Available online at www.sciencedirect.com

SCIENCE DIRECT®

Real-Time Imaging 9 (2003) 215–228

www.elsevier.com/locate/rti

Performance of three recursive algorithms for fast space-variant Gaussian filtering

Sovira Tan*, Jason L. Dale, Alan Johnston

Department of Psychology, University College London, Gower Street, London WC1E 6BT, UK

Abstract

Animal visual systems have solved the problem of limited resources by allocating more processing power to central than peripheral vision. Foveation considerably reduces the amount of data per image by progressively decreasing the resolution at the periphery while retaining a sharp center of interest. This strategy has important applications in the design of autonomous systems for navigation, tracking and surveillance. Central to foveation is a space-variant Gaussian filtering scheme that gradually blurs out details as the distance to the image center increases. Unfortunately Gaussian convolution is a computationally expensive operation, which can severely limit the real-time applicability of foveation. In the space-variant case, the problem is even more difficult as traditional techniques such as the fast Fourier transform cannot be employed because the convolution kernel is different at each pixel. We show that recursive filtering, which was introduced to approximate Gaussian convolution, can be extended to the space-variant case and leads to a very simple implementation that makes it ideal for that application. Three main recursive algorithms have emerged, produced by independent derivation methods. We assess and compare their performance in traditional filtering applications and in our specific space-variant case. All three methods drastically cut down the cost of Gaussian filtering to a limited number of operations per pixel that is independent of the scale selected. In addition we show that two of those algorithms have excellent accuracy in that the output they produce differs from the output obtained performing real Gaussian convolution by less than 1%.

1. Introduction

Gaussian filtering has established itself as one of the most widely used image processing operations in computer vision as well as biological vision. It is often a pre-processing step in many feature extraction techniques [1, 2]. Gaussian functions also play a predominant part in biological vision modeling [3–6]. The need for a multiscale image representation has long been recognised in both computer and biological vision. The recently developed scale-space theory heavily relies on smoothing Gaussian kernels [7, 8]. Typically in a multiscale representation framework, an image is convolved with Gaussian kernels of different sizes to produce a set of images at different resolutions.

In the present paper we address a particular multiresolution scheme that builds a multiscale representation within the same image. The centre of the image, where the point of interest is supposed to be located, has high resolution while the sampling appears increasingly coarser as we go towards the periphery. Such a model is biologically motivated and is designed to emulate the size and disposition of receptive fields in the fovea and retina of primates. This technique possesses attractive features for real-time visual systems intended to work at video rates. It allows one to keep a central high-resolution region of interest while retaining a wide field of view. It considerably reduces the amount of data to be processed [9]. Among the many applications that can benefit from such a concise image representation are vision systems that are both computationally intensive and critically time constrained due to their dynamic nature. Image foveation has notably been proposed for navigation systems [10–12], video surveillance [13] and target tracking [14, 15]. The important bandwidth reduction that can be achieved by foveation has also spurred research on transmission-based applications such as videoconferencing or videophones [16, 17]. The large body of work that has already been carried out includes research on algorithms [18–20] but also on hardware.

The foveation process can be implemented directly by specifically designed hardware. Although some effort

*Corresponding author.
E-mail addresses: sovira.tan@ucl.ac.uk (S. Tan), j.dale@ucl.ac.uk (J.L. Dale), a.johnston@ucl.ac.uk (A. Johnston).

1077-2014/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S1077-2014(03)00040-8

was directed at special lenses [21], most of the research has concentrated on the sensor array level. CMOS technology [22–24] has in general been preferred to CCD [25] mainly because of advantages such as the possibility of random access to individual pixels, low cost of production, higher frame rate, easier integration of sensing and computation on the same chip, etc. [26]. However the design of a CMOS foveated array of sensors is far from trivial and difficulties include low fill factor, fixed pattern noise, etc. [23]. While dedicated hardware has not yet reached commercial stage, conventional CCD cameras are widely available and have proven high-quality imaging capabilities. Although they capture images on a uniformly sampled Cartesian grid, the foveation can be digitally implemented provided sufficiently fast algorithms can be applied.

A straightforward method for implementing foveation consists of filtering the image with Gaussian kernels in a space-variant manner and then sampling the resulting image non-uniformly, on a log-polar grid for example. Each point on this grid can be considered as the center of a receptive field. The problem with this scheme is the high computational cost of performing convolutions. For a convolution kernel of size $n \times n$ the cost is $n \times n$ multiplications per pixel. For a separable kernel, as the Gaussian is, this number can be reduced to $2n$. This figure can still be too high especially when the scale, σ, of the Gaussian is large. For a Gaussian of scale σ it is customary to use a kernel of size at least 8σ. This means for example that a Gaussian of scale $\sigma = 4$ would already require $2 \times 8 \times 4 = 64$ multiplications per pixel. This computational burden could be too heavy for a real-time visual system that has to perform, in addition to foveation, other demanding tasks such as motion computation, object recognition, etc.

Consequently much work has been done on speeding up Gaussian filtering [27–30]. Among the various methods investigated, the recursive approximation to Gaussian filtering has proved the most promising for the following reasons [31–36]:

(1) High degree of accuracy.
(2) Low number of operations per pixel.
(3) The number of operations per pixel is constant and does not depend on the scale.
(4) The ease with which it is possible to synthesize a Gaussian of any scale. This is crucial for our particular application since we want to be able to vary the scale within the same image in a continuous manner.

Recursive methods have recently been extended to include Gabor filtering [37] and anisotropic Gaussian filtering [38]. Research on recursive Gaussian filtering has given rise to three main techniques [33,35,36]. The three techniques were derived using very different methods. Deriche [33] approximates the Gaussian function in the space domain, Jin et al. [35] in the z domain and Vliet et al. [36] in the Fourier domain. They all use different optimisation techniques.

The objective of this paper is twofold:

(1) First we extend the applicability of recursive Gaussian filtering to the space-variant case. Deriche, Jin et al. and Vliet et al. only treated the case where the whole image is convolved with the same Gaussian kernel. We show that by treating the filters' coefficients as variables of space it is possible to approximate space-variant Gaussian filtering.
(2) We assess and compare the performance of the three recursive techniques. Our main criteria are accuracy compared to performing the filtering with real Gaussian kernels and speed taken as the number of operations required per pixel.

The next section provides a general background on recursive filtering and describes the filters derived by Deriche, Jin et al. and Vliet et al. Section 3 explains how it is possible to make recursive filtering space-variant. The performance of the three filters are evaluated and compared in Section 4 and finally Section 5 draws the general conclusion.

2. Recursive filtering

Although the three methods differ considerably in the way they were derived and the results they yield, they all rely on some properties of the z-transform that form the basis of recursive filtering theory [39].

The filtering of an input $x(n)$ by the kernel $h(n)$ is conveniently written as a multiplication in the z domain:

$$Y(z) = H(z)X(z), \tag{1}$$

where $X(z)$ and $H(z)$ are the z-transforms respectively of $x(n)$ and $h(n)$ and $Y(z)$ is the z-transform of the output $y(n)$. If, in addition, the filter $H(z)$ can be written as a ratio of polynomials,

$$H(z) = \frac{\sum_{n=0}^{N} a_n z^{-n}}{1 + \sum_{m=1}^{M} b_m z^{-m}}, \tag{2}$$

then, in the space domain, the filtering process can be written as:

$$y(n) = \sum_{i=0}^{N} a_i x(n-i) - \sum_{i=1}^{M} b_i y(n-i) \tag{3}$$

which is called a difference equation. It should be noted that in Eq. (2) the denominator of $H(z)$ corresponds to a recursion while the numerator corresponds to a convolution. The problem is that, following the definition of the z-transform,

$$H(z) = \sum_{n=-\infty}^{+\infty} h(n) z^{-n}, \tag{4}$$

a Gaussian kernel can easily be written as a convolution only filter, that is, with no denominator.

The challenge of recursive Gaussian filtering is to approximate the z-transform of a Gaussian by a ratio of polynomials. This would introduce a recursion and it is hoped that the recursion will considerably reduce the convolutive part.

