

An Accurate and Efficient Method for Smoothly Space-Variant Gaussian Blurring

T. Popkin, A. Cavallaro and D. Hands

Abstract—This paper presents a computationally efficient algorithm for smoothly space-variant Gaussian blurring of images. The proposed algorithm uses a specialized filter bank with optimal filters computed through principal component analysis. This filter bank approximates perfect space-variant Gaussian blurring to arbitrarily high accuracy and at greatly reduced computational cost compared to the brute force approach of employing a separate low-pass filter at each image location. This is particularly important for spatially variant image processing such as foveated coding. Experimental results show that the proposed algorithm provides typically 10 to 15 dB better approximation of perfect Gaussian blurring than the blended Gaussian Pyramid blurring approach when using a bank of just eight filters.

Index Terms—Filtering, foveation filtering, multiresolution.

I. INTRODUCTION

SPACE-VARIANT blurring, such as *foveation filtering* [1], is an image processing effect whereby different parts of an image are blurred to different extents. Foveation filtering is used in lossy picture coding [2]–[7], to reduce bitrate by exploiting the fact that the focal centre of the retina (the fovea centralis) has a greater concentration of photoreceptors than the periphery [8, p. 236]. Since the retinal sampling varies smoothly, ideal foveation filtering employs a *smooth* variation in the local level of blurring [1]. Space-variant blurring allows bitrate reduction in other ways, such as by aiming to synthesize depth of field effects incurred by eye or camera lenses [9], [10]. Finally, space-variant blurring can be used to synthesize depth of field for rendering in computer graphics [11].

Existing approaches to space-variant blurring include filter banks, in which the image is typically filtered using a number of parallel band-pass or low-pass filters, whose outputs undergo a space-variant combination [1]. However, this results in a discretisation of blur levels (frequency bands), which precludes *smooth* space-variance. Another approach is to apply a spatial co-ordinate transformation, such as log polar mapping [12], and then apply uniform blurring, before applying the reverse transformation [13]. However, the log polar mapping approach can only generate foveal blur maps, and not general blur maps as with the other techniques. The simplest approach is the *summed area table* [14] (or *integral image* [15]) approach, but its square blurring gives it poor frequency domain characteristics which are detrimental to compression performance.

The linear filters in a filter bank can be implemented by number of approaches, the simplest being a non-recursive convolution, implemented as a direct finite-impulse response (FIR) filter. Alternatively, a recursive, infinite impulse response

(IIR) filter can be employed [16]. Recursive filter approximations of Gaussian blurring have also been adapted to be space-variant [17]. However, high-precision FIR and IIR filters require a large number of taps, and there will always be a point at which a fast convolution technique [18, p. 538] will be more efficient (e.g., data lengths in the range of 20 to 50 points in the case of 1-D filtering, dependent on implementation [19, p. 8-2]). Hierarchical techniques, which construct and employ a pyramid of blurred versions of the original image at differing levels of resolution, are highly efficient. Of these, the blended *Gaussian Pyramid* approach [20], [21] performs a smoothly space-variant approximation of Gaussian blurring. However, this approximation has limited accuracy. Smoothly space-variant blurring with a non-trivial filter and a general map of blur levels is extremely expensive in terms of computational cost, potentially requiring a different low-pass filter for each image location [1]. In the case of 2-D Gaussian functions, the filtering at each level can be performed by separate vertical and horizontal convolution by a 1-D Gaussian, but to do this in a general space-variant manner is too expensive for a real-time system [17]. However, we demonstrate in this paper that perfect smoothly space-variant Gaussian blurring can be computed to arbitrary precision without prohibitive computational cost.

The contribution of this paper is as follows. An optimal filter bank algorithm for high-precision space-variant Gaussian blurring is proposed, using an optimal basis computed through principal component analysis and a fast, symmetric convolution technique. The proposed approach bridges the cost-versus-accuracy performance gap between faster, less accurate approaches such as blended Gaussian Pyramid [20], [21], and the prohibitively slow, high-quality approach of a different low-pass filter at each location [1]. We demonstrate that the space of Gaussian filters over a finite range of bandwidths is closely spanned by very small number of basis functions, with rapid, exponential convergence as the number of functions employed is increased. Finally, the proposed algorithm is evaluated in terms of the trade-off between cost and accuracy when employing different numbers of basis functions.

The remainder of this paper is organised as follows. Section II presents the proposed algorithm. Section III evaluates and compares the accuracy and cost of the proposed technique. Section IV concludes the paper.

II. PROPOSED APPROACH

A. Core algorithm

Let \mathbf{I} be a W -by- H colour image and b a blur map of the same size. Let \mathbf{S}_b be a space-variant Gaussian-blurred version

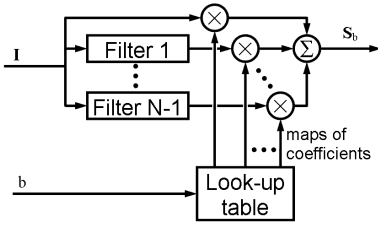


Fig. 1. Block diagram of the proposed approach. The input image, \mathbf{I} is separately convolved with a number of predefined filters and each convolved image is multiplied pixel-by-pixel by a spatial map of coefficients and the results are summed, giving the output image, \mathbf{S}_b . Each coefficient, at each pixel location, is given by a predefined look-up table according to the value in the blur map, b , at that location. The proposed approach is a filter bank method, except that normal filter banks use bandpass or lowpass filters, whereas the proposed approach uses specially-derived basis functions for the space of Gaussian functions.

of \mathbf{I} . Fig. 1 gives a block diagram of the proposed approach. The image and blur map are functions such that $\mathbf{I}, \mathbf{S}_b : D \rightarrow \mathbb{R}^3$ and $b : D \rightarrow \mathbb{R}$, where domain $D = \{(x_1, x_2) \in \mathbb{Z}^2 : 0 \leq x_1 < W, 0 \leq x_2 < H\}$ is the set of possible pixel locations and \mathbb{Z} is the set of all integers. Consider also a domain \check{D} , of size L -by- L , centred around zero, and defined as $\check{D} = \{(x_1, x_2) \in \mathbb{Z}^2 : \lceil \frac{-L}{2} \rceil \leq x_1, x_2 < \lceil \frac{L}{2} \rceil\}$, where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denote upward integer rounding. Consider a Gaussian point spread function (PSF), $G_\sigma : \check{D} \rightarrow \mathbb{R}$, defined for all $\mathbf{x} = (x_1, x_2) \in \check{D}$ as follows:

$$G_\sigma(x_1, x_2) = \begin{cases} k_\sigma \exp\left(\frac{x_1^2 + x_2^2}{-2\sigma^2}\right) & \text{if } \sigma \neq 0 \\ \delta(x_1^2 + x_2^2) & \text{if } \sigma = 0, \end{cases} \quad (1)$$

where δ is the delta function and each normalised constant k_σ is defined as

$$k_\sigma = 1 \bigg/ \sum_{(x_1, x_2) \in \check{D}} \exp(-(x_1^2 + x_2^2)/2\sigma^2). \quad (2)$$

