Recursive Gaussian filters

Dave Hale
Center for Wave Phenomena, Colorado School of Mines, Golden CO 80401, USA

ABSTRACT
Gaussian or Gaussian derivative filtering is in several ways optimal for applications requiring low-pass filters or running averages. For short filters with lengths of a dozen samples or so, direct convolution with a finite-length approximation to a Gaussian is the best implementation. However, for longer filters such as those used in computing running averages, recursive implementations may be much more efficient. Based on the filter length, we select one two popular methods for designing and implementing recursive Gaussian filters.

Key words: digital signal processing

1 INTRODUCTION

Gaussian filtering is useful in at least two different contexts in digital signal processing. One context is *low-pass filtering*. In this context, we typically wish to attenuate high-frequency noise. For example, when detecting edges or computing the orientation of features in digital images, we might compute partial derivatives of image sample values (van Vliet and Verbeek, 1995). Because derivatives amplify high-frequency (often noisy) components of signals, we might also apply a low-pass filter before or after computing those derivatives.

An equivalent and more efficient process is to apply a filter that combines both differentiation and low-pass filtering. When the low-pass filter is a Gaussian, this combination yields a filter with a smooth impulse response that approximates a derivative of a Gaussian.

Another context in which Gaussian filtering is useful is in computing *running averages*. We often use running averages to estimate parameters from signals with characteristics that vary with space and time. A typical example looks something like this:

$$y(t) = \frac{1}{2T}\int_{t-T}^{t+T} \mathrm{d}s\; x(s). \tag{1}$$

Here, we estimate a parameter y that varies with time t by averaging nearby values of some other function $x(t)$. Ideally, we choose the window length $2T$ large enough to obtain meaningful estimates of the parameter y, but small enough that we can detect significant variations in that parameter with time t.

Running averages like that in equation 1 appear often in seismic data processing. For example, in coda-wave interferometry (Snieder, 2006) the function $x(t)$ is the product of two seismograms, one shifted relative to the other. In seismic velocity estimation, the function $x(t)$ might be the numerator or denominator sum in a semblance calculation (Neidell and Taner, 1971). A trivial example is automatic gain control, in which $y(t)$ might be an rms running average of seismic amplitude computed as a function of recording time t.

In each of these applications, we might ask whether the averaging in equation 1 is sensible. For each estimate $y(t)$, this equation implies that all values $x(s)$ inside the window of length $2T$ have equal weight, but that values just outside this window have no weight at all.

This weighting makes no sense. In estimating $y(t)$, why should the values $x(t)$ and $x(t+T-\epsilon)$ get equal weight, but the value $x(t+T+\epsilon)$ get zero weight? A more sensible window would give values $x(s)$ for s near t higher weight than values farther away. In other words, we should use a weighted average, a window with non-constant weights. And the uncertainty relation (e.g.; Bracewell, 1978) tells us that *the optimal window is Gaussian*.

A sampled version of equation 1 is

$$y[n] = \frac{1}{2M+1}\sum_{m=n-M}^{n+M} x[m]. \tag{2}$$

Despite the shortcoming described above, this sort of running average may be popular because we can easily and efficiently implement it by solving recursively the following linear constant-coefficient difference equation:

$$y[n] = y[n-1] + (x[n+M] - x[n-M-1])/(2M+1), \quad (3)$$

with some appropriate initial and end conditions. For long windows (large M) we must take care to avoid excessive accumulation of rounding errors as we add and subract small sample values $x[n+M]$ and $x[n-M-1]$ from the previous output value $y[n-1]$. For short windows (small M) we might use the non-recursive running average of equation 2. But it is difficult to achieve a computational cost lower than that of equation 3.

The two contexts described above — low-pass filtering and running averages — are of course one and the same. A running average is a low-pass filter. In practice, these contexts differ only in the length of the filter required. Gaussian derivative filters typically have short lengths of less than 10 samples. Gaussian filters used in running averages tend to be longer, sometimes with lengths of more than 100 samples.

In this paper, we compare two recursive implementations of Gaussian and Gaussian derivative filters. As in equation 3, these filters solve difference equations. Numerical tests indicate that one implementation is best for short filters (derivatives), while the other is best for long filters (running averages). For long filters, especially, recursive implementations are much more efficient than direct convolution with truncated or tapered sampled Gaussians.

2 THE GAUSSIAN AND DERIVATIVES

The Gaussian function is defined by

$$g(t;\sigma) \equiv \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-t^2/2\sigma^2}, \quad (4)$$

where the parameter σ denotes the Gaussian half-width. Figure 1 displays this Gaussian function and its 1st and 2nd derivatives for $\sigma = 1$.

The Fourier transform of the Gaussian function is also a Gaussian:

$$G(\omega;\sigma) \equiv \int_{-\infty}^{\infty} \mathrm{d}t \; \mathrm{e}^{-i\omega t} g(t;\sigma) = \mathrm{e}^{-\omega^2\sigma^2/2}. \quad (5)$$

Figure 2 displays this Fourier transform and those of the 1st and 2nd Gaussian derivatives for $\sigma = 1$. Equations 4 and 5 imply that convolution of a signal with a Gaussian is equivalent to low-pass filtering of that signal. Indeed, we choose the scale factor in equation 4 so that the frequency response of this low-pass filter at zero frequency is $G(\omega = 0;\sigma) = 1$, as illustrated in Figure 2.