2.1. Deriche filtering [33]

The standard definition of a 1Dimensional (1D) Gaussian function of scale σ is:

$$g_\sigma(n) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{n^2}{2\sigma^2}\right). \tag{5}$$

Deriche approximates the causal part (i.e. $n \geqslant 0$) of $g_\sigma(n)$ by a function of the form:

$$h_\sigma^+(n) = k\left\{a_0 \cos\left(\frac{w_0}{\sigma}n\right) + a_1 \sin\left(\frac{w_0}{\sigma}n\right)\right\}\exp\left(-\frac{b_0}{\sigma}n\right) + k\left\{c_0 \cos\left(\frac{w_1}{\sigma}n\right) + c_1 \sin\left(\frac{w_1}{\sigma}n\right)\right\}\exp\left(-\frac{b_1}{\sigma}n\right), \tag{6}$$

where $k = 1/\sigma\sqrt{2\pi}$ is the normalization constant and $a_0, a_1, b_0, b_1, c_0, c_1, w_0, w_1$ are constants determined using the optimization procedure E04FCF of the lnag library. Deriche then uses the property that functions of the form $r^n\cos(\omega_0 n)$ or $r^n\sin(\omega_0 n)$ have exact closed-form z-transforms that are ratio of polynomials [39]. Deriche investigated approximations of the second, third and fourth order. The higher the order, the higher the accuracy of the approximation but the higher the number of operations needed. For our purpose we found that a third order approximation was enough. The corresponding z-transform can be written:

$$H_\sigma^+(z) = \frac{n_0^+ + n_1^+ z^{-1} + n_2^+ z^{-2}}{1 + d_1^+ z^{-1} + d_2^+ z^{-2} + d_3^+ z^{-3}}, \tag{7}$$

where the coefficients n_0^+, n_1^+, n_2^+, d_1^+, d_2^+, d_3^+ are functions of the parameters a_0, a_1, b_0, b_1, c_0, c_1, w_0, w_1.

It should be noted that the + sign in $h_\sigma^+(n)$ and $H_\sigma^+(z)$ denotes the fact that those approximations are for the positive (causal) part of the Gaussian. Deriche determines the coefficients for the negative (anti-causal) part by symmetry and taking into account that the point $n = 0$ should not be counted twice. The z-transform of the anti-causal part of the filter takes the form:

$$H_\sigma^-(z) = \frac{n_1^- z + n_2^- z^2 + n_3^- z^3}{1 + d_1^- z + d_2^- z^2 + d_3^- z^3}, \tag{8}$$

where the coefficients n_1^-, n_2^-, n_3^-, d_1^-, d_2^-, d_3^- can be deduced from the causal coefficients n_0^+, n_1^+, n_2^+, d_1^+, d_2^+, d_3^+.

The complete z-transform of the Gaussian approximation is the sum of the causal and anti-causal parts:

$$H_\sigma(z) = H_\sigma^+(z) + H_\sigma^-(z). \tag{9}$$

This point is important and determines how the filter is to be implemented.

The filtering process in the space domain can easily be derived from the equivalence between Eqs. (2) and (3). $H_\sigma^+(z)$ determines a left-to-right recursion:

$$y^+(n) = n_0^+ x(n) + n_1^+ x(n-1) + n_2^+ x(n-2) - d_1^+ y^+(n-1) - d_2^+ y^+(n-2) - d_3^+ y^+(n-3) \tag{10}$$

$H_\sigma^-(z)$ determines a right-to-left recursion:

$$y^-(n) = n_1^- x(n+1) + n_2^- x(n+2) + n_3^- x(n+3) - d_1^- y^-(n+1) - d_2^- y^-(n+2) - d_3^- y^-(n+3) \tag{11}$$

The complete result of the filtering operation is the sum of the left-to-right and right-to-left recursions:

$$y(n) = y^+(n) + y^-(n). \tag{12}$$

For the readers interested in implementing the filter for themselves we give the numerical values for the coefficients in Appendix A. They are derived from the values provided by Deriche in Ref. [33].

2.2. Jin et al. filtering [35]

While Deriche looked for an approximation to the Gaussian function in the space domain, Jin et al. carried out their approximation procedure in the z domain. Following the definition given in Eqs. (4) and (5), the z-transform of a Gaussian can readily be obtained:

$$G_\sigma(z) = k \sum_{n=-\infty}^{+\infty} \alpha^{n^2} z^{-n}, \tag{13}$$

where $k = 1/\sigma\sqrt{2\pi}$ and $\alpha = \exp(-1/2\sigma^2)$. Eq. (13) is however not in the form of a ratio of polynomials and thus translates into a pure convolution in the space domain.

Jin et al. write the z-transform of the function they intend to use as a Gaussian approximation in the form of a ratio of polynomials:

$$S^+(z) = k\frac{1 + a_1 z^{-1} + a_2 z^{-2}}{(1 - pz^{-1})^3}, \tag{14}$$

where a_1, a_2, and p are coefficients to be determined so as to best approximate the Gaussian's z-transform. As for Deriche's filter the sign + denotes the causal part of the filter. It should be noted that the form of the denominator, $(1 - pz^{-1})^3$, means that Jin et al. choose the recursion to be of order 3 and that there is a unique pole of order 3. This guarantees, according to them, a maximum region of convergence.

Then using polynomial division, Jin et al. are able to write $S^+(z)$ as an infinite series in powers of z. The z-transform of a Gaussian being also written as an infinite series in powers of z (Eq. (13)), it is possible to compare the coefficients of $S^+(z)$ and those of the positive part of $G_\sigma(z)$. As there are only three unknown coefficients in the expression of $S^+(z)$, three points of $G_\sigma(z)$ are sufficient to provide a set of equations with solutions. It would be possible to determine the coefficients by a least-mean-squares technique if more points were used.

In this paper we carried out the derivation for three points because it does not require any numerical optimization procedure and provides tractable closed-form expressions for the filter's coefficients. It can be verified that the three first terms of the series (excluding $n = 0$) yield the set of equations:

$$3p + a_1 = \alpha, \tag{15a}$$

$$6p^2 + 3a_1p + a_2 = \alpha^4, \tag{15b}$$

$$6a_1p^2 + 3a_2p + 10p^2 = \alpha^9. \tag{15c}$$

Substituting the first two equations in the third we obtain the third order equation in p:

$$p^3 - 3\alpha p^2 + 3\alpha^4 p - \alpha^9 = 0. \tag{16}$$

This equation admits three solutions including $p = \alpha^3$. However, upon testing, the best solution was found to be the following:

$$p = \frac{\alpha}{2}\left(3 - \alpha^2 - \sqrt{9 - 6\alpha^2 - 3\alpha^4}\right). \tag{17}$$

All of the filter's coefficients can be deduced from p.

As for the Deriche filter, the complete Gaussian approximation filter $S(z)$ is the sum of the causal part $S^+(z)$ and the anti-causal part $S^-(z)$ that can be deduced from the causal part by symmetry.

For the readers interested in implementing Jin et al.'s filter for themselves, we provide the difference equations together with the expression of all the filter's coefficients in Appendix B.

2.3. *Vliet et al. filtering [36]*

While Deriche approximates the Gaussian function in the space domain and Jin et al. in the z domain, Vliet et al. carry out the procedure in the Fourier domain. They take advantage of the fact that a Gaussian has a simple closed-form Fourier transform (FT), which is, in fact, another Gaussian.

Like Jin et al., Vliet et al. start by writing their approximation function in the z domain. The manner in which they do that, however, differs from Jin et al.'s. Vliet et al. lay emphasis on the filter's poles rather than on the polynomial's coefficients, although it is simple to recover the coefficients from the poles:

$$H(z) = H^+(z)H^-(z) \tag{18}$$

with

$$H^+(z) = \prod_{i=1}^{N} \frac{d_i - 1}{d_i - z^{-1}} \text{ and } H^-(z) = (-1)^N \prod_{i=1}^{N} \frac{d_i - 1}{z - d_i},$$

where the d_i s are the poles of $H(z)$ and N is the order of the filter.

An inspection of Eq. (18) reveals the main differences between Vliet et al.'s filter compared with Deriche's and Jin et al.'s:

(1) Vliet et al.'s filter consists of the multiplication of the causal and anti-causal parts while for Deriche and Jin et al. it is the sum. This point has important consequences for the implementation of the filter.
(2) Vliet et al. force their filter to be purely recursive by having no polynomial in z in the numerator.

The FT of the filter is obtained from its z-transform by setting $z = \exp(i\Omega)$. The approximation process consists of finding the poles d_i that minimize the error measure:

$$L^2 = \sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi} (G_\sigma(\Omega) - H(\Omega))^2 d\Omega}, \tag{19}$$

where $G_\sigma(\Omega)$ is the FT of the Gaussian function to be approximated.

