Recursive implementation of the Gaussian filter

Article *in* Signal Processing · June 1995

DOI: 10.1016/0165-1684(95)00020-E · Source: CiteSeer

2 authors:

Ian T Young
Delft University of Technology
211 PUBLICATIONS **5,963** CITATIONS
SEE PROFILE

Lucas J Van Vliet
Delft University of Technology
340 PUBLICATIONS **5,752** CITATIONS
SEE PROFILE

ELSEVIER

Signal Processing 44 (1995) 139–151

SIGNAL PROCESSING

Recursive implementation of the Gaussian filter

Ian T. Young*, Lucas J. van Vliet

Pattern Recognition Group, Faculty of Applied Physics, Lorentzweg 1, Delft University of Technology, NL-2628 CJ Delft, Netherlands

Received 29 September 1994; revised 25 January 1995

Abstract

In this paper we propose a recursive implementation of the Gaussian filter. This implementation yields an infinite impulse response filter that has six MADDs per dimension independent of the value of σ in the Gaussian kernel. In contrast to the Deriche implementation (1987), the coefficients of our recursive filter have a simple, closed-form solution for a desired value of the Gaussian σ. Our implementation is, in general, faster than (1) an implementation based upon direct convolution with samples of a Gaussian, (2) repeated convolutions with a kernel such as the uniform filter, and (3) an FFT implementation of a Gaussian filter.

Zusammenfassung

Dieser Beitrag schlägt eine rekursive Implementierung von Gaussfiltern vor. Die Implementierung führt auf ein rekursives (IIR) Filter mit sechs MADDs pro Dimension, unabhängig vom Wert σ des Gaussimpulses. Im Gegensatz zur Implementierung nach Deriche (1987) läßt sich hier für die Filterkoeffizienten eine einfache geschlossene Lösung in Abhängigkeit von σ angeben. Das vorgeschlagene Verfahren ist im allgemeinen schneller als (1) eine Implementierung durch direkte Faltung mit Abtastwerten eines Gaussimpulses, (2) wiederholte Faltungen mit einem Basisfilter wie z.B. einem Rechteckfilter und schneller als (3) eine FFT-Implementierung des Gaussfilters.

Résumé

Nous propsons dans cet article une implantation récursive du filtre gaussien. Cette implantation produit un filtre à réponse impulsionnelle infinie ayant six MADD par dimension indépendamment de la valeur de σ dans le noyau gaussien. En contraste avec l'implantation de Deriche (1987), les coefficients de notre filtre récursif ont une forme analytique simple pour une valeur de gaussienne σ désirée. Notre implantation est, en général, plus rapide que (1) une implantation basée sur une convolution directe avec les échantillons d'une gaussienne, (2) des convolutions répétées avec un noyau tel que celui d'un filtre uniforme, et (3) une implantation FFT du filtre gaussien.

Keywords: Gaussian filtering, IIR filters; Recursive filtering; Multi-dimensional filtering; Smoothed derivative operators

*Corresponding author. Tel.: + 31-15-278-5390. Fax: + 31-15-278-6740. E-mail: young@ph.tn.tudelft.nl.

1. Introduction

Gaussian filters have assumed a central role in image filtering because of research in models of human vision [6], methods for edge detection [2, 7], results in scale space [5, 14], and techniques for accurate measurement of analog quantities based on digital data [11, 12].

The implementation of the Gaussian filter in one or more dimensions has typically been done as a convolution with samples of the required Gaussian (Eq. (1)), as repeated convolutions with a simpler filter such as a uniform filter (Eq. (2)), or as a recursive filtering with an approximation to the Gaussian that requires a complicated procedure to determine the filter coefficients [3]. The repeated convolution approach appeals to the central limit theorem which shows that, in the limit, repeated convolutions with an impulse response such as a simple uniform filter lead to an equivalent convolution with a Gaussian filter [9].

The discrete convolution with a sampled Gaussian is given per dimension by

$$\begin{aligned}\text{out}[n] &= \sum_{k=-N_0}^{k=+N_0} \text{in}[k-n]g[k] \\ &= \text{in}[n]\otimes g[n], \end{aligned} \tag{1a}$$

where

$$g[n] = g(x\,|\,\sigma)|_{x=n} = \frac{1}{\sqrt{2\pi}\sigma}\exp(-x^2/2\sigma^2)|_{x=n},$$
$$n = \ldots, -2, -1, 0, 1, 2, \ldots, \tag{1b}$$

with σ real and N_0 an integer. N_0 is typically chosen as $N_0 \approx \lceil 5\sigma \rceil$. At this value of N_0, the continuous Gaussian $g(x)$ is down by a factor of 3.7×10^{-6} from its value at $g(x=0)$.

The implementation based upon repeated convolutions with a "square" kernel is given per dimension by

$$\begin{aligned}\text{out}[n] = \text{in}[n]\otimes\{&\text{unif}[n]\otimes\text{unif}[n]\otimes\ldots \\ &\otimes\text{unif}[n]\},\end{aligned} \tag{2a}$$

where

$$\text{unif}[n] = \begin{cases} 1/(2N_0+1), & |n| \leqslant N_0, \\ 0, & |n| > N_0. \end{cases} \tag{2b}$$

It is common to use three convolutions of a uniform filter to approximate a Gaussian. For a desired σ, this will then require $N_0 \approx \lceil \sigma \rceil$. Only a limed number of Gaussian filters can be constructed in this way because we are constrained to use integer values of N_0.

The technique developed by Deriche involves a complicated design step for every choice of σ [3, 4]. For this reason Deriche developed an alternative non-Gaussian recursive filter that has an impulse response per dimension given by $h[n] = k(\alpha|n|+1)e^{(-\alpha|n|)}$. While this filter does have some impressive properties it also has some disadvantages – it is non-isotropic (in 2-D not circularly symmetric) and it is not the target filter, the Gaussian.

