An efficient method for computing local cross-correlations of multi-dimensional signals

Dave Hale

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

ABSTRACT

Consider two multi-dimensional digital signals, each with N_s samples. For some number of lags $N_l \ll N_s$, the cost of computing a single cross-correlation of these two signals is proportional to $N_s \times N_l$.

By exploiting several properties of Gaussian windows, we can compute N_s *local* cross-correlations, again with computational cost proportional to $N_s \times N_l$. Here, *local* means the cross-correlation of signals after applying a Gaussian window centered on a single sample. Computational cost is independent of the size of the window.

Key words: digital signal processing, image processing

1 INTRODUCTION

Cross-correlations are ubiquitous in digital signal processing. We use cross-correlations to estimate relative shifts between two signals, and to compute filters that shape one signal to match another. We use auto-correlations (a special case of cross-correlations) to compute prediction error filters, and to estimate the orientations of features in multi-dimensional images.

In such applications we must assume that the quantities estimated do not vary significantly for the duration of the signals cross-correlated. But those quantities often do vary, and the variations can be important.

For example, attenuation of seismic waves implies that prediction error filters derived from seismograms should vary continuously with time. In other words, estimates of prediction error filters that vary with time contain information about seismic attenuation (Hale, 1982).

Likewise, multi-dimensional prediction or prediction error filters contain information about the orientations of features in multi-dimensional seismic images. Such filters may be used to attenuate noise (Abma and Claerbout, 1995; Gülünay, 2000) or to interpolate missing samples (Crawley et al., 1999). In any multi-dimensional image of interest, the orientations and the corresponding filters are not constant.

Time shifts estimated with cross-correlations and applied to seismograms to compensate for near-surface variations may also vary with time (e.g., MacKay et al, 2003). And those variations contain information about the near surface.

In seismic interferometry, time shifts estimated with cross-correlation of seismic coda may change with time. We may use that change to estimate seismic wave velocities (Pacheco and Snieder, 2003).

In time-lapse seismic imaging, we may cross-correlate two 3-D images to derive corrections for differences in seismic acquisition and processing (e.g., Rickett and Lumley, 2001). Because those differences vary spatially, a single cross-correlation does not yield adequate corrections.

In all of these applications, we cross-correlate signals to obtain estimates of important parameters that cannot be constant for the duration of those signals.

To account for variations in estimated parameters, we might compute *local* cross-correlations. We might first truncate or taper our signals to zero outside some specified window, and then cross-correlate those windowed signals. We might compute a suite of local cross-correlations by repeating these window-and-correlate steps for multiple, perhaps overlapping, windows. In this process, we must choose the number of windows and their shape and size.

Our choice of window size is an unavoidable compromise. Ideally, we use windows large enough to provide meaningful estimates of parameters, but small enough that we may reasonably assume those param-

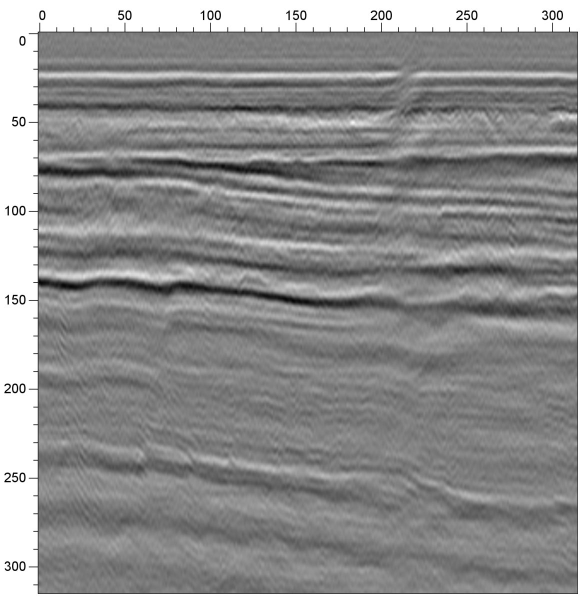

Figure 1. A 2-D seismic image, with 315×315 samples.

eters are constant within each window. The uncertainty relation (e.g.; Bracewell, 1978) constrains this choice.