Consider image $\mathbf{U}_\sigma : D \rightarrow \mathbb{R}^3$, defined as a uniformly Gaussian-blurred version of \mathbf{I} , as follows:

$$\mathbf{U}_\sigma(\mathbf{x}) = (\mathbf{I} * G_\sigma)(\mathbf{x}) = \sum_{\mathbf{y} \in \check{D}} \tilde{\mathbf{I}}(\mathbf{x} - \mathbf{y}) G_\sigma(\mathbf{y}), \quad (3)$$

for all $\mathbf{x} \in D$, where $\mathbf{I} * G_\sigma$ is the symmetric convolution of \mathbf{I} with G_σ and $\tilde{\mathbf{I}}$ is the symmetric extension of \mathbf{I} over the whole of \mathbb{Z}^2 . In precise terms, $\tilde{\mathbf{I}}(2pW - \frac{1}{2} \pm (x_1 + \frac{1}{2}), 2qH - \frac{1}{2} \pm (x_2 + \frac{1}{2})) = \mathbf{I}(x_1, x_2)$ for all $(x_1, x_2) \in D$ and all $p, q \in \mathbb{Z}$.

Now, given the blur map $b : D \rightarrow \mathbb{R}$, define image $\mathbf{S}_b : D \rightarrow \mathbb{R}^3$, a space-variant Gaussian-blurred version of \mathbf{I} , as follows:

$$\mathbf{S}_b(\mathbf{x}) = \mathbf{U}_{b(\mathbf{x})}(\mathbf{x}) \quad (4)$$

$$= \sum_{\mathbf{y} \in \check{D}} \tilde{\mathbf{I}}(\mathbf{x} - \mathbf{y}) G_{b(\mathbf{x})}(\mathbf{y}). \quad (5)$$

Here, $\mathbf{U}_{b(\mathbf{x})}$ is the uniformly-blurred image as given by Eq. (3) and $G_{b(\mathbf{x})}$ is the PSF as given by Eq. (1), but with $b(\mathbf{x})$ substituted for σ in both cases.

To compute each $\mathbf{S}_b(\mathbf{x})$ value by interpreting Eq. (5) verbatim would be prohibitive, as this would require a sum of L^2 terms for every pixel. However, a close approximation

of this computation can be performed using a faster approach which will now be described.

Given minimum and maximum blur levels, $m \in \mathbb{R}$ and $M \in \mathbb{R}$, consider a family, $\Gamma = \{G_\sigma : \sigma \in [m, M]\}$, of Gaussian PSFs. Consider the equivalent family, $\check{\Gamma} = \{\check{G}_\sigma : \sigma \in [m, M]\}$ of PSFs that have been modified to be orthogonal to the delta function, δ . That is, for all $\mathbf{x} \in \check{D}$,

$$\check{G}_\sigma(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} = \mathbf{0} \\ G_\sigma(\mathbf{x}) & \text{if } \mathbf{x} \neq \mathbf{0}. \end{cases} \quad (6)$$

Consider a tensor $Z : \check{D} \times \check{D} \rightarrow \mathbb{R}$, defined as follows:

$$Z(\mathbf{x}, \mathbf{y}) = \int_m^M \frac{\check{G}_\sigma(\mathbf{x}) \check{G}_\sigma(\mathbf{y})}{\sigma} d\sigma, \quad (7)$$

for all $\mathbf{x}, \mathbf{y} \in \check{D}$. Note that Z is symmetric with respect to its arguments; that is, $Z(\mathbf{x}, \mathbf{y}) = Z(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}, \mathbf{y} \in \check{D}$. Since the set \check{D} is finite (with L^2 members), Z can be handled numerically as a matrix. This matrix, being symmetric, can be diagonalised and has an orthonormal basis of eigenvectors [18, p. 459]; that is

$$Z(\mathbf{x}, \mathbf{y}) = \sum_{n=1}^{L^2} \beta_n(\mathbf{x}) \lambda_n \beta_n(\mathbf{y}), \quad (8)$$

for some eigenvalues $\lambda_1, \dots, \lambda_{L^2} \in \mathbb{R}$ and some orthonormal set of eigenfunctions, $\beta_1, \dots, \beta_{L^2} : \check{D} \rightarrow \mathbb{R}$. It happens that all except a small number of the eigenvalues of Z are very close to zero, the result being that each $\check{G}_\sigma \in \check{\Gamma}$ can be approximated closely by a linear combination of the first few eigenfunctions (assuming the eigenfunctions to be arranged in descending order of eigenvalue). Furthermore, suppose that this basis is extended by adding an extra function β_0 , defined to be the delta function; that is

$$\beta_0(\mathbf{x}) = \delta(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{0} \\ 0 & \text{if } \mathbf{x} \neq \mathbf{0} \end{cases} \quad (9)$$

for all $\mathbf{x} \in \check{D}$. Recall that each \check{G}_σ is orthogonal to β_0 , and hence so are the β_1, β_2, \dots which span the space of \check{G}_σ functions. Now, each Gaussian PSF $G_\sigma \in \Gamma$ can be approximated by a linear combination of the first N basis functions, $\beta_0, \dots, \beta_{N-1}$, for a suitably chosen N . That is, for every $\sigma \in [m, M]$ and every $\mathbf{x} \in \check{D}$,

$$G_\sigma(\mathbf{x}) \approx \sum_{n=0}^{N-1} c_n(\sigma) \beta_n(\mathbf{x}), \quad (10)$$

where each coefficient function $c_n : [m, M] \rightarrow \mathbb{R}$ is defined as follows:

$$c_n(\sigma) = \sum_{\mathbf{x} \in \check{D}} \beta_n(\mathbf{x}) G_\sigma(\mathbf{x}) \quad (11)$$

for all $\sigma \in [m, M]$. That is, thinking of the functions as vectors, each scalar value $c_n(\sigma)$ is the component of vector G_σ in the direction of basis vector β_n . For each n and N , consider also a normalised coefficient function $\hat{c}_{n,N} : [m, M] \rightarrow \mathbb{R}$ defined as follows:

$$\hat{c}_{n,N}(\sigma) = c_n(\sigma) \bigg/ \sum_{\mathbf{x} \in \check{D}} \sum_{n=0}^{N-1} c_n(\sigma) \beta_n(\mathbf{x}) \quad (12)$$

