CWP-567

Local dip filtering with directional Laplacians

Dave Hale

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

ABSTRACT

Local dip filters attenuate or enhance features with a specified dip that may vary for each image sample. Because these multi-dimensional filters change with each sample, they should have a small number of coefficients that can be computed efficiently from local dips. They should handle features that are vertical as well as horizontal. They should have efficient and stable inverses that facilitate the design and application of more discriminate notch filters. Local dip filters constructed from approximations to directional Laplacians have these properties and are easily implemented in any number of dimensions.

Key words: seismic image processing

1 INTRODUCTION

In seismic imaging of the earth's subsurface, we often describe the orientations of locally planar features by dip angles θ and, for 3-D images, azimuthal angles ϕ. Dip filters attenuate or enhance planar features based on their dips and azimuths, and *local dip filters* are those that can adapt locally to sample-to-sample changes in those parameters.

Orientations of locally planar features may also be described by reflection slopes. Fomel (2002) describes a method for implementing *plane-wave destruction filters* with numerous applications, including estimation of local slopes σ. Most of the applications described by Fomel are for images that have not been migrated, for which the vertical axis is time, and for which slopes are limited by seismic wave velocities.

After migration, slopes of features in seismic images may be infinite. Consider the dip of the flank of a salt dome or a fold or the dip of a fault plane. Robust local dip filters discriminate among features that are vertical as well as horizontal, without special handling of infinities. They are best parameterized by dips θ instead of slopes σ.

Local dip filters should be invertible. From inverses we can construct better dip filters and notch filters that surgically remove features with a specified local dip without attenuating other coherent features having slightly different dips.

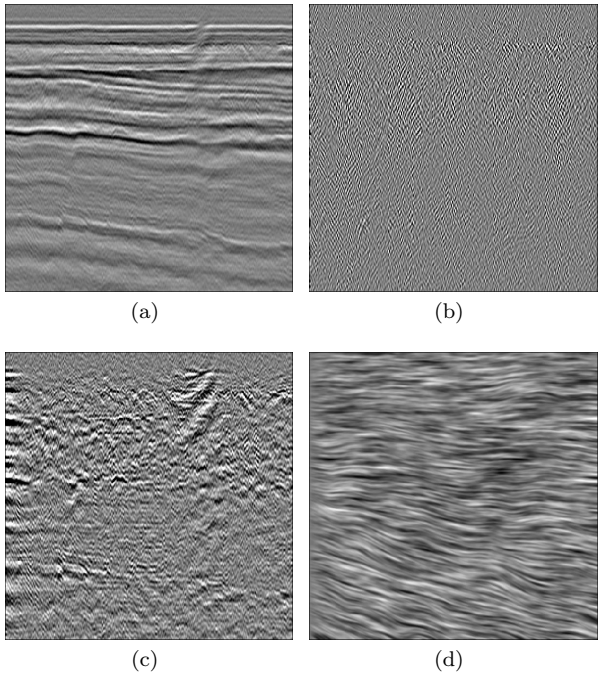
Figures 1 show an example. I first applied a local dip filter to the image of Figure 1a to obtain the image of Figure 1b. To this image I then applied the inverse of a slightly modified local dip filter to create a notch filter and the image of Figure 1c. Whereas both dip and notch filters have removed strong coherent events, the notch filter has preserved weaker but interesting coherent features in Figure 1c.

Inverses of local dip filters are also useful for regularization in seismic inverse problems. Instead of simply requiring that solutions to such problems be smooth, we may require that they be smooth in some spatially varying directions. For example, those directions might correspond to geologic dip (Clapp et al., 2004; Fomel and Guitton, 2006). Figure 1d shows an example of such anisotropic smoothing.

In this paper I describe invertible local dip filters that are based on approximations to directional derivatives of images. These robust filters handle features that are vertical as well as horizontal, and have inverses that can be used to construct notch filters. The directions for the derivatives and, hence, coefficients of the filters depend on estimates of dips of locally planar features.

1.1 Estimating local dips

To apply a local dip filter or its inverse, we need estimates of local dips. In all of the examples of this paper, I estimate local dips using local structure tensors, which are also called gradient-square tensors (van Vliet and



(a) (b) (c) (d)

Figure 1. A seismic image (a) after local dip-filtering to remove the dominant locally linear feature found at each sample. Filtering a broad range of dips (b) eliminates these features and many others as well, leaving only high spatial frequencies. Local notch filtering (c) is more discriminate, preserving many weaker but locally coherent features of interest. Applying the inverse of a local dip-filter to random noise yields a texture (d) that shows the local orientation estimated for each sample in (a).

Verbeek, 1995). For 2-D images, a structure tensor is a 2×2 matrix:

$$G = \begin{bmatrix} < g_1^2 > & < g_1 g_2 > \\ < g_1 g_2 > & < g_2^2 > \end{bmatrix}, \tag{1}$$

where g_1 and g_2 denote vertical and horizontal components of the gradient of an image, and $< \cdot >$ denotes 2-D Gaussian smoothing.

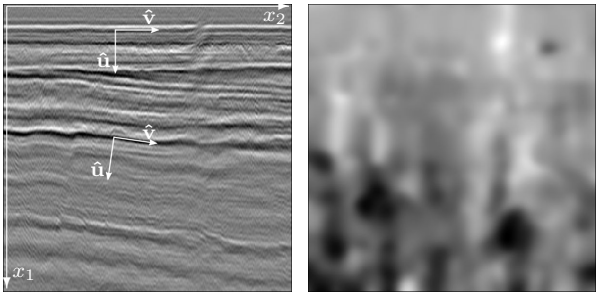
As shown by van Vliet and Verbeek (1995), the orthogonal unit eigenvectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ of the positive-semidefinite matrix G describe the orientation of locally linear features. Specifically, for each sample the vector $\hat{\mathbf{u}}$ corresponding to the largest eigenvalue is orthogonal to the locally dominant linear feature at that sample.

Figures 2 show examples. The components of the unit vectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ are related to local dips θ by

$$u_1 = \cos\theta \quad \text{and} \quad u_2 = -\sin\theta,$$

$$v_1 = \sin\theta \quad \text{and} \quad v_2 = \cos\theta.$$

By convention the vertical component u_1 of $\hat{\mathbf{u}}$ is non-negative; that is, $-\pi/2 \le \theta \le \pi/2$.



(a) (b)

Figure 2. Unit vectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ (a) define a coordinate system aligned with the dominant dip estimated for every image sample. By convention vertical components u_1 (not shown) of the local normal vectors $\hat{\mathbf{u}}$ are always non-negative and in this example are close to one for most samples. Horizontal components u_2 (b) are positive (white) for features dipping upward to the right ($\theta < 0$), and negative (black) for features dipping downward to the right ($\theta > 0$).

2 FOUR BASIC FILTERS

I begin by describing four basic local dip filters. The second and third filters are derived from the first filter, which was proposed by Claerbout (1992). The fourth filter is Fomel's (2002) plane-wave destruction filter, which I describe here for comparison and also because its implementation is almost identical to that of the third filter.