Because of the separability of multi-dimensional Gaussian filters (that is, $\exp(-(n^2+m^2)) = \exp(-n^2)\exp(-m^2)$, it is appropriate for us to examine the computational complexity of each algorithm one dimension at a time. For each dimension the use of direct convolution (Eq. (1a)) implies $(2N_0+1)$ MADDs (multiplications and additions). Taking advantage of symmetry in the Gaussian filter can reduce this to $2N_0$ additions and (N_0+1) multiplications. The use of repeated convolutions (Eq. (2a)) implies $2N_0$ additions. The Deriche approximation to the Gaussian [3] involves 12 MADDs per dimension and his alternative recursive filter (given above) requires eight MADDs per dimension.

In this paper we propose an alternative implementation of the Gaussian filter that is based upon a recursive structure. This implementation yields an infinite impulse response (IIR) filter that has six MADDs per dimension independent of the value of σ in the Gaussian kernel. The result is in general faster than either of the above-mentioned implementations and does not sacrifice accuracy.

2. Specification in the Laplace domain

Our approach is based upon a rational approximation to the Gaussian that is given by [1, Eq. 26.2.20]

$$g(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$$

$$= \frac{1}{a_0 + a_2 t^2 + a_4 t^4 + a_6 t^6} + \varepsilon(t), \qquad (3a)$$

where

$$a_0 = 2.490895, \quad a_2 = 1.466003, \qquad (3b)$$
$$a_4 = -0.024393, \quad a_6 = 0.178257.$$

The error $\varepsilon(t)$ is limited to $|\varepsilon(t)| < 2.7 \times 10^{-3}$. Relative to the maximum value of $g(t)$, which occurs at $t = 0$, this maximum error is $|\varepsilon(t)|/g(0) < 0.68\%$. A more complex approximation to $g(t)$ involving a tenth-order polynomial is also presented in [1] but the version presented here will be shown to be sufficient.

We will treat the approximation not as the Gaussian impulse response but rather as an approximation to its Fourier transform. That is, with $G(\omega) = \mathscr{F}\{g(x|\sigma)\}$, we have the well-known result

$$G(\omega) = e^{-\sigma^2\omega^2/2} = \mathscr{F}\left\{\frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}\right\}. \qquad (4)$$

For reasons that will become apparent later, we will use q instead of σ giving the rational approximation:

$$G_q(\omega) = \frac{A_0}{a_0 + a_2(q\omega)^2 + a_4(q\omega)^4 + a_6(q\omega)^6} \qquad (5a)$$

and, with $s = j\omega$, the equivalent Laplace transform is

$$G_q(s) = \frac{A_0}{a_0 - (a_2q^2)s^2 + (a_4q^4)s^4 - (a_6q^6)s^6}. \qquad (5b)$$

The expression $G_q(s)$ can be factored into the product of two terms, $G_q(s) = G_L(s)G_R(s)$: $G_L(s)$ with poles in the left-half plane and $G_R(s)$ with poles in the right-half plane.

$$G_L(s) = \frac{A_1}{(1.1668 + qs)(3.20373 + 2.21567qs + q^2s^2)} \qquad (6a)$$

and

$$G_R(s) = \frac{A_1}{(1.1668 - qs)(3.20373 - 2.21567qs + q^2s^2)}. \qquad (6b)$$

Table 1
Poles of Eqs. (6a) and (6b)

Poles	$G_L(s)$	$G_R(s)$
s_0	$-1.1668/q$	$+1.1668/q$
s_1	$(-1.10783 + 1.40586j)/q$	$(+1.10783 + 1.40586j)/q$
s_2	$(-1.10783 - 1.40586j)/q$	$(+1.10783 - 1.40586j)/q$

The poles of Eqs. (6a) and (6b) are located as shown in Table 1.

Independent of the choice of q (with $q > 0$), both filters $G_L(s)$ and $G_R(s)$ have poles in the complex plane.

3. Representation in the Z-domain

Eq. (6a) represents a causal, stable *differential* equation that can be transformed to a causal, stable *difference* equation, from $G_L(s)$ to an $H_L(z)$. Eq. (6b) represents an *anti-causal*, stable *differential* equation that can be transformed to an *anti-causal*, stable *difference* equation, from $G_R(s)$ to an $H_R(z)$. The standard technique for transforming a differential equation into a difference equation is to use the bilinear transform $s = (1 - z^{-1})/(1 + z^{-1})$ [8]. That technique leads, however, to zeroes in the transfer functions $H_L(z)$ and $H_R(z)$, a side effect we would like to avoid. In addition the bilinear transform technique has other disadvantages. See Appendix A.

We choose, instead, to use the backward difference technique [8] which approximates the derivative dy/dt by $(y[n] - y[n-1])/T$ and thus replaces s by $s = (1 - z^{-1})/T$. Like the bilinear transform, this approach has the property that the causal, stable $G_L(s)$ is mapped into a causal, stable $H_L(z)$. Further, no zeroes are introduced into $H_L(z)$ or $H_R(z)$. To generate $H_R(z)$ we use the forward difference equation approximation $dy/dt \approx (y[n+1] - y[n])/T$. The complex frequency s is replaced by $s = (z - 1)/T$. Again, this approach has the property that the anti-causal, stable $G_R(s)$ is mapped into an anti-causal, stable $H_R(z)$. Setting $T = 1$ gives

$$H_L(z) = G_L(s)|_{s=1-z^{-1}} = \frac{A_0}{(1.1668 + q(1 - z^{-1}))(3.20373 + 2.21567q(1 - z^{-1}) + q^2(1 - z^{-1})^2)} \tag{7a}$$

and

$$H_R(z) = G_R(s)|_{s=z-1} = \frac{A_0}{(1.1668 - q(z - 1))(3.20373 - 2.21567q(z - 1) + q^2(z - 1)^2)}. \tag{7b}$$

Both equations can be rewritten as standard polynomials in z^{-1} and z, respectively, giving

$$H_L(z) = \frac{A_2}{b_0 - b_1 z^{-1} - b_2 z^{-2} - b_3 z^{-3}} \tag{8a}$$

and

$$H_R(z) = \frac{A_2}{b_0 - b_1 z^1 - b_2 z^2 - b_3 z^3}, \tag{8b}$$

where

$$\begin{aligned} b_0 &= 1.57825 + (2.44413q) + (1.4281q^2) \\ &\quad + (0.422205q^3), \\ b_1 &= (2.44413q) + (2.85619q^2) + (1.26661q^3), \\ b_2 &= -((1.4281q^2) + (1.26661q^3)), \\ b_3 &= 0.422205q^3. \end{aligned} \tag{8c}$$