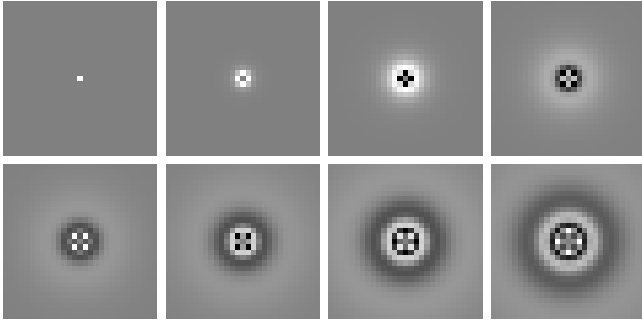


Fig. 2. Top (left to right): first four basis functions of the family of Gaussian PSFs, when $L = 81$, $m = \frac{1}{3}$ and $M = 10$ (see Eq. (7)); bottom: next four basis functions. Mid-gray represents zero; dark shades represent negative; light shades represent positive.

for all $\sigma \in [m, M]$. Note that because $\sum_{n=0}^{N-1} c_n(\sigma)\beta_n(\mathbf{x})$ converges to $G_\sigma(\mathbf{x})$ as $N \rightarrow \infty$, coupled with the fact that $\sum_{\mathbf{x} \in \check{D}} G_\sigma(\mathbf{x}) = 1$, it is a fact that $\sum_{n=0}^{N-1} \hat{c}_{n,N}(\sigma)\beta_n(\mathbf{x})$ converges to $G_\sigma(\mathbf{x})$. Therefore,

$$G_\sigma(\mathbf{x}) \approx \sum_{n=0}^{N-1} \hat{c}_{n,N}(\sigma)\beta_n(\mathbf{x}), \quad (13)$$

for all $\mathbf{x} \in \check{D}$. This is important because the use of $\hat{c}_{n,N}$ rather than c_n will ensure that the approximated Gaussian will always sum to unity.

Substituting Eq. (13), with $\sigma = b(\mathbf{x})$, into Eq. (5) gives the following:

$$\begin{aligned} \mathbf{S}_b(\mathbf{x}) &\approx \sum_{\mathbf{y} \in \check{D}} \tilde{\mathbf{I}}(\mathbf{x} - \mathbf{y}) \sum_{n=0}^{N-1} \hat{c}_{n,N}(b(\mathbf{x}))\beta_n(\mathbf{y}) \\ &= \sum_{n=0}^{N-1} \hat{c}_{n,N}(b(\mathbf{x})) \sum_{\mathbf{y} \in \check{D}} \tilde{\mathbf{I}}(\mathbf{x} - \mathbf{y})\beta_n(\mathbf{y}) \\ &= \sum_{n=0}^{N-1} \hat{c}_{n,N}(b(\mathbf{x})) (\mathbf{I} * \beta_n)(\mathbf{x}), \end{aligned}$$

where $\mathbf{I} * \beta_n$ is the symmetric convolution of \mathbf{I} with β_n . Therefore,

$$\mathbf{S}_b(\mathbf{x}) \approx \sum_{n=0}^{N-1} \hat{c}_{n,N}(b(\mathbf{x})) \psi_n(\mathbf{x}), \quad (14)$$

where each filtered image $\psi_n : D \rightarrow \mathbb{R}$ is defined as a convolution, $\psi_n = \mathbf{I} * \beta_n$. That is,

$$\psi_n(\mathbf{x}) = \sum_{\mathbf{y} \in \check{D}} \tilde{\mathbf{I}}(\mathbf{x} - \mathbf{y})\beta_n(\mathbf{y}). \quad (15)$$

The reason Z and its eigenfunctions are relevant is because these linear combinations of the N chosen eigenfunctions can be used in approximating the Gaussian PSFs, as stated in equations (10) and (13), and hence they form an approximate basis for the space, Γ , of possible PSFs. Effectively, this derivation of eigenfunctions is a form of principal component analysis on the set $\tilde{\Gamma}$. The eigenvectors are the optimal basis of $\tilde{\Gamma}$ in terms of providing the lowest expected value of the sum-of-squared-errors of the approximated Gaussians (i.e., the

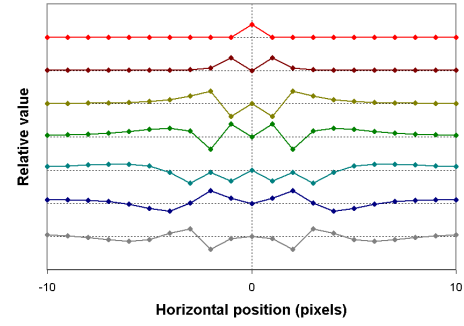


Fig. 3. Horizontal cross-sections of the first seven basis functions (as shown in Fig. 2). To aid visibility, all functions have here been scaled to have the same maximum absolute value.

lowest statistical mean squared error across a representative range of test images). This statistical perspective intrinsically involves an *a priori* probability assumption of the distribution of σ values. The assumption associated with the $1/\sigma$ factor in Eq. (7) is a log-uniform distribution of σ ; that is, a uniform *a priori* probability of $\log(\sigma)$ values. This is desirable because over the very low sigma values, the Gaussian bell curves vary greatly between close σ values, whereas over the higher σ values, there is far less variation. The $1/\sigma$ factor in Eq. (7) is effectively a weighting factor which reduces the significance of higher σ values which would dominate the integral in Eq. (7) at the expense of the lower σ values. This reduces the worst-case error of approximated Gaussian curves. For example, applying the proposed approach with $N=8$ to a random 256×256 black & white image with a uniform blur map, with σ at different multiples of 0.1, gives a worst-case error of 49.4dB at $\sigma=0.2$. However, the worst-case error if the $1/\sigma$ factor is removed from Eq. (7) is 42.8dB, also at $\sigma=0.2$.

It has been found empirically that with $m = \frac{1}{3}$, $M = 10$, $L = 81$ and just 7 basis functions (i.e., $N=7$), the worst root-mean-squared error of any of the approximated G_σ functions is roughly 0.00005. This implies, in the trivial example of a grayscale image with all pixels zero except for one pixel with gray level 255, that the sum of square errors will be $0.00005^2 \times 81^2$, so the worst possible gray level error of any pixel in the blurred image cannot be greater than $255 \times 0.00005 \times 81 = 1.03275$. However, for general images with most pixels non-zero, the overall errors will be greater and will be dependent on the image itself, as shown by the results in section III.