Because the gradient iterative minimization procedure Vliet et al. use to find the poles is a numerical method, they have to perform it for a chosen value of σ and the poles obtained are valid only for that particular value. In their paper they chose $\sigma = 2$. However Vliet et al. are able to extrapolate their results to any other value of σ by the use of the scaling transformation:

$$h\left(\frac{n}{q}\right) \leftrightarrow (d_i)^{1/q} \tag{20}$$

which means that scaling the filter $h(n)$ by a factor q in the space domain is equivalent to performing the transformation $d_i \rightarrow (d_i)^{1/q}$ on the poles of the filter's z-transform [40].

In theory, if we want to obtain a Gaussian of scale σ from a reference Gaussian of scale $\sigma = 2$, we should use the transformation factor $q = \sigma/2$. In practise this formula does not work well, a fact acknowledged by Vliet et al. in their paper. However the function obtained is a Gaussian, although of a scale different from the one expected. It is possible to directly compute the real scale of the obtained Gaussian using:

$$\sigma^2 = \sum_{n=-\infty}^{+\infty} n^2 h(n). \tag{21}$$

Trying a set of values for q and computing the effective scale σ obtained, we were able to determine the following relation between q and σ:

$$q = 0.0001\sigma^4 - 0.0021\sigma^3 + 0.0207\sigma^2 + 0.3797\sigma + 0.1763. \tag{22}$$

It should be noted that the fitting was done for the range $\sigma = [1, 4]$. For another range, a new appropriate fitting should be done.

Once the new poles are determined using the transformation factor q, the z-transform written as a function of its poles (Eq. (18)) has to be put in the orm of Eq. (2) so that the difference equation can be easily deduced. In this paper we will limit the order of the filter to 3 because higher order entails more operations and order 3 was found to yield sufficient accuracy. For $N = 3$ the causal and anti-causal filters can be written:

$$H^+(z) = \frac{\alpha}{1 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}, \tag{23a}$$

$$H^-(z) = \frac{\alpha}{1 + b_1 z + b_2 z^2 + b_3 z^3}, \tag{23b}$$

where the coefficients b_1, b_2, b_3 and α can be expressed as functions of the poles:

$$\begin{aligned} b_1 &= -b(d_1 d_2 + d_1 d_3 + d_2 d_3),\\ b_2 &= b(d_1 + d_2 + d_3),\\ b_3 &= -b,\\ b &= \frac{1}{d_1 d_2 d_3},\\ \alpha &= 1 + b_1 + b_2 + b_3. \end{aligned} \tag{24}$$

As a summary we give a step-by-step procedure to obtain a recursive Gaussian approximation using Vliet et al.'s method:

(1) Compute the transformation factor q from the chosen scale σ using Eq. (22) (which is for the range $\sigma = [1, 4]$).
(2) Compute the transformed poles $d_i = \left(d_i^0\right)^{1/q}$ from the original poles d_i^0. Those are for the value $\sigma = 2$ and are provided by Vliet et al.:

$$\begin{aligned} d_1^0 &= 1.4165 + \mathrm{i}1.00829,\\ d_2^0 &= 1.4165 - \mathrm{i}1.00829,\\ d_3^0 &= 1.86543. \end{aligned} \tag{25}$$

(3) Perform the left-to-right filtering using the causal difference equation:

$$v(n) = \alpha x(n) - b_1 v(n-1) - b_2 v(n-2) - b_3 v(n-3). \tag{26}$$

The coefficients α, b_1, b_2 and b_3 are obtained from the poles d_i using Eq. (24).
(4) Perform the right-to-left filtering on the output $v(n)$ of the left-to-right filtering. The final output of the filtering is given by

$$y(n) = \alpha v(n) - b_1 y(n+1) - b_2 y(n+2) - b_3 y(n+3). \tag{27}$$

This last step is radically different from Deriche and Jin et al.'s filters for which the final output is the sum of the causal and anti-causal parts. This is because Vliet et al. write the filter as the product of the causal and anti-causal parts (Eq. (18)) while Deriche and Jin et al. write it as the sum. This difference can have important consequences in a real-time implementation of the filters. With Deriche and Jin et al.'s filters, the causal and anti-causal filters can be implemented simultaneously in parallel while for Vliet et al.'s filter we have to wait for the output of the causal filter before we can start the anti-causal one.

3. Extension to the space-variant case

First it should be noted that the preceding exposition was 1D for the sake of simplicity. The extension to the 2D case is straightforward thanks to the separability of the Gaussian function. Once the 1D filtering has been carried out on every line of the processed image, the same 1-D filtering should be performed on every column of the resulting image replacing left-to-right by top-to-bottom and right-to-left by bottom-to-top. As for the Vliet et al.'s method the line and column filtering have to be implemented sequentially and not simultaneously because the 2D z-transform of a separable function is the product of two 1D z-transforms.

In the preceding chapters we have assumed, following the papers we referenced, that the filters' coefficients are constants. That means filtering the whole image with the same Gaussian, which is effectively what is required in most cases. In the special application we are investigating, however, the scale of the Gaussian needs to be changed at each pixel according to its location relative to the center of the fovea. Performing a full convolution with the desired Gaussian at each pixel would require prohibitive amounts of computations. We therefore want to investigate if recursive filtering can be modified to perform space-variant filtering and how accurate it can be.

The method we propose is to modify the difference equation (Eq. (3)) to treat the coefficients of the filter a_i and b_i as variables of the spatial coordinates (x, y). The coefficients depend only on the scale σ of the Gaussian, which we can make a space-variant function i.e.

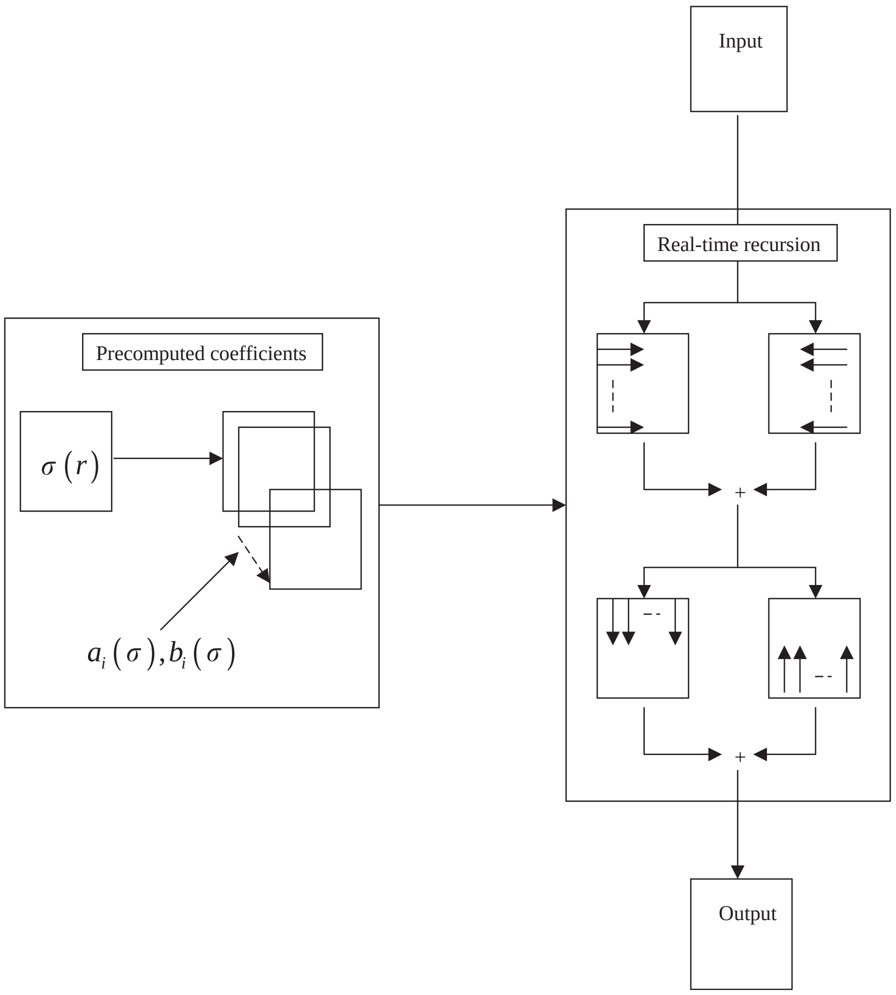

Fig. 1. Implementation of Deriche and Jin et al.'s filter.