This implementation suggests the following filtering strategy. The input data are first filtered in the *forward* direction as suggested by the difference equation corresponding to Eq. (8a). The output of this result – let us call it $w[n]$ – is then filtered in the *backward* direction according to the difference equation corresponding to Eq. (8b). The difference equations are

forward:

$$w[n] = B\,\mathrm{in}[n] + (b_1 w[n-1] + b_2 w[n-2] + b_3 w[n-3])/b_0 \tag{9a}$$

and

backward:

$$\mathrm{out}[n] = Bw[n] + (b_1 \mathrm{out}[n+1] + b_2 \mathrm{out}[n+2] + b_3 \mathrm{out}[n+3])/b_0. \tag{9b}$$

Both filters have the same normalization constant, B, that can be specified by using the constraint that the transfer function of the filter should be 1.0 for the frequency $\omega = \Omega = 0$. This leads to

$$B = 1 - ((b_1 + b_2 + b_3)/b_0). \tag{10}$$

Example. Let us now look at an example of the use of this procedure. We begin by choosing $q = 5.0$. Using Eq. (8c) this leads to $b_1/b_0 = 2.36565$, $b_2/b_0 = -1.89709$, $b_3/b_0 = 0.51601$, and $B = 0.01543$. The impulse response, $h[n]$, of this recursive implementation of the Gaussian filter is shown in Fig. 1(a) along with a continuous curve representing a true Gaussian with $\sigma = 6.09$ as defined in Eq. (1b). An enlargement of the scale in order to enhance the tails of the Gaussian is shown in Fig. 1(b).

Several observations associated with this example are useful. Eqs. (6a) and (6b) and the associated poles (listed in Table 1) suggest that the impulse response of each separate term (forward and backward) will oscillate. No such behavior, however, is observed in the impulse response $h[n]$ of the total Gaussian recursive filter. (This is not the case when the bilinear transform implementation is used. See Appendix A.)

The standard deviation associated with the impulse response of the recursive Gaussian filter is not the value q as suggested by Eqs. (4), (5a) and (5b). While a value of $q = 5.0$ was used in the above example, a minimum mean-square-error fit of a true Gaussian to the observed impulse response gives a value of $\sigma = 6.09$. The root-mean-square error for $q = 5.0$ is rms $= 2.8 \times 10^{-4}$, the maximum absolute error is $|\varepsilon| = 1.7 \times 10^{-3}$, and the root-square error (described below) is rs $= 5.6 \times 10^{-3}$ when measured over the interval -33σ to $+33\sigma$. We discuss this matter below.

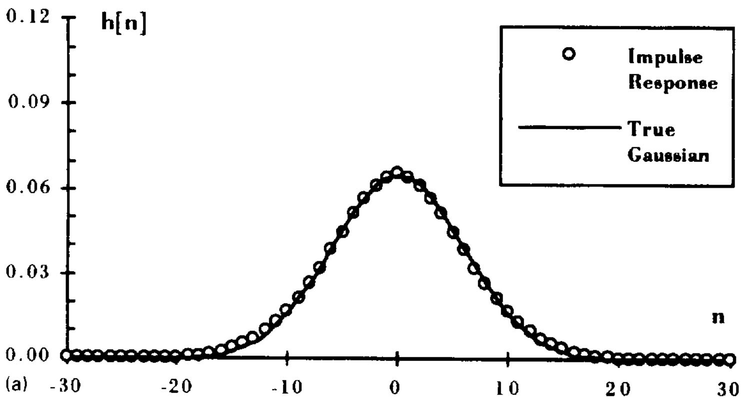

Fig. 1(a). Impulse response $h[n]$ resulting from the application of Eqs. (9a) and (9b) to the input signal in$[n] = \delta[n]$. The continuous curve is $g(x|\sigma = 6.1)$. The value of q used in Eq. (8c) is $q = 5.0$.

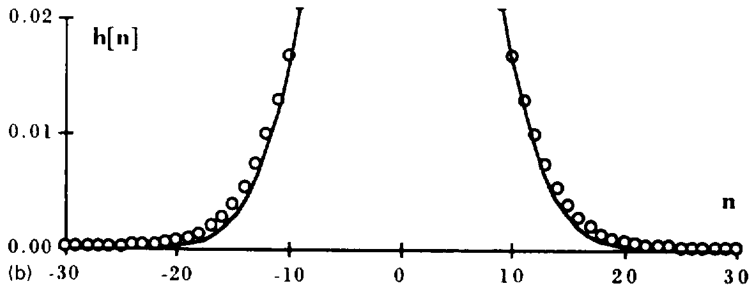

Fig. 1(b). Enlargement of $h[n]$ resulting from the application of Eqs. (9a) and (9b) to the input signal in$[n] = \delta[n]$ with $q = 5.0$. Note the values of the vertical scale.

4. Choosing q

The difference between the value of q and the value of σ can be attributed to two causes. First, we have chosen to use the approximation in the frequency domain and test the approximation in the spatial (or time) domain. Second, the use of the forward and backward difference approximations means a known source of error due to the fact that neither approximation gives a correct mapping from the ω-axis in the s-plane to the unit circle in the z-plane. The forward and backward difference approximations are particularly crude in this regard and only work when the signals concerned are 'oversampled' so that their spectral energy is confined to a tight band around $\Omega = 0$. This oversampling is equivalent to taking many samples of the desired Gaussian which should translate into greater accuracy in the approximation when we move to wider Gaussians, that is, when σ increases.

