

Fault surfaces and fault throws from 3D seismic images

Dave Hale

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

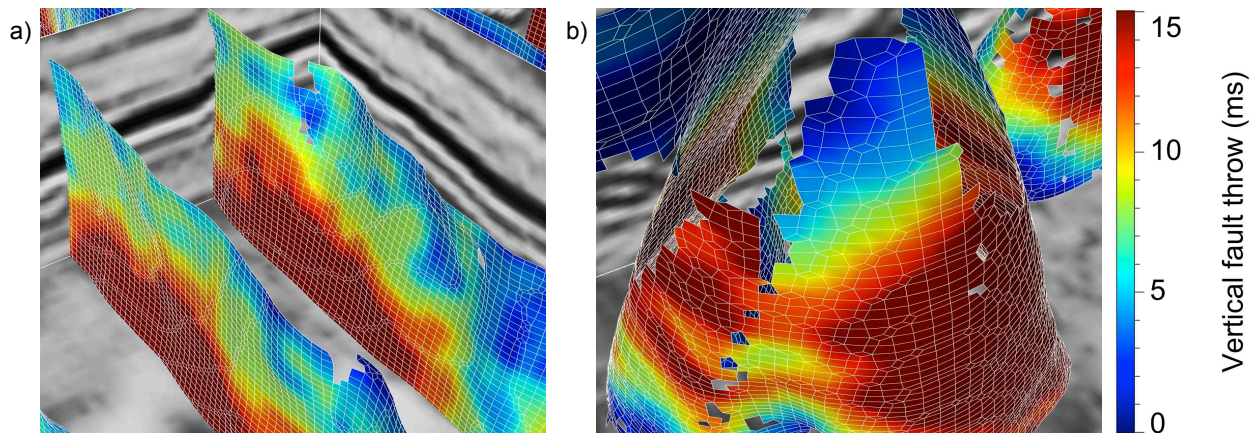


Figure 1. Roughly planar (a) and conical (b) fault surfaces and fault throws computed automatically from a 3D seismic image. Vertical and horizontal image slices are shown in the background. Vertical fault throws are measured in ms because the vertical axis of the image is time. Each quadrilateral intersects exactly one edge in the 4 ms by 25 m by 25 m image-sampling grid.

ABSTRACT

A new method for processing 3D seismic images yields images of fault likelihoods and corresponding fault strikes and dips. A second process automatically extracts from those images fault surfaces represented by meshes of quadrilaterals. A third process uses differences between seismic image sample values alongside those fault surfaces to automatically estimate fault throw vectors. While some of the faults found in one 3D seismic image have an unusual conical shape, displays of unfaulted images illustrate the fidelity of the estimated fault surfaces and fault throw vectors.

Key words: seismic geologic faults surfaces throws

1 INTRODUCTION

Fault surfaces like those displayed in Figure 1 are an important aspect of subsurface geology that we can derive from seismic images. Fault displacements, also shown in Figure 1, are important as well, as they enable correlation across faults of subsurface properties.

In the context of exploration geophysics, fault throw, relative displacement up or down the dip of a fault, is usually more significant than fault heave, displacement along the strike of a fault. Moreover, fault throw vectors are usually more perpendicular to geologic layers, and therefore easier to estimate, than are fault heave vectors.

As described by Luo and Hale (2012), we can use estimated fault throw vectors to undo faulting. Figure 2 displays multiple fault surfaces and corresponding fault throws computed for a 3D seismic image, before and after this *unfaulting* process. After unfaulting, seismic reflections are more continuous across faults, suggesting that estimated fault throws are generally consistent with true fault displacements.

Before this unfaulting, we must first compute images of faults, extract fault surfaces from those images, and estimate fault throws.

