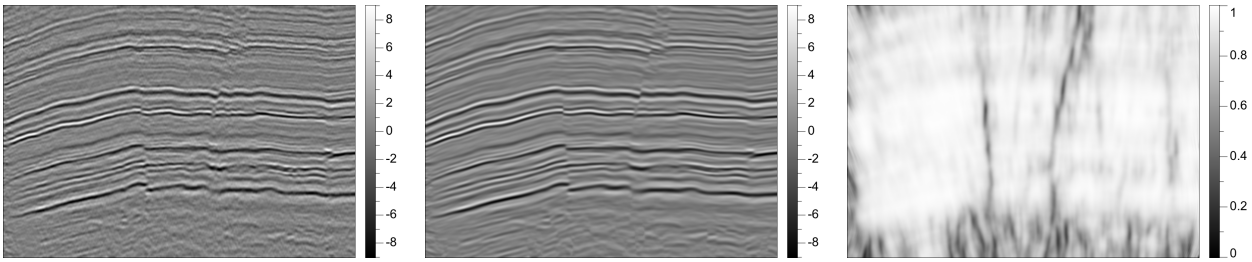
Structure-oriented smoothing and semblance

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



(a) (b) (c)

Figure 1. A seismic image (a) after applying structure-oriented smoothing (b) and semblance (c) filters.

ABSTRACT
Smoothing along structures apparent in seismic images can enhance these structural features while preserving important discontinuities such as faults or channels. Filters appropriate for such smoothing must seamlessly adapt to variations in the orientation and coherence of image features. I describe an implementation of smoothing filters that does this and is both computationally efficient and simple to implement.
Structure-oriented filters lead naturally to the computation of structure-oriented semblance, an attribute commonly used to highlight discontinuities in seismic images. Semblance is defined in this paper as simply the ratio of a squared smoothed-image to a smoothed squared-image. This definition of semblance generalizes that commonly used today, because an unlimited variety of smoothing filters can be used to compute the numerator and denominator images in the semblance ratio. The smoothing filters described in this paper yield an especially flexible method for computing structure-oriented semblance.

Key words: seismic image smoothing semblance

1 INTRODUCTION

Images like those displayed in Figure 1 are familiar in the context of exploration seismology, where spatial sampling is sufficiently uniform to enable the application of a variety of generic image-processing techniques. For example, coherency-enhancing anisotropic diffusion filters, as described by Weickert (1997, 1999), have been adapted by Fehmers and Höcker (2003) for structure-oriented filtering of seismic images to enhance their interpretation. Figure 1b illustrates a similar *structure-oriented smoothing* process that smooths along coherent reflections while preserving faults.

Faults apparent in Figure 1a are highlighted in Figure 1c using a process that (in the sense of Fehmers and Höcker, 2003) I call *structure-oriented semblance*. As shown here, semblance is a normalized measure of coherence with values between zero and one, where zero corresponds to no coherence. I used the semblance image in structure-oriented filtering to inhibit smoothing across the faults in Figure 1b. However, semblance images like that in Figure 1c are often used directly in seismic interpretation to construct geologic models of faults and stratigrathic features such as channels (e.g., Bahorich and Farmer, 1995; Marfurt et al., 1998).

The purpose of this paper is to describe new methods for computing smoothed and semblance images like

those in Figure 1. The proposed smoothing filter is both simple to implement and computationally efficient, and leads naturally to a semblance filter that is more generally useful than methods commonly used today to compute semblance.

1.1 Structures and orientations

Coefficients of structure-oriented smoothing and semblance filters depend on the orientations of coherent structures in images. The required orientations can be estimated by scanning over a set of sampled orientations and choosing those for which semblance or some comparable attribute is maximized (e.g., Marfurt et al., 1998; Marfurt, 2006). Orientation scanning can provide simultaneously both semblance and the orientations required for structure-oriented smoothing.

However, searching for the optimal orientation can be costly, particularly when we do not know before scanning whether imaged structures have locally linear, planar, or other shapes. For example, buried channels tend to appear as curvilinear features in seismic images, whereas geologic horizons appear curviplanar (well approximated by curved surfaces). We should expect both structure-oriented smoothing and semblance to honor the varying dimensionalities of structures apparent in seismic images, and scanning for both orientation and dimensionality is computationally costly. The cost is especially high for 3D seismic images.

1.2 Structure tensors

An alternative to scanning over orientations of image features that could be linear or planar (or ...) is to compute *structure tensors* (van Vliet and Verbeek, 1995; Weickert, 1997; Fehmers and Höcker, 2003). Structure tensors are simply smoothed outer products of image gradients.

In all of the examples shown in this paper, I approximated image gradients and smoothed the products of the components of those gradients using Gaussian derivative and smoothing filters, respectively (Deriche, 1992; van Vliet et al., 1998; Hale, 2006). Specifically, in computing structure tensors for the 2D image of Figure 1a, I used Gaussian derivative and smoothing filters with radii $\sigma = 1$ and $\sigma = 8$, respectively.

In this way we may construct an entire field of structure tensors that typically vary from sample to sample in an image. Let $\mathbf{T}$ denote the structure tensor corresponding to one image sample. For a 2D image like that shown in Figure 1a, each structure tensor $\mathbf{T}$ is a 2×2 symmetric positive-semidefinite matrix

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} \\ t_{12} & t_{22}. \end{bmatrix}. \tag{1}$$

As shown by Fehmers and Höcker (2003), the eigendecomposition of each structure tensor $\mathbf{T}$ provides the measures of orientation and dimensionality that we require in structure-oriented filtering. For 2D images, this eigen-decomposition is

$$\mathbf{T} = \lambda_u \mathbf{u}\mathbf{u}^T + \lambda_v \mathbf{v}\mathbf{v}^T, \tag{2}$$

where λ_u and λ_v are the eigenvalues and $\mathbf{u}$ and $\mathbf{v}$ the corresponding eigenvectors of $\mathbf{T}$.

By convention we label the eigenvalues and eigenvectors of $\mathbf{T}$ so that $\lambda_u \geq \lambda_v \geq 0$. This convention implies that eigenvectors $\mathbf{u}$ indicate directions in which image gradients are highest and will therefore be orthogonal to linear features. The eigenvectors $\mathbf{v}$, which correspond to the smaller eigenvalues λ_v, will be parallel to such features.

The eigenvalues λ_u and λ_v of each structure tensor $\mathbf{T}$ provide measures of isotropy and linearity. Specifically, we may compute for each image sample the nonnegative ratios

$$\begin{aligned} &\textit{isotropy:} \quad \lambda_0 = \lambda_v/\lambda_u \\ &\textit{linearity:} \quad \lambda_1 = (\lambda_u - \lambda_v)/\lambda_u. \end{aligned} \tag{3}$$

defined here such that $\lambda_0 + \lambda_1 = 1$.