In contrast, we often choose the number of windows and their shape simply to reduce computational costs. For example, we might avoid a Gaussian window shape (although that shape is optimal with respect to the uncertainty relation) because the Gaussian function is never zero. We might also choose a small number of windows because the cost of the repeated window-and-correlate process is proportional to that number.

Here I describe an efficient method for computing a local cross-correlation for *every sample* in a multi-dimensional signal. The number of windows equals the number of signal samples, and their shape is Gaussian.

Figures 2 and 3 display local auto-correlations for the small 2-D seismic image shown in Figure 1. The method described in this paper computes a local auto-correlation for every sample in that image. Figure 2 shows one lag $(5, 5)$ from each of those auto-correlations. Figure 3 shows $225 = 15 \times 15$ lags, but for only $1/225$ of the auto-correlations computed.

Note that the 2-D local auto-correlations displayed in Figure 3 vary significantly. Each auto-correlation contains local information about the image of Figure 1.

Surprisingly, the cost of computing a local auto-correlation for each sample in this image *is independent of the size of the Gaussian window*. Indeed, the cost (number of floating-point operations) of computing almost $100,000$ local auto-correlations in this example is only about 32 times greater than that of computing a single non-local auto-correlation of the entire image.

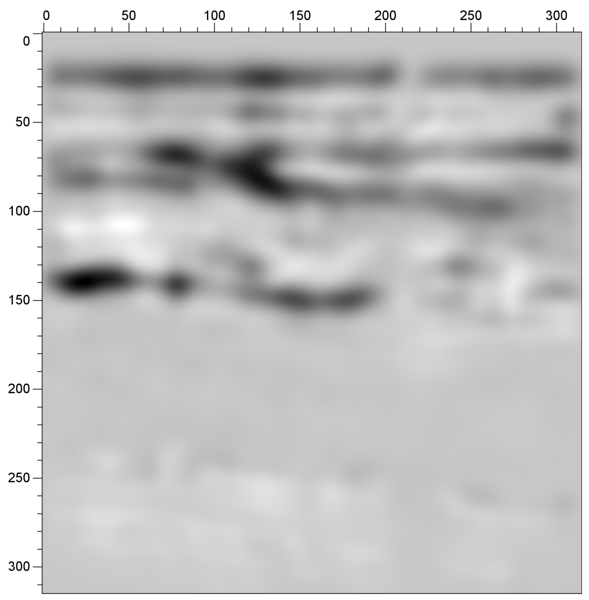

Figure 2. Local auto-correlations of the image in Figure 1 for a single lag (5,5) — five samples in both the horizontal and vertical directions. The local window is a two-dimensional Gaussian function with radius $\sigma = 8$ samples.

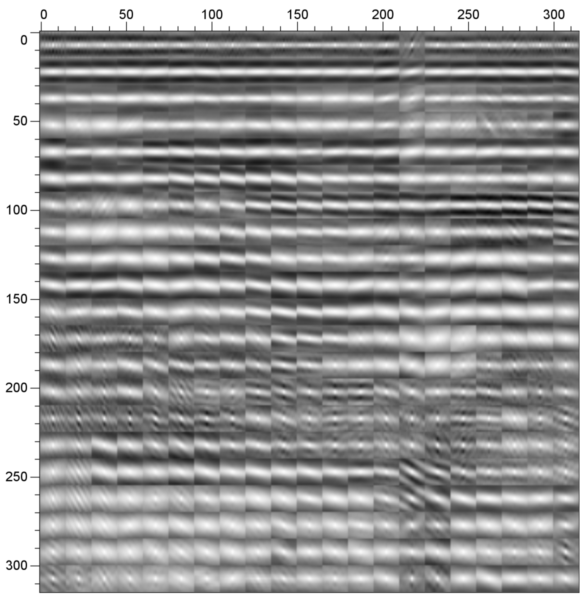

Figure 3. A subset of the local auto-correlations of the image in Figure 1. Shown here are $225 = 15 \times 15$ lags for only $1/225$ of the local auto-correlations computed. Each auto-correlation is normalized by its value at lag (0,0).