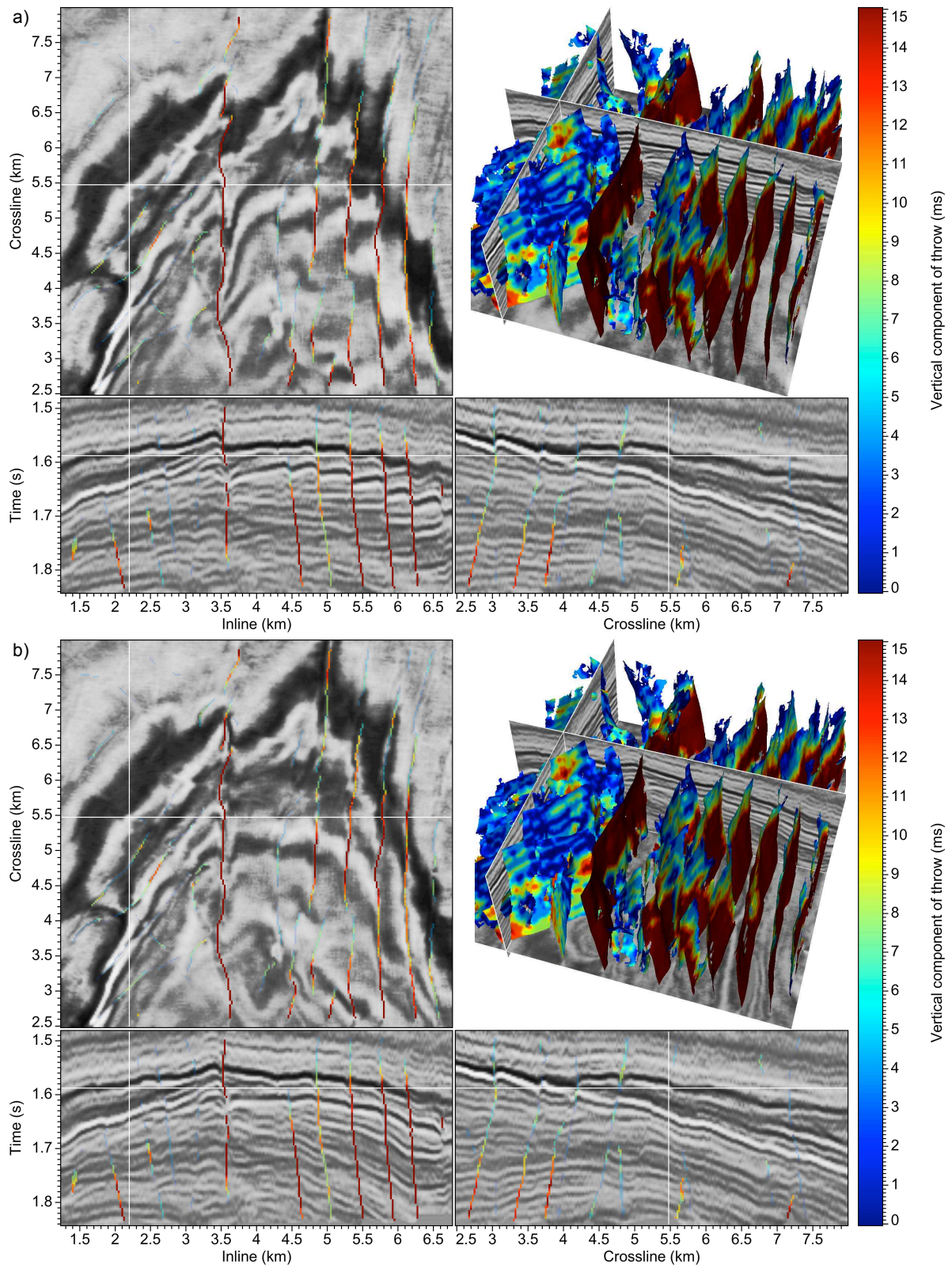


Figure 2. Fault surfaces and fault throws for a 3D seismic image before (a) and after (b) unfaulting.

1.1 Fault images

Several methods for highlighting faults, that is, for computing 3D images of faults from 3D seismic images, are commonly used today. Some compute a measure of the continuity of seismic reflections, such as semblance (Marfurt et al., 1998) or other forms of coherence (Marfurt et al., 1999). Others compute a measure of discontinuity, such as variance (Randen et al., 2001; Van Bemmelen and Pepper, 2011), entropy (Cohen et al., 2006), or gradient magnitude (Aqrabi and Boe, 2011). All of these methods are based on the observation that faults may exist where continuity in seismic reflections is low or, equivalently, where discontinuity is high.

However, in small regions within 3D seismic images, continuity may be low for reasons unrelated to faults. Stratigraphic features such as buried channels are well highlighted in seismic images by low continuity. Low continuity is also caused by incoherent noise that is stronger than weak seismic reflections. Even when a fault is present, seismic events may appear to be highly continuous when fault throws are approximately equal to the dominant period (or wavelength) of those events. Event continuity alone is insufficient to distinguish faults.

For these reasons, Gersztenkorn and Marfurt (1999) noted that any measure of continuity or discontinuity must include some form of averaging within vertical windows that should be longer when detecting faults than when detecting stratigraphic features. In effect, these averaging windows smooth together small regions of low continuity that are vertically aligned along faults with significant vertical extent. More recently, Aqrabi and Boe (2011) noted that such vertical smoothing of image gradient magnitudes (computed via Sobel filters) is desirable when highlighting faults.

However, faults are seldom vertical. When averaging any seismic attribute used to highlight faults, we should vary the orientation of this averaging to coincide with the strikes and dips of those faults. Neff et al. (2000) and Cohen et al. (2006) do this in their computation of fault images, as they scan over a range of fault orientations for each sample in a 3D seismic image. The computational cost of such scans can be high when, for each 3D image sample and for each possible fault orientation, one must process many samples [over 1300, in the example of Cohen et al. (2006)] within some box-shaped neighborhood.

1.2 Fault surfaces

To extract fault surfaces like those shown in Figures 1 and 2 from 3D images of faults requires additional processing, which again has been performed in various ways.