The first eight basis functions, when $m = \frac{1}{3}$ and $M = 10$, are represented in Figs 2 and 3. A graph showing the exponential nature of the descent of the eigenvalues is given in Fig. 4. This illustrates the nature of the space of eigenvector as being of low approximate dimensionality. In rough terms, each eigenvalue can be seen as the square of the width of the space when measured in the direction of the given eigenvector. The square error induced by discarding the least significant eigenvectors will, in general, be roughly proportional to the sum of the eigenvalues of these discarded eigenvectors. The mean ratio between any adjacent pair within the first 20 eigenvalues as shown is 4.39; i.e., the width of the space roughly halves in the direction of each new eigenvector.

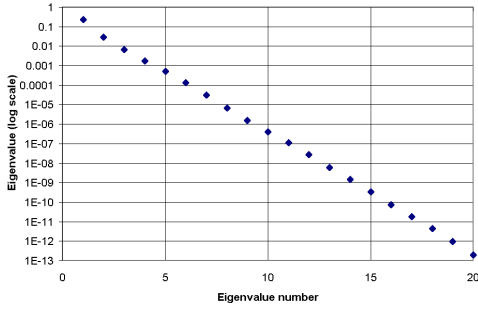


Fig. 4. Plot of the first 20 eigenvalues of the computed basis functions. The vertical scale is logarithmic, clearly showing the exponential decrease of the eigenvalues, which corresponds with an exponential convergence of any approximated Gaussian generated by different numbers of basis functions.

B. Implementation details

Eq. (14) is the top-level stage of the proposed algorithm. Note that being a weighted sum of convolutions (i.e., filters) makes it a special instance of the filter bank method [1]. Because β_0 is the delta function, the first of these $\psi_n = \mathbf{I} * \beta_n$ convolutions, with $n = 0$, is the identity operation. That is, $\psi_0 = \mathbf{I}$, so no work needs to be done here. However, each of the remaining $N - 1$ convolutions can be performed using any fast convolution technique [18, p. 538], using fast Fourier transforms, fast *number theoretic transforms*, or fast discrete cosine transforms (DCTs). All these approaches in their simplest form restrict the input image to array dimensions which are exact powers of two, and therefore, in its simplest form as presented herein, the proposed approach follows the same restriction. The approach can be extended to other image sizes by zero padding of the image.

In the specific implementation reported in this paper, the convolution performed was a symmetric convolution, performed using *convolution form* DCTs [22]. Using Martucci's terminology [22], the aim was *half-sample symmetry* around image boundaries, and *whole-sample symmetry* around the point spread function's origin point. This requires involved applying a Type I DCT to a quadrant of the basis function and a Type II DCT to the image, followed by per-element multiplication then an inverse Type II DCT to obtain the convolved image. This involves converting each basis function β_n to a $(W + 1)$ -by- $(H + 1)$ domain size by zero padding up to $x_1 = W$ and $x_2 = H$ and discarding $\beta_n(x_1, x_2)$ values for negative x_1 and x_2 . This loses no information, since $\beta_n(x_1, x_2) = \beta_n(\pm x_1, \pm x_2)$ for all n , x_1 and x_2 , due to every Gaussian PSF G_σ having the same symmetry. Each 2-D DCT was computed by applying the corresponding type of 1-D DCT firstly replacing each row of the image or PSF with its DCT then replacing each column with its DCT. Every 1-D DCT was performed using a verbatim implementation of the sparse matrix decompositions described in [23]. These classic DCT operations were converted into *convolution form* DCTs by appropriately weighting the array elements before and after the classic DCT, as prescribed in [22].

The proposed algorithm assumes that all the eigenfunctions have been precomputed and their Type I discrete cosine transforms have been stored (for the fast convolution stage),

along with a look-up table approximation of each coefficient function, $\hat{c}_{n,N}$. The width L of the PSF domain \check{D} should be chosen to be sufficiently large that every G_σ function is sufficiently close to zero for the desired accuracy. In practice, $L \geq 6M$ is sufficient due to the fact that a Gaussian is near zero beyond three standard deviations from its mean, but in section III of this paper, $L = 81$ was used with $M = 10$. Furthermore, L should be an odd number so that every G_σ function has symmetry about its central point.

The eigenfunctions $\beta_1, \beta_2, \beta_3, \dots$ were computed by diagonalisation of the matrix representation of Z , which was numerically approximated by a discrete summation equivalent of the integral in Eq. (7). Because of the $1/\sigma$ factor, this integral was approximated by summing $\check{G}_\sigma(\mathbf{x})\check{G}_\sigma(\mathbf{y})$ samples over a discrete range of σ values with density decreasing in proportion to $1/\sigma$. Specifically,

$$\int_m^M \frac{\check{G}_\sigma(\mathbf{x})\check{G}_\sigma(\mathbf{y})}{\sigma} d\sigma \approx \frac{\log_e(M/m)}{Q+1} \sum_{q=0}^Q \check{G}_{\sigma_q}(\mathbf{x})\check{G}_{\sigma_q}(\mathbf{y}), \quad (16)$$

where each σ_q is defined as $\sigma_q = ((M/m)^{q/Q})m$. In the specific implementation of this paper, $Q = 99$ was used, so that each $Z(\mathbf{x}, \mathbf{y})$ computation was approximated by a sum of 100 terms.

The computation of Z and its eigenvectors are computationally expensive processes in which some savings are possible. As these are precomputed, this expense does not effect the cost of the core algorithm. However, a computational saving has been made for the purpose of this paper by reducing the \check{G}_σ arrays in size, by restricting each \check{G}_σ function to a one-eighth segment of the domain \check{D} , so as to exploit the $\check{G}_\sigma(x_1, x_2) = \check{G}_\sigma(\pm x_1, \pm x_2) = \check{G}_\sigma(\pm x_2, \pm x_1)$ symmetries of the Gaussian functions. This restriction was done before eigenvector decomposition. After this, the reverse process was applied to reconstruct the basis functions over the whole of \check{D} , followed by normalisation of the basis. In order to produce exactly the same resulting eigenfunctions, it is necessary to apply appropriate weightings to each boundary point in proportion to the square root of the number of identical-shaped eighth-part segments which share the boundary point (and to apply the inverse weightings to the corresponding locations afterwards). That is, each function used was represented by a triangle of points as shown below:

$$\begin{pmatrix} \frac{1}{\sqrt{8}}g_{0,0} & & & & \\ \frac{1}{\sqrt{2}}g_{1,0} & \frac{1}{\sqrt{2}}g_{1,1} & & & \\ \frac{1}{\sqrt{2}}g_{2,0} & g_{2,1} & \frac{1}{\sqrt{2}}g_{2,2} & & \\ \frac{1}{\sqrt{2}}g_{3,0} & g_{3,1} & g_{3,2} & \frac{1}{\sqrt{2}}g_{3,3} & \\ \vdots & \vdots & \vdots & \ddots & \\ \frac{1}{\sqrt{2}}g_{h,0} & g_{h,1} & g_{h,2} & \cdots & \frac{1}{\sqrt{2}}g_{h,h} \end{pmatrix} \quad (17)$$

where each $g_{y,x}$ represents $\check{G}_\sigma(x, y)$, and $h = (L - 1)/2$, with L an odd number.