Figure 2 illustrates eigenvectors $\mathbf{v}$ for a small subset of the structure tensors computed for every sample in two different 2D images. The lengths of these vectors have been scaled by linearity λ_1. The eigenvectors $\mathbf{v}$ are well aligned with image structures and appear longest where those structures are most coherent and linear. Elsewhere, as in the lower right corner of Figure 2b, image features are more isotropic.

2 STRUCTURE-ORIENTED SMOOTHING

Eigenvectors derived from a structure-tensor field enable us to design filters that smooth along linear or planar features, but that do not smooth across them. We may choose from among a wide variety of filters.

2.1 Slope-based smoothing

For example, to smooth along the features in Figure 2a, we could use Fomel's plane-wave destruction filters (Fomel, 2002). (For smoothing we would simply subtract from the input image the output of a plane-wave destruction filter.) Coefficients of these filters depend on reflection slopes, which are simply ratios $p = v_1/v_2$ of components of the eigenvectors $\mathbf{v}$. Figure 2a suggests that these slopes are typically less than 1, because coherent features in Figure 2a tend to be more horizontal than vertical.

However, slopes $p = v_1/v_2$ can be infinite, as implied by the eigenvectors illustrated in Figure 2b, which contains both horizontal and vertical features. The words "horizontal" and "vertical" here refer only to this display of what is in fact a truly horizontal slice from a 3D seismic image. Still, a filter parameterized

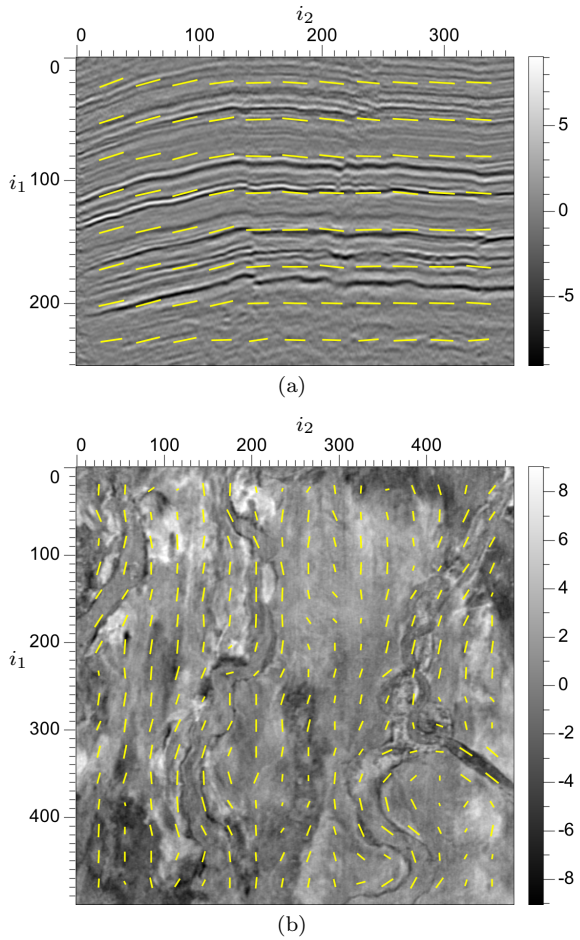


Figure 2. Eigenvectors $\mathbf{v}$ for 2D slices of two different 3D seismic images of faulted geologic layers (a) and buried channels (b).

by reflection slope cannot be used to smooth along the features apparent in Figure 2b.

2.2 Smoothing with anisotropic diffusion

A more generally useful alternative proposed by Weickert (1997, 1999) and Fehmers and Höcker (2003) (and others) is to parameterize a structure-oriented smoothing filter by a tensor field. For an input image $f(\mathbf{x})$ and a corresponding structure-tensor field $\mathbf{T}(\mathbf{x})$, these authors propose the solution of an anisotropic diffusion equation

$$\frac{\partial g(\mathbf{x};\tau)}{\partial \tau} = \boldsymbol{\nabla} \cdot \mathbf{D}(\mathbf{x}) \, \boldsymbol{\nabla} \, g(\mathbf{x};\tau) \tag{4}$$

with initial condition $g(\mathbf{x};\tau=0) = f(\mathbf{x})$. For any time $\tau > 0$, the solution $g(\mathbf{x};\tau)$ is a smoothed version of the input image $f(\mathbf{x})$.

This anisotropic diffusion equation is parameterized by a diffusion-tensor field $\mathbf{D}(\mathbf{x})$ that shares the eigenvectors of the structure-tensor field $\mathbf{T}(\mathbf{x})$. To smooth along the eigenvectors $\mathbf{v}(\mathbf{x})$ illustrated in Figure 2, we might set $\lambda_u(\mathbf{x}) = 0$ and $\lambda_v(\mathbf{x}) = 1$ for all tensors in the field $\mathbf{D}(\mathbf{x})$. More generally, we might let the eigenvalues of $\mathbf{D}(\mathbf{x})$ depend on the measures of isotropy and linearity computed from $\mathbf{T}(\mathbf{x})$ according to equations 3.

Unfortunately, as noted by Fehmers and Höcker (2003), straightforward explicit and stable numerical solutions to the anisotropic diffusion equation 4 may require a prohibitively large number of time steps. So they instead use an unspecified solution method that they liken to a "rudimentary multigrid implementation".

2.3 An anisotropic smoothing filter

When considering the efficiency of structure-oriented smoothing with anistropic diffusion, Fehmers and Höcker (2003) suggest that it is unnecessary to model the diffusion process exactly. Consistent with their suggestion, I propose here the numerical solution of a different partial differential equation

$$g(\mathbf{x}) - \alpha \, \boldsymbol{\nabla} \cdot \mathbf{D}(\mathbf{x}) \, \boldsymbol{\nabla} \, g(\mathbf{x}) = f(\mathbf{x}). \tag{5}$$

Here, $f(\mathbf{x})$ represents the input image and $g(\mathbf{x})$ represents the output smoothed image. The constant filter parameter α could be absorbed into the smoothing-tensor field $\mathbf{D}(\mathbf{x})$, but is kept separate to provide a convenient control for the extent of smoothing. For $\alpha = 0$, we have $g(\mathbf{x}) = f(\mathbf{x})$, and no smoothing is performed.

Unlike the solution of the diffusion equation 4, the solution to the smoothing equation 5 does not depend on time τ. However, smoothing with equation 5 does require the numerical solution of a large sparse system of equations. (Numerical solution of equation 5 is equivalent to a single but possibly large time step in an unconditionally stable implicit numerical solution of the diffusion equation 4.) Fortunately, simple finite-difference approximations suffice to obtain a sparse symmetric positive-semidefinite system of equations that we may solve efficiently using conjugate-gradient iterations.