For example, Pedersen et al. (2002, 2003) and Pedersen (2007, 2011) developed the method of ant tracking

to merge together small regions of low continuity in 3D fault images into larger fault surfaces.

Gibson et al. (2005) propose a multistage method of constructing larger fault surfaces by merging smaller ones, beginning with small surfaces that correspond to “local discontinuities” in 3D seismic images. Different methods for growing large fault surfaces from small initial surfaces have also been proposed by Admasu et al. (2006) and Kadlec et al. (2008); Kadlec (2011). In such methods, seismic interpreters can specify seed points from which to begin growing fault surfaces.

In a more general context, Schultz et al. (2010) describe a direct method for extracting so-called crease surfaces from 3D images without seed points. In one example, they extract surfaces corresponding to ridges in a 3D image of fractional anisotropy, which is computed from 3D diffusion tensor magnetic resonance images (DT-MRI) of the human brain. Their method of extracting surfaces works well for 3D images with ridges that are well-defined and continuous.

1.3 Fault throws

Methods for computing fault surfaces lead naturally to the problem of estimating relative displacements of geologic layers alongside such surfaces. Solutions to this problem are not trivial, in part because of the sinusoidal character of seismic waveforms alongside faults, which can cause apparent horizontal alignment of seismic events across faults even when fault throws are significant. Another difficulty is that fault throws typically vary within the spatial extent of any fault surface. Nevertheless, several authors have described solutions to the problem of estimating fault throws.

For example, Aurnhammer and Tönnies (2005) demonstrate the use of local crosscorrelations computed in rectangular windows and a genetic algorithm with geological and geometrical constraints to match horizons extracted from both sides of faults in 2D seismic images.

Liang et al. (2010) also used local crosscorrelations to estimate fault throws, while simultaneously scanning over fault dips to determine the locations and orientations of faults in 2D seismic images.

Admasu (2008) addressed the problem of estimating fault throws from 3D images through a Bayesian matching of seismic horizons extracted alongside faults in vertical 2D image slices, with the matching for one 2D slice used as a guide for the matching in adjacent slices. This method requires that faults surfaces are approximately orthogonal to the 2D image slices used to compute the fault throws.

In a 3D solution to the problem, Borgos et al. (2003) correlated seismic horizons across faults by clustering into classes local extrema in various attributes computed from 3D seismic images. Carrillat et al. (2004) and Skov et al. (2004) show examples of analyzing fault displacements computed using this method. In another

3D solution, Bates et al. (2009) demonstrated a “geo-model time differential analysis method” for computing fault throws after automatic horizon tracking.

1.4 This paper

This paper contributes to solutions of all three of the problems described above: (1) computing 3D fault images, (2) extracting fault surfaces, and (3) estimating fault throws. The sequence of solutions proposed here was used to compute the fault surfaces and throws displayed in Figures 1 and 2. Although each of these three solutions was designed in conjunction with the others in the sequence, aspects of any one of them could be adapted to enhance other methods summarized above.

I first compute 3D fault images of an attribute I call fault likelihood. Much like Cohen et al. (2006), I scan over multiple fault strikes and dips to maximize this semblance-based attribute. However, the computational cost of the algorithm I use to perform this scan is independent of the number of samples used in the averaging performed for each fault orientation. In other words, I improve computational efficiency by eliminating the factor (of 1300 or more) equal to the number of samples in the windows described by Cohen et al. (2006).

I then use the resulting 3D images of fault likelihoods, dips and strikes to extract fault surfaces using a method that is similar to that proposed by Schultz et al. (2010). The fault surfaces shown in Figures 1 and 2 are ridges in 3D images of fault likelihood, and are represented by meshes of quadrilaterals. I have made no attempt to fill any of the small holes apparent in these surfaces, although such a filling process would be easy to implement because every quadrilateral is linked to its neighbors. The fact that holes are small is due to the continuity of ridges in the 3D images of fault likelihood.

Finally, I compute fault throws from differences in values of samples extracted from 3D seismic images alongside fault surfaces. The algorithm I use to compute fault throws is derived from a classic dynamic programming solution (Sakoe and Chiba, 1978) to a problem in speech recognition. That solution today is often called *dynamic time warping* and is here extended to find a spatial warping that best aligns samples of 3D seismic images alongside faults, as illustrated in Figure 2.