Convolution of a signal with a Gaussian derivative is equivalent to differentiating that signal before or after low-pass filtering. This combination of differentiation with Gaussian low-pass filtering is an efficient way to perform both operations simultaneously. It computes the derivative of lower-frequency signal while attenuating higher-frequency noise.

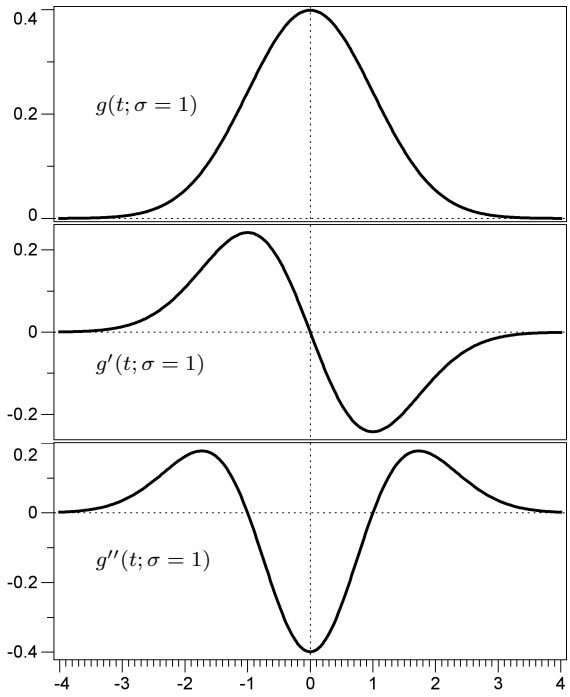

Figure 1. The Gaussian function $g(t;\sigma = 1)$ and its derivatives.

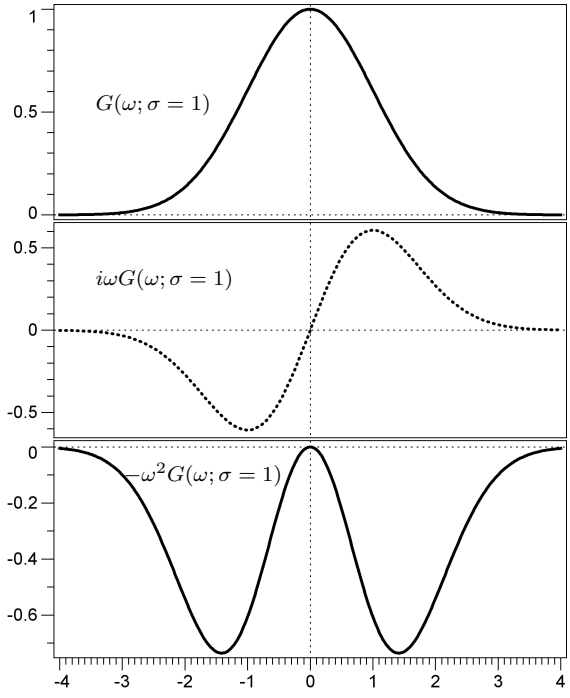

Figure 2. The Fourier transforms of the Gaussian function $g(t;\sigma = 1)$ and its 1st and 2nd derivatives. The Fourier transform of the Gaussian 1st derivative $g'(t;\sigma)$ is purely imaginary, as indicated here by the dashed curve.

2.1 Sampling and aliasing

For digital filtering, we must sample the Gaussian function $g(t;\sigma)$ and its derivatives. Because the Fourier transform $G(\omega;\sigma)$ is nowhere zero, the Gaussian function cannot be sampled without aliasing. However, aliasing can in most applications be negligible.

For definiteness, assume a unit sampling interval $\Delta t = 1$. Then, for half-width $\sigma = 1$ and the Nyquist frequency $\omega = \pi$, the Fourier transform is $G(\pi;\sigma = 1) \approx 0.0072$. (See Figure 2.) This small magnitude implies that the Gaussian function $g(t;\sigma \geq 1)$ can be sampled with unit sampling interval without significant aliasing.

The impact of aliasing is application dependent, but we typically do not sample Gaussians for which $\sigma < 1$. For applications requiring almost no aliasing, we might choose $\sigma \geq 2$, because $G(\pi;\sigma \geq 2) < 10^{-8}$.

As illustrated in Figure 2, the Fourier transforms of the derivatives $g'(t;\sigma)$ and especially $g''(t;\sigma)$ do not approach zero so quickly. In some applications, Gaussian derivatives may require a larger half-width $\sigma > 1$ to avoid significant aliasing.

2.2 FIR approximations

Just as we may reasonably assume that the Fourier transform $G(\omega;\sigma)$ has finite bandwidth, so may we assume that the Gaussian function itself has finite support. Figure 1 suggests that the Gaussian function and its derivatives are approximately zero for $|t| > 4\sigma$. For example, $g(t;\sigma) < 0.0004$ for $|t| > 4\sigma$.

Again, assume a unit sampling interval, and let $h[n]$ denote a sampled Gaussian function that has been truncated or tapered to zero for $|n| > M \approx 4\sigma$. Gaussian filtering can then be performed via direct convolution:

$$y[n] = \sum_{m=n-M}^{n+M} h[n-m]x[m].$$

This convolution is reminiscent of equation 2.

Exploiting symmetry in the filter $h[m]$, this *finite-impulse-response* (FIR) approximation to Gaussian filtering requires roughly $1 + 4\sigma$ multiplications and 8σ additions per output sample. The computational cost of applying an FIR Gaussian or Gaussian derivative filter grows linearly with the Gaussian half-width σ.