I use the bilinear transform (e.g., Oppenheim et al., 1999) to approximate the partial derivatives in equation 5. As a simple example, consider the 1D version of equation 5 with constant coefficients

$$g(x) - \alpha \, g''(x) = f(x). \tag{6}$$

Using z-transform notation, the bilinear transform of a derivative is the substitution

$$g'(x) \Rightarrow 2\frac{1-z^{-1}}{1+z^{-1}} \, G(z) \tag{7}$$

or, equivalently,

$$-g'(x) \Rightarrow 2\frac{1-z}{1+z} \, G(z). \tag{8}$$

With this approximation, the z-transform of equation 6

becomes

$$\left(z^{-1}+2+z\right)G(z)-4\alpha\left(z^{-1}-2+z\right)G(z) = \left(z^{-1}+2+z\right)F(z), \quad (9)$$

which corresponds to the finite-difference approximation

$$(1-4\alpha)\,g[i-1]+(2+8\alpha)\,g[i]+(1-4\alpha)\,g[i+1] = f[i-1]+2f[i]+f[i+1]. \quad (10)$$

Given N input samples $f[i]$, $i=0,1,\ldots,N-1$, and suitable (e.g., zero-slope) boundary conditions, we can easily solve this tri-diagonal system of N equations for N smoothed output samples $g[i]$.

While other finite-difference approximations to the smoothing equation 6 may seem more straightforward, I chose the bilinear transform because the smoothing filter implied by equation 10 has a zero at the Nyquist frequency for all $\alpha > 0$. This smoothing filter is in fact a cascade of two (one causal and one anti-causal) 1st-order Butterworth filters (e.g., Oppenheim et al., 1999).

The same bilinear transform works as well for each partial derivative in 2D and 3D versions of equation 5 with variable tensor coefficients $\mathbf{D}$. The derivation is somewhat more tedious and the resulting system of equations is not tri-diagonal as in the 1D version. However, as noted above, the system of equations remains sparse, symmetric and positive-semidefinite, and may be solved efficiently with conjugate-gradient iterations.

Let $\mathbf{f}$ denote a vector containing all samples $f[i_1,i_2,\ldots,i_n]$ of an n-dimensional input image $f(\mathbf{x})$, and let $\mathbf{g}$ denote a corresponding vector of samples $g[i_1,i_2,\ldots,i_n]$ of the smoothed output image $g(\mathbf{x})$. After bilinear transform of equation 5, the sparse system of equations to be solved has the form

$$(\mathbf{B}^T\mathbf{B}+\mathbf{A}^T\mathbf{D}\mathbf{A})\mathbf{g} = \mathbf{B}^T\mathbf{B}\mathbf{f} \quad (11)$$

In this equation $\mathbf{D}$ now represents a sparse matrix with non-zero elements corresponding to coefficients in the smoothing-tensor field $\mathbf{D}(\mathbf{x})$ described above. Sparse matrices $\mathbf{A}$ and $\mathbf{B}$ correspond to finite-difference approximations obtained with the bilinear transform.

In each conjugate-gradient iteration, we must compute the matrix-vector product $\mathbf{s}=(\mathbf{B}^T\mathbf{B}+\mathbf{A}^T\mathbf{D}\mathbf{A})\mathbf{r}$ for some temporary vectors $\mathbf{r}$ and $\mathbf{s}$. Here is a small computer program (written in C, C++ or Java) that does this for 2D image smoothing, where the vectors $\mathbf{r}$ and $\mathbf{s}$ represent values $r[i_1,i_2]$ and $s[i_1,i_2]$ stored in 2D arrays.

```
// Zero all elements of the array s; then
for (i2=1; i2<n2; ++i2) {
for (i1=1; i1<n1; ++i1) {
  e11 = alpha*d11[i2][i1]; // smoothing
  e12 = alpha*d12[i2][i1]; // tensor
  e22 = alpha*d22[i2][i1]; // coefficients
  r00 = r[i2  ][i1  ];
  r01 = r[i2  ][i1-1];
  r10 = r[i2-1][i1  ];
  r11 = r[i2-1][i1-1];
  rs = 0.25f*(r00+r01+r10+r11); // for B'B
  ra = r00-r11;
  rb = r01-r10;
  r1 = ra-rb; // two components of
  r2 = ra+rb; // the gradient of r
  s1 = e11*r1+e12*r2; // multiplied by the
  s2 = e12*r1+e22*r2; // smoothing tensor
  sa = s1+s2;
  sb = s1-s2;
  s[i2  ][i1  ] += sa+rs;
  s[i2  ][i1-1] -= sb-rs;
  s[i2-1][i1  ] += sb+rs;
  s[i2-1][i1-1] -= sa-rs;
}}
```

For 2D images, this program is the simplest and most efficient way to ensure a symmetric positive-definite implementation of the matrix operator $\mathbf{B}^T\mathbf{B}+\mathbf{A}^T\mathbf{D}\mathbf{A}$ on the left-hand side of equation 11. (Note that this program does not require an explicit representation of that left-hand-side matrix. A similar program for smoothing 3D images is provided in Appendex A.) An even simpler program can be used to apply the right-hand side matrix operator $\mathbf{B}^T\mathbf{B}$ once to the input image vector $\mathbf{f}$ before beginning conjugate-gradient iterations.

Figure 3 illustrates anisotropic smoothing with constant parameter $\alpha=18$ and smoothing tensors $\mathbf{D}=\mathbf{v}\mathbf{v}^T$. The eigenvalues of these smoothing tensors are constant: $\lambda_u=0$ and $\lambda_v=1$. Therefore, unlike the smoothing shown in Figure 1b, that shown here in Figure 3 does not preserve discontinuities across faults.

I used 27 and 33 conjugate-gradient iterations to compute the smoothed images shown in Figures 3a and 3b, respectively. When I changed $\alpha=18$ to $\alpha=5$, the number of iterations decreased by roughly a factor of two, to 13 and 16 iterations, respectively. These results and further experiments suggest that the number of iterations required is independent of image dimensions and is roughly proportional to $\sqrt{\alpha}$.

The smoothing filters implied by equation 11 do not correspond to any weighted averaging of image pixels along linear trajectories like the line segments in Figures 2.

To illustrate this point, Figure 4 shows the weights implicitly applied to input samples $f[i_1,i_2]$ when computing a small subset of output samples $g[i_1,i_2]$ like those shown in Figure 3b, but here for $\alpha=70$ to make the weights more visible. These smoothing-filter weights are largest along *curvilinear* trajectories that would be difficult and costly to construct explicitly. That construction is unnecessary, because we may instead simply solve the anisotropic smoothing equation 11.