2 FAULT IMAGES

Whereas seismic horizons appear in 3D seismic images as coherent events, a fault appears less prominently as a curvilinear surface on which seismic events are discontinuous, yet correlated, with some displacement, from one side of the fault to the other. Therefore, a useful first step in extracting fault surfaces and estimating

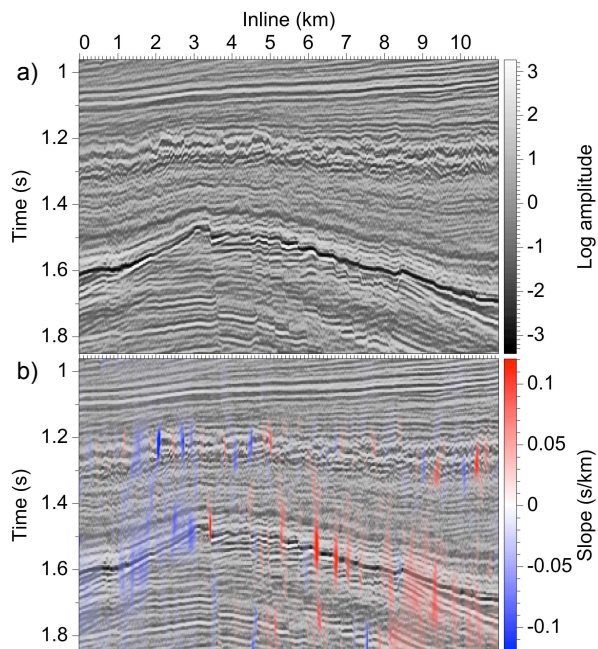


Figure 3. A 2D seismic image $g[i_t, i_x]$ after gain (a), with local reflection slopes $p[i_t, i_x]$ (b) displayed in color.

fault throws is to first compute 3D fault images in which faults are most prominent.

2.1 Semblance

The method I use for this first step is based on semblance (Taner and Koehler, 1969), and is therefore similar to methods proposed by Marfurt et al. (1998). Like Marfurt et al. (1999), I compute semblances from small numbers (3 in 2D, 9 in 3D) of adjacent seismic traces, after aligning those traces so that any coherent events are horizontal.

This process is illustrated for 2D seismic image shown in Figure 3a. Let $g[i_t, i_x]$ denote such an image, an array indexed by two integers: i_t , for time or depth, and i_x , for inline distance. To enhance the visibility of weaker features in this image, I applied the gain function $\text{sgn}(\cdot) \log(1 + |\cdot|)$ to every image sample.

Using structure tensors (e.g., van Vliet and Verbeek, 1995; Weickert, 1999), I first compute for the gained seismic image g a corresponding image of local reflection slopes p , displayed in color in Figure 3b. I then define and compute structured-oriented semblance as

$$s[i_t, i_x] = \frac{\langle s_n[i_t, i_x] \rangle}{\langle s_d[i_t, i_x] \rangle}, \quad (1)$$

where $\langle \cdot \rangle$ denotes some sort of smoothing (discussed be-

low), and

$$\begin{aligned}\tilde{g}[i_t, i_x; j_x] &= g(i_t + p[i_t, i_x] j_x, i_x + j_x), \\ s_n[i_t, i_x] &= \left\{ \frac{1}{2M_x + 1} \sum_{j_x=-M_x}^{M_x} \tilde{g}[i_t, i_x; j_x] \right\}^2, \\ s_d[i_t, i_x] &= \frac{1}{2M_x + 1} \sum_{j_x=-M_x}^{M_x} \{\tilde{g}[i_t, i_x; j_x]\}^2.\end{aligned}\quad (2)$$

Structure-oriented semblance is therefore simply the square of an average of slope-aligned sample values \tilde{g} divided by an average of the squares of those same values. The number of traces in the local windows used to compute these averages is $2M_x + 1$; I choose $M_x = 1$ so that only three traces must be aligned when computing semblance numerators s_n and denominators s_d .

This definition of structure-oriented semblance is easily extended to 3D images $g[i_t, i_x, i_y]$. After computing local inline slopes p_x and crossline slopes p_y , I compute semblance numerators and denominators using local windows of $9 = 3 \times 3$ slope-aligned traces.

2.2 Smoothing

The smoothing denoted by $\langle \cdot \rangle$ in equation 1 is an essential part of the semblance computation for two reasons. First, without this smoothing, semblances are unstable where the denominators in equation 1 are nearly zero, that is, where slope-aligned values \tilde{g} are nearly zero.

The second reason is that discontinuities in seismic images corresponding to faults are most significant for strong reflections that may be separated by multiple periods or wavelengths. Some sort of smoothing is necessary to link together these localized regions in which semblance numerators s_n are much smaller than semblance denominators s_d .

It is for this second reason that Gersztenkorn and Marfurt (1999) recommend the use of longer vertical smoothing windows when highlighting structural features such as faults, and shorter windows when highlighting stratigraphic features such as channels. In proposing a different gradient-based measure of discontinuity, Aqrabi and Boe (2011) likewise use a vertical smoothing of that measure for the same reason.