To test this hypothesis we have for a range of values of q looked at (1) the value of σ provided by our Gaussian recursive filter, (2) the maximum absolute error, and (3) the root-square error. The relationship between q and σ was investigated by generating the impulse response $h[n]$ for a range of values of q given by $0.5 \leqslant q \leqslant 20.0$. This range of q covers values from where the Gaussian is clearly undersampled to values where it is certainly oversampled. The non-linear Levenberg–Marquardt algorithm was used to find a minimum mean-square-error fit between the data as given by $h[n]$ and the function $g(x|\sigma_0) = A\exp(-x^2/2\sigma_0^2)$ with (A, σ_0) as the parameters to be found [10, 15]. In Fig. 2 we see the resulting relationship between q and σ_0.

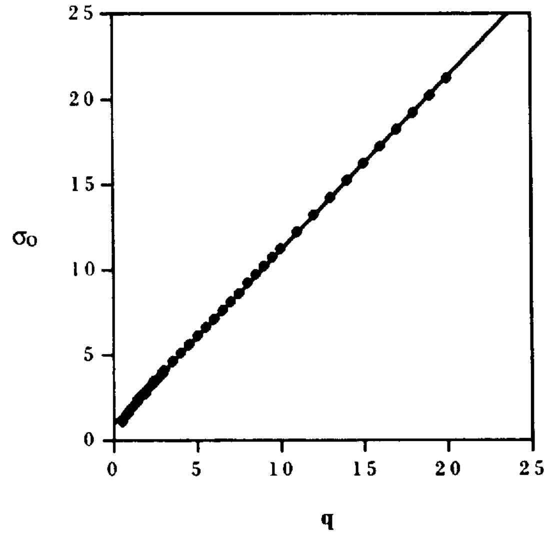

Fig. 2. Levenberg–Marquardt estimate of σ_0 for each value of q plus the regression line.

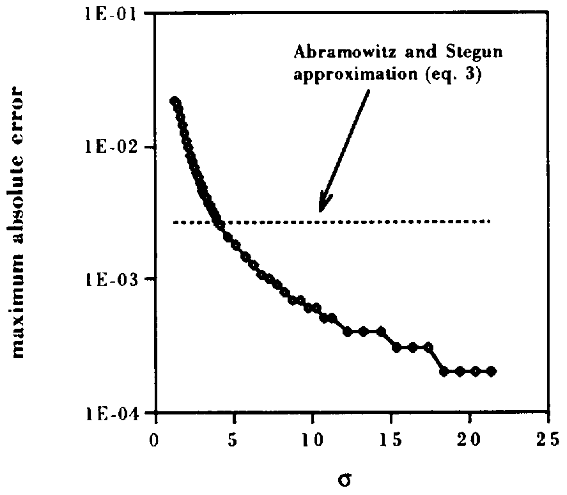

Fig. 3. The maximum absolute error $|\varepsilon(n)|$ of $h[n]$ for each value of σ.

Although the behavior of this curve is dominated by a linear relationship between q and σ_0 for a significant range of q, for small values ($0.5 \leqslant q \leqslant 1.5$) the curve has a much better representation as a second-order polynomial. A regression analysis with q as the independent variable and σ_o as the dependent variable gives:

$$\sigma_0 = \begin{cases} 0.97588 + (1.01306q), & q \geqslant 1.5, \\ 0.30560 + (1.71881q) - (0.21639q^2), \\ \quad 0.5 \leqslant q \leqslant 1.5. \end{cases} \tag{11a}$$

Inverting these two forms gives

$$q = \begin{cases} 0.98711\sigma_0 - 0.96330, & \sigma_0 \geqslant 2.5, \\ 3.97156 - 4.14554\sqrt{1 - 0.26891\sigma_0}, \\ \quad 0.5 \leqslant \sigma_0 \leqslant 2.5. \end{cases} \tag{11b}$$

The correlations (ρ) associated with the regression equations (11a) are both greater than 0.9999 and all coefficients are significant at $p < 0.0001$. Using the regression equations (11a), the value of $q = 5.0$ (used in the previous example) leads to the value $\sigma_0 = 6.04$.

5. Accuracy

The maximum absolute error associated with the use of the recursive Gaussian filter can be evaluated by looking at the maximum value of $|\varepsilon(n)| = |g(n|\sigma_0) - h[n]|$. The result is shown in Fig. 3. For values of σ greater than 4.0 the maximum absolute error $|\varepsilon|$ is *smaller* than the maximum error of the approximation given in Eq. (3). Fig. (3) clearly supports our assertion that the accuracy increases as σ (and thus, q) increases.

The maximum absolute error 'ε_ω' associated with Eq. (3) takes place in the frequency domain. The error evaluation ε_n in Fig. 3 takes place in the domain of $h[n]$ where ε_ω is 'smeared out'.

Further, we can compare the accuracy associated with various implementations of a Gaussian filter. A measure for comparison is based upon the following analysis. For an arbitrary input $x[n]$ the output associated with a 'true' Gaussian is given by $y_g[n] = g(n|\sigma_0) \otimes x[n]$ and for an approximation by $y_h[n] = h[n] \otimes x[n]$. The error in the output *per sample* is given by

$$\begin{aligned} \varepsilon_y[n] &= y_g[n] - y_h[n] = \{g(n|\sigma_0) - h[n]\} \otimes x[n] \\ &= \varepsilon[n] \otimes x[n]. \end{aligned} \tag{12}$$

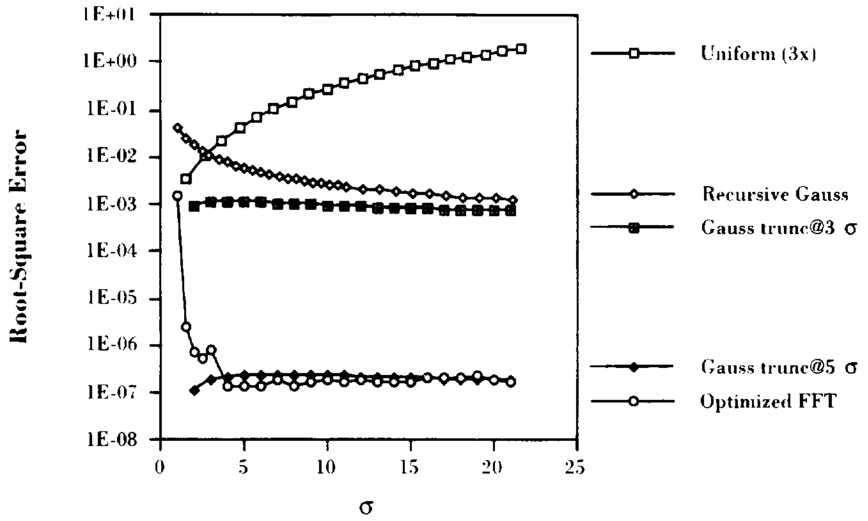

Fig. 4. The root-square error $\sqrt{\sum_{n=-\infty}^{+\infty}|\varepsilon[n]|^2}$ for each value of σ.