3 STRUCTURE-ORIENTED SEMBLANCE

We can use smoothing filter weights like those shown in Figure 4 to define a locally weighted semblance. The key

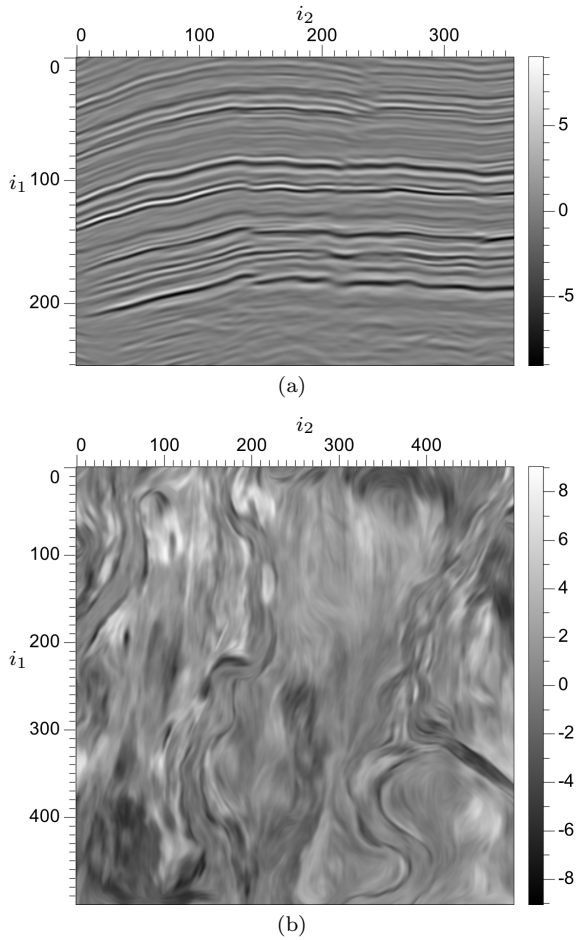


Figure 3. Anisotropic smoothing (for $\alpha = 18$) along the eigenvectors $\mathbf{v}$ of Figure 2 for two seismic images.

step is to recognize that sums in conventional definitions of semblance perform a sort of smoothing. We can then replace those sums with the weighted sums of smoothing filters.

3.1 Conventional semblance

Semblance was first defined by Taner and Koehler (1969) and developed further by Neidell and Taner (1971) in the context of velocity spectra, where they computed semblance as a function of time for CMP gathers after normal moveout correction. An equivalent definition that is consistent with the notation used elsewhere in this paper is

$$s[i_1] = \frac{\displaystyle\sum_{j_1=i_1-M_1}^{i_1+M_1} \left(\sum_{j_2=0}^{N_2-1} f[j_1,j_2] \right)^2}{N_2 \displaystyle\sum_{j_1=i_1-M_1}^{i_1+M_1} \sum_{j_2=0}^{N_2-1} \left(f[j_1,j_2]\right)^2}. \qquad (12)$$

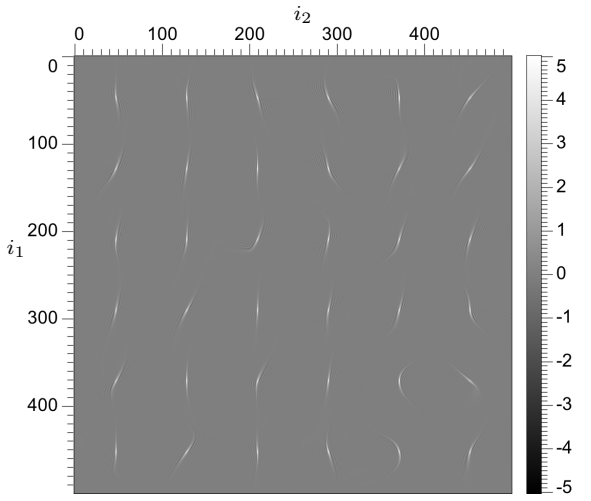


Figure 4. Anisotropic smoothing-filter weights for $\alpha = 70$. These weights conform to the eigenvector field $\mathbf{v}$ of Figure 2b, and imply curvilinear smoothing paths. Weights shown here are scaled by a factor of 100 for display.

For all indices i_1, semblance $s[i_1]$ is maximized and equals one when the values $f[j_1, j_2]$ do not vary with the index j_2. For Taner and Koehler's velocity spectra, the index j_2 corresponds to a trace number and the index j_1 corresponds to a time sample. The inner sums over j_2 correspond to N_2 NMO-corrected traces in a CMP gather. The outer sums over j_1 correspond to a window of $2M_1 + 1$ time samples centered at the sample i_1 for which semblance $s[i_1]$ is to be computed.

Marfurt et al. (1998) used a similar definition of semblance as the ratio of sums like these to compute local measures of coherence. Instead of scanning over different NMO velocities to maximize semblance, they instead scanned over different planar reflection slopes in local windows of seismic traces. A simplified version (omitting Hilbert tranforms) of their semblance measure for 2D images would be

$$s[i_1,i_2] = \frac{\displaystyle\sum_{j_1=i_1-M_1}^{i_1+M_1} \left(\sum_{j_2=i_2-M_2}^{i_2+M_2} f[j_1,j_2] \right)^2}{(2M_2+1) \displaystyle\sum_{j_1=i_1-M_1}^{i_1+M_1} \sum_{j_2=i_2-M_2}^{i_2+M_2} \left(f[j_1,j_2]\right)^2}. \qquad (13)$$

In this definition, $f[j_1, j_2]$ corresponds to a window of $(2M_1 + 1) \times (2M_2 + 1)$ image samples after shifting in time to flatten planar reflections for a specified slope. Semblance is maximized if that specified slope matches the slope of a planar reflection in the image.

Semblance computed by equation 13 is stored at indices i_1 and i_2 that correspond to the centers of the summation windows. Therefore, the sums in the numerator and denominator of this equation work much like symmetric local smoothing filters. However, these smoothing filters have constant weights, implying that