Figure 4a shows structure-oriented semblances computed using a vertical two-sided exponential smoothing filter. This smoothing filter is efficient and trivial to implement. An implementation in the programming language C++ (or Java) for input array x and output array y , both of length n , is as follows:

```
float b = 1.0f-a;
float yi = y[0] = x[0];
for (int i=1; i<n-1; ++i)
    y[i] = yi = a*yi+b*x[i];
y[n1-1] = yi = (a*yi+x[n1-1])/(1.0f+a);
for (int i=n1-2; i>=0; --i)
    y[i] = yi = a*yi+b*y[i];
```

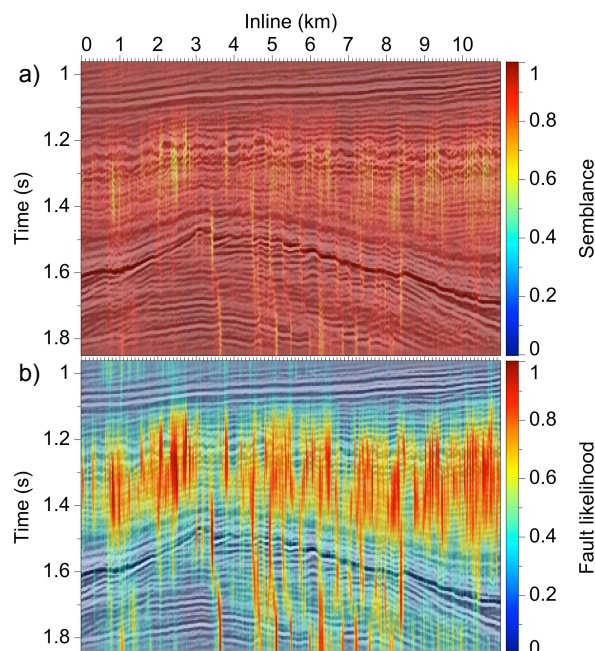


Figure 4. Semblances s (a) and fault likelihoods f (b).

The extent of smoothing is controlled by the parameter a . In the example shown in Figure 4, $a = 0.93$, which for low frequencies approximates a Gaussian filter with half-width $\sigma = 20$ samples. (In practice, I specify σ and compute the corresponding parameter a .) As for a Gaussian filter, the impulse response for a two-sided exponential filter is infinitely long but decays smoothly to zero.

This vertical smoothing of semblance numerators and denominators accounts for the vertical extent of features with low semblance s apparent in Figure 4a. To accentuate these features I define an attribute *fault likelihood* f by

$$f \equiv 1 - s^8. \quad (3)$$

The choice of power 8 is somewhat arbitrary; it simply increases the contrast between samples with low and high fault likelihoods, as shown in Figure 4b.

Although features in semblance and fault likelihood images shown in Figure 4 have significant vertical extent, these features are not well aligned with faults, because the faults are not vertical. To improve the fault likelihood attribute f , we must instead smooth along the faults. Our problem is that we have not yet determined the locations or orientations of the faults.

2.3 Scanning

This sort of problem is common in seismic data processing, for example, when we must perform normal moveout corrections without knowing the moveout velocities.

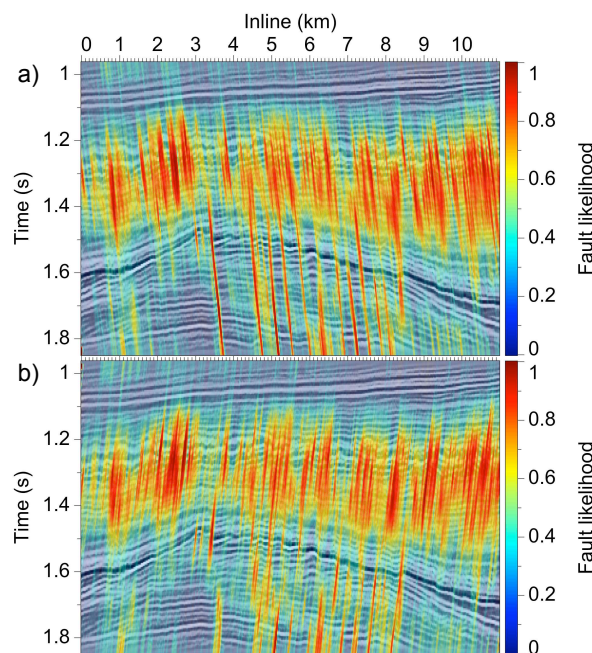


Figure 5. Fault likelihoods computed for two different fault dips θ , one positive (a) and the other negative (b), in the scan used to estimate fault dips.

A common solution is to perform a scan for multiple velocities to find the velocities that maximize some (often semblance-based) measure of alignment. Here I scan over fault dips θ to find those dips that maximize fault likelihoods f .

Figure 5 illustrates the results of non-vertical smoothing for two different fault dips θ in this scan. These examples show that fault likelihoods tend to be largest when smoothing of semblance numerators and denominators is performed along the faults, which are not vertical.

To perform this non-vertical smoothing efficiently for each fault dip θ , I (1) shear both semblance numerator and denominator images horizontally to make faults with that dip appear to be vertical, (2) apply the simple vertical smoothing filter described above, and (3) unshear the smoothed images before computing their ratio.

A fault that is vertical after horizontal shearing is shorter than it was before shearing. I therefore scale the half-width σ of the vertical smoothing filter by $\cos \theta$, to compensate for this shortening.