2.1 Claerbout's wavekill filter A

Let f denote a sampled image like that in Figure 2a, and let g denote the output of a local dip filter A applied to f. In directions parallel to the vectors $\hat{\mathbf{v}}$ in Figure 2a, the image f changes slowly, and so derivatives in those directions will be small. Hence, a simple local dip filter A can be constructed from a local directional derivative:

$$g = \hat{\mathbf{v}} \cdot \vec{\nabla} f,$$

or

$$A = \hat{\mathbf{v}} \cdot \vec{\nabla} = \hat{\mathbf{v}}^T \vec{\nabla}.$$

A simple finite-difference approximation to the gradient $\vec{\nabla}$ has components

$$\frac{\partial}{\partial x_1} \approx \begin{array}{|c|c|}\hline -\frac{1}{2} & -\frac{1}{2} \\ \hline \frac{1}{2} & \frac{1}{2} \\ \hline \end{array} \quad \text{and} \quad \frac{\partial}{\partial x_2} \approx \begin{array}{|c|c|}\hline -\frac{1}{2} & \frac{1}{2} \\ \hline -\frac{1}{2} & \frac{1}{2} \\ \hline \end{array}$$

where x_1 denotes the vertical spatial coordinate increasing downward and x_2 denotes the horizontal spatial coordinate increasing to the right. The filter A is then

$$A = \begin{array}{|c|c|}\hline -\frac{v_1+v_2}{2} & -\frac{v_1-v_2}{2} \\ \hline \frac{v_1-v_2}{2} & \frac{v_1+v_2}{2} \\ \hline \end{array},$$

where v_1 and v_2 are vertical and horizontal components of the unit vector $\hat{\mathbf{v}}$ aligned with the features that we wish to attenuate.

Alternatively, we can express the filter A in terms of vertical and horizontal components of the normal vector $\hat{\mathbf{u}}$:

$$A = \begin{array}{|c|c|} \hline -\frac{u_1-u_2}{2} & \frac{u_1+u_2}{2} \\ \hline -\frac{u_1+u_2}{2} & \frac{u_1-u_2}{2} \\ \hline \end{array} . \tag{2}$$

This is the stencil for the *wavekill filter* proposed by Jon Claerbout (1992). When applied to an image f, this filter attenuates features that are parallel to the vector $\hat{\mathbf{v}}$ and perpendicular to the vector $\hat{\mathbf{u}}$.

2.1.1 Implementing A

As the vectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ vary from sample to sample, so do the coefficients of this filter. The computational cost of computing those coefficients for each sample is small, due to their simplicity and the filter's compact 2×2 stencil.

If we let

$$m = \frac{u_1 - u_2}{2} \quad \text{and} \quad p = \frac{u_1 + u_2}{2}, \tag{3}$$

(m for minus, p for plus) then the wavekill filter stencil becomes

$$A = \begin{array}{|c|c|} \hline -m & p \\ \hline -p & m \\ \hline \end{array} . \tag{4}$$

An equation that implements this local dip filter is

$$\begin{aligned} g[i_1,i_2] \;=\; & m[i_1,i_2] \times f[i_1,i_2] \\ & + \; p[i_1,i_2] \times f[i_1-1,i_2] \\ & - \; p[i_1,i_2] \times f[i_1,i_2-1] \\ & - \; m[i_1,i_2] \times f[i_1-1,i_2-1] \end{aligned} \tag{5}$$

for all image sample indices i_1 and i_2.

In this implementation I have chosen the lower-right corner of the filter stencil as the output sample for the filter. My choice is somewhat arbitrary. The stencil has no sample about which it is symmetric, so any corner will do.

Choosing the lower-right corner makes A a causal quarter-plane filter in the sense that the output $g[i_1,i_2]$ depends only on present and past input samples in the upper-left quarter plane.

2.1.2 Implementing A^{-1}

If a causal filter has a causal and stable inverse, then we say it is minimum-phase. (For an extension of the minimum-phase concept to multi-dimensional filters, see Claerbout, 1998.) The filter A is not minimum-phase; its causal inverse is unstable. We obtain that causal inverse by rewriting equation 5 to solve recursively for

$$\begin{aligned} f[i_1,i_2] \;=\; & \big(g[i_1,i_2] \\ & - \; p[i_1,i_2] \times f[i_1-1,i_2] \\ & + \; p[i_1,i_2] \times f[i_1,i_2-1] \\ & + \; m[i_1,i_2] \times f[i_1-1,i_2-1] \big) / m[i_1,i_2]. \end{aligned} \tag{6}$$

A necessary (but insufficient) condition for stability is that the divisor $m[i_1,i_2]$ is never zero. For a dip $\theta = -45$ degrees this inverse filter is clearly unstable, for then $u_1 = u_2$ and $m = 0$. In fact this causal inverse filter is unstable for all negative dips for which $m < p$.

2.1.3 Amplitude spectra of A

To assess the fidelity of the forward filter A we may look at its 2-D amplitude spectra for various dips, as shown in Figure 3.

For small wavenumbers less than half-Nyquist, these filters have the desired amplitude response, with the greatest attenuation along a (dark blue) line in the direction of the normal vector $\hat{\mathbf{u}}$. For higher wavenumbers, contours of constant amplitude are no longer linear, and the wavekill filter attenuates dips that are not parallel to $\hat{\mathbf{v}}$. This dispersion is caused by the finite-difference approximation to the gradient $\vec{\nabla}$.

2.2 Symmetric filter $A^T A$

From the simple wavekill filter $A = \hat{\mathbf{v}} \cdot \vec{\nabla} = \hat{\mathbf{v}}^T \vec{\nabla}$ we can construct a symmetric filter

$$\begin{aligned} A^T A \;&=\; \vec{\nabla}^T \hat{\mathbf{v}} \hat{\mathbf{v}}^T \vec{\nabla} \\ &=\; \vec{\nabla}^T (I - \hat{\mathbf{u}} \hat{\mathbf{u}}^T) \vec{\nabla}, \end{aligned} \tag{7}$$

where $I = \hat{\mathbf{u}}\hat{\mathbf{u}}^T + \hat{\mathbf{v}}\hat{\mathbf{v}}^T$ is a 2×2 identity matrix. Since A is a directional derivative, $A^T A$ is like a directional second derivative, or a *directional Laplacian*. More precisely, $A^T A$ is the negative of a directional Laplacian, because $\vec{\nabla}^T \vec{\nabla} = -\vec{\nabla}^2$.

2.2.1 Implementing $A^T A$

An obvious way to apply this filter is to first apply the linear filter A and then apply its transpose A^T. The transpose filter A^T is easy to implement if we think of equation 5 as multiplication by a sparse matrix, with the columns of the input and output images f and g arranged end to end in tall column vectors.

Thinking of the filter A^T in this way leads us to the following observation. Whereas equation 5 *gathers* four weighted input samples f to compute one output sample g, its transpose *scatters* one input sample into four output samples with the same weights. This gather-scatter symmetry can be seen in any software that carefully implements the transpose of a linear filter.

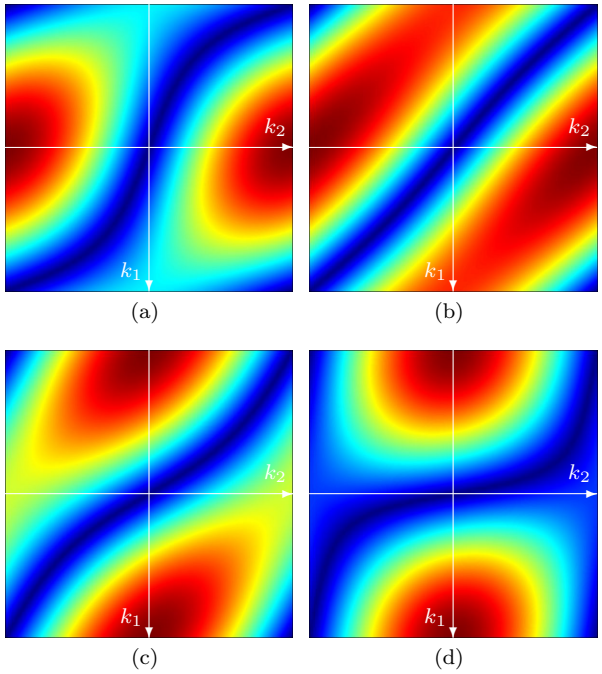


Figure 3. 2-D amplitude spectra of Claerbout's (1998) wavekill filters A for dips of (a) 20, (b) 40, (c) 60, and (d) 80 degrees, and for $-\pi \le k_1 \le \pi$ and $-\pi \le k_2 \le \pi$. Dark blue denotes zero. Dark red denotes the maximum amplitude, which varies for different filters.