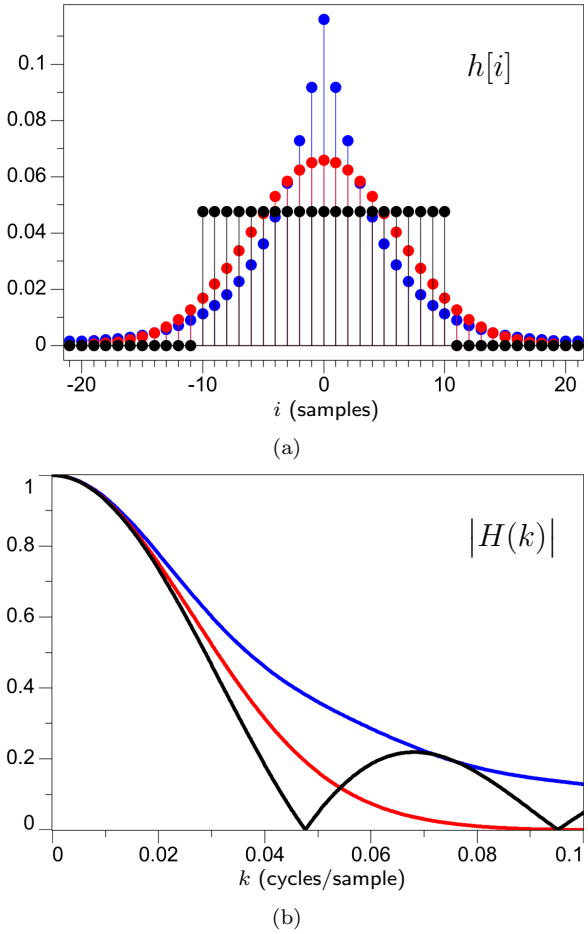


Figure 5. Three weighting sequences $h[i]$ (a) and their Fourier amplitude spectra $|H(k)|$ (b). Weights shown are boxcar (black), Gaussian (red), and smoothed exponential (blue).

all samples inside the windows are equally significant, but that samples just outside these windows are worthless. Intuitively, this sort of weighting makes no sense. In practice, it leads to visible artifacts where strong image features enter and leave such windows.

3.2 Weighting sequences

Better windows would give more weight to samples near the centers of windows and less weight to those near window edges.

Figure 5 illustrates three different weighting sequences $h[i]$ with their Fourier amplitude spectra $|H(k)|$. For zero frequency, $k = 0$, all three weighting sequences have the same Fourier amplitude $|H(0)|$ and curvature $|H''(0)|$. Therefore, in their responses to low frequencies, these weighting sequences are comparable, even though their shapes differ significantly.

The boxcar sequence $h[i]$ corresponds to the conventional definition of semblance computed for constant-weighted samples in a window with finite duration. The boxcar weighting sequence illustrated in Figure 5a has half-width $M = 10$ samples.

The Gaussian sequence $h[i]$ is proportional to $\exp\left(-i^2/2\sigma^2\right)$. This sequence is smoothest and is nowhere zero, but in practice may be truncated where weights are insignificant. The Gaussian sequence in Figure 5a has half-width $\sigma = \sqrt{M(M+1)/3}$, and the choice $M = 10$ makes it comparable to the boxcar sequence shown there.

The smoothed exponential sequence $h[i]$ is likewise nowhere zero, and is the impulse response of the filter implied by solution of equation 10 for smoothing parameter $\alpha = M(M+1)/6$. Again, the choice $M = 10$ makes this weight sequence comparable to the boxcar sequence in Figure 5a.

3.3 Weighted semblance

We may use weighting sequences like those shown in Figure 5a to generalize the conventional definition of semblance to what I call *weighted semblance*.

Let us first consider a single weighted-semblance value s computed for a 1D sequence $f[j]$:

$$s = \frac{\left(\sum_j h[j]f[j]\right)^2}{\sum_j h[j]\left(f[j]\right)^2}. \tag{14}$$

Here, unspecified limits for sums are assumed to include all indices j for which both $f[j]$ and $h[j]$ are defined.

As for conventional semblance, we want weighted semblance s to be a normalized measure of coherence such that $0 \le s \le 1$. We may obtain such a measure for any set of weights $h[j]$ that have two properties:

$$\sum_j h[j] = 1 \quad \text{and} \quad h[j] \ge 0 \text{ for all } j. \tag{15}$$

In other words, the weights $h[j]$ must be normalized and non-negative. All three weighting sequences displayed in Figure 5 have these two properties.

The proof that $0 \le s \le 1$ is simple. Because all weights $h[j]$ are non-negative, the denominator in equation 14 must be non-negative as well. Since the numerator is non-negative for any weights, the semblance ratio is bounded below by $0 \le s$.

To see that semblance is bounded above by $s \le 1$, let us rewrite equation 14 as

$$s = \frac{\left[\sum_j \left(\sqrt{h[j]}\right)\left(\sqrt{h[j]}f[j]\right)\right]^2}{\left[\sum_j \left(\sqrt{h[j]}\right)^2\right]\left[\sum_j \left(\sqrt{h[j]}\right)f[j]\right)^2\right]}. \tag{16}$$

The square roots are real valued because all weights $h[i]$

are non-negative. The leftmost term in the denominator is unity because it equals the sum of those weights. Then, by the Cauchy-Schwarz inequality, we have $s \le 1$.

Equation 16 is written carefully to show that weighted semblance is equivalent to a normalized correlation coefficient for two sequences $\sqrt{h[j]}$ and $\sqrt{h[j]}f[j]$. This coefficient is unity if the sequence $f[j]$ is constant. When this sequence is not constant, more weight is given to the values $f[j]$ for which the weights $h[j]$ are largest.

To compute weighted semblance in sliding and seamlessly overlapping windows of the sequence $f[j]$, we simply write

$$s[i] = \frac{\Big(\sum\limits_j h[i-j]f[j]\Big)^2}{\sum\limits_j h[i-j]\big(f[j]\big)^2}. \tag{17}$$

Both numerator and denominator in this ratio include convolution with a smoothing filter like those shown in Figure 5a. In this sense, *semblance is the ratio of a squared smoothed-sequence to a smoothed squared-sequence.*

However, the smoothing filter need not be shift invariant. An even more general form of equation 17 is

$$s[i] = \frac{\Big(\sum\limits_j h[i;j]f[j]\Big)^2}{\sum\limits_j h[i;j]\big(f[j]\big)^2}. \tag{18}$$

In other words, smoothing-filter coefficients $h[i;j]$ may vary with index i, provided that the properties in equations 15 are satisfied for all i.

Before extending the notion of weighted semblance to 2D and 3D images, it will help to simplify notation further by letting $\langle\cdot\rangle$ denote smoothing of whatever is inside the angle brackets, so that semblance becomes simply

$$s = \frac{\langle f\rangle^2}{\langle f^2\rangle}. \tag{19}$$

3.4 2D structure-oriented semblance

For 2D images, we have the opportunity to include a second smoothing, as in the conventional definition of semblance in equation 13. Using the concise notation described above, we may rewrite this equation as

$$s_{2,1} = \frac{\langle\, \langle f\rangle_2^2\, \rangle_1}{\langle\, \langle f^2\rangle_2\, \rangle_1}, \tag{20}$$

where $\langle\cdot\rangle_1$ denotes smoothing along the 1st image axis, and $\langle\cdot\rangle_2$ denotes smoothing along the 2nd axis. The outer smoothing $\langle\cdot\rangle_1$ helps to stabilize semblance values where the inner smoothing $\langle\cdot\rangle_2$ accumulates only very small and perhaps noisy values. Depending on expected orientations of image features, we might switch the 1st and 2nd smoothing directions.