2.3 IIR approximations

In constrast, the cost of an *infinite-impulse-response* (IIR) approximation to Gaussian filtering is independent of the half-width σ. IIR Gaussian and Gaussian derivative filters solve recursively a sequence of difference equations like this one:

$$\begin{aligned} y[n] = b_0 x[n] \; &+ \; b_1 x[n-1] + b_2 x[n-2] \\ &- \; a_1 y[n-1] - a_2 y[n-2], \end{aligned} \tag{6}$$

where the filter coefficients b_0, b_1, b_2, a_1, and a_2 are some function of the Gaussian half-width σ. This difference equation is reminiscent of equation 3.

The recursive solution of a single difference equation cannot well approximate Gaussian filtering. With only five parameters, we cannot shape the impulse response of this system to fit well the Gaussian function. Moreover, the impulse response of the system of equation 6 is one-sided, not symmetric like the Gaussian function in Figure 1. IIR symmetric filtering requires a sequence of solutions to both *causal and anti-causal* systems.

The z-transform of the IIR filter implied by equation 6 is

$$H^+(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}.$$

Assuming that all poles of $H^+(z)$ lies inside the unit circle in the complex z-plane, this 2nd-order system is both causal and stable. A corresponding stable and anti-causal 2nd-order system is

$$H^-(z) = \frac{b_0 + b_1 z + b_2 z^2}{1 + a_1 z + a_2 z^2},$$

for which all poles lie outside the unit circle.

Neither of these 2nd-order recursive systems is sufficient for Gaussian filtering. But by combining causal and anti-causal systems like these, we can construct higher-order recursive systems with symmetric impulse responses that closely approximate the Gaussian function.

3 TWO METHODS

Deriche (1992) and van Vliet et al. (1998) describe different methods for designing recursive Gaussian and Gaussian derivative filters. Both methods are widely used in applications for computer vision.

In their designs, both methods exploit the scaling properties of Gaussian functions and their Fourier transforms. Specifically, from equations 4 and 5, we have

$$g(t;\sigma) = \frac{\sigma_0}{\sigma} g(t\sigma_0/\sigma;\sigma_0)$$

and

$$G(\omega;\sigma) = G(\omega\sigma/\sigma_0;\sigma_0).$$

This scaling property implies that poles and zeros computed for a Gaussian filter with half-width σ_0 can be used to quickly construct Gaussian filters for any half-width σ.

To understand this construction, consider one factor for one pole of a Gaussian filter designed for half-width σ_0:

$$H(z;\sigma_0) = \frac{1}{1 - \mathrm{e}^{i\omega_0} z^{-1}} \times \tilde{H}(z;\sigma_0),$$

where $\tilde{H}(z;\sigma_0)$ represents all of the other factors corresponding to the other poles and zeros of this system. The factor highlighted here has a single complex pole at

$$z = \mathrm{e}^{i\omega_0} = \mathrm{e}^{i(\nu_0 + i\mu_0)} = \mathrm{e}^{-\mu_0}\mathrm{e}^{i\nu_0}.$$

For this factor to represent a causal stable system, this pole must lie inside the unit circle in the complex z-plane, so we require $\mu_0 > 0$. In other words, stability of the causal factor requires that the pole lies in the upper half of the complex ω-plane.

Substituting $z = \mathrm{e}^{i\omega}$, the frequency response of this system is

$$H(\omega;\sigma_0) = \frac{1}{1 - \mathrm{e}^{i(\omega_0 - \omega)}} \times \tilde{H}(\omega;\sigma_0).$$

Now use the scaling property to design a new system with half-width σ:

$$H(\omega;\sigma) \approx H(\omega\sigma;\sigma_0) = \frac{1}{1 - \mathrm{e}^{i(\omega_0 - \omega\sigma)}} \times \tilde{H}(\omega\sigma;\sigma_0).$$

The scaling property here holds only approximately, because the frequency response $H(\omega;\sigma)$ only approximates the frequency response $G(\omega;\sigma)$ of an exact Gaussian filter.

The pole of the new system is at $\omega = \omega_0/\sigma$ in the complex ω-plane or at

$$z = \mathrm{e}^{i\omega_0/\sigma} = \mathrm{e}^{-\mu_0/\sigma}\mathrm{e}^{i\nu_0/\sigma}$$

in the complex z-plane. A similar scaling applies to all poles of the new system.

Note that increasing σ causes the poles of the system $H(z;\sigma)$ to move closer to the unit circle, yielding a longer impulse response, consistent with a wider Gaussian.

3.1 Deriche

Deriche (1992) constructs recursive Gaussian filters as a *sum* of causal and anti-causal systems:

$$H(z) = H^+(z) + H^-(z).$$

For Deriche's 4th-order filters, the causal system is

$$H^+(z) = \frac{b_0^+ + b_1^+ z^{-1} + b_2^+ z^{-2} + b_3^+ z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4}},$$

and the anti-causal system is

$$H^-(z) = \frac{b_1^- z + b_2^- z^2 + b_3^- z^3 + b_4^- z^4}{1 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4}.$$

To implement the composite system $H(z)$, we apply both causal and anti-causal filters to an input sequence $x[n]$, and accumulate the results in an output sequence $y[n]$. We apply the causal and anti-causal filters in *parallel*.

The coefficients a_1, a_2, a_3, and a_4 depend only on the locations of the filter poles, and these coefficients are the same for both the causal system $H^+(z)$ and the anti-causal system $H^-(z)$.