But

$$|\varepsilon_y[n]| \leqslant |\varepsilon[n]| \otimes |x[n]|. \tag{13}$$

Applying the Cauchy–Schwarz inequality to the right-hand side of Eq. (13) gives

$$|\varepsilon_y[n]| \leqslant \sqrt{\sum_{n=-\infty}^{+\infty} |\varepsilon[n]|^2} \sqrt{\sum_{n=-\infty}^{+\infty} |x[n]|^2} = \sqrt{E_\varepsilon}\sqrt{E_x}, \tag{14}$$

where E_ε is the energy in the error signal and E_x is the energy in the input signal. Because the choice of $x[n]$ is arbitrary, the only way to guarantee that the error $|\varepsilon_y[n]|$ is small is to choose $h[n]$ such that $\sqrt{E_\varepsilon}$ (the root-square error) is acceptably small.

We have evaluated the root-square error for one-dimensional filtering with the recursive Gaussian filter, FIR convolution with Gaussian kernels (Eq. (1a)), a triple convolution with a uniform filter (Eq. (2a)), and with an FFT implementation of the Gaussian filter. In each case σ is estimated via the Levenberg–Marquardt method and the residual root-square error is then computed. The results are shown in Fig. 4.

For typical values of σ, such as $\sigma = 2.1$, the root-square error for the recursive implementation of the Gaussian is 1.82×10^{-2}. This value should be viewed as extremely conservative because the bound from Eq. (14) depends on two inequalities. Neither is likely to achieve its maximum value as the error signal $\varepsilon[n]$ has negative values (Eq. (13)) and it is quite unlikely that $|x[n]| = k|\varepsilon[n]|$ – the condition for equality in the Cauchy–Schwartz inequality (Eq. (14)).

6. Speed

We have evaluated the improvement in computational speed for various values of σ of a two-dimensional, recursive Gaussian filter in comparison to an FIR convolution with a Gaussian kernel (Eq. (1a)), a triple convolution with uniform filters (Eq. (2a)) with a square support and with a circular support, and with an FFT implementation of the Gaussian filter. All evaluations were done on a 256 × 256 image.

The computation time of the recursive Gaussian filter is 100 ms on a Silicon Graphics Indigo R4400 computer independent of σ. The other (non-recursive) filters were implemented with the most efficient algorithms that we have. The Gaussian FIR uses an implementation that takes advantage of the symmetry in the impulse response to reduce the number of MADDs. The uniform filters (both

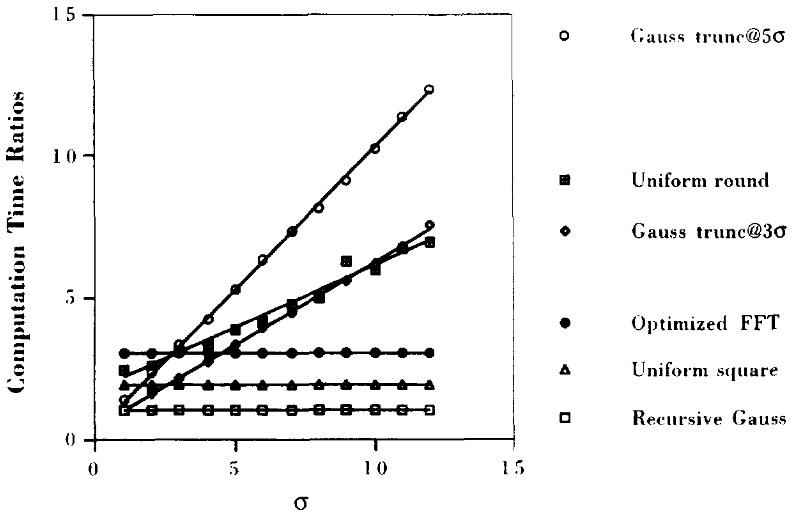

Fig. 5. Ratio of computation times of various filter algorithms to the fastest implementation – the recursive Gaussian – as a function of σ.

square and circular) use an incremental updating process for improved speed [13]. The FFT implementation uses a real FFT and look-up tables for the sine and cosine values. The overall results are shown in Fig. 5.

For all values of $\sigma > 1.0$, the recursive algorithm is the fastest. At $\sigma = 5.0$, for example, the Gaussian FIR convolutions are 3.3 and 5.3 times *slower* when the filter coefficients are truncated at 3σ and 5σ, respectively. A square uniform filter is 1.9 times slower and a circular uniform filter is 3.9 times slower. The FFT implementation, whose computation time is independent of σ, is 3.1 times slower than the recursive implementation. Further, the use of FFT techniques requires that image sizes be highly composite numbers such as powers of two.

7. Isotropy

An important aspect of the Gaussian filter, alluded to earlier, is its isotropy in N-dimensions. To illustrate this issue we have computed the iso-brightness contours associated with five different implementations of Gaussian filtering – our IIR recursive Gaussian (Eq. (9)), the FIR Gaussian convolution (Eq. (1)), the triple convolution with a 2-D *circular* uniform filter (Eq. (2a)), the triple convolution with a 2-D *square* uniform filter (Eq. (2a)), and the Deriche filter given by $h[n] = k(\alpha|n| + 1)e^{(-\alpha|n|)}$. The results are shown in Fig. 6.