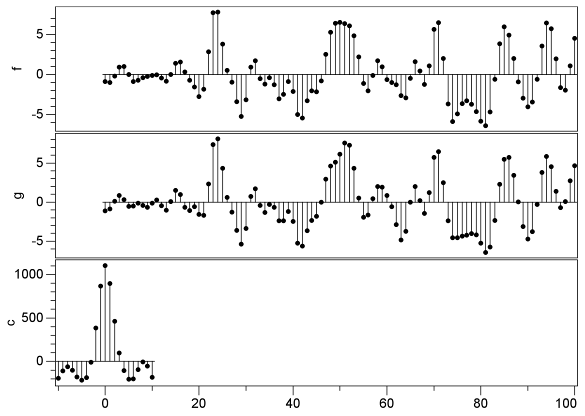

Figure 4. Cross-correlation c of two sequences f and g.

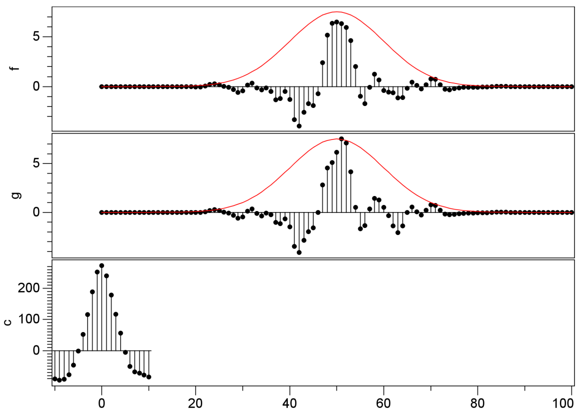

Figure 5. Cross-correlation c of two Gaussian-windowed sequences f and g.

2 CROSS-CORRELATION

We first consider the familiar cross-correlation of two 1-D analog signals f and g, defined by:

$$c(u) = (f \star g)(u) \equiv \int_{-\infty}^{\infty} \mathrm{d}x \; f(x) \; g(x+u), \tag{1}$$

where u denotes the cross-correlation lag. With the change of variables $x \to x - u/2$, we obtain a centered definition

$$c(u) = (f \star g)(u) \equiv \int_{-\infty}^{\infty} \mathrm{d}x \; f(x-u/2) \; g(x+u/2) \tag{2}$$

that will be useful in our derivation of local cross-correlations below.

From the centered form of equation 2, we can easily confirm the relation

$$c(u) = (f \star g)(u) = (g \star f)(-u).$$

A cross-correlation is generally not a symmetric function. Of course, if the signals f and g are identical, then $c(u)$ is an auto-correlation

$$r(u) = (f \star f)(u) = (f \star f)(-u) = r(-u),$$

which is symmetric in the lag variable u. In our local cross-correlations below, it is important that we preserve these relations.

The sampled version of equation 1 is

$$c[l] = (f \star g)[l] \equiv \sum_{j=-\infty}^{\infty} f[j] \; g[j+l], \tag{3}$$

where the new lag variable l is constrained to be an integer. Figure 4 shows an example of the sampled cross-correlation c of two sequences f and g. Note that the number of lags $N_l = 21$ for which we have computed the cross-correlation c is significantly less than the number of samples $N_s = 101$ in the two sequences f and g. This scenario $N_l \ll N_s$ is common in digital signal processing. When $N_l \approx N_s$, we might more efficiently compute the cross-correlation via fast Fourier transforms, after padding the sequences f and g sufficiently with zeros to avoid aliasing.

However, for $N_l \ll N_s$, the most efficient way to compute a cross-correlation is to simply evaluate the sum as written in equation 3. In this case, the computational complexity of cross-correlation is clearly $O(N_s \times N_l)$. This means, for example, that the cost of cross-correlation doubles if we double either the number of samples N_s or the number of lags N_l.