The coefficients b_1^+, b_2^+, b_3^+, and b_4^+ in the causal system $H^+(z)$ depend on the locations of the filter zeros.

The coefficients b_1^-, b_2^-, b_3^-, and b_4^- in $H^-(z)$ are easily computed from the other coefficients, because the composite filter $H(z)$ must be symmetric. That is, we require $H(z) = H^+(z) + H^-(z) = H(z^{-1})$. Therefore,

$$\begin{aligned} b_1^- &= b_1^+ - b_0^+ a_1 \\ b_2^- &= b_2^+ - b_0^+ a_2 \\ b_3^- &= b_3^+ - b_0^+ a_3 \\ b_4^- &= \quad - b_0^+ a_4. \end{aligned}$$

With these relations, the eight coefficients of the causal system $H^+(z)$ determine completely the symmetric impulse response $h[n]$ of the composite system $H(z)$. Those eight coefficients, in turn, depend on the poles and zeros of the causal system $H^+(z)$.

Deriche computes these poles and zeros to minimize a sum

$$E = \sum_{n=0}^{N} [h[n] - g(n;\sigma_0)]^2$$

of squared differences between the impulse response $h[n]$ and the sampled Gaussian function with half-width σ_0. Deriche chooses $\sigma_0 = 100$ and $N = 10\sigma_0 = 1000$.

A solution to this non-linear least-squares minimization problem is costly, but need be computed only once. After computing poles and zeros to approximate a Gaussian with half-width σ_0, Deriche uses the scaling properties to obtain poles and zeros for other σ.

Deriche solves a similar minimization problem to obtain poles and zeros for systems with impulse responses that approximate Gaussian 1st and 2nd derivatives.

3.2 van Vliet, Young, and Verbeek

van Vliet, Young and Verbeek (1998) construct recursive Gaussian filters as a *product* of causal and anti-causal systems:

$$H(z) = H^+(z) \times H^-(z).$$

For van Vliet et al.'s 4th-order filters, the causal system is

$$H^+(z) = \frac{b_0}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4}},$$

and the anti-causal system is

$$H^-(z) = \frac{b_0}{1 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4}.$$

To implement the composite system $H(z)$, we first apply the causal filter to an input sequence $x[n]$ to obtain an intermediate output sequence $y^+[n]$. We then apply the anti-causal filter to that sequence $y^+[n]$ to obtain a final output sequence $y[n]$. In this implementation, we apply the causal and anti-causal filters in *series*.

Once again, the coefficients a_1, a_2, a_3, and a_4 depend only on the locations of the filter poles, and these

coefficients are the same for both the causal system $H^+(z)$ and the anti-causal system $H^-(z)$. The symmetry of this system is easily verified; that is, $H(z) = H^+(z) \times H^-(z) = H(z^{-1})$.

van Vliet et al. compute the filter poles in two ways. They minimize either the rms error

$$L^2 = \left\{ \frac{1}{2\pi} \int_{-\pi}^{\pi} \mathrm{d}\omega \, [H(\omega;\sigma_0) - G(\omega;\sigma_0)]^2 \right\}^{1/2}$$

or the maximum error

$$L^\infty = \max_{|\omega|<\pi} |H(\omega;\sigma_0) - G(\omega;\sigma_0)|$$

in the frequency responses $H(\omega;\sigma)$ of the filters. For either error, they compute poles for $\sigma_0 = 2$, and then scale those poles to obtain Gaussian filters for other σ.

For Gaussian filters, van Vliet et al. show that the rms and maximum errors do not differ significantly for poles computed by minimizing either error.

For Gaussian derivative filters, van Vliet et al. propose the application of centered finite-difference filters before or after Gaussian filtering. The 1st-finite-difference system is

$$D_1(z) = \frac{1}{2}(z - z^{-1}),$$

and the 2nd-finite-difference system is

$$D_2(z) = z - 2 + z^{-1}.$$

These finite-difference filters only approximate differentiation, and the approximation is best for low frequencies where

$$D_1(\omega) = i \sin\omega \approx i\omega$$

and

$$D_2(\omega) = -2(1 - \cos\omega) \approx -\omega^2.$$

For higher frequencies, the error in these approximations may be significant.

3.3 Serial or parallel implementation

An advantage highlighted by van Vliet et al. is that their system has only poles, no zeros, and therefore requires fewer multiplications and additions than does Deriche's parallel system. A serial cascade of causal and anti-causal all-pole IIR filters is less costly than a comparable parallel (or serial) system of IIR filters with both poles and zeros.

Unfortunately, a disadvantage of any system that cascades causal and anti-causal filters lies in filtering sequences that have finite-length. To see this, assume an input sequence $x[n]$ with length N samples. Specifically, the input sequence $x[n]$ is non-zero for only $0 \le n < N$. Typically, we want a filtered output sequence $y[n]$ with the same length.

To compute the filtered output sequence $y[n]$, we first apply the causal recursive filter of our cascade system to the finite-length sequence $x[n]$ to obtain an intermediate sequence $y^+[n]$ that has *infinite length.* The intermediate sequence $y^+[n]$ may be non-zero for $0 \le n < \infty$. If we simply truncate that sequence $y^+[n]$ to zero for $n \ge N$, we discard samples that should be input to the second anti-causal recursive filter of our cascade system. The output sequence $y[n]$ is then incorrect, with significant errors for sample indices $n \approx N$. Figure 3 illustrates these errors with a simple example.