As an example application of the proposed algorithm, consider foveation filtering. In this scenario, the blur map would be a spatial map of cut-off frequencies, using knowledge of the point of human fixation combined with a model of the

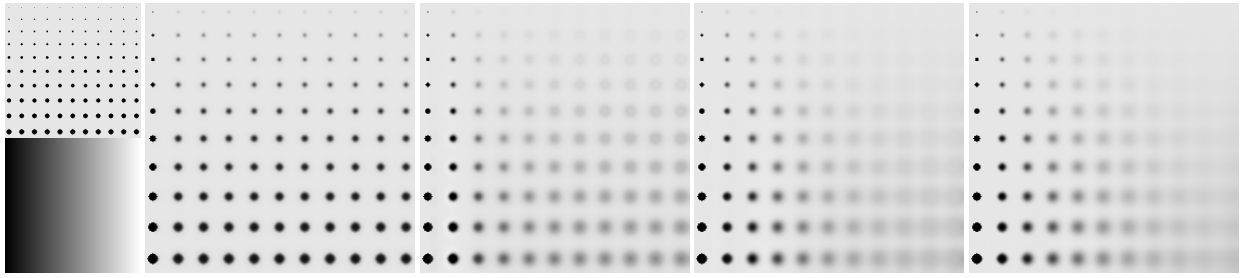


Fig. 5. Examples of applying the proposed algorithm to a 256×256 synthetic image, with varying number, N , of basis functions employed. Top left: original image. Bottom left: blur map (σ increasing left to right from 0 to 10). Remaining images, left to right: output of the proposed approach with $N = 2, 4, 6, 8$. The most significant basis functions (with the highest eigenvalues) tend to correspond to the higher frequencies, which can be seen in the fact that the lower-blur regions in the above change less than the higher-blur regions as the number of basis functions increases.



Fig. 6. Top left: raw image (*Mandrill*). Bottom left: blur map based on the visual field of a glaucoma patient [20]; white = maximum ($\sigma = 10$); black = zero. Right: space-variant Gaussian-blurred image according to the blur map, using the proposed algorithm with $N = 8$.



Fig. 7. Top left: raw image (leftmost 512×512 portion of 512×768 image *Kodim08*). Bottom left: blur map, with blur increasing steadily from right to left; white = maximum ($\sigma = 10$); black = zero. Right: space-variant Gaussian-blurred image according to the blur map, using the proposed algorithm with $N = 8$.

human visual system, such as a contrast threshold formula [24]. Each cut-off frequency f is then converted into a σ value by employing the common convention of treating the cut-off frequency of a filter as its 3dB point. This would give $\sigma = \sqrt{(2 \log_e(\sqrt{2}))}/f$.

Fig. 5 shows example output of the algorithm on a synthetic image, for a range of numbers of basis functions employed. Fig. 6 shows example output of the algorithm on a real image in a vision research application. This allows normally-sighted people to visualise the effects of sight problems such as glaucoma. Fig. 7 shows example output in variable resolution rendering. With all these examples, the same set of basis functions were used, as generated for a range $[m, M] = [\frac{1}{3}, 10]$ of σ values.

C. Computational cost

Applying Eq. (14) for a single location \mathbf{x} is an order $\mathcal{O}(N)$ operation as it costs only N look-up operations (one for each $\hat{c}_{n,N}$), N multiplications and $N-1$ additions. The dominant cost in the algorithm is the fast DCTs in computation of the filtered images $\psi_1, \dots, \psi_{N-1}$, each of which will be of order $\mathcal{O}(HW \log(HW))$, which only need to be done once for each image \mathbf{I} . Therefore, the overall cost of these convolutions is of order $\mathcal{O}(NHW \log(HW))$, which can be regarded as

$\mathcal{O}(HW \log(HW))$ since N is fixed. This compares with $\mathcal{O}(H^2W^2)$ for the approach of applying an independent filter for every pixel, as is necessary with the reference method in the extreme case of a different blur level for every pixel. As an additional comparison, the fastest space-variant blurring approach is the integral image approach, whose cost is of order $\mathcal{O}(HW)$ for an W -by- H image. The blended Gaussian Pyramid approach is also $\mathcal{O}(HW)$, assuming a fixed maximum blur level (and hence a fixed number of hierarchy levels) as H and W increase. The blended Gaussian Pyramid cost therefore increases at roughly the same relative rate as the integral image approach as H and W become large.

III. EVALUATION AND COMPARISONS

A. Evaluation of the proposed approach

The proposed approach approximates space-variant Gaussian blurring to arbitrarily high accuracy, permitting any number N of basis functions to be employed. This allows a trade-off between computational cost and blurring accuracy, which is evaluated in this section in terms of PSNR. To this end, we have applied the algorithm with a varying number of basis functions, from one to fifteen. In order to provide an implementation-dependent measure of computational cost, a count of the total number of arithmetic operations

(floating point additions, subtractions and multiplications) was employed. Average computational times per pixel were also recorded when running the (non-optimised) Java code, on a 3.19 GHz Pentium D machine running Microsoft Windows XP and the Java Runtime Environment 1.6.0. All computations in the evaluations were performed using 64-bit floating point arithmetic, to ensure a high accuracy ceiling for the experiments.

For simplicity, the test images [25]–[27] were selected to be a power of two in width and height (512×512): (1) 5.2.08; (2) top-left 512×512 portion of *Barbera*; (3) *F-16*; (4) leftmost 512×512 portion of *Kodim12*; (5) topmost 512×512 portion of *Kodim17*; (6) topmost 512×512 portion of *Kodim18*; (7) leftmost 512×512 portion of *Kodim23*; (8) *Lena*; (9) a synthetic white noise image, *Rand512*, with each RGB sample randomly taken as either $(0,0,0)$ or $(255,255,255)$ with equal probability. For simplicity, all images were treated as full-colour RGB images, including the grayscale images 5.2.08 and *Rand512*, which were treated as colour images during experiments, for consistency with the other images. Therefore each measured cost of the implementation was precisely three times what it would have been for a grayscale image of the same size.

For all test images, a simplified foveal blur map was employed, defined for all pixel locations $(x_1, x_2) \in D$ as

$$b(x_1, x_2) = 2M \sqrt{\frac{(x_1 - c_1)^2 + (x_2 - c_2)^2}{H^2 + W^2}},$$

where $(c_1, c_2) \in D$ is the centre of the W -by- H image, M is the maximum blur level, and $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote integer downward rounding. Note that this map has zero blurring at the image centre and max blur level of 10 at each image corner. This blur map is depicted in Figs 9 and 10.