Because the stencil for the filter A is small, the stencil for the filter $A^T A$ is only slightly larger

$$A^T A = \begin{array}{|c|c|c|} \hline -m^2 & 2mp & -p^2 \\ \hline -2mp & 1 & -2mp \\ \hline -p^2 & 2mp & -m^2 \\ \hline \end{array}, \tag{8}$$

where m and p are defined by equations 3. This stencil is simply the 2-D auto-correlation of that in equation 4.

I have momentarily assumed that m and p are constants. When they vary spatially the coefficients in this stencil are not centrosymmetric (not symmetric about its center) and the central coefficient may not equal one.

It might be tempting to implement this filter by using $m[i_1, i_2]$ and $p[i_1, i_2]$ for the indices i_1 and i_2 of the central sample in this stencil to compute the filter coefficients for the eight adjacent image samples. But this approach does not yield a symmetric positive-semidefinite composite filter $A^T A$.

As described above, one proper way to implement $A^T A$ is to first apply the filter A for variable coefficients, and then to apply the filter A^T for variable coefficients. The impulse response of the composite filter $A^T A$ will vary with the location of the impulse, but it will not generally be centrosymmetric like the stencil of equation 8 above.

A more efficient way to achieve $A^T A$ is suggested by equation 7. We may first apply the gradient filter $\vec{\nabla}$, then multiply by the 2×2 matrix $\hat{\mathbf{v}}\hat{\mathbf{v}}^T = I - \hat{\mathbf{u}}\hat{\mathbf{u}}^T$, and finally apply the transpose of the gradient filter $\vec{\nabla}^T$.

These three steps can all be performed in a single pass over the input and output images. Here is a fragment of a C, C++ or Java computer program that implements the filter of equation 7 in one-pass:

```
for (int i2=1; i2<n2; ++i2) {     // i2=0?
  for (int i1=1; i1<n1; ++i1) {   // i1=0?
    float u2i = u2[i2][i1];
    float u1i = sqrt(1.0f-u2i*u2i);
    float a11 = 1.0f-u1i*u1i;
    float a12 =     -u1i*u2i;
    float a22 = 1.0f-u2i*u2i;
    float fa = f[i2  ][i1  ]-f[i2-1][i1-1];
    float fb = f[i2  ][i1-1]-f[i2-1][i1  ];
    float f1 = 0.5f*(fa-fb);
    float f2 = 0.5f*(fa+fb);
    float g1 = a11*f1+a12*f2;
    float g2 = a12*f1+a22*f2;
    float ga = 0.5f*(g1+g2);
    float gb = 0.5f*(g1-g2);
    g[i2  ][i1  ]  = ga;
    g[i2-1][i1-1] -= ga;
    g[i2  ][i1-1] -= gb;
    g[i2-1][i1  ] += gb;
  }
}
```

This simple fragment does not compute the first row $i_1 = 0$ or first column $i_2 = 0$ of the output image g; those cases are easily handled by assuming zero values outside array bounds.

2.2.2 *Implementing* $(A^T A)^{-1}$

In applications requiring inverse filters, a symmetric positive-semidefinite $A^T A$ is especially useful. For if we try to apply $(A^T A)^{-1} = A^{-1} A^{-T}$ using a cascade of fast recursions as in equation 6, we encounter the same instability that we have seen before.

However, because $A^T A$ is symmetric positive-semidefinite and sparse, we can apply inverse filters by solving systems of equations $A^T A f = g$ by the iterative method of conjugate gradients. This method requires only three extra arrays, each the size of the images f and g. For 3-D images, this relatively low memory requirement can be an important consideration.

An alternative to conjugate-gradient iterations is Cholesky decomposition of $A^T A$. For variable coefficients this matrix decomposition may be more costly than the method of conjugate gradients.

However, an approximation to Cholesky decomposition may be adequate. The approximation is Wilson-Burg factorization (Fomel et al., 2003), a method for computing a minimum-phase filter from its auto-correlation. The Wilson-Burg method computes a minimum-phase filter $\tilde{A}$ such that

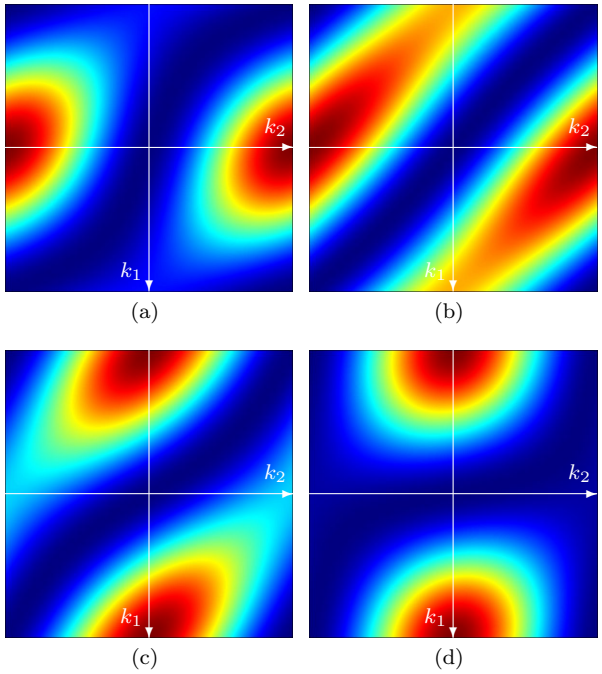


Figure 4. 2-D amplitude spectra of symmetric filters $A^T A$ for dips of (a) 20, (b) 40, (c) 60, and (d) 80 degrees, and for $-\pi \le k_1 \le \pi$ and $-\pi \le k_2 \le \pi$. These amplitudes are the square of those in Figures 3. Dark blue denotes zero. Dark red denotes the maximum amplitude, which varies for different filters.

$$\tilde{A}^T \tilde{A} \approx A^T A. \tag{9}$$

In my approximations I computed minimum-phase filters $\tilde{A}$ with 14 non-zero coefficients such that $\tilde{A}^T \tilde{A}$ approximates $A^T A$ in the stencil of equation 8 above.

I have tabulated such filters as a function of dip θ, and then applied $\tilde{A}$ for variable coefficients by selecting for each output sample the most appropriate filter from the table. Because each of the tabulated filters $\tilde{A}$ is minimum-phase, both $\tilde{A}$ and $\tilde{A}^T$ have stable inverses, and those inverses $\tilde{A}^{-1}$ and $\tilde{A}^{-T}$ can be implemented efficiently as recursive filters.

In practice the differences between $A^T A$ and $\tilde{A}^T \tilde{A}$ are insignificant; the approximation in equation 9 is adequate. Differences in the inverses however may be more significant. Even then the filter $\tilde{A}^{-1}\tilde{A}^{-T}$ is useful as a preconditioner (approximate inverse) in the method of conjugate gradients when applying $(A^T A)^{-1}$.

2.2.3 Amplitude spectra of $A^T A$

For constant coefficients we may compute amplitude spectra of the centrosymmetric stencil $A^T A$ of equation 8 for different dips. These are shown in Figures 4.

Amplitude spectra for $A^T A$ are simply the square of those for A. (Compare Figures 3 and 4.) Squaring the amplitudes broadens the valleys of attenuation. The filter $A^T A$ attenuates the specified dip but significantly attenuates many nearby dips as well. In this respect, the filter $A^T A$ is less discriminant than A.