We might instead compute an extended intermediate sequence $y^+[n]$ for sample indices $0 \le n < N + L$, where L is the *effective length* of the first causal IIR filter. Beyond some length L, the impulse response of that filter may be approximately zero, and so may be negligible. For Gaussian filters with large half-widths σ, this effective length may be significant and costly in both memory and computation.

An alternative to estimating and using an effective length is to simply convert van Vliet et al.'s cascade system into an equivalent parallel system, using the method of partial fractions (e.g., Oppenheim and Schafer, 1999). In other words, we may use the cascade all-pole design of van Vliet et al. (1998) with the parallel poles-and-zeros implementation of Deriche (1992). Figure 4 illustrates this alternative parallel implementation.

A 4th-order parallel implementation of either Deriche's or van Vliet et al.'s recursive Gaussian filter requires 16 multiplications and 14 additions per output sample. *This cost is independent of the Gaussian half-width σ.*

Compare this cost with that for a non-recursive FIR filter obtained by sampling a Gaussian function $g(t;\sigma)$ that has been truncated to zero for $|t| > 4\sigma$: $1+4\sigma$ multiplications and 8σ additions. Assuming that the computational cost of a multiplication and addition are the same, a parallel recursive implementation is more efficient for $\sigma > 2.5$.

4 NUMERICAL TESTS

To test the design methods of Deriche and van Vliet et al., we used both methods to compute filter poles and zeros for 4th-order Gaussian and Gaussian derivative filters. We implemented both filters as parallel systems. While a parallel implementation of van Vliet et al.'s filter is no more efficient than Deriche's, it eliminates the truncation errors described above.

Figures 5 and 6 show both impulse responses and amplitude spectra for Gaussian (0th-derivative) filters, for three different half-widths σ. For small $\sigma \approx 1$, Deriche's filter is more accurate than van Vliet et al.'s filter, and the difference is most significant for higher frequencies.

However, for large $\sigma \approx 64$, Deriche's filter is much less accurate than van Vliet et al.'s filter; the accuracy of Deriche's filter degrades quickly with increasing σ.

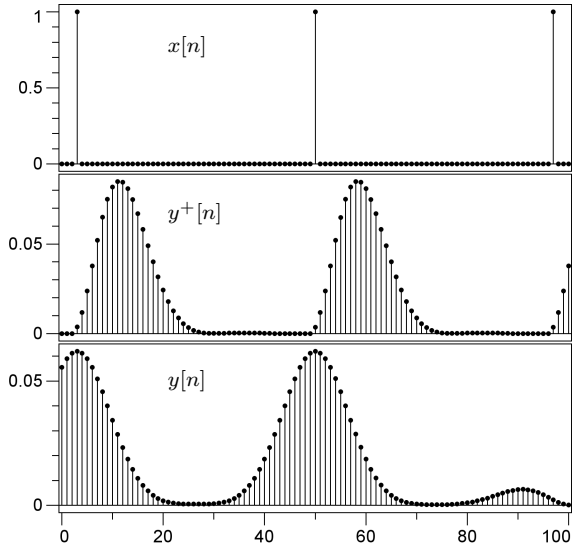

Figure 3. Series implementation of a recursive Gaussian filter applied to a sequence $x[n]$. After applying the first causal filter, samples with indices $n > 100$ of the intermediate output sequence $y^+[n]$ have been lost, and are therefore not seen by the second anti-causal filter applied to $y^+[n]$ to obtain the sequence $y[n]$. Compare with Figure 4.

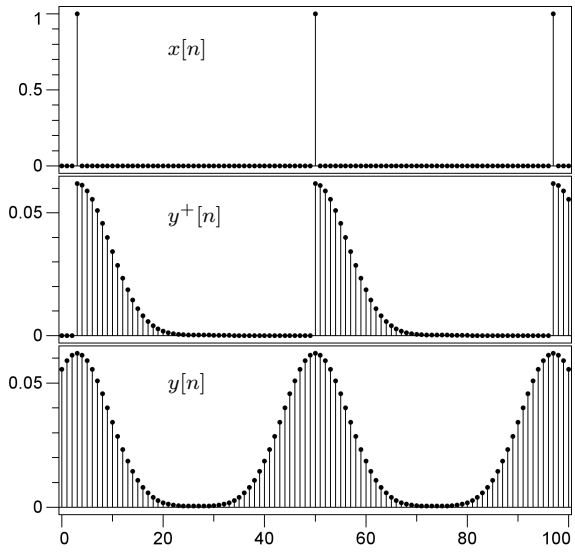

Figure 4. Parallel implementation of a recursive Gaussian filter applied to a sequence $x[n]$. Applying the first causal filter yields an intermediate output sequence $y^+[n]$. Applying the second anti-causal filter yields another intermediate output sequence $y^-[n]$ not shown. Accumulating these two intermediate sequences yields the output sequence $y[n]$. Compare with Figure 3.

We have used successfully van Vliet et al.'s filter for $\sigma > 1000$. For such large half-widths σ, Deriche's filter is useless.

For intermediate $\sigma \approx 8$, the errors in both methods appear to be insignificant in the scale of Figures 5 and 6. For $\sigma \approx 32$ (not shown), the rms errors for the two filters are equal and less than 3.4×10^{-4}.

The wide range of half-widths σ that we used to test Gaussian filters reflects differences between their applications. For low-pass filters, we would typically use smaller $\sigma < 32$, for which Deriche's filter is more accurate. For running averages, larger σ are more typical, and for $\sigma > 32$, van Vliet et al.'s filter is more accurate.