For structure-oriented semblance, I simply replace axis-aligned smoothing with structure-oriented smoothing. When computing semblance for 2D images, we may define the smoothing filters using eigenvectors $\mathbf{u}$ and $\mathbf{v}$ computed from structure tensors. Structure-oriented semblance is then

$$s_{v,u} = \frac{\langle\, \langle f\rangle_v^2\, \rangle_u}{\langle\, \langle f^2\rangle_v\, \rangle_u}. \tag{21}$$

The inner smoothing $\langle\cdot\rangle_v$ is along image features, and the outer smoothing $\langle\cdot\rangle_u$ is across those features.

I computed the structure-oriented semblance shown in Figure 1c using structure-oriented smoothing filters with $M = 4$ ($\alpha = M(M+1)/6$) for inner smoothing $\langle\cdot\rangle_v$ along image features and $M = 16$ for outer smoothing $\langle\cdot\rangle_u$ across those features. As expected, semblance is low near faults and near the bottom where the image f is less coherent.

The structure-oriented smoothing filters used to compute semblance in this and other examples shown in this paper do not strictly satisfy the second requirement of equation 15 that all weights be non-negative. Recall the smoothing-filter weights shown in Figure 4, which are mostly positive, but on close inspection exhibit negative values. Non-negative weights are easy to obtain for 1D smoothing; e.g., the smoothed exponential sequence in Figure 5a. However, I have been unable to guarantee non-negative weights in useful finite-difference approximations of the more general anisotropic smoothing equation 5.

Nevertheless, these smoothing filters yield a useful semblance measure. I simply clip the values of $s_{v,u}$ so that values less than zero are replaced with zero and values greater than one are replaced with one. In my experience, computing structure-oriented semblance for both 2D and 3D images, this clipping occurs rarely. In the example shown in Figure 1c, no such clipping was necessary.

Figure 6 shows the same semblance image again for comparison with a more conventional slope-based semblance image. I computed this slope-based semblance for a sliding window of nine traces ($M = 4$). Within each such window, I used sinc interpolation of each trace to flatten the nine-trace image before computing the inner horizontal sums with constant (boxcar) weights. I then computed the outer vertical sums by applying vertical Gaussian smoothing for $M = 16$ ($\sigma = \sqrt{M(M+1)/3}$), before finally computing the semblance ratios. Although smoothing filters varied, this example of slope-based semblance is comparable to that of structure-oriented semblance because I chose consistent half-widths $M = 4$ and $M = 16$ in both examples.

A notable difference between the two semblance images in Figure 6 is the appearance of faults as piecewise vertical features in the slope-based semblance image. This vertical bias is caused by the outer Gaussian filtering that in the conventional slope-based method smooths semblance numerators and denominators only

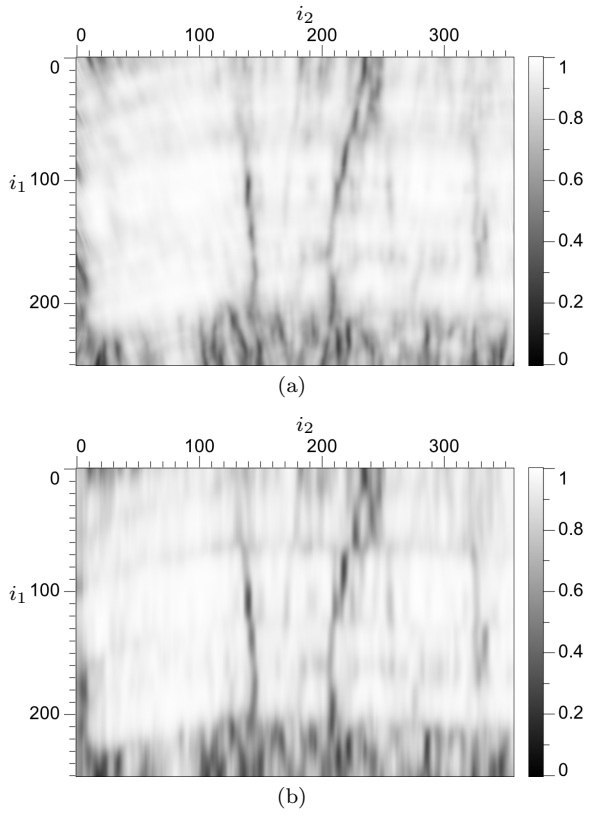


Figure 6. Structure-oriented semblance $s_{v,w}$ (a) computed via structure-oriented smoothing and a more conventional slope-based semblance (b) for the image of Figure 2a.

vertically. This bias would be even more apparent had I used boxcar smoothing instead of Gaussian smoothing.

A second disadvantage of a slope-based semblance method is that (like slope-based smoothing) it cannot be readily applied to images having features with both horizontal and vertical orientations, like those illustrated in Figure 2b. In contrast, structure-oriented semblance is easy to implement for arbitrary orientations of the structure eigenvectors $\mathbf{u}$ and $\mathbf{v}$.

Figure 7 shows an example of structure-oriented semblance $s_{v,u}$ computed for the image of channels using structure-oriented $\langle\cdot\rangle_v$ (inner) and $\langle\cdot\rangle_u$ (outer) smoothings, both with $M = 4$. Recall that the image in Figure 7a is actually a horizontal 2D slice extracted from a 3D image. The semblance image in Figure 7b therefore contains numerous spurious features that occur where linear features cross contours of zero amplitude in the 2D slice of Figure 7a. Because the numerator of the semblance ratio in equation 21 contains a local weighted average of f, semblance is low near all such zero crossings. To obtain a more meaningful semblance image, we must process the 3D image.