2.3 Folded filter B

A better filter would have the amplitude spectrum of A and a stable inverse that does not require solution of a sparse system of linear equations. I obtained such a filter by folding the stencil of $A^T A$ in equation 8 from right to left symmetrically about its center:

$$B = \begin{array}{|c|c|c|} \hline -2m^2 & 2mp & \\ \hline -4mp & 1 & \\ \hline -2p^2 & 2mp & \\ \hline \end{array} .$$

In folding, I centrosymmetrically added coefficients on the right side of the stencil for $A^T A$ to those on the left side, leaving the central column of coefficients unchanged.

To understand why such folding might provide a useful dip filter, imagine a dipping feature that passes through the central sample of the stencil for $A^T A$. Because this stencil is centrosymmetric, the products of the dipping feature and coefficents on the right are the same as products obtained for coefficients on the left. So we can simply double the left-side products and omit the right-side products.

Another way to derive the stencil for B is to construct a weighted sum of wavekill filters with the goal of making that sum invertible. Recall that the inverse filter of equation 6 is unstable for $m = 0$. In other words, for $m = 0$, the filter of equation 5 is not invertible. In this case, that wavekill filter has zero weight in the weighted sum:

$$B = 2m \times \begin{array}{|c|c|} \hline -m & p \\ \hline -p & m \\ \hline 0 & 0 \\ \hline \end{array} + 2p \times \begin{array}{|c|c|} \hline 0 & 0 \\ \hline -m & p \\ \hline -p & m \\ \hline \end{array} .$$

In this sum, the left-hand stencil handles the positive dips $\theta > 0$ for which $u_2 < 0$ and $m \neq 0$, while the right-hand stencil handles the negative dips $\theta < 0$ for which $u_2 > 0$ and $p \neq 0$. The scale factor 2 makes this sum the same as the stencil obtained by folding:

$$B = \begin{array}{|c|c|} \hline -2m^2 & 2mp \\ \hline -4mp & 1 \\ \hline -2p^2 & 2mp \\ \hline \end{array} . \tag{10}$$

2.3.1 Implementing B

Implementation of the folded filter B with six coefficients is much like that for the wavekill filter A with four coefficients. We let the middle-right coefficient with value 1 in this stencil be the central sample for the filter B. Then, for each output sample, we simply multiply coefficients in this stencil by corresponding input samples and sum the products.

When the coefficients vary spatially this operation is not convolution; but it is linear, and we may again think of B as a large sparse matrix with which we compute an output image $g = Bf$ from an input image f.

2.3.2 Implementing B^{-1}

Because the central sample for the filter B is 1, and therefore never 0, we might hope that this filter is easily inverted. Indeed, this potential motivated the weighted sum used to derive B. However, unlike the wavekill quarter-plane filter A, the folded half-plane filter B is not causal, due to the non-zero lower-right coefficient $2mp$ that is generally non-zero.

Therefore, given $g[i_1, i_2]$ we cannot simply solve for the central sample $f[i_1, i_2]$ in terms of previously computed adjacent samples, as I did in equation 6. Specifically, the sample $f[i_1, i_2]$ is coupled by the right column of the stencil for B to the samples above and below it.

However, if we have already computed $f[i_1, i_2 - 1]$ for all indices i_1, then we may compute $f[i_1, i_2]$ for all i_1 by solving a tridiagonal system of equations. Unlike more general sparse systems, tridiagonal systems can be solved efficiently without iterations.

In summary, we may apply the inverse filter B^{-1} to an image by recursively solving tridiagonal systems of equations from left to right. We begin with $i_2 = 0$ and assume that $f[i_1, -1] = g[i_1, -1] = 0$. We then solve recursively for $f[i_1, 0]$, $f[i_1, 1]$, and so on.

2.3.3 Amplitude spectra of B

Amplitude spectra for the filter B are shown in Figures 5. Let us again focus our attention on small wavenumbers near the centers of these spectra, where the filters should be most accurate.

For small dips the amplitude spectra for B resemble those for the wavekill filter A in Figures 3. For the largest dip $\theta = 80$ degrees the amplitude spectrum of B is more like that for $A^T A$ in Figure 4d.

These differences in amplitude spectra are caused by folding in one direction. Folding horizontally makes the horizontal part of the directional second-derivative $A^T A$ more like a first-derivative, but leaves the vertical part like a second-derivative.

One might wonder whether folding both horizontally and vertically would reduce these differences. However, the resulting quarter-plane filter would treat features with positive dips differently than those with negative dips. Therefore I folded only horizontally to obtain the half-plane filter B.

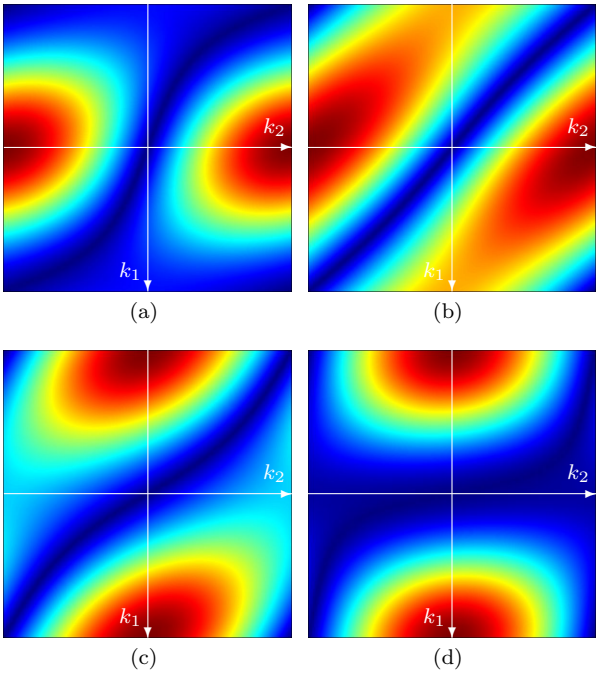


Figure 5. 2-D amplitude spectra of folded filters B for dips of (a) 20, (b) 40, (c) 60, and (d) 80 degrees, and for $-\pi \le k_1 \le \pi$ and $-\pi \le k_2 \le \pi$. Dark blue denotes zero. Dark red denotes the maximum amplitude, which varies for different filters.

2.4 Fomel's plane-wave destruction filter C

Our search for a filter with a stencil like that in equation 10 was motivated by Fomel's *plane-wave destruction filter* (2002), which has a similar stencil:

$$C = \begin{array}{|c|c|} \hline -\frac{(1+\sigma)(2+\sigma)}{12} & \frac{(1-\sigma)(2-\sigma)}{12} \\ \hline -\frac{(2+\sigma)(2-\sigma)}{6} & \frac{(2+\sigma)(2-\sigma)}{6} \\ \hline -\frac{(1-\sigma)(2-\sigma)}{12} & \frac{(1+\sigma)(2+\sigma)}{12} \\ \hline \end{array}. \tag{11}$$

where $\sigma = v_1/v_2 = -u_2/u_1 = \tan\theta$ is the slope of the feature to be attenuated.

The coefficients in the left and right columns of this stencil approximate quadratic interpolations of three samples with indices $i_1 - 1$, i_1, and $i_1 + 1$, evaluated at $i_1 - \sigma/2$ and $i_1 + \sigma/2$, respectively. By subtracting the interpolated value on the right from the one on the left, this filter annihilates features with slope σ.

The accuracy of the interpolation decreases with increasing $|\sigma|$. For vertical features, σ and the coefficients of C in equation 11 are infinite, and this filter is unstable.

