = \frac{K'(r_0)}{2r_0} \tag{9}$$

Then the central part is

$$K_c(r) = \begin{cases} K(r) - \alpha - \beta r^2, & r \leq r_0 \\ 0, & r \geq r_0 \end{cases} \tag{10}$$

and the convolution takes the form

$$\bar{f}(x,y) = \underbrace{\sum_{x',y'} K_c(x'-x,y'-y)f(x',y')}_{\bar{f}_c(x,y)}$$
$$+ \underbrace{\sum_{X',Y'} \sum_{\xi=-M}^{M} \sum_{\eta=-M}^{M} K_p((X'-X)\Delta' + \xi\Delta, (Y'-Y)\Delta' + \eta\Delta) f(X'+\xi, Y'+\eta)}_{\bar{f}_p(X,Y)}$$

So far we did not introduced any approximation but just regrouped the original expression and named the results of convolution with the central respectively peripheral part of the kernel as $\bar{f}_c(x,y)$ and $\bar{f}_p(x,y)$.

Now let us assume that r_0 is so large that inside one pixel of 2nd level (i.e. for all $(2M+1)\times(2M+1)$ pixels of the 1st level it comprises) K_p can be approximated by the linear function (effectively the first terms of the Taylor series)

$$\begin{aligned}
&K_p((X'-X)\Delta' + \xi\Delta, (Y'-Y)\Delta' + \eta\Delta) \\
&\approx K_p((X'-X)\Delta', (Y'-Y)\Delta') + (X'-X)D((X'-X)\Delta', (Y'-Y)\Delta')\xi \\
&+(Y'-Y)D((X'-X)\Delta', (Y'-Y)\Delta')\eta
\end{aligned}$$

where

$$D(r) \equiv \Delta'\Delta \times r^{-1}\frac{\partial K_p}{\partial r} = (2M+1)\Delta^2 \times r^{-1}\frac{\partial K_p}{\partial r} = 2(2M+1)\Delta^2 \times \frac{\partial K_p}{\partial (r^2)}$$

Then introducing

$$F_{m,n}(X,Y) \equiv \sum_{\xi=-M}^{M} \sum_{\eta=-M}^{M} \xi^m \eta^n f(X+\xi, Y+\eta)$$

we have in the 2nd level pixel (X,Y),

$$\begin{aligned}
\bar{f}_p(X,Y) \approx &\sum_{X',Y'} K_p((X'-X)\Delta', (Y'-Y)\Delta' F_{0,0}(X',Y') \\
&+ \sum_{X',Y'} (X'-X)D((X'-X)\Delta', (Y'-Y)\Delta' F_{1,0}(X',Y') \\
&+ \sum_{X',Y'} (Y'-Y)D((X'-X)\Delta', (Y'-Y)\Delta' F_{0,1}(X',Y') \quad (11)
\end{aligned}$$

which is nothing but the convolution on the coarse pixel grid of 2nd level of resolution.

Deviation of the kernel from its linear approximation can be estimated through the second derivatives. For $r \geq r_0$ the relative error is

$$\leq \frac{(\Delta')^2}{2} \frac{\left|\frac{\partial K}{\partial r^2}\right| + 2r^2 \left|\frac{\partial}{\partial r^2}\frac{\partial K}{\partial r^2}\right|}{K} \tag{12}$$

while within the central zone it is

$$\leq \frac{(\Delta')^2}{4} \left.\frac{\left|\frac{\partial K}{\partial r^2}\right|}{K}\right|_{r=r_0} \tag{13}$$

The use of the derivative in r^2, not in r, is convenient when the kernel is given as an analytic function of r^2, as it is in human vision. If the kernel is defined as a tabulated function of r,

$$\frac{\partial K}{\partial (r^2)} = \frac{1}{2r}\frac{\partial K}{\partial r}. \tag{14}$$

Usually $\frac{\left|\frac{\partial K}{\partial (r^2)}\right| + 2r^2 \left|\frac{\partial}{\partial (r^2)}\frac{\partial K}{\partial (r^2)}\right|}{K}$ decreases for $r \to \infty$, so the relative error in *any* pixel is bounded by

$$\left.\frac{(\Delta')^2}{2} \frac{\left|\frac{\partial K}{\partial (r^2)}\right| + 2r^2 \left|\frac{\partial}{\partial (r^2)}\frac{\partial K}{\partial (r^2)}\right|}{K}\right|_{r=r_0} \tag{15}$$

Then r_0 can be found as the radius for which the above expression equals the error of approximation. It can be done with e.g. bisection because of monotone decrease.

Construction of the *third level* i.e. decomposition of now K_p into *its* central and peripheral parts is done similarly, just taking K_p instead of K. The central part of K_p becomes level 2 in the pyramid of filters, while peripheral part of K_p is again decomposed, *its* central part becoming level 3 of the pyramid of filters and so on.