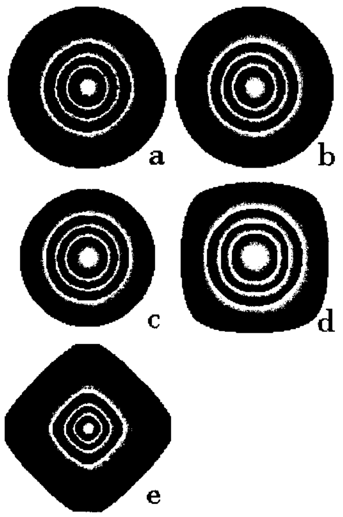

Fig. 6. Impulse response of various filters: (a) IIR recursive Gaussian from Eqs. (9a) and (9b) with $\sigma = 20$; (b) FIR Gaussian filter from Eqs. (1a) and (1b) with $\sigma = 20$; (c) triple convolution with a 2-D uniform filter with a circular (pillbox) support and diameter 41; (d) triple convolution with a 2-D uniform filter with a square support and width 41; (e) IIR recursive Deriche filter [4] with $\alpha = 0.1$.

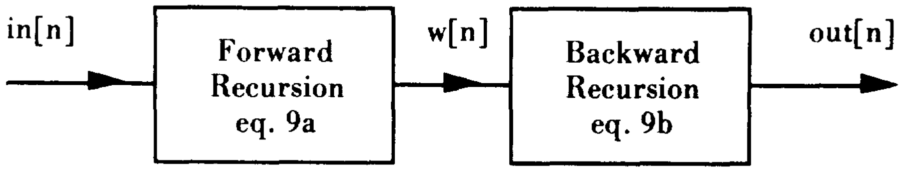

Fig. 7. Gaussian filter as concatenation of two filters.

As shown in Figs. 6(a)–(c), our recursive Gaussian filter, the FIR convolution with a Gaussian, and the triple convolution with a circular (pillbox) uniform filter lead to an isotropic (circularly symmetric) impulse response. The other filters, as shown in Figs. 6(d) and (e), lead to anisotropic impulse responses.

8. Derivative filters

In a variety of image processing and computer vision applications, Gaussian filters are frequently used in combination with derivative filters. Independent of what order derivative is required, we consider it essential that the derivative operations have zero phase in order to avoid displacing edges in images. This zero phase condition can be realized either by using a derivative filter with an even impulse response, $h_d[n] = h_d[-n]$, or by using a forward filter in concatenation with a backward filter such that the phase terms cancel each other. The recursive implementation for the Gaussian that we have developed is illustrated in Fig. 7.

The implementation of derivative filters involves replacing either Eq. (9a) or Eqs. (9a) and (9b).

First derivative – Eq. (9a) is replaced by Eq. (15),

$$w[n] = (B/2)(\mathrm{in}[n+1] - \mathrm{in}[n-1]) + (b_1 w[n-1] + b_2 w[n-2] + b_3 w[n-3])/b_0, \tag{15}$$

and Eq. (9b) is left unchanged.

Second derivative – Eqs. (9a) and (9b) are replaced by Eqs. (16a) and (16b):

forward:

$$w[n] = B(\mathrm{in}[n] - \mathrm{in}[n-1]) + (b_1 w[n-1] + b_2 w[n-2] + b_3 w[n-3])/b_0 \tag{16a}$$

and

backward:

$$\mathrm{out}[n] = B(w[n+1] - w[n]) + (b_1 \mathrm{out}[n+1] + b_2 \mathrm{out}[n+2] + b_3 \mathrm{out}[n+3])/b_0. \tag{16b}$$

Third derivative – Eqs. (9a) and (9b) are replaced by Eqs. (17a) and (17b):

forward:

$$w[n] = B(\mathrm{in}[n+1] - 2\mathrm{in}[n] + \mathrm{in}[n-1]) + (b_1 w[n-1] + b_2 w[n-2] + b_3 w[n-3])/b_0 \tag{17a}$$

and

backward:

$$\mathrm{out}[n] = (B/2)(w[n+1] - w[n-1]) + (b_1 \mathrm{out}[n+1] + b_2 \mathrm{out}[n+2] + b_3 \mathrm{out}[n+3])/b_0. \tag{17b}$$

The low-pass nature of the Gaussian filter helps suppress the noise that is amplified by the derivative process. When higher-order derivatives are used the value of σ_0 (and hence q) should be increased accordingly to provide additional noise suppression.

9. Summary

In this paper we have developed a straightforward recursive implementation of the Gaussian filter. We have examined various alternatives and shown that simple forward and backward difference approximations in concatenation lead to the required IIR filter. We have shown that our implementation is the fastest of those evaluated and that the result in 2-D (and higher dimensions as well) is isotropic. We have developed a simple, algebraic closed-form procedure for choosing the five coefficients of the recursive filter given the desired σ_0. The 'recipe' (per dimension) is as follows:

1. Choose σ_0 based upon the desired goal of the filtering.
2. Use Eq. (11b) to determine q.
3. Use Eq. (8c) to determine b_0, b_1, b_2, and b_3.
4. Use Eq. (10) to determine B.
5. Implement the forward filter with Eq. (9a).
6. Implement the backward filter with Eq. (9b).
7. Apply.

If derivative filters are desired then the appropriate equation(s) from the sets (15)–(17) can be substituted in steps 5 and 6.

Acknowledgments

This work was partially supported by the Netherlands Organization for Scientific Research (NWO) Grant 900-538-040, the Foundation for Technical Sciences (STW) Project 2987, the ASCI PostDoc program, and the Rolling Grants program of the Foundation for Fundamental Research in Matter (FOM).