Fomel describes higher order interpolations that could be used instead, but these too will fail for vertical or near vertical features. The problem here lies in choosing one direction for interpolation, the vertical x_1 direction. For features with slopes $|\sigma| > 1$ we should instead be interpolating in the horizontal x_2 direction.

2.4.1 Implementing C

Implementation of the plane-wave destruction filter for any slope is the same as that for filter B described above, and vice-versa. Indeed, one of my motives for designing filter B was to make it easy to insert B into any existing implementation of filter C.

Infinities for vertical features can be eliminated by simply multiplying the coefficients of this filter by u_1 to obtain

$$C_2 = \begin{array}{|c|c|} \hline -\frac{(u_1-u_2)(2u_1-u_2)}{12} & \frac{(u_1+u_2)(2u_1+u_2)}{12} \\ \hline -\frac{(2u_1-u_2)(2u_1+u_2)}{6} & \frac{(2u_1-u_2)(2u_1+u_2)}{6} \\ \hline -\frac{(u_1+u_2)(2u_1+u_2)}{12} & \frac{(u_1-u_2)(2u_1-u_2)}{12} \\ \hline \end{array}. \tag{12}$$

Coefficients of this normalized filter are finite for all dips.

2.4.2 Implementing C^{-1}

I have not used the normalized filters C_2 in the examples shown in this paper, partly because they are not the more familiar filters proposed by Fomel (2002), and also because normalization does not help with the implementation of inverse filters.

For slopes $|\sigma| < 1$ we can implement inverse filters C^{-1} the same way we implement B^{-1}. That is, we can recursively construct and solve tridiagonal systems of equations from left to right when applying C^{-1}.

However, for slopes $|\sigma| > 1$, the corresponding tridiagonal matrix is not diagonally dominant, and this left-to-right recursion becomes unstable.

2.4.3 Amplitude spectra of C

Amplitude spectra for the plane-wave destruction filter C of equation 11 are shown in Figures 6. For smaller dips, the amplitudes resemble those of the other filters described above.

For larger dips, these amplitude spectra are aliased. This aliasing may be useful when attempting to remove aliased dipping events from images, but in such cases the filter C will for some wavenumbers also remove unaliased events having different smaller dips.

3 EXAMPLES

Qualities of the four local dip filters described above are best illustrated with examples. To compare and contrast these filters, I applied them to images with small dips, large dips, and a test image with a complete range of all possible dips.

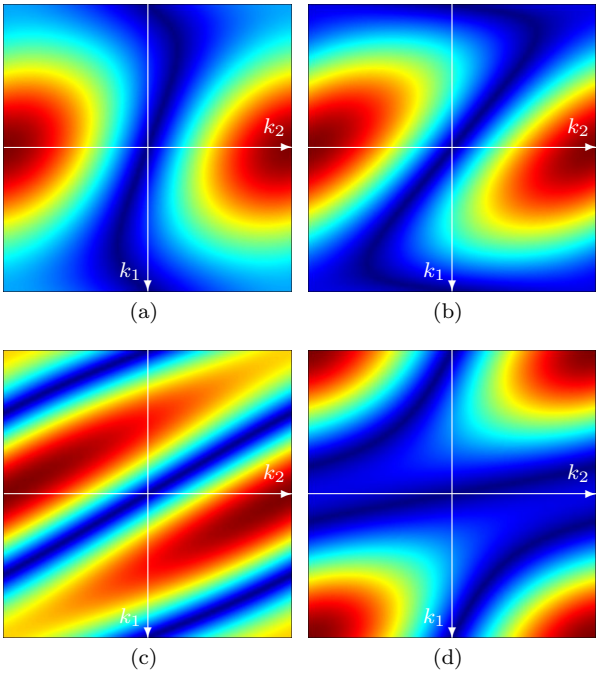


Figure 6. 2-D amplitude spectra of Fomel's plane-wave destruction filters C for dips of (a) 20, (b) 40, (c) 60, and (d) 80 degrees, and for $-\pi \le k_1 \le \pi$ and $-\pi \le k_2 \le \pi$. Dark blue denotes zero. Dark red denotes the maximum amplitude, which varies for different filters.

In all examples, I first estimated local dips from local structure tensors, and then used those dips to compute the coefficients for all four filters.

3.1 Small dips

The first example is the input image of Figure 1a (also Figure 2a). Dips of dominant features in this image are small, with $|\theta| < 45$ degrees.

Output images for all four filters — A, B, C and A^TA — are displayed in Figures 7. All filters attenuate the locally planar events in these images. The output images for filters A, B and C are almost identical as we would expect from similarities in their amplitude spectra for small dips.

The output for filter A^TA is notably different, as it effectively differentiates the input image twice instead of once. This filter therefore further amplifies high wavenumbers while attenuating a broader swath of dips for low wavenumbers. Again this output is consistent with the amplitude spectra for small dips shown in Figures 4a and 4b.

To test the inverses of the four filters for small dips, I applied them to an image containing isotropically ban-

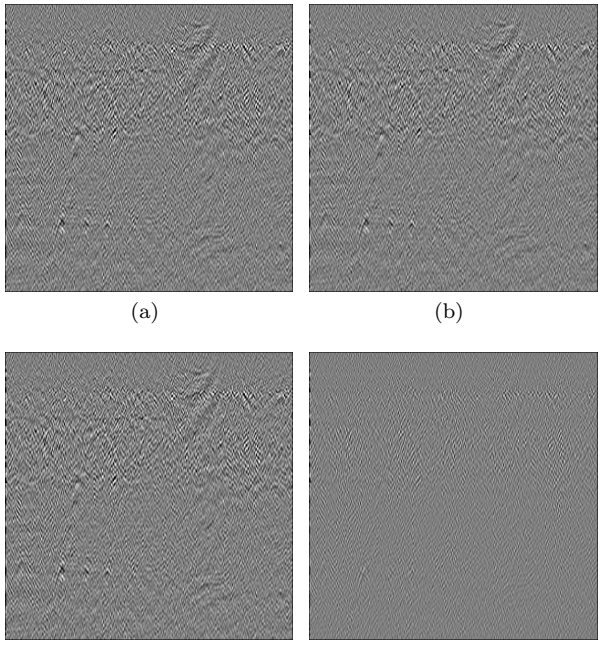


Figure 7. Output images for local dip filters (a) A, (b) B, (c) C and (d) A^TA applied to the image of Figure 1a. All filters attenuate locally coherent dipping features, but leave only features with dips that differ significantly from the predominate dip. For small dips, the outputs for the folded filter B and Fomel's plane-wave destruction filter C appear almost identical to that for the wavekill filter A.

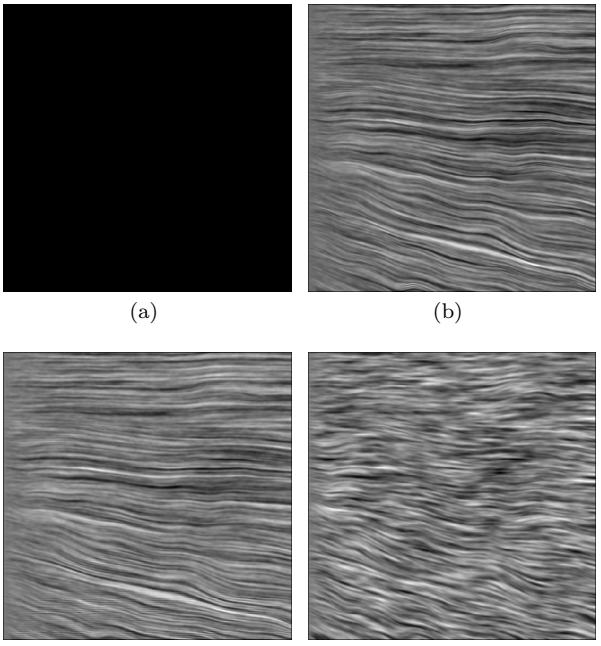


Figure 8. Application to a random-noise image of inverses of local dip filters (a) A, (b) B, (c) C and (d) A^TA. Causal inverses of wavekill filters A are unstable. Inverses for the folded filter B and Fomel's filter C are obtained by recursively solving tridiagonal systems of equations from left to right. The inverse of the symmetric filter A^TA is computed by conjugate-gradient iteration.

dlimited random noise. Inverses that are unstable for such random images are also unstable for real images, because pseudo-random rounding errors are created in the application of inverse filters to any real image.