The generalization to multi-dimensional signals is straightforward. For two-dimensional signals, equation 3 becomes

$$c[l_1,l_2] = \sum_{j_1=-\infty}^{\infty} \sum_{j_2=-\infty}^{\infty} f[j_1,j_2] \; g[j_1+l_1,j_2+l_2]. \tag{4}$$

In two dimensions, lags have two components l_1 and l_2. The computational complexity is again $O(N_s \times N_l)$, where N_s is the total number of samples in the 2-D signals, and N_l is the number of 2-D lags for which we compute the cross-correlation.

3 GAUSSIAN WINDOWS

To obtain a local cross-correlation, we might apply a smooth window function w to each of the sequences f and g, before computing the cross-correlation via equation 3. Figure 5 illustrates this process. In this example, the window function w is a Gaussian defined by

$$w(x) \equiv \mathrm{e}^{-x^2/2\sigma^2}, \tag{5}$$

with Fourier transform

$$W(k) \equiv \int_{-\infty}^{\infty} \mathrm{d}x \; \mathrm{e}^{-ikx} w(x) = \sqrt{2\pi}\sigma \; \mathrm{e}^{-k^2\sigma^2/2}.$$

We choose a Gaussian window because it has four desirable properties:

(i) Optimal resolution. Only the Gaussian window minimizes the resolution product $\Delta k \times \Delta x$, where Δx and Δk denote consistently-defined widths of $w(x)$ and $W(k)$, respectively.

(ii) Isotropic *and* separable in N dimensions. Only the Gaussian window is both isotropic and separable.

(iii) Fast recursive-filter implementation. The cost of applying a filter with an approximately Gaussian impulse response is independent of the filter length.

(iv) The product of any two Gaussians with equal widths is a Gaussian.

Property (i) follows from the uncertainty relation. Multiplication by any window function $w(x)$ corresponds to convolution in the wavenumber k domain, and this convolution smears the Fourier transforms $F(k)$ and $G(k)$ of the signals $f(x)$ and $g(x)$. This smearing implies a loss of resolution in those Fourier spectra. For the Gaussian window, with $\Delta x = \sigma$, we have $\Delta k = 1/\sigma$, and the product $\Delta k \times \Delta x = 1$. No other window function $w(x)$ has a smaller resolution product (e.g., Bracewell, 1978). The Gaussian window smears the Fourier spectra of $f(x)$ and $g(x)$ less than does any other.

Property (ii) highlights two features that are important when processing multi-dimensional signals: isotropy and separability. Many window functions have one or the other of these two features, but only the Gaussian window has both (Sahoo and Kannappan, 1992). Isotropic windows have a width and shape that is the same in all directions. Isotropy implies no directional bias in multi-dimensional correlations.

Separability implies computational efficiency. An N-dimensional Gaussian window is simply the product of N one-dimensional Gaussian windows. For two dimensions, the Gaussian window $w(x)$ of equation 5 leads to

$$\begin{aligned} w(x_1,x_2) &\equiv \mathrm{e}^{-(x_1^2+x_2^2)/2\sigma^2} \\ &= \mathrm{e}^{-x_1^2/2\sigma^2} \times \mathrm{e}^{-x_2^2/2\sigma^2} \\ &= w(x_1) \times w(x_2). \end{aligned}$$

From this result, we can readily show that convolution with an N-dimensional Gaussian window (filter) can be performed by applying a sequence of one-dimensional Gaussian filters. For two dimensions,

$$\begin{aligned} & w(x_1,x_2) * f(x_1,x_2) \\ &\quad \equiv \int_{-\infty}^{\infty} \mathrm{d}\xi_1 \int_{-\infty}^{\infty} \mathrm{d}\xi_2 \; w(x_1-\xi_1, x_2-\xi_2) f(\xi_1,\xi_2) \\ &\quad = \int_{-\infty}^{\infty} \mathrm{d}\xi_1 \; w(x_1-\xi_1) \int_{-\infty}^{\infty} \mathrm{d}\xi_2 \; w(x_2-\xi_2) f(\xi_1,\xi_2) \\ &\quad = w(x_1) *_1 w(x_2) *_2 f(x_1,x_2), \end{aligned}$$

where $*_1$ and $*_2$ denote one-dimensional convolutions in the 1st and 2nd dimensions, respectively.