Appendix A

Examples using the bilinear transform

Starting with Eqs. (6a) and (6b) we use the bilinear transform $s = (1-z^{-1})/(1+z^{-1})$ [8]. This leads to the equations

$$H_L(z) = A_2\frac{1+3z^{-1}+3z^{-2}+z^{-3}}{c_0-c_1z^{-1}-c_2z^{-2}-c_3z^{-3}} \tag{A.1a}$$

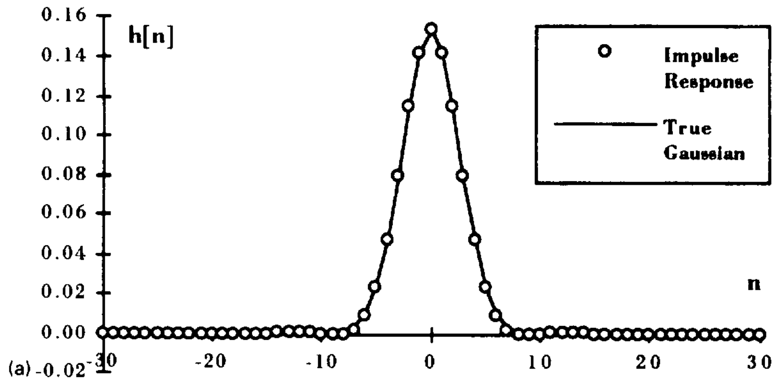

Fig. 8(a). Impulse response $h[n]$ resulting from using the bilinear transform version of the recursive Gaussian filter. The continuous curve is $g(x\,|\,\sigma = 2.6)$. The value of q used in Eq. (A.1c) is $q = 5.0$.

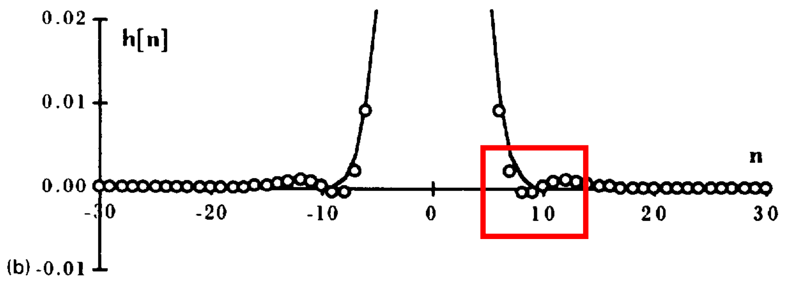

Fig. 8(b). Enlargement of $h[n]$. Note the ringing around $x = n = 9$.

and

$$H_R(z) = A_2 \frac{1 + 3z + 3z^2 + z^3}{c_0 - c_1 z - c_2 z^2 - c_3 z^3}, \tag{A.1b}$$

where

$$\begin{aligned} c_0 &= +3.73811 + (5.78897q) + (3.38247q^2) + q^3, \\ c_1 &= -11.2143 - (5.78897q) + (3.38247q^2) + 3q^3, \\ c_2 &= -11.2143 + (5.78897q) + (3.38247q^2) - 3q^3, \\ c_3 &= -3.73811 + (5.78897q) - (3.38247q^2) + q^3. \end{aligned} \tag{A.1c}$$

The associated difference equations are:

forward:

$$\begin{aligned} w[n] = {} & C(\text{in}[n] + 3\text{in}[n-1] + 3\text{in}[n-2] \\ & + \text{in}[n-3]) + (c_1 w[n-1] \\ & + c_2 w[n-2] + c_3 w[n-3])/c_0 \end{aligned} \tag{A.2a}$$

and

backward:

$$\begin{aligned} \text{out}[n] = {} & C(w[n] + 3w[n+1] + 3w[n+2] \\ & + w[n+3]) + (c_1 \text{out}[n+1] \\ & + c_2 \text{out}[n+2] + c_3 \text{out}[n+2])/c_0. \end{aligned} \tag{A.2b}$$

The normalization constant, C, can be specified by again using the constraint that the transfer function of the filter should be 1.0 for the frequency $\omega = \Omega = 0$. This leads to

$$C = \frac{\{c_0 - (c_1 + c_2 + c_3)\}}{8c_0}. \tag{A.3}$$

Two examples

Example 1. For $q = 5.0$, Eqs. (A.1c) and (A.3) give $c_1/c_0 = 1.73132$, $c_2/c_0 = -1.12575$, $c_3/c_0 = 0.27099$, and $C = 0.01543$. (Note that the bilinear

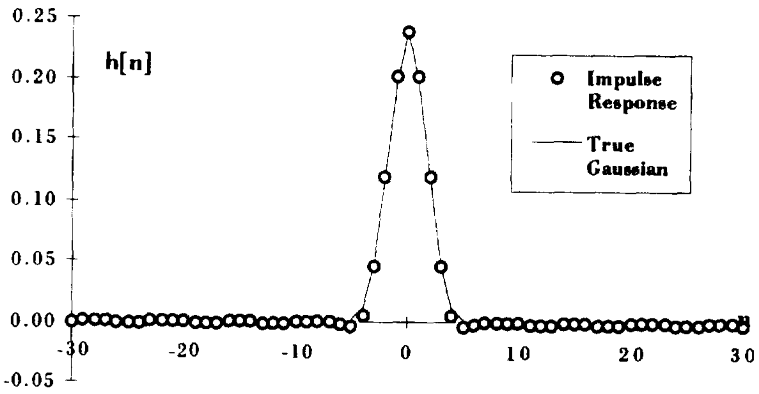

Fig. 9(a). Impulse response $h[n]$ resulting from using the bilinear transform version of the recursive Gaussian filter. The continuous curve is $g(x \mid \sigma = 1.67)$. The value of q used in Eq. (A.1c) is $q = 3.0$.

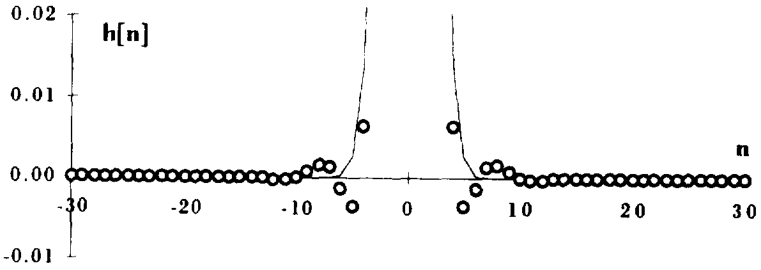

Fig. 9(b). Enlargement of $h[n]$. Note the ringing around $x = n = 5$.

transform and the forward/backward difference approximation both have the same normalization constant. This is because at $z = 1$ ($\Omega = \omega = 0$) both transformations are the same.)