$\sigma \rightarrow \sigma(x, y)$. In this case a_i and b_i become functions of the spatial coordinates too. In this paper we will assume circular symmetry so that the scale, and therefore the filter coefficients, are only variables of $r = \sqrt{x^2 + y^2}$. We can thus rewrite the general difference equation:

$$y(n) = \sum_{i=0}^{N} a_i(r)x(n-i) - \sum_{i=1}^{M} b_i(r)y(n-i). \quad (28)$$

Practically this means that we have to change the coefficients of the filter at each pixel. In the preceding chapters, we have expressed the coefficients of the three recursive filters as functions of σ (Eqs. (A), (B.1)–(B.3) and 24). Those equations allow us to synthesize Gaussians of any scales so that we can easily make the scale vary continuously from the center of the image to the periphery.

It would also be time-consuming to compute the coefficients at each point in realtime. However, it should be stressed that they can be precomputed, stored and then accessed in a look-up table manner. Only the difference equations need to be performed on the fly.

Figs. 1 and 2 are diagrams summarising the different steps in the implementation of the filters. It can be seen that Deriche and Jin et al.'s filters (Fig. 1) can lead to a more parallel implementation than Vliet et al.'s (Fig. 2).

4. Performance evaluation and comparison

For a more complete description of the filters, we measure their accuracy as Gaussian approximations in three different ways:

(1) We first examine the impulse response of the filters and compare them with real Gaussian functions.
(2) We consider the outputs of the filters obtained with a test image. We also convolve this test image with a real Gaussian kernel, which provides the reference for accuracy comparison. At this stage we do not

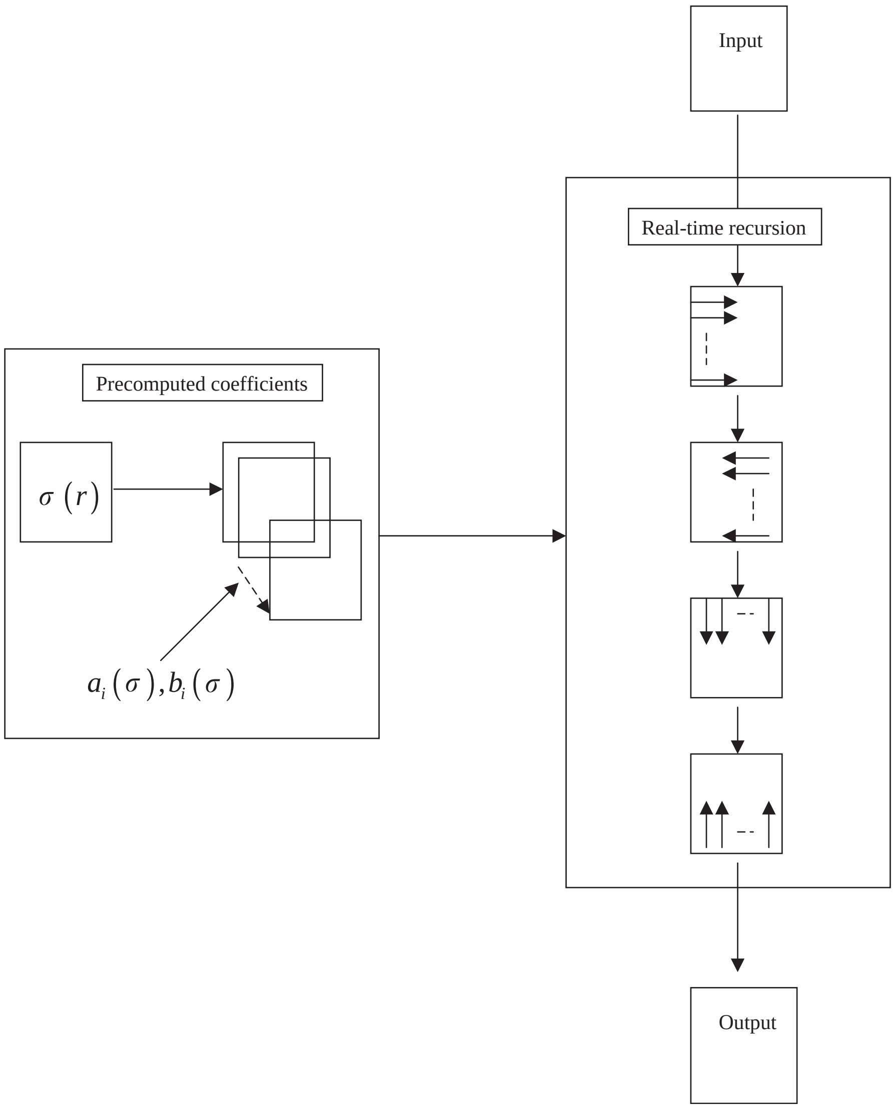

Fig. 2. Implementation of Vliet et al.'s filter.

yet vary the scale with space. The whole image is filtered with the same scale σ.

(3) Finally we test the filters in the space-variant case. Within the same image we make the scale vary from $\sigma = 1$ in the center to $\sigma = 4$ at the corners. We also perform the filtering painstakingly with different Gaussian kernels at each point to provide the comparison reference.

The impulse responses of the filters should be Gaussian functions. Plotting them against the true Gaussians they are designed to approximate give us a qualitative idea of their performance. Figs. 3, 4 and 5 respectively show the 1D impulse responses of Deriche's, Jin et al.'s and Vliet et al.'s filters for four values of σ. The true Gaussian is plotted in diamonds. It can be seen that Deriche's filter (Fig. 3) appears to provide a good fit at all scales. Jin et al.'s filter (Fig. 4) seems to fit the Gaussian perfectly at low scales but its results degrade as the scale increases. A possible explanation for that behavior is that Jin et al.'s filter is derived on matching only three points of the true Gaussian. For a small scale a few points are enough to describe the Gaussian completely because it decays rapidly. However as the scale increases, it takes more points for the Gaussian to decay and those additional points are not taken in account by Jin et al.'s filter. This is corroborated by Fig. 4 where it can be seen that at $\sigma = 4$ the impulse response of Jin et al.'s filter does not decay as much as the true Gaussian. As for Vliet et al.'s filter (Fig. 5), some degree of mismatch is noticeable at $\sigma = 1$ but the filter's accuracy improves as σ increases. It is also useful to visualize the 2D impulse responses of the filters as, eventually, it is a 2D filter we wish to apply. Fig. 6 provides 3D views of the filters as well as a 2D section