The causal recursive inverse A^{-1} is unstable and produces no output. For the small dips in this example, the recursive tridiagonal inverses B^{-1} and C^{-1} produce almost identical textures.

Dips in the texture for the inverse $(A^TA)^{-1}$ are less well defined than those for B^{-1} and C^{-1}. Because the filter A^TA attenuates a wider range of dips, its inverse $(A^TA)^{-1}$ amplifies a wider range of dips instead of a single sharply defined dip at each sample.

3.2 Large dips

A simple way to test local dip filters for large dips is to transpose the image used in the previous examples, so that horizontal features become vertical. This transposed image is shown in Figure 9a.

Figure 9b shows a synthetic test image with small and large, negative and positive dips. For any sample in this image, there exists only one coherent event with one local dip, and the output for an ideal local dip filter should be zero.

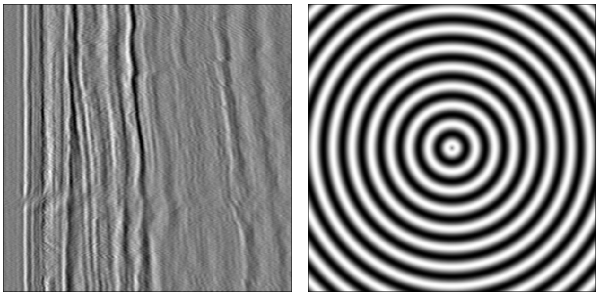


Figure 9. Test image (a) with vertical features is the transpose of the image of Figure 1a. Test image (b) is a synthetic image with all dips.

Filter outputs for the transposed input are shown in Figures 10. For this example, the coefficients of the plane-wave destruction filter C approach infinity, and this accounts for the high amplitudes in Figure 10c.

The similarity of the outputs for the folded filter B (Figure 10b) and the symmetric filter A^TA (Figure 10d) is consistent with their amplitude spectra for the largest dip in Figures 4d and 5d. For dips near 90 degrees, both

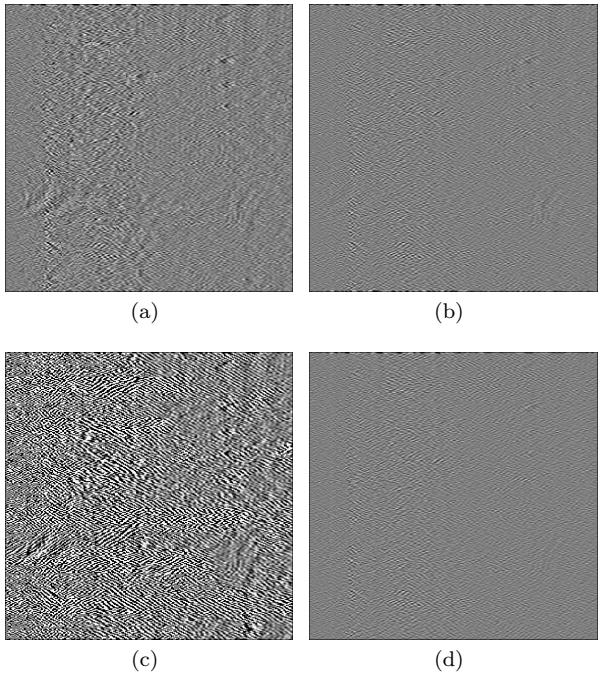


Figure 10. Output images for local dip filters (a) A, (b) B, (c) C and (d) A^TA applied to the transposed image in Figure 9a. For large dips, the output for Fomel's filter C goes to infinity.

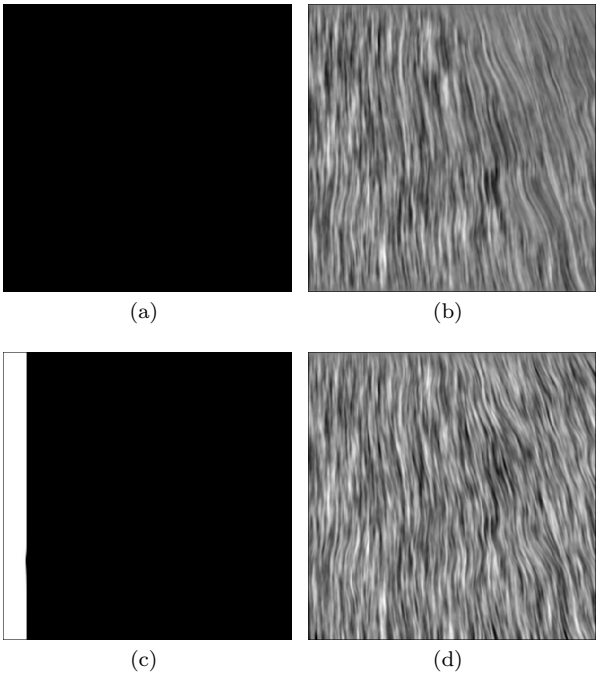


Figure 11. Application to a random-noise image of inverses of local dip filters (a) A, (b) B, (c) C and (d) A^TA for dips obtained from the transposed image in Figure 9a. Causal inverses of wavekill filters A are unstable. Inverses for the folded filter B and Fomel's filter C are obtained by recursively solving tridiagonal systems of equations from left to right. The inverse of the symmetric filter A^TA is computed by conjugate-gradient iteration.

of these filters approximate a second derivative in the vertical direction.

I applied the inverses of these four filters to a random-noise image to obtain the textures shown in Figures 11. The inverse A^{-1} for the wavekill filter is again unstable, as is the inverse C^{-1} for the plane-wave destruction filter.

For near-vertical events the inverses B^{-1} and $(A^TA)^{-1}$ exhibit similar textures in Figures 11b and 11d, consistent with the similarity of the filters outputs in Figures 10b and 10d.

Outputs for the circular synthetic input with all dips are shown in Figures 12. The most obvious difference in these output images is the instability of the plane-wave destruction filter C for large dips.

Textures for the four inverse filters are shown in Figures 13. The inverses A^{-1} and C^{-1} are again unstable. Texture for the inverse $(A^TA)^{-1}$ is most consistent for all dips.

4 USEFUL COMBINATIONS

We can combine basic filters like B and A^TA and their inverses to obtain notch filters or dip filters that are more useful than the basic filters alone.

To simplify notation, let H denote the filter A^TA defined by equation 7:

$$\begin{aligned} H \equiv A^TA &= \vec{\nabla}^T \hat{\mathbf{v}}\hat{\mathbf{v}}^T \vec{\nabla} \\ &= \vec{\nabla}^T (I - \hat{\mathbf{u}}\hat{\mathbf{u}}^T) \vec{\nabla}. \end{aligned}$$

If we neglect errors due to finite-difference approximations of derivatives, then the Fourier transform of this basic filter is

$$H(k_1, k_2) = (v_1 k_1 + v_2 k_2)^2.$$

Contours of constant amplitude $H(k_1, k_2)$ are parallel lines corresponding to constant $v_1 k_1 + v_2 k_2$. These parallel contours are apparent near the origins of the spectra in Figures 4, where wavenumbers and finite-difference errors are small.