Notice the filter of the n-th (for $n > 1$) level is not scalar, but its value is a "tuple" $\left\{K_p, r^{-1}\frac{\partial K_p}{\partial r}\right\}$. The source *image* at this level also keeps a "triad" $\{F_{0,0}, F_{0,1}, F_{1,0}\}$ of sums over sub-pixels (i.e. of $(2M+1)\times(2M+1)$ pixels of the previous level) in each its color channel.

3 Factorization/separability

These words mean a function of several arguments is a product of univariate function. Namely, let us look at the famous Gaussian filter kernel

$$K = C^2 e^{-(r/a)^2}. \tag{16}$$

It is a product of two univariate functions

$$K = Ce^{-(x/a)^2} \times Ce^{-(y/a)^2} \equiv k(x)k(y) \tag{17}$$

so the convolution with it is just two successive *one-dimensional* convolutions

$$\bar{f}(x,y) = \sum_{x'} k(x'-x)g(x',y') \tag{18}$$

$$g(x',y') = \sum_{y'} k(y'-y)f(x',y') \tag{19}$$

For a filter of radius 100 pixels a *direct* calculation of a 2D convolution requires about 100^2 operations per pixel, while two 1D convolutions need only 200 operations, i.e. 50 times less. For larger radius the gain can be giant. So it would be great if this method can be somehow extended to work with more general kernels besides Gaussian.

It happens that really a 2D convolution can be calculated as a series of 1D ones for an isotropic kernel $K(r)$ and even just a "Hermitian" kernel $K(x,y) = K(y,x)$, this is a generalization of "separable filtering" from [10], [1].

Indeed, a Hermitian matrix $K(x,y) = K(y,x)$ of dimension $N \times N$ can be expanded as a sum of "eigenprojectors"

$$K(x,y) = \sum_{m=0}^{N-1} \lambda_m \psi_m(x)\psi_m(y) \tag{20}$$

where ψ_m — is a normalized eigenvector, λ_m being the corresponding eigenvalue. Notice that a matrix with nonnegative elements can have some negative eigenvalues.

The most contribution in (20) comes from eivenvectors with larger $|\lambda|$ so that we order in the absolute value of the eigenvalues and terminate the series (20) when the sum of *remaining* λ_m^2 becomes less than the desired accuracy. Notice for a smooth kernel, the first eigenvectors are also smooth, while the last oscillate rapidly, changing the sign from pixel to pixel. This is a usual property.

For kernels used in simulation of the human vision, a good accuracy about 1% is achieved already for just 2 – 4 (first) eigenvectors. The convolution now is calculated as:

$$\bar{f}(x,y) = \sum_{m} \sum_{x'} \psi_m(x'-x)g_m(x',y') \tag{21}$$

$$g_m(x',y') = \sum_{y'} \lambda_m \psi_m(y'-y)f(x',y') \tag{22}$$

It is not necessary to keep all the images g_m for all pixels. For a filter of size $N \times N$ it is enough to keep just N rows of each of them.

4 Combining factorization and pyramid of resolutions

We can combine the two approaches. Either we first construct the pyramid of filters and then each its level is factored.

In principle the opposite order is also possible, i.e. we first decompose the *original* kernel with (20) and then construct the pyramid for each of its ψ_m. But for this order

we should need to calculate eigenvectors (at least the main) of the matrix of size about 40000x25000 which is too expensive.

For all levels beyond the 1st the filter is a tuple $\left\{K_p, r^{-1}\frac{\partial K_p}{\partial r}\right\}$ so one must factor *both* its components *and* so that the number of eigenvectors retained be *the same*. We find it from the obvious condition that K_p *itself* is approximated well enough, because the gradient component, $r^{-1}\frac{\partial K_p}{\partial r}$ is used for improvement of accuracy to the 2nd order and is thus less essential than the main term K_p.

5 Results

The method was applied to HDRI image of size 4000x2500 shown in Figure 2 (left). Filter was $K = \frac{1}{2\pi a^2}\frac{1}{(1+(r/a)^2)^{3/2}}$ with $a = 12.5$ pixels and was defined in area 2500x2500 pixels.

Fig. 2. Left: The source image used for benchmarking of the method. Right: its exact filtering (FFT).

We compared our methods with the exact filtering based on FFT, shown in Figure 2 (right). The results of our methods are shown in Figure 3. All methods and their combinations give very similar, acceptable results. The difference between images obtained by the 5 filtering methods is below 0.67%.