In applications of Gaussian derivative filters, smaller σ are typical. Figures 7 and 8 show both impulse responses and amplitude spectra for Gaussian 1st-derivative filters, for three different half-widths σ.

Recall that van Vliet et al. do not design filters explicitly for derivatives. Instead, they design Gaussian filters for use with centered finite-difference operators. The errors in van Vliet et al.'s 1st-derivative filters are due largely to errors in the 1st finite-difference operator. These errors decrease with increasing σ.

Likewise, the errors in van Vliet et al.'s 2nd-derivative filters are due largely to errors in the 2nd finite-difference operator. Figures 9 and 10 show both impulse responses and amplitude spectra for Gaussian 2nd-derivative filters, for three different half-widths σ. Again, the errors in van Vliet et al.'s derivative filters decrease with increasing σ.

In both Figures 8 and 10, for both 1st- and 2nd-derivative filters and $\sigma = 1$, we observe significant errors in Deriche's filters for frequencies near the Nyquist frequency. These errors are due to aliasing of the frequency responses $i\omega G(\omega; \sigma = 1)$ and $-\omega^2 G(\omega; \sigma = 1)$. As illustrated in Figure 2, these Fourier spectra are only approximately bandlimited to $|\omega| < \pi$, and the aliasing caused by unit sampling of the Gaussian derivatives $g'(t; \sigma = 1)$ and $g''(t; \sigma = 1)$ yields the errors at high frequencies seen in Figures 8 and 10.

5 CONCLUSION

Gaussian and Gaussian derivative filtering can be well approximated by recursive filters. We tested two design methods. Deriche's (1992) design is more accurate for small Gaussian half-widths $\sigma < 32$, and his parallel system accurately computes output samples near the ends of finite-length input signals. Gaussian derivative filters typically have small widths, and Deriche's design is best for those.

van Vliet et al.'s (1998) design is more accurate for large Gaussian half-widths $\sigma \geq 32$. To compute accurately the ends of finite-length signals, we convert their serial all-pole system to a parallel poles-and-zeros system like that of Deriche.

When computing derivatives, typical Gaussian half-widths might be as small as $\sigma < 3$. Then, we may simply convolve with a sampled finite-length approximation to a Gaussian derivative. Indeed, although both Deriche and van Vliet et al. focus specifically on Gaussian derivatives, we find their Gaussian (0th-derivative) filters for larger σ to be most useful.

In summary, the results of our numerical tests lead us to suggest the following implementations of Gaussian filtering for different half-widths σ:

$\sigma < 3$:	Direct convolution with an FIR filter obtained by sampling a Gaussian function $g(t;\sigma)$ truncated to zero for $\lvert t\rvert > 4\sigma$.
$3 \leq \sigma < 32$:	Recursive filtering with a parallel IIR filter designed by the method of Deriche (1992).
$\sigma \geq 32$:	Recursive filtering with a parallel IIR filter designed by the method of van Vliet et al. (1998).

Source code for our implementations of both Deriche's and van Vliet et al.'s filters is available in the *Mines Java Toolkit*.

REFERENCES

Bracewell, R., 1978, The Fourier transform and its applications (2nd edition): McGraw-Hill.

Deriche, R., 1992, Recursively implementing the Gaussian and its derivatives: Proceedings of the 2nd International Conference on Image Processing, Singapore, p. 263–267.

Neidell, N.S. and Taner M.T., 1971, Semblance and other coherency measures for multichannel data: Geophysics, v. 36, no. 3, p. 482–497.

Oppenheim, A.V., and Schafer, R.W., 1999, Discrete-time signal processing (2nd edition): Prentice-Hall.

Snieder, R., 2006, The theory of coda wave interferometry: Pure and Applied Geophysics, v. 163, p. 455–473.

van Vliet, L., Young, I., and Verbeek, P. 1998, Recursive Gaussian derivative filters: Proceedings of the International Conference on Pattern Recognition, Brisbane, p. 509–514.

van Vliet, L.J. and Verbeek, P.W., 1995, Estimators for orientation and anisotropy in digitized images: ASCI'95, Proceedings of the First Annual Conference of the Advanced School for Computing and Imaging, p. 442–450.