Each blurred image under test was computed using the proposed algorithm, according to the approximation in Eq. (14), for the given number, N , of basis functions. The basis functions as described in Section II were generated with a maximum blur level $M = 10$ and a minimum $m = \frac{1}{3}$, and they were generated on a reduced-size domain of size 81×81 , then padded with zeros to the full 512×512 size.

The reference method for perfect Gaussian blurring worked by computing each target image using Eq. (4), after separately computing every uniformly Gaussian-blurred image U_σ , for 101 discrete σ values, $\sigma \in \{\frac{0}{10}, \frac{1}{10}, \frac{2}{10}, \dots, \frac{99}{10}, \frac{100}{10}\}$. Each computation was performed using the same fast convolution technique as for the proposed approach except for the trivial case of $\sigma = 0$, which was dealt with by simply assigning $U_0 = I$, the input image. To ensure consistency across techniques, the same restriction to this discrete set of σ values was applied to the proposed approach.

The PSNR accuracy figures of the proposed approach using an increasing number of basis functions are shown in Table I. The convergence of the proposed approach to perfect Gaussian blurring can be seen as the number of basis functions increases. At $N = 15$, for all test images, the PSNR exceeds 70 dB.

The cost figures in terms of floating point operation counts and average computational times per pixel are shown in Table

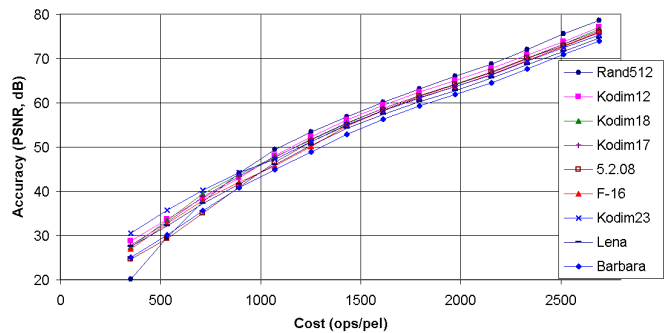


Fig. 8. Cost-versus-accuracy plot (PSNR) for the proposed algorithm using varying numbers (2 to 15) of basis functions, when applied to the test images according to the corresponding foveal blur map shown in Figs 9 and 10. Cost is measured in terms of the count of arithmetic operations per pixel. Accuracy is measured as PSNR relative to the perfectly Gaussian blurred image.

II. Fig. 8 visualizes the relationship between cost and PSNR. Each floating point count given in the graph and table is k/HW , where k is the total number of arithmetic operations used by the proposed algorithm to blur the given image, and HW is the number of pixels in the image. The PSNR improves at a steady rate on the logarithmic scale. The average improvement is 3.8 dB per basis function for the mean PSNR across images. The increase in cost is a fixed 180.1 ops/pel, and the average cost increase for each decibel of improvement to the mean accuracy is 50.4 ops/pel/dB.

Figures 9 and 10 show the output of the proposed technique for *Lena* and *Kodim18*, respectively, when using a differing number of basis functions ($N = 2, 5, 8$). In the $N = 2$ images, the image is sharp inside a narrow region around the foveation point (at the image centre), whereas outside this region, the level of blur is roughly uniform. This reflects the fact that only two basis functions are employed here, one of which is the delta function (no blurring) and the other of which cannot provide blurring greater than roughly $\sigma = 2$ (see the second basis function cross-section in Fig. 3). The remaining images show the result converging to the true space-variant Gaussian blurring as the number of basis functions employed increases, with the visible peripheral blurring increasing.

B. Comparison with blended Gaussian Pyramid

In this section we compare the results of the proposed approach with those of a blended Gaussian Pyramid approach [20] in the same tests as described in the previous subsection. The main part of the blended Gaussian Pyramid approach is the Gaussian Pyramid technique itself [21], which works as follows. Firstly, the RGB image is subsampled by factor of two vertically and horizontally, each time preceded by a 5-tap filter (i.e., weighted average) of $(\frac{1}{20}, \frac{1}{4}, \frac{2}{5}, \frac{1}{4}, \frac{1}{20})$ in each direction. This gives a multi-level pyramid of low-resolution representations of the original, each with $1/4$ as many pixels as the previous level. Then, each level is upsampled back up to the original size, by repeated upsampling by a factor of two in each direction. Each upsampling is done as if by zero padding followed by averaging, in each direction, by $(0, \frac{1}{2}, 0, \frac{1}{2}, 0)$ at inter-pixel (odd) locations, and $(\frac{1}{10}, 0, \frac{4}{5}, 0, \frac{1}{10})$ at even locations. For consistency with the other approaches employed,



Fig. 9. Examples of foveal blurring of *Lena*. Top left: raw image. Bottom left: blur map (black: $\sigma=0$; white: $\sigma=10$). The remaining images give the output of the proposed approach as it converges towards perfect Gaussian blurring (left to right: $N = 2, 5, 8$).



Fig. 10. Examples of foveal blurring of *Kodim18*. Top left: raw image. Bottom left: blur map (as for Fig. 9). The remaining images give the output of the proposed approach as it converges towards perfect Gaussian blurring (left to right: $N = 2, 5, 8$).

the image is assumed to extend symmetrically beyond its boundaries. The result is a sequence of increasingly blurred images, the first of which is unblurred and each successive one is twice as blurred as the previous. To use the Gaussian Pyramid approach for smoothly space-variant blurring, inter-level blending is necessary. For this, the scheme of Perry & Geisler [20] is used, which works by taking, for each pixel, a linear combination between the two corresponding pixels of the images blurred to levels above and below the desired blur level for the given pixel. This combination of pixels $\mathbf{p}_{i_r}, \mathbf{p}_{i_r-1} \in \mathbb{R}^3$ of the blurred images at levels i_r and $i_r - 1$ of the pyramid is computed as $B(r)\mathbf{p}_{i_r-1} + (1 - B(r))\mathbf{p}_{i_r}$, where $B(r) \in \mathbb{R}$ is the blending factor, computed as

$$B(r) = (0.5 - T_{i_r}(r)) / (T_{i_r-1}(r) - T_{i_r}(r)),$$

T_{i_r} and T_{i_r-1} are the transfer functions (i.e., frequency responses) of the blurring at levels i_r and $i_r - 1$, and r is the half-amplitude frequency of the desired Gaussian curve, which was computed as $r = \sqrt{\log_e 4 / 2\pi b(\mathbf{x})}$ at a given location \mathbf{x} . Each level i_r is chosen as the value for which level $i_r - 1$ gives too little blurring and i_r gives too much. That is, for each r ,

$$i_r = 1 + \max\{j : j \in \mathbb{Z} \text{ and } T_j(r) \geq 0.5\}.$$

The transfer functions were computed by applying the Gaussian Pyramid method to a simple impulse function and com-