The impulse response of this recursive implementation of the Gaussian filter is shown in Fig. 8(a) and an enlarged version in Fig. 8(b).

Example 2. For $q = 3.0$, Eqs. (A.1c) and (A.3) give $c_1/c_0 = -1.05492$, $c_2/c_0 = 0.565331$, $c_3/c_0 = -0.129687$, and $C = 0.04759$. The impulse response of this implementation is shown in Fig. 9(a) and an enlarged version in Fig. 9(b).

The bilinear transformation version shows a considerable amount of ringing, an undesirable side-effect when one is filtering images. This ringing is not seen when the forward/backward technique is used even at relatively low values of q such as $q = 3.0$. This is shown in Fig. 10.

The final result for the forward/backward technique is better than one might expect given the error analyses presented in Figs. 3 and 4. Further, the computational complexity goes from six MADDs in the forward/backward technique to 12 MADDs with the bilinear technique. For $\sigma = 4.1$, the root-square error is $\text{rs} = 9.0 \times 10^{-3}$ with the forward/backward technique and $\text{rs} = 2.9 \times 10^{-3}$ with the bilinear technique. For $\sigma = 6.2$, the values are 5.6×10^{-3} (forward/backward) and 2.25×10^{-3} (bilinear). The small improvement in the error bound rs for the bilinear technique over the forward/backward technique is due to the avoidance of aliasing when the bilinear technique is used. In summary, the bilinear technique is somewhat more

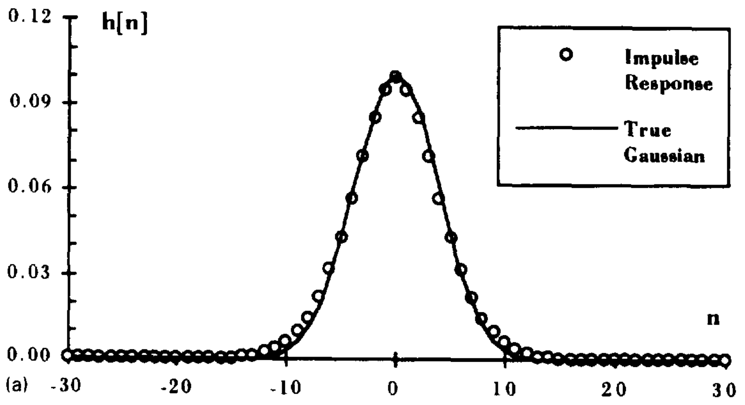

Fig. 10(a). Impulse response $h[n]$ resulting from the application of Eqs. (9a) and (9b) to the input signal $\text{in}[n] = \delta[n]$. The continuous curve is $g(x|\sigma = 4.0)$. The value of q used in Eq. (8c) is $q = 3.0$.

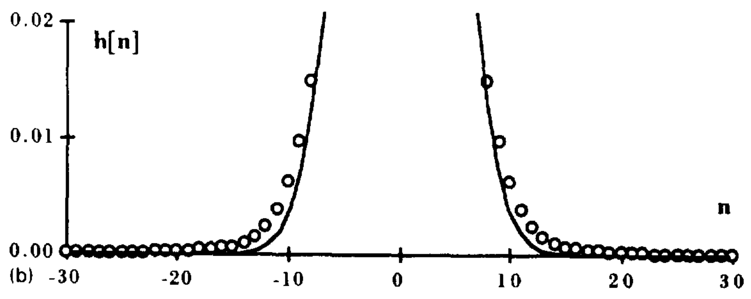

Fig. 10(b). Enlargement of $h[n]$ resulting from the application of Eqs. (9a) and (9b) to the input signal $\text{in}[n] = \delta[n]$ with $q = 3.0$. Note the values of the vertical scale.

accurate but the forward/backward technique is faster and does not have ringing in $h[n]$.

References

[1] M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions*, Dover, New York, 1965.
[2] J. Canny, "A computational approach to edge detection", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-8, 1986, pp. 679–698.
[3] R. Deriche, "Separable recursive filtering for efficient multi-scale edge detection". *Proc. Internat. Workshop on Industrial Applications of Machine Vision and Machine Intelligence*, Tokyo, 1987, pp. 18–23.
[4] R. Deriche. "Fast algorithms for low-level vision", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-12, 1990, pp. 78–87.
[5] J.J. Koenderink, "The structure of images". *Bio. Cyb.*, Vol. 50, 1984, pp. 363-370.
[6] D. Marr, *Vision*, Freeman, New York, 1982.
[7] D. Marr and E.C. Hildreth, "Theory of edge detection", *Proc. R. Soc. London Ser. B*, Vol. 207, 1980, pp. 187–217.
[8] A.V. Oppenheim, A.S. Willsky and I.T. Young, *Systems and Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[9] A. Papoulis, *The Fourier Integral and its Applications*, McGraw-Hill, New York, 1962.
[10] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Veterling, *Numerical Recipes in C*, Cambridge Univ. Press, Cambridge, UK, 1988.
[11] L.J. Van Vliet, Grey-scale measurements in multi-dimensional digitized images, Ph.D Thesis, Delft University of Technology, 1993.
[12] P.W. Verbeek and L.J. Van Vliet, "Estimators of 2D edge length and position, 3D surface area and position in sampled grey-valued images", *BioImaging*, Vol. 1, 1993, pp. 47–61.
[13] P.W. Verbeek, H.A. Vrooman and L.J. Van Vliet, "Low-level image processing by max–min filters", *Signal Processing*, Vol. 15, No. 3, October 1988, pp. 249–258.
[14] A. Witkin, "Scale-space filtering", *Proc. Internat. Joint Conf. on Artificial Intelligence*, Karlsruhe, Germany, 1983, pp. 1019–1021.
[15] S. Wolfram, *Mathematica, A System for Doing Mathematics by Computer*, Addison-Wesley, Redwood City, CA, 2nd Edition, 1991.