4.1 Notch filters

To construct a notch filter, we first define a perturbed basic filter

$$H(\epsilon) \equiv \vec{\nabla}^T \hat{\mathbf{v}}\hat{\mathbf{v}}^T \vec{\nabla} + \epsilon I.$$

Then a notch filter is the composite filter defined by

$$H_n = H^{-1}(\epsilon)\ H(0).$$

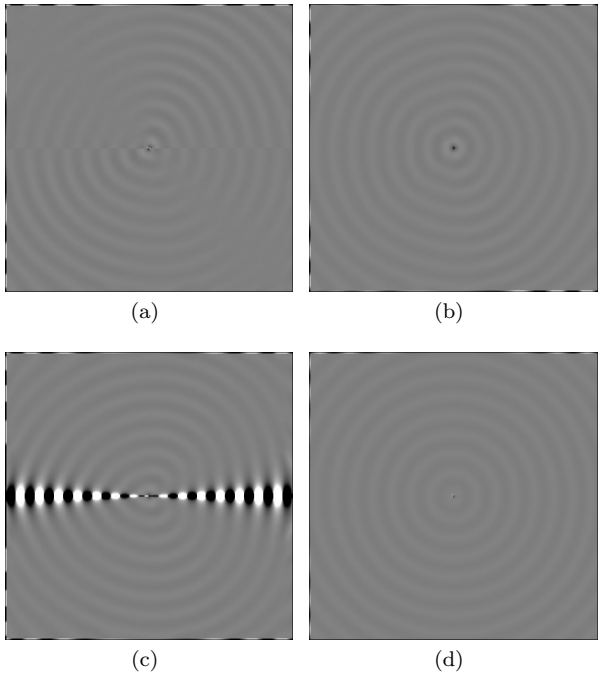


Figure 12. Output images for local dip filters (a) A, (b) B, (c) C and (d) A^TA applied to the test image in Figure 9b. For large dips, the output for Fomel's filter C goes to infinity.

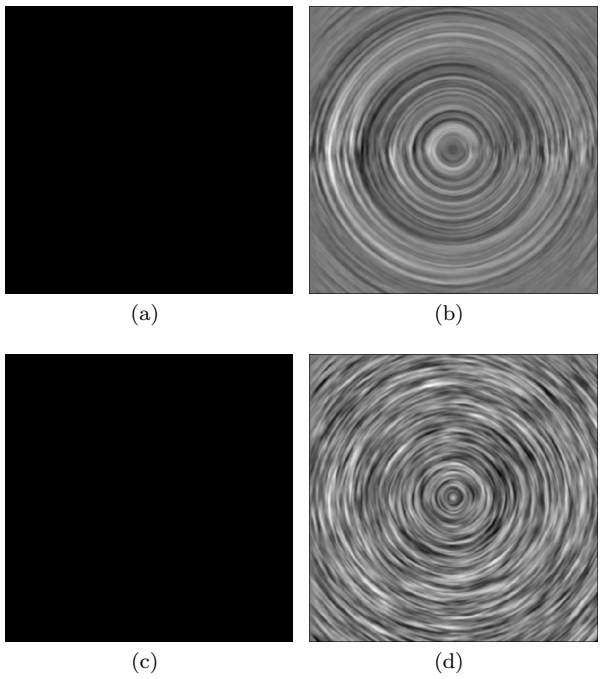


Figure 13. Application to a random-noise image of inverses of local dip filters (a) A, (b) B, (c) C and (d) A^TA for dips obtained from the synthetic test image in Figure 9b.

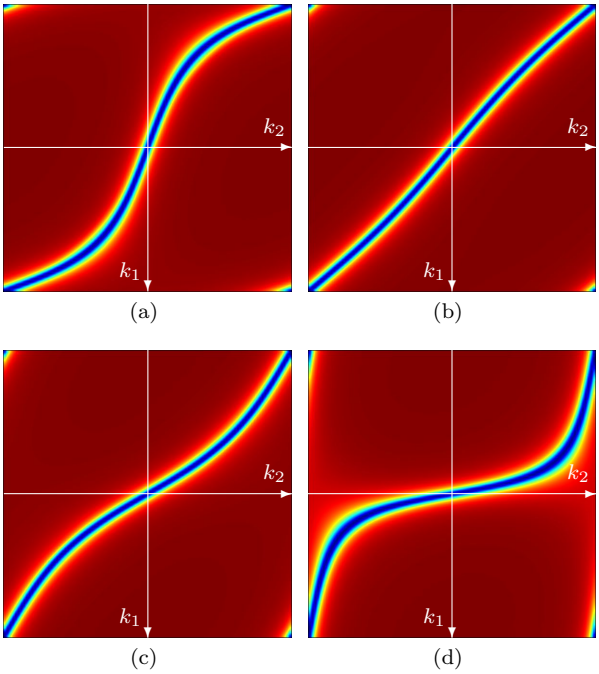


Figure 14. 2-D amplitude spectra of notch filters H_n for dips of (a) 20, (b) 40, (c) 60, and (d) 80 degrees, and $\epsilon = 0.01$. Dark blue and red denote amplitudes of zero and one, respectively.

Neglecting finite-difference errors, the Fourier transform of this notch filter is

$$H_n(k_1,k_2) = \frac{(v_1k_1+v_2k_2)^2}{(v_1k_1+v_2k_2)^2+\epsilon}.$$

Contours of constant amplitude $H_n(k_1,k_2)$ are again parallel lines corresponding to constant $v_1k_1+v_2k_2$.

Amplitude spectra for notch filters with $\epsilon = 0.01$ are displayed in Figures 14. Comparing these spectra with those of Figures 4, we see how notch filters H_n can be more discriminate than the basic filters $H = A^TA$ in their attenuation of specified dips.

The parameter ϵ controls the width of each notch. If we choose $\epsilon = 0$, then $H_n = 1$ and the filters do nothing. By increasing ϵ slightly we create a narrow notch at the dip to be zeroed, and the width of this notch grows with ϵ. Far from the notch, spectral amplitudes approach one.

A small positive ϵ has a side benefit. It increases the eigenvalues of $H(\epsilon)$, thereby reducing the number of iterations required when the method of conjugate gradients is used to apply $H^{-1}(\epsilon)$ in the notch filter H_n. We can easily modify any implementation of H^{-1} to implement $H^{-1}(\epsilon)$.

The difference between the basic filter $H = A^TA$ and the notch filter H_n is highlighted above in Figures 1b and 1c, respectively. Both filters attenuate the stronger coherent events in the image of Figure 1a. The basic filter H attenuates much more, leaving only high-

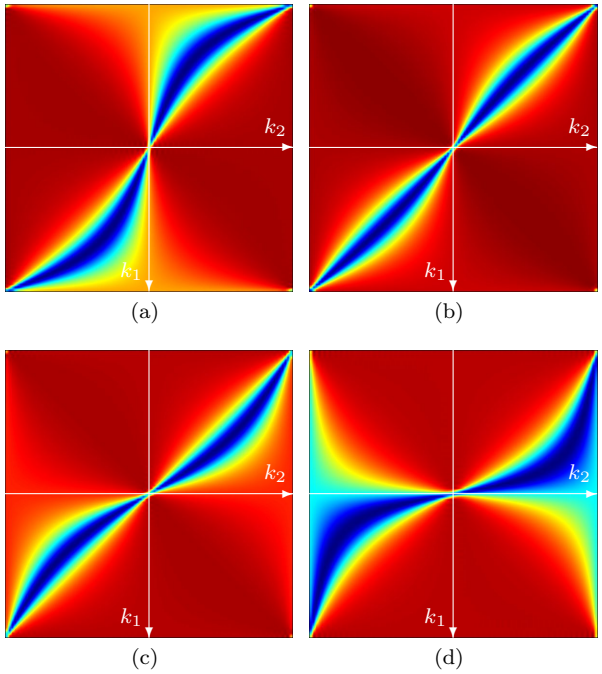


Figure 15. 2-D amplitude spectra of dip filters H_d for dips of (a) 20, (b) 40, (c) 60, and (d) 80 degrees, and $\epsilon = 0.05$. Dark blue and red denote amplitudes of zero and one, respectively.

wavenumber and mostly incoherent energy. The notch filter H_n is more discriminate, preserving many coherent and interesting features, while surgically removing the stronger events.