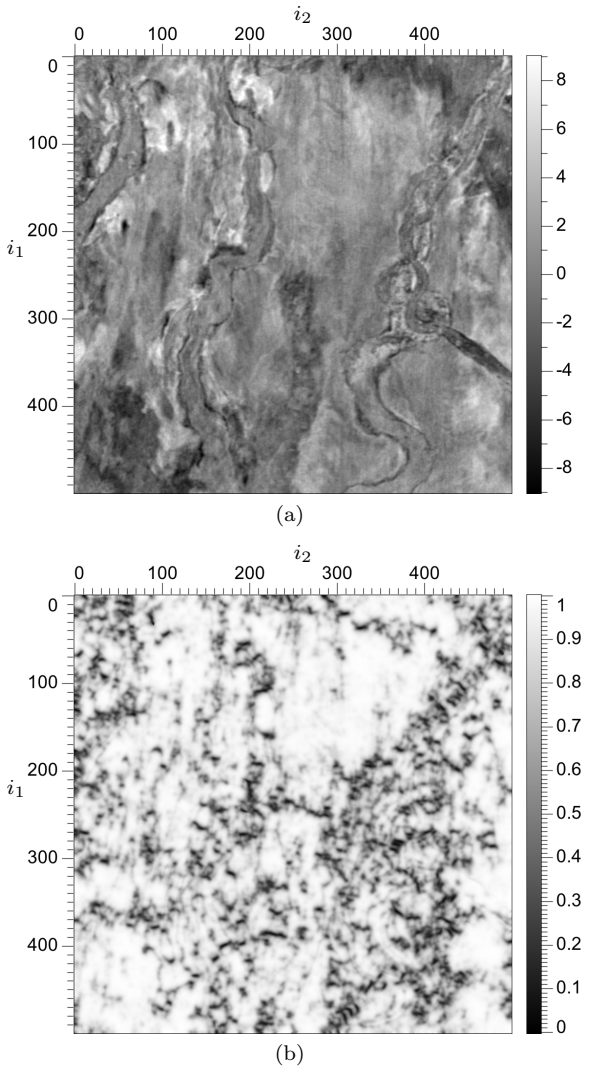


Figure 7. Features in a 2D horizontal slice (a) of a 3D image of channels have varying orientations. Structure-oriented semblance $s_{v,u}$ (b) highlights amplitude variations along the linear trends of these features.

3.5 3D structure-oriented semblance

To compute structure-oriented semblance for 3D images, I use 3D structure-oriented smoothing. These smoothing filters can again be parameterized by smoothing tensors derived from structure tensors. Each structure tensor is a 3×3 matrix with eigendecomposition

$$\mathbf{T} = \lambda_u \mathbf{u}\mathbf{u}^T + \lambda_v \mathbf{v}\mathbf{v}^T + \lambda_w \mathbf{w}\mathbf{w}^T, \tag{22}$$

where λ_u, λ_v and λ_w are the eigenvalues and $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ the corresponding eigenvectors of $\mathbf{T}$.

I again label the eigenvalues and eigenvectors of $\mathbf{T}$ so that $\lambda_u \geq \lambda_v \geq \lambda_w \geq 0$. Eigenvectors $\mathbf{u}$ again indicate directions in which image gradients are highest,

orthogonal to linear or planar features. The eigenvectors $\mathbf{w}$, which correspond to the smallest eigenvalues λ_w, will be aligned with linear features, such as the channels in Figure 7a. Both eigenvectors $\mathbf{v}$ and $\mathbf{w}$ will lie within the planes of any planar features.

Structure-oriented semblance measured within local planes defined by eigenvectors $\mathbf{v}$ and $\mathbf{w}$ is then simply

$$s_{vw,u} = \frac{\langle\, \langle f \rangle_{vw}^2 \,\rangle_u}{\langle\, \langle f^2 \rangle_{vw} \,\rangle_u}. \tag{23}$$

This equation defines a *planar semblance*. For the inner curviplanar smoothing, we use smoothing tensors $\mathbf{D} = \mathbf{v}\mathbf{v}^T + \mathbf{w}\mathbf{w}^T$. For the outer orthogonal curvilinear smoothing, we use $\mathbf{D} = \mathbf{u}\mathbf{u}^T$. Unlike slope-based semblance, planar semblance remains well defined for steeply dipping, even vertical, features in 3D seismic images.

Choosing the eigenvalues of smoothing tensors $\mathbf{D}$ in a different way, we can likewise define a *linear semblance*

$$s_{w,uv} = \frac{\langle\, \langle f \rangle_{w}^2 \,\rangle_{uv}}{\langle\, \langle f^2 \rangle_{w} \,\rangle_{uv}}. \tag{24}$$

As its name implies, this linear semblance measures coherence along curvilinear paths within an image.

Planar and linear semblance are two extremes in a continuum of semblance measures we may define by choosing the eigenvalues of smoothing tensors $\mathbf{D}$ in different ways. We may, for example, choose the eigenvalues of $\mathbf{D}$ to be functions of the eigenvalues of the corresponding structure tensors $\mathbf{T}$, perhaps using the following measures of isotropy, linearity and planarity:

$$\begin{aligned} \textit{isotropy:} \quad & \lambda_0 = \lambda_w/\lambda_u \\ \textit{linearity:} \quad & \lambda_1 = (\lambda_v - \lambda_w)/\lambda_u \\ \textit{planarity:} \quad & \lambda_2 = (\lambda_u - \lambda_v)/\lambda_u, \end{aligned} \tag{25}$$

defined here such that $\lambda_0 + \lambda_1 + \lambda_2 = 1$. In any case, the outer smoothing we perform in the semblance calculation is an orthogonal complement to the inner smoothing.

Figure 8 shows examples of both planar and linear semblance. All smoothing filters in these examples have half-width $M = 2$ samples. (Shorter filters are more suitable for 3D images than for 2D images because of the extra dimension in which smoothing can be performed.) Planar semblance highlights (with low values) all features that are not planar, such as the channels, which are linear. Linear semblance highlights variations within those channels, features that are neither linear nor planar, but may be significant.

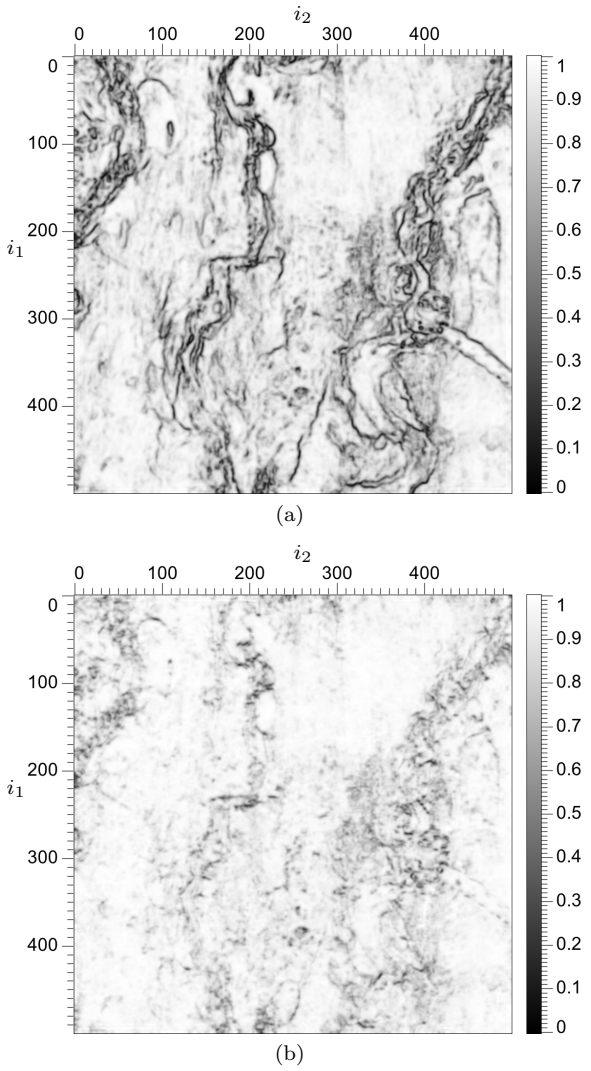


Figure 8. 3D structure-oriented planar semblance $s_{vw,u}$ (a) and linear semblance $s_{w,uv}$ (b) computed for a 3D seismic image of buried channels. Shown here are horizontal 2D slices (coincident with the 2D slices shown in Figure 7) extracted from 3D semblance images.