Property (iii) implies that application of those one-dimensional Gaussian filters is efficient, with computational cost independent of the parameter σ. Specifically, for half-widths $\sigma > 3$ samples, a recursive implementation requires fewer multiplications and additions than does convolution with a typical finite-length approximation to a Gaussian filter (Deriche, 1992; van Vliet et al., 1998; Hale, 2006b). A recursive implementation is also more efficient than one based on fast Fourier transforms.

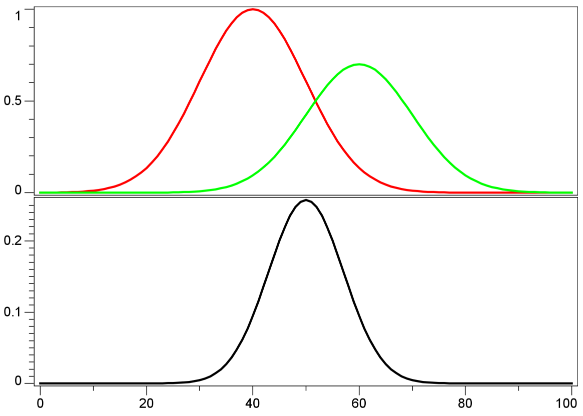

Figure 6. The product of two Gaussian windows (upper) with the same half-widths σ is a Gaussian window (lower) with half-width $\sigma/\sqrt{2}$.

The last property (iv), though trivial to prove, is perhaps less well-known than the first three. Using the Gaussian window $w(x)$ defined by equation 5, the simple proof is:

$$\begin{aligned} & a\ w(x-u/2) \times b\ w(x+u/2) \\ &\quad = \ a\ \mathrm{e}^{-(x-u/2)^2/2\sigma^2} \times b\ \mathrm{e}^{-(x+u/2)^2/2\sigma^2} \\ &\quad = \ ab\ \mathrm{e}^{-u^2/4\sigma^2}\ \mathrm{e}^{-x^2/\sigma^2}. \end{aligned} \tag{6}$$

Figure 6 illustrates this property, for $u = 20$. Equation 6 shows that changing the separation u of the peaks of two Gaussians changes only the amplitude, not the width, of their Gaussian product. Property (iv) is essential in the efficient computation of local cross-correlations described below.

4 LOCAL CROSS-CORRELATIONS

Consider now the local cross-correlation process illustrated in Figure 5. This figure shows the computation of a single local cross-correlation. If we slide the Gaussian window to the right or left, we obtain a different local cross-correlation. Indeed, we could compute N_l lags of a local cross-correlation for each of the N_s samples in the sequences f and g by centering a Gaussian window on each of those samples.

Described this way, we might expect the computational complexity of the repeated process of windowing and cross-correlation to be $O(N_s \times N_w \times N_l)$, where N_w is the number of non-zero samples in each Gaussian window, and N_s is the number of samples on which we can center that window. Because the Gaussian window is never zero, $N_w = N_s$. In practice, however, we might truncate to zero each Gaussian window for samples far from its central peak.

A computational complexity of $O(N_s \times N_w \times N_l)$ would be costly, especially for large windows of multi-dimensional signals. However, using properties (ii), (iii), and (iv) of Gaussian windows listed above, we can reduce this complexity to only $O(N_l \times N_s)$.