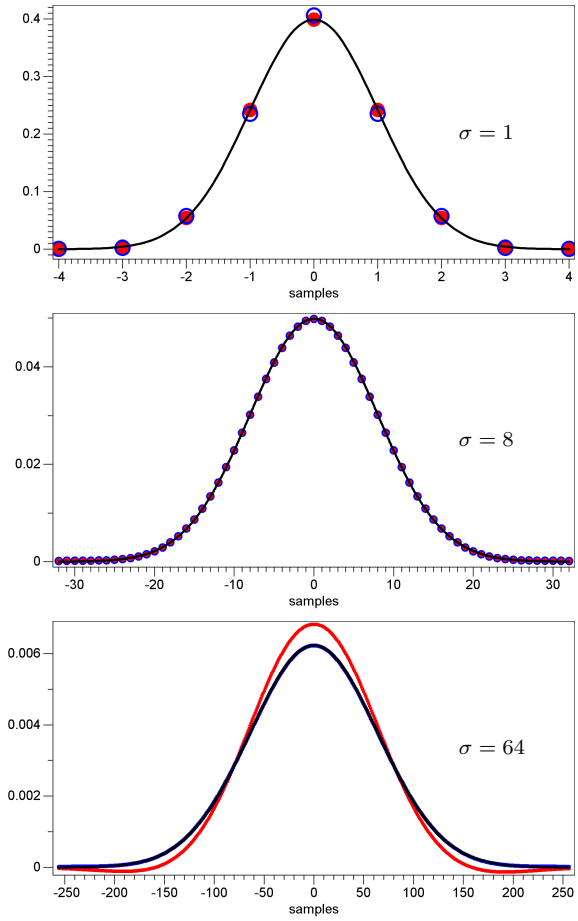

Figure 5. Impulse responses of recursive Gaussian filters designed by the methods of Deriche (light red filled circles) and van Vliet et al. (dark blue hollow circles) for three different values of σ. The solid black curve is the exact Gaussian function.

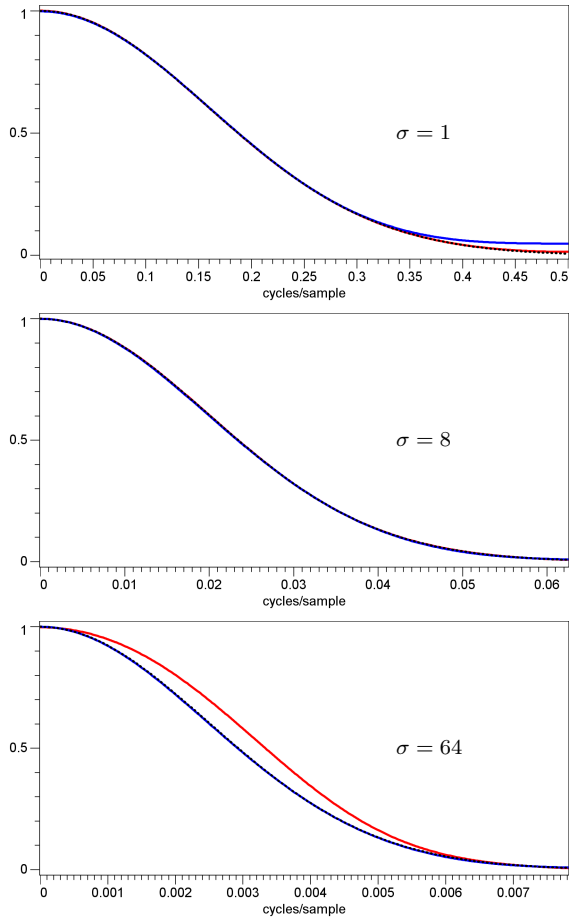

Figure 6. Amplitude responses of recursive Gaussian filters designed by Deriche (light red) and by van Vliet et al. (dark blue) for three different values of σ. For comparison, the dotted black curve is the exact Gaussian function, which closely matches the lower curve in all three plots.

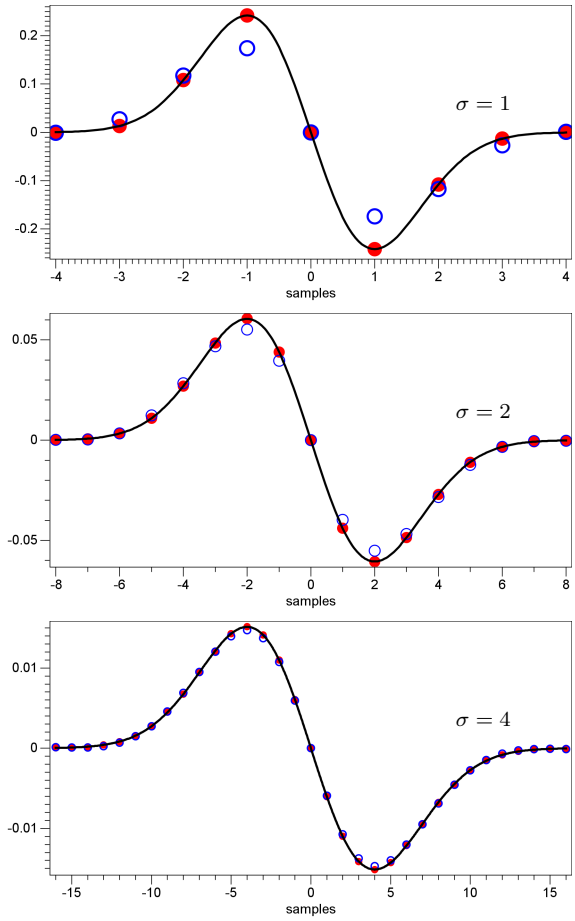

Figure 7. Impulse responses of recursive Gaussian 1st-derivative filters designed by the methods of Deriche (light red filled circles) and van Vliet et al. (dark blue hollow circles) for three different values of σ. The solid black curve is the exact Gaussian 1st-derivative.