Note that the cost of the scan over fault dips does not depend on the extent of smoothing, which is controlled by the parameter a in the recursive smoothing filter. This recursive filter is largely responsible for reducing the computational cost of this scan, relative to those described by Neff et al. (2000) and Cohen et al. (2006).

This cost reduction is especially significant when

scanning over both fault dips θ and strikes ϕ for 3D seismic images. In these scans smoothing of semblance numerators and denominators must be two-dimensional, within planes spanned by fault strike and dip vectors.

In scanning over fault strikes, for each strike angle ϕ , I rotate the semblance numerator and denominator images, to align the fault strike direction with either of the horizontal image axes. I then smooth the rotated images once horizontally along the fault strike direction before scanning over fault dips θ . The computational costs of rotation and horizontal smoothing for each fault strike ϕ are therefore negligible compared to the cost of the scans over fault dips θ . The cost of an entire scan over fault strikes and dips for a 3D image is dominated by a sequence of scans over fault dips for multiple 2D images.

An alternative to the sequence of rotation, horizontal smoothing, shearing and vertical smoothing described above is to implement the smoothing filters $\langle \cdot \rangle$ with a fast Fourier transform (FFT). In my implementations, such FFT-based smoothing filters are simpler, but about three times slower, than the sequence described above.

Computational cost is also a factor in my choice of semblance, which requires smoothing of only numerator and denominator images s_n and s_d . Alternatives such as the normalized correlation coefficient (Rodgers and Nicewander, 1988) or the eigen-structure-based coherence described by Gersztenkorn and Marfurt (1999) would require smoothing of more images for each fault strike and dip in the scan.

Sampling of fault strike and dip angles in the scan requires computation of angle sampling intervals and specification of lower and upper bounds. Because the units for axes of seismic images are often different — time versus distance — I measure angles in sample coordinates, so that an angle of forty-five degrees corresponds to a slope of one sample per sample.

Suitable sampling intervals that avoid undersampling are $\Delta\phi = \frac{1}{2\sigma_\phi}$ and $\Delta\theta = \frac{1}{2\sigma_\theta}$, both measured in radians, where σ_ϕ and σ_θ denote half-widths of the smoothing filters in the strike and dip directions, respectively.

When scanning to compute fault likelihoods for all examples shown in this paper, I chose $\sigma_\phi = 4$ samples and $\sigma_\theta = 20$ samples. These smoothing filter half-widths yield sampling intervals $\Delta\phi \approx 7.2$ degrees and $\Delta\theta \approx 1.4$ degrees. Minimum and maximum fault strikes were -90 and 90 degrees, and minimum and maximum fault dips (measured from vertical) were -15 and 15 degrees, so that the numbers of fault strikes and dips scanned were $N_\phi = 26$ and $N_\theta = 22$, respectively. The size of the product $N_\phi N_\theta = 572$ highlights the practical need to reduce the computational cost of computing fault likelihoods for each of the fault orientations in the scan over possible fault orientations.

Recall that fault dip angles are measured with re-

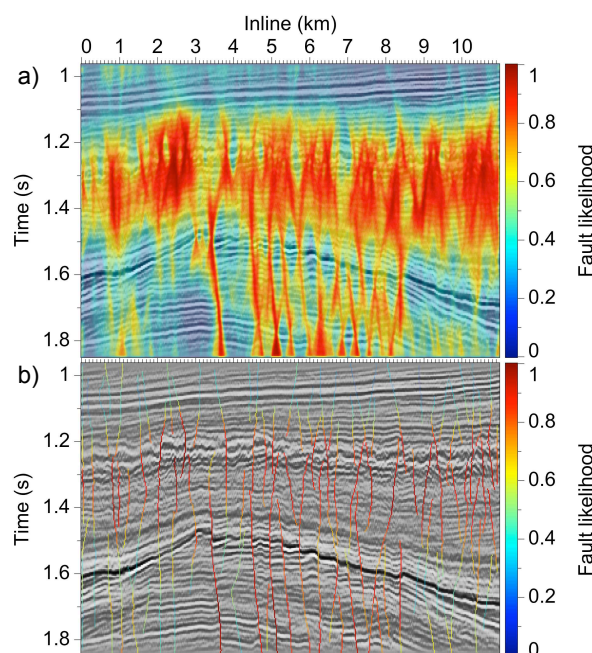


Figure 6. Fault likelihoods computed by scanning over fault dips θ , before (a) and after (b) thinning.

spect to sample coordinates. Using the simple approximation that one millisecond of time corresponds approximately to one meter of depth, the true magnitudes of dips for many of the faults apparent in Figure 2 are roughly 45 degrees. Using the same approximation, the maximum fault dip scanned was roughly 60 degrees.

2.4 Fault likelihoods

The purpose of the scan over fault strikes and dips is to find, for each image sample, the angles ϕ and θ that maximize the fault likelihood f . I begin with a fault likelihood image $f = 0$. Then, for each orientation (ϕ, θ) in the scan, where the fault likelihood $f_{(\phi, \theta)}$ exceeds the maximum likelihood stored in f , I update f and also save the corresponding strike ϕ and dip θ . When complete, the results of this scan are images of maximum fault likelihoods and corresponding fault strikes and dips.