For the moment, consider f and g to be continuous functions of x, and define the windowed signals

$$\begin{aligned} f(x;y) &\equiv f(x)w(y-x) \\ g(x;y) &\equiv g(x)w(y-x). \end{aligned}$$

Here, $w(y-x)$ is a Gaussian window centered at location $x = y$. Then define cross-correlation of these windowed signals by

$$\begin{aligned} c(y;u) &\equiv \int_{-\infty}^{\infty} \mathrm{d}x\, f(x;y)\ g(x+u;y) \\ &= \int_{-\infty}^{\infty} \mathrm{d}x\, f(x-u/2;y)\ g(x+u/2;y) \\ &= \int_{-\infty}^{\infty} \mathrm{d}x\, f(x-u/2)\ g(x+u/2) \\ &\qquad \times w(y-x+u/2)\ w(y-x-u/2) \\ &= \int_{-\infty}^{\infty} \mathrm{d}x\, f(x-u/2)g(x+u/2)\ v(y-x;u), \end{aligned} \tag{7}$$

where equations 6 and 7 define

$$\begin{aligned} v(x;u) &\equiv w(x+u/2) \times w(x-u/2) \\ &= \mathrm{e}^{-u^2/4\sigma^2}\mathrm{e}^{-x^2/\sigma^2}. \end{aligned} \tag{8}$$

For any lag u, equation 7 shows that we can compute local cross-correlations $c(y;u)$ for all y by *convolving* the Gaussian filter $v(x;u)$ with the product of shifted signals $f(x-u/2)g(x+u/2)$.

In practice, we must compute the cross-correlation $c(y;u)$ for sampled integer values of y and u. For odd integer lags u, the arguments $x \mp u/2$ of f and g in equation 7 suggest that interpolation of those signals may be necessary. To avoid that interpolation, we decompose any integer lag $u = l$ into two integer parts:

$$l_f \equiv \lfloor l/2 \rfloor, \quad l_g \equiv \lceil l/2 \rceil,$$

where $\lfloor x \rfloor$ denotes the largest integer not greater than x, and $\lceil x \rceil$ denotes the smallest integer not less than x. Then $l = l_g + l_f$, $|l| \bmod 2 = l_g - l_f$ and, with a change of variables, equation 7 becomes

$$\begin{aligned} c(y;l) &= \int_{-\infty}^{\infty} \mathrm{d}x\, f(x-l_f)\ g(x+l_g) \\ &\qquad \times v(y-x-(|l| \bmod 2)/2;l). \end{aligned}$$

Finally, integer sampling of the variables x and y yields

$$\begin{aligned} c[k;l] &= \sum_{j=-\infty}^{\infty} f[j-l_f]\ g[j+l_g] \\ &\qquad \times v(k-j-(|l| \bmod 2)/2;l). \end{aligned} \tag{9}$$

For any integer lag l, equation 9 suggests that local cross-correlations $c[k;l]$ be computed in two steps:

(i) Compute a sample-by-sample product of shifted sequences f and g:

$$h[j;l] \equiv f[j-l_f]\ g[j+l_g]. \tag{10}$$

(ii) Filter that product by convolving with a Gaussian window, which for odd lags l is shifted by half a sample:

$$c[k;l] = \sum_{j=-\infty}^{\infty} h[j;l]v(k-j-(|l| \bmod 2)/2;l). \tag{11}$$

As discussed above, recursive filters exist that approximate well this convolution with a Gaussian window. To handle the case of odd lags l, we might incorporate the half-sample shift in equation 11 into our design of those recursive filters. Alternatively, we simply approximate the shift by linear midpoint interpolation of two Gaussian windows:

$$v(k-(|l| \bmod 2)/2;l) \approx (v[k;l] + v[k-|l| \bmod 2;l])/2,$$

which leads to

$$\begin{aligned} &c[k;l] \\ &= \frac{1}{2}\sum_{j=-\infty}^{\infty} h[j;l]\ (v[k-j;l] + v[k-j-|l| \bmod 2;l]) \\ &= \frac{1}{2}\sum_{j=-\infty}^{\infty} (h[j;l] + h[j-|l| \bmod 2;l])\ v[k-j;l] \\ &= \sum_{j=-\infty}^{\infty} \tilde{h}[j;l]\ v[k-j;l], \end{aligned} \tag{12}$$

where

$$\tilde{h}[j;l] \equiv (f[j-l_f]\ g[j+l_g] + f[j-l_g]\ g[j+l_f])/2.$$

Note that $\tilde{h}[j;l]$ is simply $h[j;l]$ for even lags l. For odd lags, $\tilde{h}[j;l]$ is a linear midpoint interpolation of two values of $h[j;l]$.