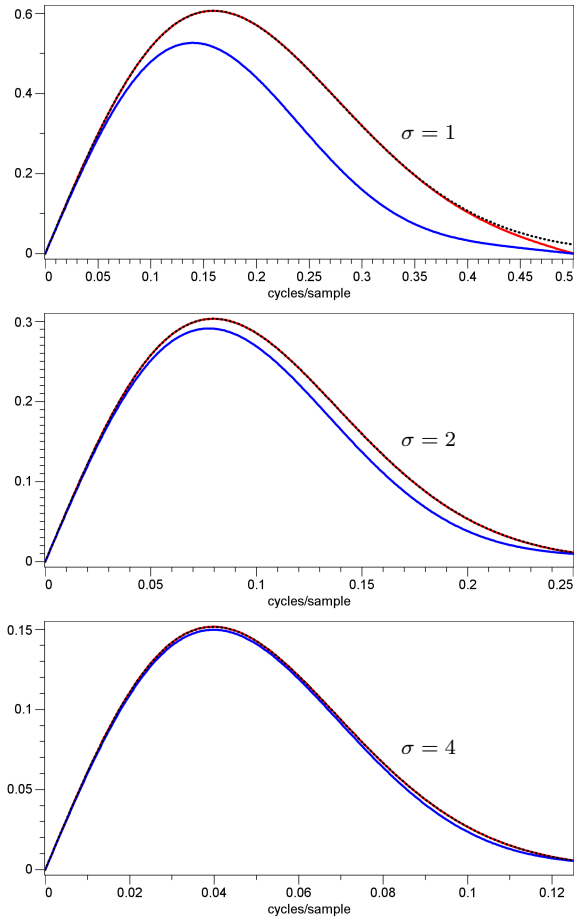

Figure 8. Amplitude responses of recursive Gaussian 1st-derivative filters designed by Deriche (light red) and by van Vliet et al. (dark blue) for three different values of σ. For comparison, the dotted black curve is the exact Gaussian 1st-derivative, which closely matches the upper (Deriche) curve in all three plots.

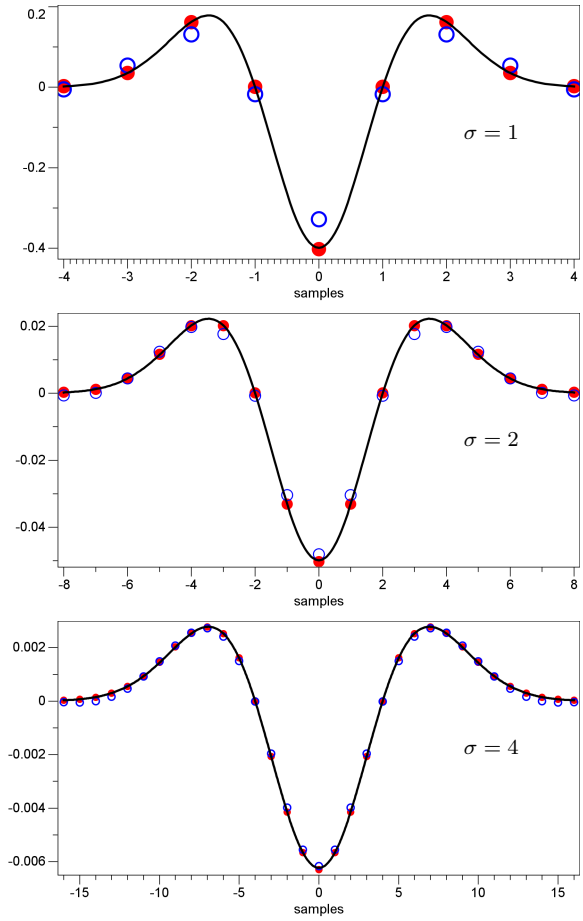

Figure 9. Impulse responses of recursive Gaussian 2nd-derivative filters designed by the methods of Deriche (light red filled circles) and van Vliet et al. (dark blue hollow circles) for three different values of σ. The solid black curve is the exact Gaussian 2nd-derivative.

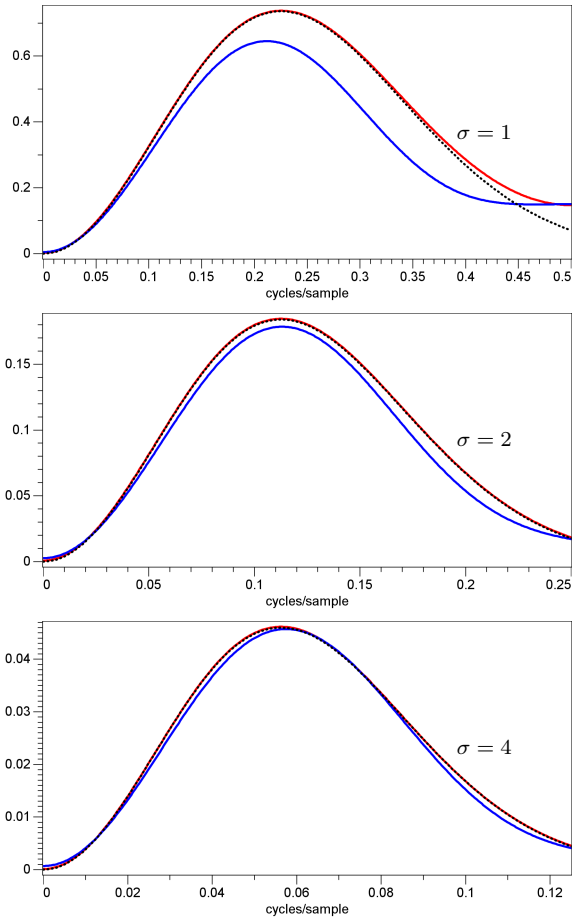

Figure 10. Amplitude responses of recursive Gaussian 2nd-derivative filters designed by Deriche (light red) and by van Vliet et al. (dark blue) for three different values of σ. For comparison, the dotted black curve is the exact Gaussian 2nd-derivative, which closely matches the upper (Deriche) curve in all three plots.