4.2 Better dip filters

To construct a better dip filter, we perturb our basic filter H in a slightly different way:

$$H(\epsilon) \equiv \vec{\nabla}^T \left[(1+\epsilon) I - \hat{\mathbf{u}}\hat{\mathbf{u}}^T \right] \vec{\nabla}.$$

Then the improved dip filter is the composite filter defined by

$$H_d = H^{-1}(\epsilon)\ H(0).$$

Neglecting finite-difference errors, the Fourier transform of this dip filter is

$$H_d(k_1, k_2) = \frac{(v_1 k_1 + v_2 k_2)^2}{(v_1 k_1 + v_2 k_2)^2 + \epsilon(k_1^2 + k_2^2)}.$$

Contours of constant amplitude $H_d(k_1, k_2)$ are lines radiating from the origin. Amplitude is a function of only the ratio k_2/k_1.

Amplitude spectra of this dip filter for $\epsilon = 0.05$ are displayed in Figures 15.

As their Fourier transforms suggest, our better dip filters H_d are simply notch filters for which the width of the notch grows with increasing wavenumbers k_1 and k_2. The parameter ϵ controls the rate at which the notch width grows or, equivalently, the range of dips attenuated.

4.3 3-D filters

Much of the design of local dip filters above can be extended to three (or more) dimensions.

We can estimate dips θ and azimuths ϕ of locally planar features from 3-D local structure tensors, 3×3 matrices computed from image gradients like those in equation 1 for two dimensions. For each image sample, the eigenvectors of these 3×3 matrices are orthogonal unit vectors $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$. The vector $\hat{\mathbf{u}}$ is normal to the best-fitting plane and corresponds to the largest eigenvalue. The vectors $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ lie within that best-fitting plane.

In three dimensions

$$A^T A = \vec{\nabla}^T (I - \hat{\mathbf{u}}\hat{\mathbf{u}}^T) \vec{\nabla},$$

which is identical to equation 7, except that vectors now have three components instead of two, and $I = \hat{\mathbf{u}}\hat{\mathbf{u}}^T + \hat{\mathbf{v}}\hat{\mathbf{v}}^T + \hat{\mathbf{w}}\hat{\mathbf{w}}^T$ is a 3×3 identity matrix. In any number of dimensions, the directional Laplacian $A^T A$ is the isotropic Laplacian $\vec{\nabla}^T \vec{\nabla}$ minus the projection of that Laplacian onto the normal vector $\hat{\mathbf{u}}$.

By subtracting the component of an isotropic Laplacian that is orthogonal to a plane, we construct local dip filters that attenuate features lying within that plane.

As discussed above for two dimensions, the most efficient way to apply $A^T A$ for any number of dimensions is to

(i) apply the gradient filter $\vec{\nabla}$,
(ii) multiply by the matrix $I - \hat{\mathbf{u}}\hat{\mathbf{u}}^T$, and
(iii) apply the transpose of the gradient filter $\vec{\nabla}^T$.

Again, these three steps can be performed in a single pass over the input and output images.

Having generalized directional Laplacian filters $A^T A$ to three or more dimensions, we can generalize composites of these filters as well. Definitions and implementations of the notch filter H_n and the dip filter H_d are almost identical in any number of dimensions.

Note that 3-D local dip filters constructed as directional Laplacians are not equivalent to a cascade of 2-D filters. To understand the difference, consider a hypothetical example in which we wish to remove horizontal planar features from a 3-D image. For these features, the normal vector $\hat{\mathbf{u}}$ points vertically downward.

In this case, the amplitude spectrum of the required directional Laplacian filter is $k_2^2 + k_3^2$, where k_2 and k_3 are wavenumbers corresponding to horizontal spatial coordinates x_2 and x_3. As expected, this amplitude is zero when both $k_2 = 0$ *and* $k_3 = 0$.

If we instead apply a 2-D filter in the x_2 direction, followed by a 2-D filter in the x_3 direction, the amplitude

spectrum of the composite filter is $k_2^2 k_3^2$. This amplitude is zero when either $k_2 = 0$ *or* $k_3 = 0$.

The difference between a 3-D directional Laplacian and a cascade of two 2-D filters lies in the difference between *and* and *or*. The filter cascade will attenuate features that appear horizontal in either constant-x_2 *or* constant-x_3 slices of a 3-D image, even when those features may be dipping in directions perpendicular to those slices. The 3-D directional Laplacian will correctly attenuate only truly horizontal features.

5 CONCLUSION

By constructing basic dip filters from directional derivatives, we obtain filters that

- adapt easily to changes in local dip,
- handle robustly all (even vertical) dips,
- can be inverted to construct notch filters, and
- extend easily to any number of dimensions.

This paper highlights two such basic dip filters B and $A^T A$.

The folded filter B was designed to fit on the stencil of Fomel's plane-wave destruction filter C. Software that uses filter C can easily be modified to use filter B, which more robustly handles steeply dipping features and has a stable and efficient inverse. The efficiency of the inverse B^{-1} is due to the efficiency with which we can solve tridiagonal systems of equations.

Folding to obtain the filter B from the symmetric filter $A^T A$ is useful in two dimensions, but less so in higher dimensions. The limitation is that folding works for only one axis. In three or more dimensions, the inverse B^{-1} of a folded filter B requires solution of a sparse system of equations that is not tridiagonal.

For 3-D images it may be simpler and more efficient to use $A^T A$ instead. The symmetric filter $A^T A$ has only a slightly larger stencil and can be implemented efficiently in only one pass over input and output images. Inverse filters can be applied either by conjugate-gradient iterations or with tables of minimum-phase filters precomputed by Wilson-Burg factorization.

Although the examples of this paper show only 2-D images, extension of the filter $A^T A$ to three and higher numbers of dimensions is straightforward. We begin with a finite-difference approximation to an isotropic N-dimensional Laplacian, and then subtract away projections of that operator corresponding to the features that we wish to preserve.

The filter $A^T A$ is not discriminate enough to be useful by itself. However, by combining this filter with inverses of slightly modified filters, we obtain notch filters and better dip filters. These combinations attenuate strong coherent signals while preserving weaker signals with slightly different dips. Such combinations may be used to enhance as well as attenuate image features.

REFERENCES

Claerbout, J.F., 1992, Earth soundings analysis — processing versus inversion: Blackwell Scientific Publications.

Claerbout, J.F., 1998, Multidimensional recursive filters via a helix: Geophysics, **63**, 1532–1541.

Clapp, R.G., B. Biondi, and J.F. Claerbout, 2004, Incorporating geologic information into reflection tomography: Geophysics, **69**, 533–546.

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

Fomel, S., and P. Sava, J. Rickett, and J.F. Claerbout, 2003, The Wilson-Burg method of spectral factorization with application to helical filtering: Geophysical Prospecting, **51**, 409–420.

Fomel, S., and A. Guitton, 2006, Regularizing seismic inverse problems by model reparameterization using plane-wave construction: Geophysics, **71**, A43–A47.

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.