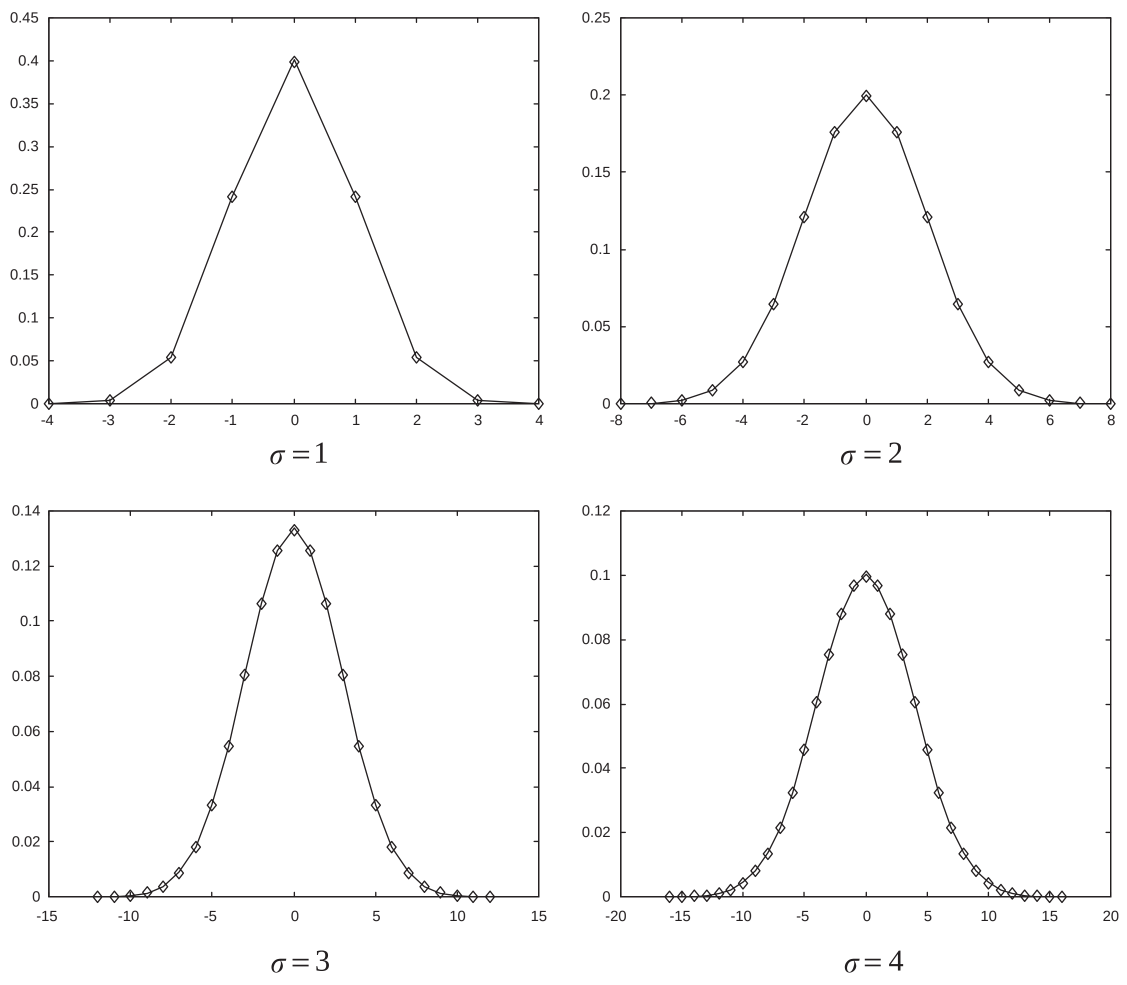

Fig. 3. 1D impulse response of Deriche's filter. The real Gaussian is plotted in diamonds.

view. The scale is $\sigma = 4$. It can be verified that all three filters have good isotropy.

We also want a quantitative measure of the filters' comparison with real Gaussians. We use a normalized root-mean-square (rms) error measure defined as:

$$\Delta_1 = \frac{1}{N_D} \sum_{(i,j)\in D} \frac{\sqrt{(h(i,j) - g_\sigma(i,j))^2}}{g_\sigma(i,j)}, \tag{29}$$

where $h(i,j)$ is the impulse response of the filter and $g_\sigma(i,j)$ is the actual Gaussian. To count only significant values of the Gaussian, the domain D is restricted to points whose distance is shorter than 3σ from the center of the Gaussian. N_D is the number of points in D. From Eq. (5) it is easy to verify that at $n = 3\sigma$ the Gaussian only retains 1.11% of its maximum value.

We evaluate Δ_1 for four different scales, $\sigma = 1, 2, 3, 4$. The results are provided in Table 1. Deriche's filter appears to be the most accurate and Jin et al.'s the least. Both Deriche and Vliet et al.'s filters yield better accuracy as the scale increases. By contrast Jin et al.'s filter has excellent accuracy for $\sigma = 1$ but its performance rapidly degrades as σ increases. The behavior of Vliet et al.'s filter could be explained in the same way as that of Jin et al. The difference is that Vliet et al. carry out their approximation procedure in the frequency domain. It is well known that a small Gaussian in the space domain will yield a large Gaussian in the Fourier domain and conversely. Using a limited number of points in the frequency domain it is therefore easier to describe a large Gaussian completely than a small one.

Another way of looking at the filters' performance is to examine their outputs obtained with a test image. This is a pragmatic approach as in the end it is the processed image that is of practical interest. The test image we used is an array of uniform random noise shown in the top left of Fig. 7. The reason for using that test image is that the effects of space-variant filtering are more visible on uniform noise. That is because uniform noise has a flat power spectrum where all frequencies are equally represented whereas for a natural image the power spectrum typically falls off rapidly as frequencies

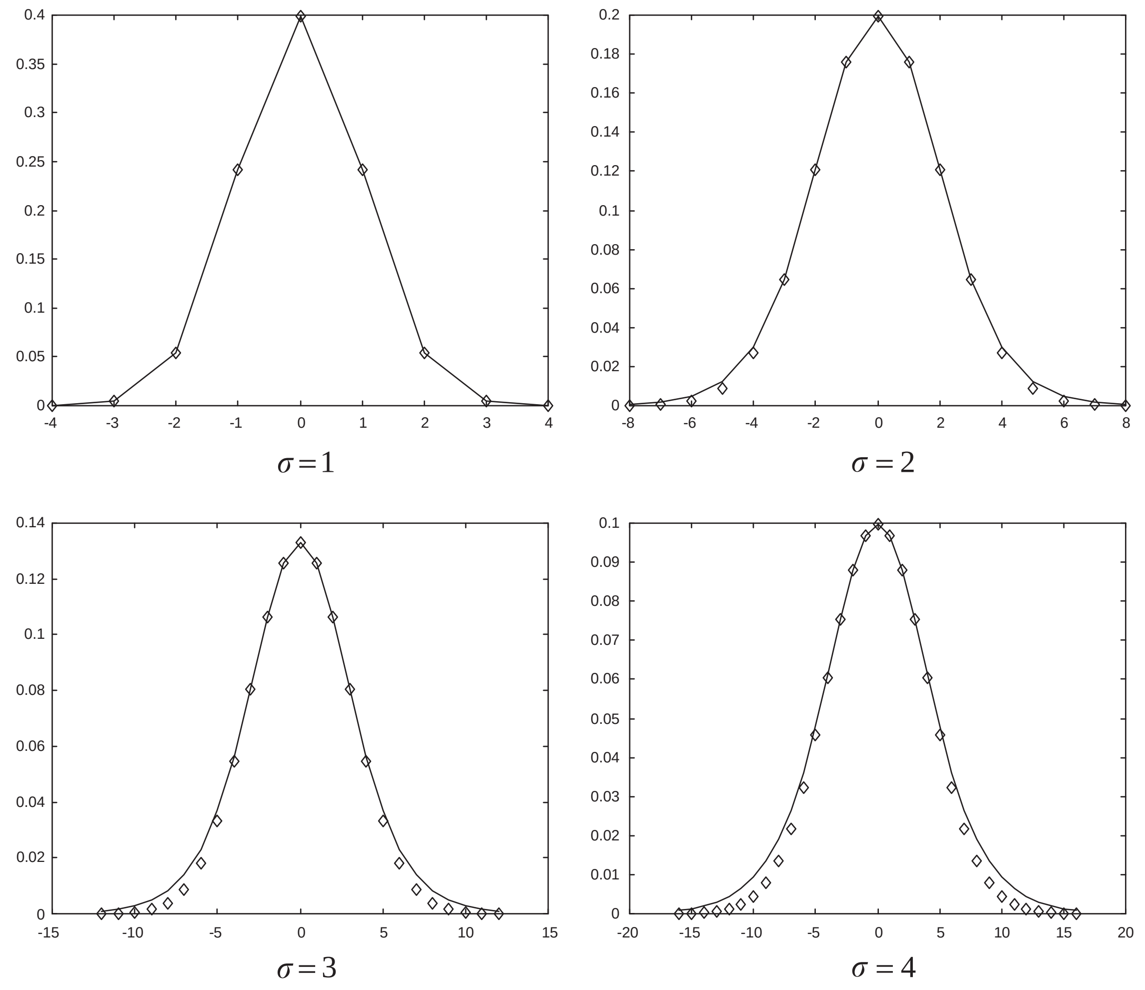

Fig. 4. 1D impulse response of Jin et al.'s filter. The real Gaussian is plotted in diamonds.

increase. As a quantitative accuracy measure, we use a normalized root-mean-square (rms) error similar to Δ_1:

$$\Delta_2 = \frac{1}{N}\sum_{(i,j)} \frac{\sqrt{\left(I_h(i,j) - I_g(i,j)\right)^2}}{I_g(i,j)}, \tag{30}$$

where I_h represents the test image processed with a recursive filter and I_g the same image processed with real Gaussian kernels. N in the total number of pixel in the image. We first investigate the non space-variant case where the whole image is filtered with the same value of σ. We use the same four values of σ as in the first test. The results are given in Table 2.