In some applications, the approximation of linear interpolation may be inadequate. This approximation corresponds to using slightly different windows for the even and odd lags of local cross-correlations. That difference can be significant in applications such as local prediction error filtering, where it may cause normal equations to become singular (Hale, 2006a). In such applications, we replace linear midpoint interpolation of two samples with an 8-sample interpolation derived from a windowed sinc function.

In any case, implementation of equation 12 is straightforward. We first compute a sum $\tilde{h}[j;l]$ of sample-by-sample products of shifted sequences f and g, and then filter that sum with a Gaussian window.

With recursive implementations, the cost of the Gaussian filter in equation 12 is proportional to the number N_s of samples to be filtered, and that cost is independent of the width of the Gaussian. Therefore, the complexity of computing $c[k;l]$ for any single lag l is $O(N_s)$.

The complexity of computing N_l lags of N_s local cross-correlations is $O(N_l \times N_s)$! In other words, the computational complexity for N_l lags of N_s local cross-correlations is the same as that for N_l lags of a *single* non-local cross-correlation.

This same computational complexity extends to multiple dimensions. In two dimensions, for any 2-D lag (l_1, l_2), we first compute the sum

$$\begin{aligned}\tilde{h}[j_1,j_2;l_1,l_2] = (&\\ &f[j_1 - l_{1f}, j_2 - l_{2f}]\ g[j_1 + l_{1g}, j_2 + l_{2g}] + \\ &f[j_1 - l_{1g}, j_2 - l_{2f}]\ g[j_1 + l_{1f}, j_2 + l_{2g}] + \\ &f[j_1 - l_{1f}, j_2 - l_{2g}]\ g[j_1 + l_{1g}, j_2 + l_{2f}] + \\ &f[j_1 - l_{1g}, j_2 - l_{2g}]\ g[j_1 + l_{1f}, j_2 + l_{2f}])/4,\end{aligned}$$

where

$$\begin{aligned} l_{1f} &\equiv \lfloor l_1/2 \rfloor, \qquad l_{1g} \equiv \lceil l_1/2 \rceil, \\ l_{2f} &\equiv \lfloor l_2/2 \rfloor, \qquad l_{2g} \equiv \lceil l_2/2 \rceil,\end{aligned}$$

and then apply a two-dimensional Gaussian filter to that sum to obtain local cross-correlations $c[k_1,k_2;l_1,l_2]$. Recall that the two-dimensional Gaussian filter is equivalent to a cascade of two one-dimensional Gaussian filters. Again, the computational complexity is $O(N_l \times N_s)$, where N_s is the total number of samples in the 2-D signals, and N_l is the number of 2-D lags for which we compute local cross-correlations. This is the process that we used to compute the local auto-correlations shown in Figures 2 and 3.

The actual cost of computing N_s local cross-correlations is of course higher than that of computing a single non-local cross-correlation, but only by a constant factor that is independent of the number of lags N_l and the number of samples N_s.

That constant factor depends primarily on the number of coefficients used in recursive approximations to Gaussian filtering. In our implementations, these filters require about 16 multiplications and 16 additions per sample per dimension filtered. A single cross-correlation requires only one multiplication and one addition per sample. So, for the 2-D image of Figure 1, the constant factor is about $32 = 16 \times 2$. For 3-D images, the constant factor would be about $48 = 16 \times 3$.

In other words, the cost in floating-point operations of computing almost 100,000 local auto-correlations for the 2-D image of Figure 1 is only about 32 times greater than that of computing a single non-local auto-correlation of that same image.