The memory and speed for computer with i7–4800MQ @ 2.7Ghz and 16Gb DDR3 RAM are shown in Table 1

Table 1. Time in seconds and memory in Mb for different methods of filtration

	pyramid 1st order	**pyramid 2nd order**	**FFT**
with factorization	12.1 sec, 276 Mb	22.3 sec, 276 Mb	n/a
no factorization	27.8 sec, 196 Mb	39.3 sec, 218 Mb	18.8 sec, 2598 Mb

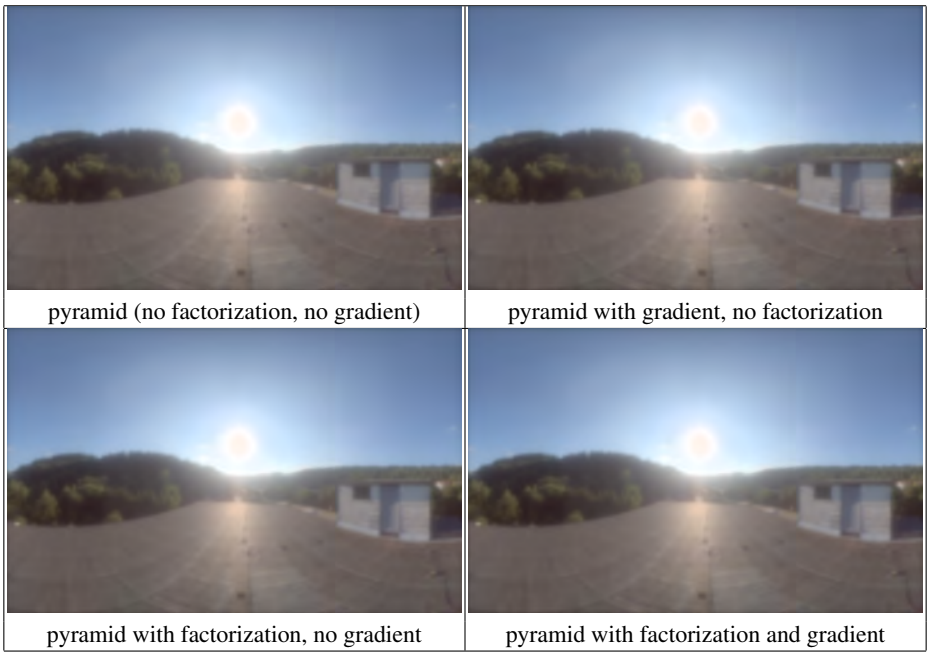

Fig. 3. Image filtered with our methods.

6 Conclusion

We elaborated and tested the method that allows to filter images of very high resolutions without FFT. The new method needs much less memory while the speed is comparable with that achieved with FFT, and in case of pyramid with factorization and no gradient even higher. The deviation from the standard exact method is meanwhile very low, below 1% which is admissible for visual applications. It can be further improved by adding support of gradients (2nd order) and manipulating the settings of the method. This however increases memory and decreases speed, so the results shown in Section 5 are just a reasonable compromise.

References

1. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer (2011).
2. Franssen, L., Coppens, J.: Straylight at the retina: scattered papers. Univ. of Amsterdam, Faculty of Medicine (2007).
3. Ginis, H.S., Pérez, G.M., Bueno, J.M., Artal, P.: The wide-angle point spread function of the human eye reconstructed by a new optical method. Journal of vision 12(3), 20:1–20:10 (2012).
4. Spencer, G., Inc, T., Shirley, P., Zimmerman, K., Greenberg, D.: Physically-based glare effects for digital images. Computer Graphics 29(08), 325–334 (1995).
5. Williamson, C., Strachan, E., Siebert, J.: Simulating the effects of laser dazzle on human vision. In: Proceedings of International Laser Safety Conference. pp. 268–277. Orlando, USA (2013).

6. Yoshida, A., Mittner, M., Mantiuk, R., Seidel, H.P.: Brightness of glare illusion. In: Proceedings of the 5th symposium on Applied perception in graphics and visualization APGV '08. pp. 83–90 (2008).
7. Press, W., Teukolsky, S., Vettering, W., Flannery, B.: Numerical Recipes in C++: The Art of Scientific Computing (2nd edn) Numerical Recipes Example Book (C++) (2nd edn) Numerical Recipes Multi-Language Code CD ROM with LINUX or UNIX single-screen license revised version. European Journal of Physics 24(3), 329 (2003).
8. Pattanaik, S.N., Ferwerda, J.A., Fairchild, M.D., Greenberg, D.P.: A multiscale model of adaptation and spatial vision for realistic image display. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH '98. pp. 287–298. ACM, New York, NY, USA (1998).
9. Dammertz, H., Sewtz, D., Hanika, J., Lensch, H.: Edge-avoiding a-trous wavelet transform for fast global illumination filtering. In: Proceedings of High Performance Graphics 2010. pp. 67–75 (2010).
10. Perona, P.: Deformable kernels for early vision. IEEE Trans. Pattern Anal. Mach. Intell. **17**, 488–499 (1995).