Although Δ_1 and Δ_2 seem to represent the same error measure, Δ_2 appears to produce dramatically lower figures than Δ_1. This is probably because in the first test, by normalizing the error, we give an equal weight to every point of the Gaussian. By contrast, when an image is convolved with a Gaussian, the resulting output at each pixel is mainly determined by the central values of the Gaussian.

We note that Deriche and Vliet et al.'s filters perform better with increasing scale while Jin et al.'s filter displays the opposite behavior. That is consistent with the first test. However, while Deriche's filter seemed more accurate in the impulse response comparison, Vliet et al.'s filter performs better when it comes to filtering actual images (except for the smallest scale $\sigma = 1$). This might be because Vliet et al.'s filter models the central part of the Gaussian function better than Deriche's. In fact, both Deriche and Vliet et al.'s filters yield very good results. The error is below 1% in all cases and can, as σ increases, become much lower. That suggests that using a recursive filter in the place of a costly Gaussian convolution can be a practical solution.

Finally we test the filters in the space-variant case. We make the scale vary from $\sigma = 1$ in the center of the same test image (top left of Fig. 7) to $\sigma = 4$ in the corners. The way in which σ varies, or, in other words, the form of the function $\sigma(r)$, can be chosen by the user. It could for example be made to fit the available data on the size of receptive fields in the retina or alternatively the cortical

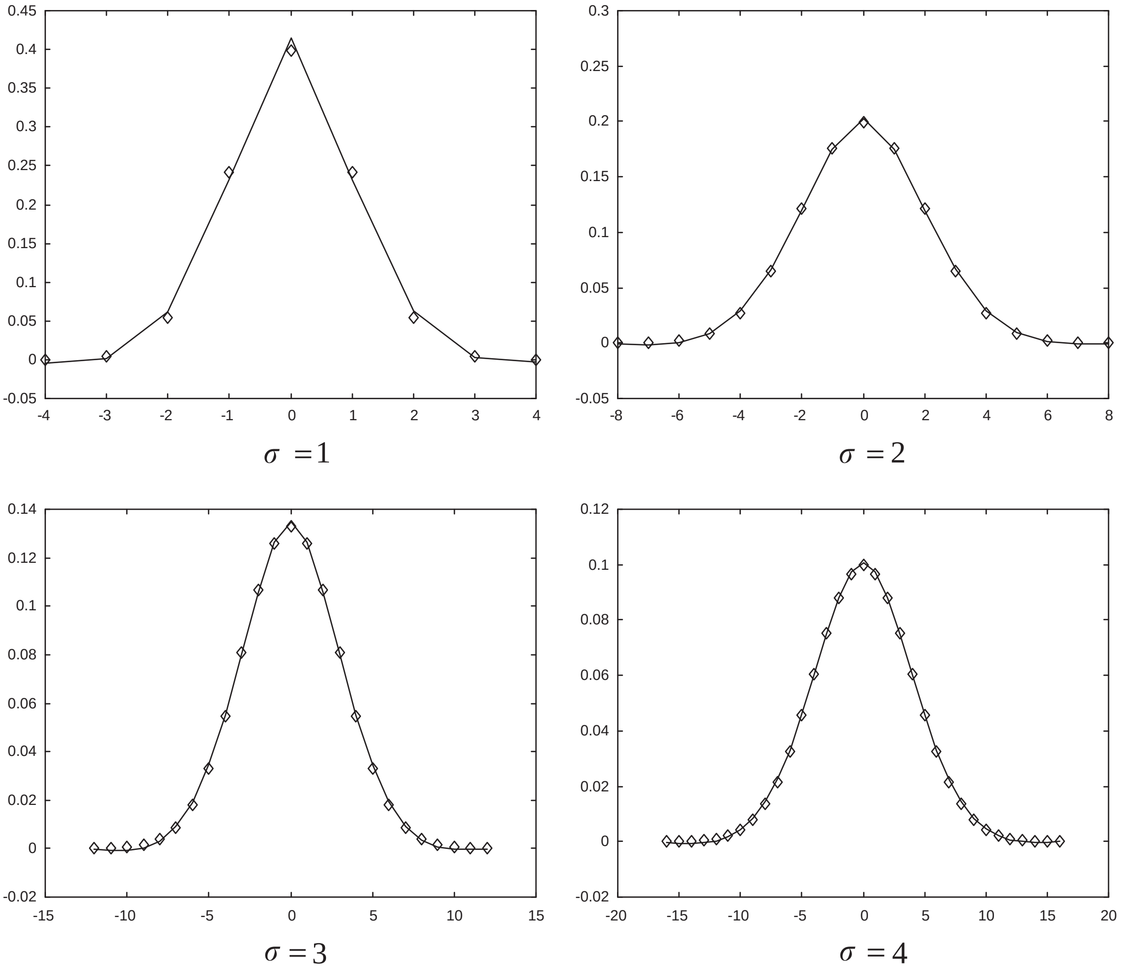

Fig. 5. 1D impulse response of Vliet et al.'s filter. The real Gaussian is plotted in diamonds.

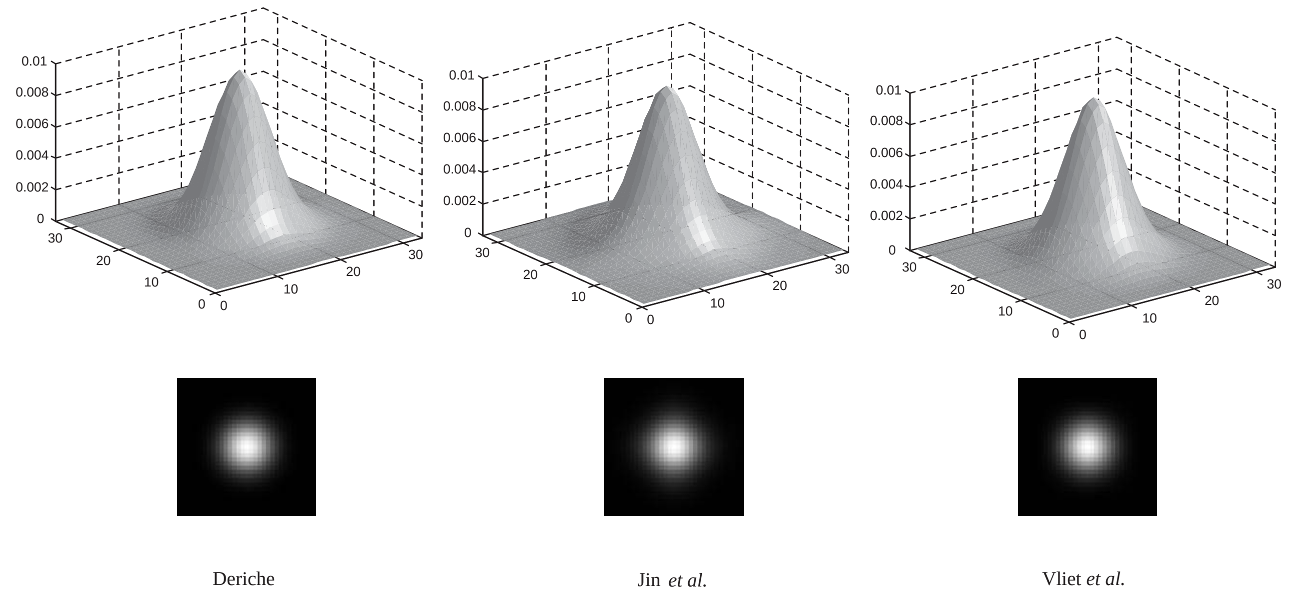

Fig. 6. 3D and section view of the 2-D impulse response of the three filters.

magnification [41–43]. For our test we chose two simple functions, one linear and one sinusoidal, that we respectively call $\sigma_1(r)$ and $\sigma_2(r)$:

$$\sigma_1(r) = 1 + 3\frac{r}{L_0}, \tag{31}$$

$$\sigma_2(r) = 2.5 + 1.5\cos\left(\pi\left(\frac{r}{L_0} - 1\right)\right), \tag{32}$$

where L_0 is the distance between the center of the image and any of its corners. The space-variant filtering is implemented in the way described in Section 3. Fig. 7 shows the results for the sinusoidal scale function. As for the convolution operation, the recursive filter is not valid for the points at the borders of the image. That is because recursive filtering is based on the output of previous points and those pixels do not have previous neighbors. Consequently those points are not shown in Fig. 7, and they are not counted in the error estimation. The original size of all images is 256×256.