4 CONCLUSION

Structure-oriented smoothing filters as described in this paper are quite general, with parameters derived mostly from structure-tensor fields. In contrast to smoothing filters parameterized by slopes of image features, this generality enables smoothing of 2D and 3D images with arbitrary orientations and dimensionalities.

Structure-oriented smoothing filters are also simple to implement with small computer programs. The most significant part of the implementation for 2D filters (not including a necessary but readily available conjugate-gradient solver) is a computer program with only 23 lines. A similar program for 3D structure-oriented smoothing consists of only 42 lines.

From structure-oriented smoothing we may define structure-oriented semblance as the ratio of a squared smoothed-image to a smoothed squared-image.

Structure-oriented semblance is a special case of weighted semblance, in which weighted sums are implied by structured-oriented filters that smooth either along or across image features. The orientation of those features is arbitrary; e.g., structure-oriented semblance has no vertical bias. And unlike the weights implied by conventional semblance computed in sliding boxcar windows, the weights used in structure-oriented semblance smoothly and seamlessly decay to zero.

ACKNOWLEDGMENTS

Thanks to WesternGeco for providing the seismic image of buried channels, and to the U.S. Department of Energy for providing the seismic image of geologic layers. Thanks also to Paul Sava, Ran Xuan and Luming Liang for helpful reviews of this paper.

APPENDIX A — 3D SMOOTHING

For 3-D structure-oriented smoothing via solution of equation 11, the following computer program (written in C, C++ or Java) computes the product $\mathbf{s} = (\mathbf{B}^T\mathbf{B} + \mathbf{A}^T\mathbf{D}\mathbf{A})\mathbf{r}$ for vectors $\mathbf{r}$ and $\mathbf{s}$ that represent values $r[i_1, i_2, i_3]$ and $s[i_1, i_2, i_3]$ stored in 3D arrays.

```
// Zero all elements of the array s; then
for (i3=1; i3<n3; ++i3) {
for (i2=1; i2<n2; ++i2) {
for (i1=1; i1<n1; ++i1) {
  e11 = alpha*d11[i3][i2][i1]; // smoothing
  e12 = alpha*d12[i3][i2][i1]; // tensor
  e13 = alpha*d13[i3][i2][i1]; // coefficients
  e22 = alpha*d22[i3][i2][i1];
  e23 = alpha*d23[i3][i2][i1];
  e33 = alpha*d33[i3][i2][i1];
  r000 = r[i3  ][i2  ][i1  ];
  r001 = r[i3  ][i2  ][i1-1];
  r010 = r[i3  ][i2-1][i1  ];
  r011 = r[i3  ][i2-1][i1-1];
  r100 = r[i3-1][i2  ][i1  ];
  r101 = r[i3-1][i2  ][i1-1];
  r110 = r[i3-1][i2-1][i1  ];
  r111 = r[i3-1][i2-1][i1-1];
  rs = 0.25f*(r000+r001+r010+r011+
              r100+r101+r110+r111); // for B'B
  ra = r000-r111;
  rb = r001-r110;
  rc = r010-r101;
  rd = r100-r011;
  r1 = ra-rb+rc+rd; // three
  r2 = ra+rb-rc+rd; // components of
  r3 = ra+rb+rc-rd; // gradient of r
  s1 = e11*r1+e12*r2+e13*r3; // multiplied by
  s2 = e12*r1+e22*r2+e23*r3; // the smoothing
  s3 = e13*r1+e23*r2+e33*r3; // tensor
  sa = s1+s2+s3;
  sb = s1-s2+s3;
  sc = s1+s2-s3;
  sd = s1-s2-s3;
  s[i3  ][i2  ][i1  ] += sa+rs;
  s[i3  ][i2  ][i1-1] -= sd-rs;
  s[i3  ][i2-1][i1  ] += sb+rs;
  s[i3  ][i2-1][i1-1] -= sc-rs;
  s[i3-1][i2  ][i1  ] += sc+rs;
  s[i3-1][i2  ][i1-1] -= sb-rs;
  s[i3-1][i2-1][i1  ] += sd+rs;
  s[i3-1][i2-1][i1-1] -= sa-rs;
}}}
```

REFERENCES

Bahorich, M.S., and S.L. Farmer, 1995, 3-D seismic coherency for faults and stratigraphic features: The coherence cube: The Leading Edge, **14** 1053–1058.

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

Fehmers, G.C., C.F.W. Höcker, 2003, Fast structural interpretations with structure-oriented filtering: Geophysics, **68**, 1286–1293.

Fomel, S., 2002, Applications of plane-wave destruction filters: Geophysics, **67**, 1946–1960.

Hale, D., 2006, Recursive Gaussian filters: CWP Report 546, 269–278.

Marfurt, K.J., 2006, Robust estimates of 3D reflector dip and azimuth: Geophysics, **71**, P29–P40.

Marfurt, K.J., R.L. Kirlin, S.L. Farmer and M.S. Bahorich, 1998, 3-D seismic attributes using a semblance-based coherence algorithm: Geophysics, **63**, 1150–1165.

Neidell, N.S. and M.T. Taner, 1971, Semblance and other coherency measures for multichannel data: Geophysics, **36**, 482–497.

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

Taner, M.T., and F. Koehler, 1969, Velocity spectra — digital computer derivation and applications of velocity functions: Geophysics, **34**, 859–881.

van Vliet, L.J., and P.W. Verbeek, 1995, Estimators for orientation and anisotropy in digitized images: Proceedings of the first annual conference of the Advanced School for Computing and Imaging ASCI'95, Heijen (The Netherlands), 442–450.

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

Weickert, J., 1997, A review of nonlinear diffusion filtering: *in* B. t.H. Romeny, L. Florack, J. Koenderink and M. Viergever, eds., Scale-Space Theory in Computer Vision, Lecture Notes in Computer Science, Springer, **1252** 3–28.

Weickert, J., 1999, Coherence-enhancing diffusion filtering: International Journal of Computer Vision, **31**, 111–127.