Figure 6a shows fault likelihoods computed with a scan over 22 fault dips for the 2D seismic image. Ridges of fault likelihood in this *fault image* generally coincide with faults apparent in the seismic image. These ridges can be found by simply scanning each row of the fault image, preserving only local maxima, and setting fault likelihoods elsewhere to zero. In effect, this process thins the fault image, reducing the number of image samples at which a fault might be considered to exist.

Figure 6b shows ridges extracted from the fault image of Figure 6a, after discarding any ridges with fewer

than $2\sigma_\theta = 40$ adjacent samples. Parts of some ridges, especially those with lower fault likelihoods, may not coincide with faults. At this stage I do not suppress these parts, although one might easily suppress some of them by thresholding fault likelihoods.

Instead, I keep all ridges with length sufficient to reliably estimate fault throws. Then, faults can be assumed to exist at locations where fault likelihoods are high and fault throws are non-zero. Because, after thinning, so few samples are involved, such filtering of fault ridges can be performed interactively.

It is significant that the scanning process used to compute images of fault likelihood also yields images of fault strikes and dips for which fault likelihood is maximized. Those fault and strike and dip angles are especially useful when extracting fault surfaces from ridges in 3D images of fault likelihoods.

3 FAULT SURFACES

One can easily imagine how to extract fault curves from 2D fault images like the one shown in Figure 6a. For example, we might simply link together samples with non-zero fault likelihood in the thinned fault image of Figure 6b. We could then use samples of the seismic image on the left and right sides of the extracted fault curves to estimate fault throws.

It is more difficult to construct fault surfaces from 3D fault images. One problem is how to best represent a fault surface, which need not be aligned with any axis of the sampling grid for the 3D seismic image. For example, the roughly conical fault displayed in Figure 1b cannot be projected onto a plane, and therefore cannot be represented by a single-valued function (such as distance) of coordinates within that plane.

Also, the resolution with which we sample fault surfaces will be important later, when we compute fault throws. In that computation, we must be able to efficiently traverse upward and downward along fault curves of constant strike as we analyze seismic image samples alongside fault surfaces. We must also be able to efficiently traverse left and right along fault traces for which time (or depth) is constant.

For these reasons, I represent each fault surface with an unstructured mesh of quadrilaterals (quads) like those shown in Figure 1.

3.1 Extracting quads from fault images

My first step in constructing quad meshes is to extract a set of quads, not yet connected, from the 3D image of fault likelihoods. That 3D image is analogous to the 2D image of fault likelihoods shown in Figure 6a.

As shown in Figure 7, each quad in a fault surface intersects exactly one edge of the 3D sampling grid for the fault image. Each of the four nodes of a quad lies

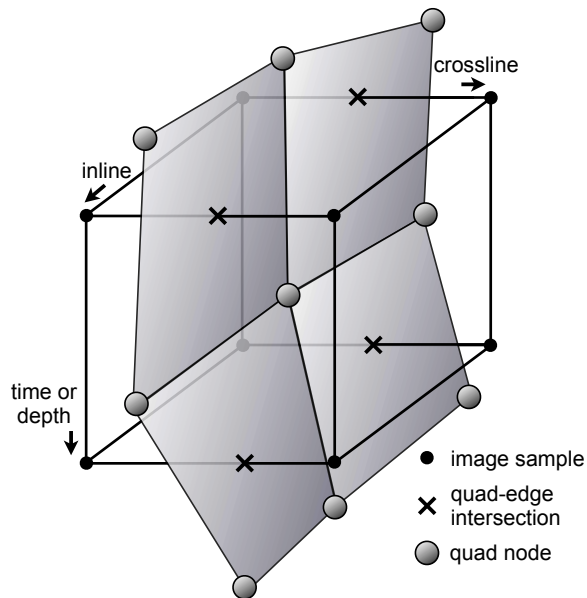


Figure 7. Four adjacent quads in a fault surface share a node that lies within one cell of the 3D fault image sampling grid. Spatial coordinates of the quad node are averages of the coordinates of intersections of the fault surface and edges of the image sampling grid.

within exactly one cell of that grid. The coordinates of a quad node within any such cell are averages of the coordinates of all quad-edge intersections for that cell. This averaging enables representation of a fault surface with sub-voxel precision. Therefore, to find the locations of the quad nodes, we must first find the intersections of the fault surface and edges of the 3D sampling grid.

I find edge intersections and compute their locations using a method similar to that described by Schultz et al. (2010). I assume that fault surfaces are ridges in 3D images of fault likelihoods, analogous to the fault curves apparent in the 2D images shown in Figure 6.