For a quantitative assessment we use the same error measure as for the second test. The results, shown in the first two lines of Table 3, are consistent with the previous test. Vliet et al.'s filter still appears to be the best performing and Jin et al.'s the worse. It should be noticed that for all three filters the error is bigger for the

Table 1
Normalized rms error (%) for the 2D impulse response of the three filters

	$\sigma=1$	$\sigma=2$	$\sigma=3$	$\sigma=4$
Deriche	2.0	0.84	0.85	0.79
Jin et al.	0	18	41	56
Vliet et al.	19	7.3	6.0	5.5

Table 2
Normalized rms error (%) of the filters measured on their outputs obtained with the test image

	$\sigma=1$	$\sigma=2$	$\sigma=3$	$\sigma=4$
Deriche	0.61	0.48	0.36	0.30
Jin et al.	0	4.7	12	18
Vliet et al.	0.93	0.18	0.10	0.082

Table 3
Summary of the main characteristics of the filters

	Deriche	Jin et al.	Vliet et al.
Space-variant rms accuracy (linear) (%)	0.43	8.0	0.18
Space-variant rms accuracy (sinusoidal) (%)	0.46	8.1	0.23
Number of multiplications per pixel	24	24	16
Possibility of parallel processing for the causal and anti-causal parts	Yes	Yes	No
Number of points used in the filter's derivation	1000	3	Not known

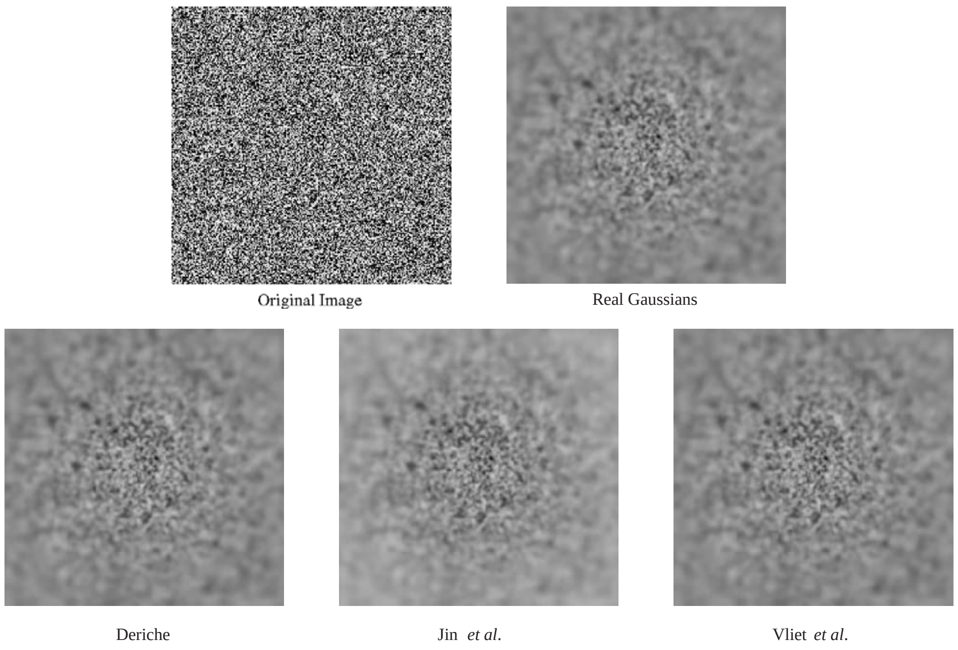

Fig. 7. Original test image and outputs of the space-variant filters with the sinusoidal scale mapping.

sinusoidal mapping of σ than for the linear one. This might be due to the steeper slope of $\sigma_2(r)$ in the middle of its curve. Jin et al.'s filter seems to be the most robust to the mapping change with a degradation of only 1.2% while Vliet et al.'s normalized rms error went up by 28%. However, even with the worst degradation Vliet et al.'s filter remains by far the most accurate filter. Even in the sinusoidal case, Vliet et al.'s error is only half of Deriche's. It should be stressed that both Deriche and Vliet et al.'s filter have very good performance in the two scale mappings with less than 0.5% of normalized rms error.

The relative weakness of Jin et al.'s filter can be explained if we recall that the derivation of its coefficients was based on only 3 points of a real Gaussian. By comparison Deriche used 1000 points in his optimization procedure. Jin et al.'s method could be used with more points but we would have to resort to a numerical procedure and would lose the benefit of a compact closed-form expression for the filter's coefficients. Of the three filters Jin et al.'s is the only one that relies uniquely on non-numerical expressions (Eq. (17) and (B.3)). Considering the fact that is was derived with only 3 points of a real Gaussian, the results of Jin et al.'s filter can in fact be viewed as rather impressive. In the two scale mappings its normalized rms error is only of 8%.

The other crucial factor to be taken into account when assessing the filters' performance is the computational complexity that will determine the speed at which the filters can be implemented. The number of multiplications required per pixel is a good indicator of a filter's complexity. By examining the filters' difference equations it is easy to count the number of multiplications required per pixel. It should be reminded that there are four passes per pixel, two horizontal and two vertical. With that in mind we obtain that both Deriche and Jin et al.'s filters require 24 multiplications per pixel whereas Vliet et al.'s needs only 16. Those figures are independent of the value of σ chosen. For comparison, performing a direct convolution (two 1D convolutions) with Gaussian kernels would require 16 multiplications per pixel for $\sigma = 1$ and 64 for $\sigma = 4$.

5. Conclusion

Table 3 sums up the main facts about the three recursive filters. First it should be stressed that both Deriche and Vliet et al.'s filters appear to be convincing candidates to approximate Gaussian convolution at high speed. The error they produce when compared to the output obtained with real Gaussian kernels is less than 0.5% in the space-variant case and less than 1% in the constant scale case. Both filters require a competitively low number of multiplications per pixel. Moreover their performance further improves with increasing scales. Vliet et al.'s filter appears to be more attractive in both accuracy and low complexity. However it should be reminded that Deriche's filter allows a more parallel implementation. This could cancel out Vliet et al.'s advantage. In addition, the first two error tests both indicate that Deriche's filter is more accurate at small scales. This point could have its importance depending on how large one wants the fovea to be.

It should be noted that we have not used Jin's method to its full potential. If one is willing to give up elegant coefficient expressions for numerical optimization procedures, Jin et al.'s filter could prove to be as competitive as Deriche's or Vliet et al.'s.

Finally, recursive filtering appears to be the natural way to perform space-variant Gaussian processing because of the ease with which the scale can be changed within the same image. Changing the scale simply entails changing the filter's coefficients and those coefficients are continuous functions of σ, which allows Gaussians of any scale to be synthesized.

Acknowledgements

The research was supported by the Project Grant ECOVISION, IST-2001-32114.

Appendix A. Coefficients for Deriche filtering

$$\begin{aligned}
d_1^+ &= d_1^- = -2\cos\left(\frac{1.475}{\sigma}\right)\exp\left(-\frac{1.512}{\sigma}\right) - \exp\left(-\frac{1.556}{\sigma}\right),\\
d_2^+ &= d_2^- = 2\cos\left(\frac{1.475}{\sigma}\right)\exp\left(-\frac{3.068}{\sigma}\right) + \exp\left(-\frac{3.024}{\sigma}\right),\\
d_3^+ &= d_3^- = -\exp\left(-\frac{4.58}{\sigma}\right),\\
n_0^+ &= 1.0051k,\\
n_1^+ &= k\left\{1.021\sin\left(\frac{1.475}{\sigma}\right) - 2.9031\cos\left(\frac{1.475}{\sigma}\right)\right\}\\
&\quad \exp\left(-\frac{1.512}{\sigma}\right) + 0.8929k\exp\left(-\frac{1.556}{\sigma}\right),\\
n_2^+ &= -k\left\{1.021\sin\left(\frac{1.475}{\sigma}\right) + 0.8929\cos\left(\frac{1.475}{\sigma}\right)\right\}\\
&\quad \exp\left(-\frac{3.068}{\sigma}\right) + 1.898k\exp\left(-\frac{3.024}{\sigma}\right),\\
n_1^- &= n_1^+ - d_1^+ n_0^+,\\
n_2^- &= n_2^+ - d_2^+ n_0^+,\\
n_3^- &= -d_3^+ n_0^+,
\end{aligned} \tag{A}$$

where $k = \dfrac{1}{\sigma\sqrt{2\pi}}$.