5 CONCLUSION

When computing N_s local cross-correlations for all N_s samples of two signals by conventional windowing, we can imagine three loops: an outermost loop over N_s windows, an inner loop over N_l lags, and an innermost loop where we multiply and sum N_w non-zero samples in each window. These three loops imply a computational complexity of $O(N_s \times N_l \times N_w)$.

The trick described in this paper is to first rearrange the loops: an outermost loop over N_l lags, an inner loop over N_s samples, and an innermost loop over N_w non-zero samples in each window. We then recognize the two inner loops over N_s and N_w as a convolution that we can replace with recursive filtering, with cost that is independent of N_w. So the computational complexity is only $O(N_l \times N_s)$.

In summary, we rearrange the computation so that Gaussian windowing becomes recursive Gaussian filtering.

In this rearrangement, we must take care to ensure that important properties of cross-correlations are preserved. For example, in local prediction error filtering, it is important that each local auto-correlation be symmetric about zero lag, and that the system of normal equations to be solved for prediction error coefficients is both symmetric and positive-definite.

It is easy to derive a local correlation-like sequence that would not have these properties. For example,

$$c[k;l] = \sum_{j=-\infty}^{\infty} f[j]\ g[j+l]\ w[k-j],$$

where $w[j]$ is a sampled Gaussian window defined by equation 5, defines a local correlation-like sequence that is simpler and easier to implement than equation 12. But this simpler definition does not yield local auto-correlations that are symmetric. The changes of variable and half-sample shifts used to derive equation 12 above are required to obtain accurate local cross-correlations.

The method described in this paper computes a local cross-correlation for every sample in a multi-dimensional signal. This capability leads us to ask two questions. Do we need to keep all of them? And, if not, how many do we need?

On the one hand, the smooth variations in Figure 2 suggest that we might subsample local auto-correlations without aliasing. That smoothness depends on our choice of Gaussian half-width σ. In that example, we used a Gaussian radius $\sigma = 8$. Had we used a larger radius, the features in Figure 2 would have been even smoother.

On the other hand, it is possible for local cross-correlations to be aliased even when the signals being correlated are not. This is because the bandwidth of the product of two signals (recall equation 10) is the sum of the bandwidths of those two signals.

We leave these questions of sampling and aliasing of local cross-correlations to another paper.

ACKNOWLEDGMENT

Thanks to Ken Larner and John Stockwell for their helpful reviews of this manuscript.

REFERENCES

Abma, R., and Claerbout, J., 1995, Lateral prediction for noise attenuation by t-x and f-x techniques: Geophysics, v. 60, no. 6, p. 1887–1896.

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

Crawley, S., Clapp, R., and Claerbout, J., 1999, Interpolation with smoothly non-stationary prediction-error filters: 69th Annual International Meeting, SEG, Expanded Abstracts, p. 1154–1157.

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

Gülünay, N., 2000, Noncausal spatial prediction filtering for random noise reduction on 3-D poststack data: Geophysics, v. 65, no. 5, p. 1641–1653.

Hale, D., 1982, Q-adaptive deconvolution: Stanford Exploration Project Report, no. 30, p. 133–158.

Hale, D., 2006a, Seamless local prediction filtering: this report.

Hale, D., 2006b, Recursive Gaussian filters: this report.

Kannappan, P., and Sahoo, P.K., 1992, Rotation invariant separable functions are Gaussian: SIAM Journal on Mathematical Analysis, v. 23, no. 5, p. 1342–1351.

MacKay, S., Fried, J., and Carvill, C., 2003, The impact of water-velocity variations on deepwater seismic data: The Leading Edge, v. 22, p. 344–350.

Pacheco, C., and Snieder, R., 2003, Time-lapse monitoring with multiply scattered waves: 73rd Annual International Meeting, SEG, Expanded Abstracts, p. 1849–1852.

Rickett, J.E., and Lumley, D.E., 2001, Cross-equalization data processing for time-lapse seismic reservoir monitoring — a case study from the Gulf of Mexico: Geophysics, v. 66, no. 4, p. 1015–1025.

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.