TABLE I
BLURRING ACCURACY RESULTS (PSNR, DECIBELS)

Image	Rand512	Lena	5.2.08	Barbara	F-16	Kodim12	Kodim17	Kodim18	Kodim23	Mean	
Gaussian Pyramid	34.9	46.1	37.1	42.5	41.8	39.1	41.8	46.9	44.1	41.6	
Proposed approach (using 2 to 15 basis functions)	2	20.1	27.6	24.7	25.0	27.0	28.7	27.9	27.4	30.5	26.5
	3	29.7	32.2	29.3	30.2	32.8	33.6	33.1	33.5	35.7	32.2
	4	37.7	37.2	35.1	35.7	38.0	38.4	38.7	39.5	40.3	37.9
	5	44.1	41.7	41.0	40.8	42.0	43.0	43.4	43.8	44.2	42.7
	6	49.4	46.0	46.5	44.9	45.8	48.1	47.6	47.8	47.2	47.0
	7	53.5	50.4	51.2	48.8	50.1	52.3	51.8	51.7	51.3	51.2
	8	56.8	54.2	54.9	52.8	54.8	56.2	55.4	55.3	55.1	55.1
	9	60.1	57.4	58.3	56.3	58.2	59.5	58.7	58.7	58.3	58.4
	10	63.2	60.2	61.5	59.3	61.2	62.5	61.6	61.7	61.1	61.4
	11	66.0	62.8	64.1	61.9	63.7	65.1	64.1	64.3	63.7	64.0
	12	68.8	65.7	66.9	64.5	66.4	67.8	66.9	67.0	66.3	66.7
	13	72.1	68.7	70.2	67.7	69.6	70.8	70.0	70.2	69.4	69.8
	14	75.7	71.7	72.9	70.9	72.7	73.9	73.2	73.3	72.4	73.0
	15	78.7	74.7	76.0	73.9	75.9	77.2	76.3	76.7	75.3	76.1

puting the magnitude of its Fourier Transform, restricted to a straight line horizontally through the zero frequency point.

Table I compares the accuracy of the proposed approach with that of the blended Gaussian Pyramid approach. The proposed approach outperforms blended Gaussian Pyramid in terms of PSNR at $N = 4$, for *Rand512*, at $N = 5$ for the mean across images and at $N = 7$ for the worst-case image

TABLE II
COMPUTATIONAL COST RESULTS

	Ref method	Gaussian Pyramid	Proposed approach (using 2 to 15 basis functions)													
			2	3	4	5	6	7	8	9	10	11	12	13	14	15
F.P. Count (ops/pel)	17576.3	148.9	351.2	531.2	711.3	891.4	1071.5	1251.6	1431.7	1611.7	1791.8	1971.9	2152.0	2332.1	2512.1	2692.2
Time (μ s/pel)	755.5	0.20	12.2	17.1	22.8	28.1	34.1	39.5	44.9	52.2	59.0	65.2	81.5	83.3	84.8	83.8

(*Lena*). At $N = 8$, the PSNR improvement is typically 10 to 15 dB, with a 22.0 dB improvement in the case of *Rand512*. We choose therefore $N = 8$ as the number of basis functions to employ for high-quality Gaussian blurring.

C. Discussion

The main reason for the limited accuracy of the blended Gaussian Pyramid approach is its slower-decaying tail in the frequency and spatial domains, when compared to perfect Gaussian blurring and the proposed technique. This is due to the fact that each effective approximated Gaussian PSF is a weighted sum of two Gaussian PSFs, one twice the width of the other. This slower decay can be expected to act as a disadvantage in foveated coding, given that the aim is to remove high frequency components. An example of a cross section of the impulse response of the blended Gaussian Pyramid approach, when applied with a fixed blur level, is given in Fig. 11. The plot shows how the impulse response deviates notably from that of true Gaussian blurring in the outer tails of the curve.

Table II compares the costs of the proposed method, the reference method (i.e., independent filter for each blur level) and the blended Gaussian Pyramid method. In all cases, the counts of arithmetic operations were the same for all images. For $N = 5$, at 891.4 ops/pel, the proposed approach costs 6 times as much as Gaussian Pyramid, while typically providing only marginally better blurring accuracy. However, with $N = 8$, for which the proposed approach gave on average 13.5 dB better accuracy, the cost is 1431.7 ops/pel, which is less than 9% of the cost of the reference method and less than 10 times the cost of the blended Gaussian Pyramid approach.

Figure 12 provides a σ -dependent comparison between the proposed technique, blended Gaussian Pyramid and the *integral image* technique [14], [15] in the context of space-variant preblurring for coding. Each curve gives the relative increase in JPEG bitrate of the given technique when compared with the equivalent using perfect Gaussian blurring. Each bitrate was taken by applying a uniform blur map of the given σ value, to a 256-by-256 white noise image and encoding to a fixed quality level, measured using PSNR (fixed at 35dB in all cases). The *integral image* approach employed a square window width computed as 3.3σ then rounded to the nearest odd number (the ratio of 3.3 minimises the mean squared difference between a square window and a 2-D Gaussian window). The graph demonstrates the advantage of a direct substitution of blended Gaussian Pyramid with the proposed approach in a foveated coding scenario, showing how the bitrate advantages in a given region of an image will be dependent on the blur level in that region. The poorer general performance of *integral image* blurring demonstrates the advantage of instead using a blurring

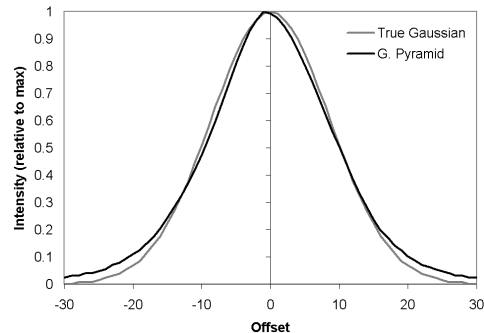


Fig. 11. An example of an impulse response of the blended Gaussian Pyramid approach and true Gaussian blurring, when $\sigma = 8$. Note the wider tails of blended Gaussian Pyramid. These were generated by applying the blurring to a synthetic image consisting of a narrow vertical bar, which had a deliberate offset from zero to demonstrate the fact that the effective impulse response of blended Gaussian Pyramid will not generally be perfectly central.

technique with a smooth point spread function. The poorer general performance of *integral image* blurring demonstrates the advantage of instead using a blurring technique with a smooth point spread function. The drop into negative percentages for *integral image* blurring below $\sigma = 1$ is a consequence of the round-to-nearest interpretation of blur levels that was applied, which causes this approach to be effectively given a higher blur level than the others, resulting in a lower bitrate. The average bitrate improvement with the proposed approach compared with blended Gaussian Pyramid, is 5.4%.