Appendix B. Coefficients for Jin et al. filtering

$$S^+(z) = k\frac{1 + a_1z^{-1} + a_2z^{-2}}{1 + b_1z^{-1} + b_2z^{-2} + b_3z^{-3}} \quad \leftrightarrow y^+(n) = k\{x(n) + a_1x(n-1) + a_2x(n-2)\} - \{b_1y^+(n-1) + b_2y^+(n-2) + b_3y^+(n-3)\}, \tag{B.1}$$

$$S^-(z) = k\frac{a_3z + a_4z^2 + a_5z^3}{1 + b_1z + b_2z^2 + b_3z^3} \quad \leftrightarrow y^-(n) = k\{a_3x(n+1) + a_4x(n+2) + a_5x(n+3)\} - \{b_1y^-(n+1) + b_2y^-(n+2) + b_3y^-(n+3)\} \tag{B.2}$$

$$\begin{aligned} b_1 &= -3p, \\ b_2 &= 3p^2, \\ b_3 &= -p^3, \\ a_1 &= \alpha - 3p, \\ a_2 &= \alpha^4 - 3\alpha p + 3p^2, \\ a_3 &= a_1 - b_1, \\ a_4 &= a_2 - b_2, \\ a_5 &= -b_3. \end{aligned} \tag{B.3}$$

As for Deriche's filter the final result is the sum of the left-to-right recursion, $y^+(n)$, and the right-to-left recursion, $y^-(n)$.

References

[1] Canny J. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 1986;8:679–98.

[2] Unser M, Eden M. Multiresolution feature extraction and selection for texture segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 1989;11:717–28.

[3] Marr D. Vision. New York. Freeman; 1982.

[4] Koenderink JJ, van Doorn AJ. Representation of local geometry in the visual system. Biological Cybernetics 1987;55: 367–75.

[5] Watson AB. The cortex transform: rapid computation of simulated neural images. Computer Vision, Graphics and Image Processing 1987;39:311–27.

[6] Johnston A, McOwan PW, Benton CP. Robust velocity computation from a biologically motivated model of motion perception. Proceedings of the Royal Society London B 1999;266:509–18.

[7] Witkins AP. Scale-space filtering. In: Proceedings of the Eighth International Joint Conference on Artificial Intellegence, Karlsruhe, West Germany, August, 1983. p. 1019–22.

[8] Lindeberg T. Scale-space theory in computer vision. Dordrecht: Kluwer Academic Publishers; 1994.

[9] Bolduc M, Levine MD. A review of biologically motivated space-variant data reduction model for robotic vision. Computer Vision and Image Understanding 1998;69:170–84.

[10] Weiman C. Log-polar vision for mobile robot navigation. In: Proceedings of the Electronic Imaging Conference, Boston, MA, November, 1990. p. 382–5.

[11] Tistarelli M, Sandini G. On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. IEEE Transactions on Pattern Analysis and Machine Intelligence 1993;15:401–10.

[12] Toepfer C, Wende M, Baratoff G, Neumann H. Robot navigation by combining central and peripheral optical flow detection on a space variant map. In: Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia, August, 1998. p. 1804–7.

[13] Comaniciu D, Berton F, Ramesh V. Adaptive resolution system for distributed surveillance. Real-Time Imaging 2002;8:427–37/ doi:10.1006/rtim.2002.0298.

[14] Weiman C, Juday RD. Tracking algorithms for log-polar mapped image coordinates. In: SPIE-Intelligent Robots and Computer Vision VIII: Algorithms and Techniques 1989;1192:843–53.

[15] Bernardino A, Santos-Victor J, Sandini G. Foveated active tracking with redundant 2D parameters. Robotics and Autonomous Systems, 2002;39:205–21.

[16] Wallace RS, Bederson BB, Schwartz EL. Voice-bandwidth visual communications through logmaps: The Telecortex. In: Proceedings of IEEE Workshop on Applications of Computer Vision, Palms Springs, CA, November, 1992. p. 4–10.

[17] Ferrari F, Nielsen J, Questa P, Sandini G. Space variant sensing for personal communication and remote monitoring. In: Proceedings of the EU-HCM Smart Workshop, Lisbon, Portugal, April 1995.

[18] Wallace RS, Ong PW, Bederson BB, Schwartz EL. Space variant image processing. International Journal of Computer Vision 1994;13:71–90.

[19] Dias J, Araujo H, Parades C, Batista J. Optical normal flow estimation on log-polar images. A solution for real-time binocular vision. Real-Time Imaging 1997;3:213–28.

[20] Bolduc M, Levine MD. A real-time foveated sensor with overlapping receptive fields. Real-Time Imaging 1997;3: 195–212.

[21] Basu A, Licardie S. Modeling fish-eye lenses. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, 1993. p. 1822–8.

[22] Wodnicki R, Roberts GW, Levine MD. A foveated image sensor in standard CMOS technology. In: Custom Integrated Circuits Conference, Santa Clara, CA, May 1995.

[23] Pardo F, Dierickx B, Scheffer D. Space-variant nonorthogonal structure CMOS image sensor design. IEEE Journal of Solid-State Circuits 1998;33:842–9.

[24] Ohtsuka Y, Hamamoto T, Aizawa K. A new image sensor with space variant sampling control on a focal plane. IEICE Transactions on Information Systems 2000;E83-D(7):1331–7.

[25] Van der Spiegel J, Kreider G, Claeys C, Debusschere I, Sandini G, Dario P, Fantini F, Belluttti P, Soncini G. A foveated retina-like sensor using CCD technology. In: Mead C, Ismael M, editors. Analog VLSI and neural network implementations. Dordrecht: Kluwer Academic Publishers; 1989.

[26] Fossum ER. CMOS image sensors: electronic camera-on-a-chip. IEEE Transactions on Electron Devices 1997;44:1689–98.

[27] Burt PJ. Fast filter for image processing. Computer Graphics and Image Processing 1981;16:20–51.

[28] Burt PJ. Fast algorithms for estimating local image properties. Computer Graphics and Image processing 1983;21:368–82.

[29] Crowley JL, Stern RM. Fast computation of the difference of low-pass transform. IEEE Transactions on Pattern Analysis and Machine Intelligence 1984;6:212–22.

[30] Wells WM. Efficient synthesis of Gaussian filters by cascaded uniform filters. IEEE Transactions on Pattern Analysis and Machine Intelligence 1986;8:234–9.

[31] Deriche R. Fast algorithms for low-level vision. IEEE Transactions on Pattern Analysis and Machine Intelligence 1990;12: 78–87.

[32] Deriche R. 1992. Recursively implementing the Gaussian and its derivatives. In: Proceedings of the Second International Conference on Image Processing, Singapore, September 1992. p. 263–7.

[33] Deriche R. Recursively implementing the Gaussian and its derivatives. Research Report 1893, INRIA, France, 1993.

[34] Young IT, van Vliet LJ. Recursive implementation of the Gaussian filter. Signal Processing 1995;44:139–51.

[35] Jin, JS, Gao Y. Recursive implementation of LoG Filtering. Real-Time Imaging 1997;3:59–65.

[36] van Vliet LJ, Young IT, Verbeek PW. Recursive Gaussian derivative filters. In: Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia, August 1998. p. 509–14.

[37] Young IT, van Vliet LJ, van Ginkel M. Recursive Gabor filtering. IEEE Transactions on Signal Processing 2002;50:2798-805/ doi:10.1190/TSP.2002.804095.

[38] Geusebroek JM, Smeulders AWM, van de Weijen J. Fast anisotropic Gauss filtering. In: 7th European Conference on Computer Vision, Copenhagen, Danemark, 2002. p. 99–112.

[39] Proakis J, Manolakis D. Digital signal processing: principles, algorithms and applications. Englewood Cliffs, NJ: Prentice-Hall; 1996.

[40] Oppenheim AV, Willsky AS, Nawab H. Signals & systems. Englewood cliffs; NJ: Prentice-Hall; 1997.

[41] Schwarz EL. Computational anatomy and functional architecture of striate cortex: a spatial mapping approach to perceptual coding. Vision Research 1980;20:645–69.

[42] Dow BM, Snyder AZ, Vautin RG, Bauer R. Magnification factor and receptive field size in foveal striate cortex of the monkey. Exp. Brain Res., 1981;44:213–28.

[43] Johnston A. The geometry of the striate topographic map. Vision Research 1989;29:1493–500.