IV. CONCLUSION

We have proposed a specialized filter bank for high-precision approximation of smoothly space-variant Gaussian blurring. The filters are implemented using a fast DCT approach, to provide convolution with symmetric extension beyond image boundaries. These filters are an optimal basis from the perspective of spanning a given range of Gaussian point spread functions, and are computed using principal component analysis. As the number of basis functions employed is increased, the resultant blurring converges rapidly to true space-variant Gaussian blurring. Arbitrarily-perfect Gaussian blurring can be obtained depending on the number of basis functions used. The cost of the algorithm is the same regardless of the number of desired blurring levels or the complexity of the blur map.

Future work includes the extension of the proposed approach to video in order to investigate its benefits in a foveated coding scenario.

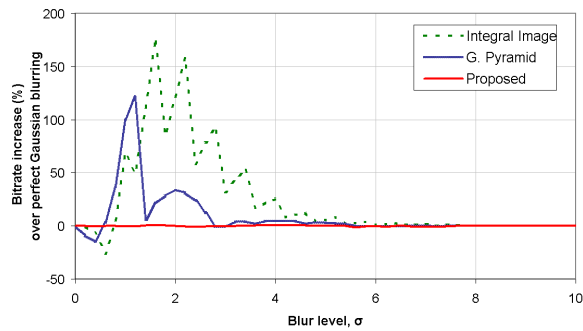


Fig. 12. A JPEG bitrate comparison between different blurring techniques as a function of blur level, σ . Each line gives the relative increase in JPEG bitrate of the given technique when compared with the equivalent using perfect Gaussian blurring.

REFERENCES

- [1] Z. Wang and A. C. Bovik, "Foveated image and video coding," in *Digital Video, Image Quality and Perceptual Coding*, H. R. Wu and K. R. Rao, Eds. CRC Press, 2006, ch. 14, pp. 431–457, ISBN 0-8247-2777-0.
- [2] S. Lee, M. S. Pattichis and A. C. Bovik, "Foveated video compression with optimal rate control," *IEEE Trans. Image Process.*, vol. 10, no. 7, pp. 911–992, Jul. 2001.
- [3] N. Dhavale and L. Itti, "Saliency-based multifoveated MPEG compression," in *Proc. 7th International Symposium on Signal Processing and Its Applications*, vol. 1, Jul. 2003, pp. 229–232.
- [4] C. Dikici, R. Civanlar and H. Isil Bozma, "Fovea based coding for video streaming," in *Image Analysis and Recognition (Proc. ICIAR)*, ser. Lecture Notes in Computer Science. Berlin / Heidelberg: Springer, Sep. 2004, vol. 3211, pp. 285–294, ISBN 978-3-540-23223-0.
- [5] A. T. Duchowski, "Acuity-matching resolution degradation through wavelet coefficient scaling," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1437–1440, Aug. 2000.
- [6] L. Itti, "Automatic foveation for video compression using a neurobiological model of visual attention," *IEEE Trans. Image Process.*, vol. 13, no. 10, pp. 1304–1318, Oct. 2004.
- [7] Y. Yitzhaky and E. Peli, "Vision-model-based image foveation and motion estimation," *Optical Engineering*, vol. 44, no. 10, pp. 107 004–1 – 107 004–11, Oct. 2005.
- [8] B. A. Wandell, *Foundations of Vision*. Sinauer Associates, 1995, ISBN 0-87893-853-2.
- [9] I. van der Linde, "Multi-resolution image compression using image foveation and simulated depth of field for stereoscopic displays," in *Proc. SPIE*, vol. 5291, 2004, pp. 71–80.
- [10] T. Popkin, A. Cavallaro and D. Hands, "Distance blurring for space-variant image coding," in *Proc. ICASSP*, Taipei, Taiwan, Apr. 2009, pp. 665–668.
- [11] J. Krivanek, J. Zara and K. Bouatouch, "Fast depth of field rendering with surface splatting," in *Proc. Computer Graphics International*, Los Alamitos, 2003, pp. 196–201.
- [12] R. S. Wallace, P.-W. Ong, B. B. Bederson and E. L. Schwartz, "Space-variant image processing," *International J. Computer Vision*, vol. 13, no. 1, pp. 71–90, Sep. 1994.
- [13] S. Lee and A. C. Bovik, "Fast algorithms for foveated video processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 2, pp. 149–162, Feb. 2003.
- [14] F. C. Crow, "Summed-area tables for texture mapping," in *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. New York, USA: ACM Press, 1984, pp. 207–212.
- [15] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004.
- [16] J. G. Proakis and D. G. Manolakis, *The Digital Signal Processing Handbook*, 2nd ed. Macmillan, 1992, ISBN 0-02-396815-X.
- [17] S. Tan, J. L. Dale and A. Johnston, "Performance of three recursive algorithms for fast space-variant Gaussian filtering," *J. Real-Time Imaging*, vol. 9, pp. 215–228, 2003.
- [18] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge University Press, 1992, ISBN 0-521-43108-5.
- [19] V. K. Madisetti and D. B. Williams, *The Digital Signal Processing Handbook*. CRC Press, 1997, ISBN 0-8493-8572-5.
- [20] J. S. Perry and W. S. Geisler, "Gaze-contingent real-time simulation of arbitrary visual fields," in *Proc. SPIE*, vol. 4662, Jun. 2002, pp. 57–69.
- [21] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Communications*, vol. COM-31, no. 4, pp. 532–540, Apr. 1983.
- [22] S. A. Martucci, "Symmetric convolution and the discrete sine and cosine transforms," *IEEE Trans. Signal Process.*, vol. 42, no. 5, pp. 1038–1051, May 1994.
- [23] Z. Wang, "Fast Algorithms for the Discrete W Transform and for the Discrete Fourier Transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-32, no. 4, pp. 1038–1051, Aug. 1984.
- [24] W. S. Geisler and J. S. Perry, "A real-time foveated multiresolution system for low-bandwidth video communication," *Proc. SPIE, Vol. 3299*, pp. 294–305, Jul. 1998.
- [25] "The USC-SIPI Image Database," <http://sipi.usc.edu/database/> (accessed Dec 18, 2008).
- [26] [Http://www.hlevkin.com/TestImages/classic.htm](http://www.hlevkin.com/TestImages/classic.htm) (accessed Dec 18, 2008).
- [27] "Kodak Lossless True Color Image Suite," <http://r0k.us/graphics/kodak/> (accessed Dec 18, 2008).