These ridges intersect edges of cells in the 3D sampling grid, and can be found by considering all such edges, one at a time. Each edge is defined by two adjacent samples in the 3D image of fault likelihood. Let f_1 and f_2 denote fault likelihoods for these two samples. We may at this point choose a threshold f_{\min} and assume that faults can exist only if both $f_1 \geq f_{\min}$ and $f_2 \geq f_{\min}$. I chose $f_{\min} = 0.5$ when extracting the fault surfaces shown in this paper.

Following Schultz et al. (2010), let \mathbf{g} and \mathbf{H} denote the gradient vector and Hessian matrix for either of two adjacent samples in the fault likelihood image, like those shown in Figure 7. I compute each gradient (vector of 1st derivatives) \mathbf{g} and Hessian (matrix of 2nd derivatives) \mathbf{H} using simple centered finite-difference approximations to partial derivatives, after Gaussian smoothing (with radius $\sigma = 1$ sample) of the fault image to attenu-

ate high frequencies for which those approximations are poor.

Now let $\mathbf{H} = \lambda_u \mathbf{u}\mathbf{u}^T + \lambda_v \mathbf{v}\mathbf{v}^T + \lambda_w \mathbf{w}\mathbf{w}^T$ denote the eigen-decomposition of \mathbf{H} , where the eigenvalues are ordered so that $\lambda_u \geq \lambda_v \geq \lambda_w$. At locations of ridges, the smallest eigenvalue λ_w should be negative, and I assume that faults can exist only between two samples for which this condition is true.

If this condition is indeed true, then the eigenvectors \mathbf{w} for these two samples should be orthogonal to any ridge that may exist between them. Like Schultz et al. (2010), I then compute for each of these two samples a vector \mathbf{h} defined by

$$\mathbf{h} = (1 - \lambda)\mathbf{w}\mathbf{w}^T \mathbf{g}, \quad (4)$$

where

$$\lambda = \begin{cases} 0 & \text{if } \lambda_v - \lambda_w > \epsilon, \\ (1 - \frac{\lambda_v - \lambda_w}{\epsilon})^2 & \text{otherwise,} \end{cases} \quad (5)$$

and ϵ is a small fraction of the square of the typical fault image sample value. For fault likelihoods in the range $[0, 1]$, I use $\epsilon = 0.01$. The purpose of this parameter is to smooth the transition of the factor λ from zero to one where the eigenvalues λ_v and λ_w are nearly equal.

Recall that the scan used to compute the 3D fault likelihood image yields corresponding estimates of fault strike and dip angles for every image sample. I therefore make two significant modifications to the process of computing vectors \mathbf{h} .

First, from the fault strike and dip I compute a fault normal vector \mathbf{n} . I then assume that a fault can exist only between two samples for which $|\mathbf{n}^T \mathbf{w}| > 1/2$. This condition ensures some consistency between two different estimates of the normal vector; the angle between \mathbf{n} and \mathbf{w} must be less than 60 degrees. This upper bound on angle is rather large because the eigenvector \mathbf{w} of \mathbf{H} tends to be a poor estimate of the fault normal vector.

Therefore, in a second modification, if this condition is satisfied, I replace the eigenvector \mathbf{w} in equation 4 with the fault normal vector \mathbf{n} computed from the estimated fault strike and dip. I experimented with using the eigenvectors \mathbf{w} instead, as in Schultz et al. (2010), and found that the fault strikes and dips obtained during the fault image scan yielded more consistent normal vectors.

Finally, like Schultz et al. (2010), I assume that ridges exist between two adjacent samples with vectors \mathbf{h}_1 and \mathbf{h}_2 that point in opposite directions, so that $\mathbf{h}_1^T \mathbf{h}_2 < 0$. Letting \mathbf{x}_1 and \mathbf{x}_2 denote the spatial coordinates of two samples for which this condition is true, I compute the location \mathbf{x}_e where a ridge intersects the edge between those two samples by linear interpolation:

$$\mathbf{x}_e = \frac{[\mathbf{h}_2^T (\mathbf{h}_2 - \mathbf{h}_1)]\mathbf{x}_1 - [\mathbf{h}_1^T (\mathbf{h}_2 - \mathbf{h}_1)]\mathbf{x}_2}{(\mathbf{h}_2 - \mathbf{h}_1)^T (\mathbf{h}_2 - \mathbf{h}_1)}. \quad (6)$$

The spacial coordinates of a quad node located within any cell of the sampling grid are the average of coor-

dinates \mathbf{x}_e computed for all quad-edge intersections in that cell.

This interpolation and averaging yields quads that coincide (with sub-voxel precision) with ridges in the 3D image of fault likelihoods, and implies that the four nodes of a quad need not be coplanar. Also, while interpolating and averaging to compute the spatial coordinates of quad nodes, I interpolate and average the corresponding fault likelihoods, strikes and dips.

By analyzing all edges in the sampling grid for 3D images of fault likelihood, strike and dip, I extract quads that intersect edges where faults may exist. Because quads extracted in this way share nodes, they may appear to be parts of larger fault surfaces when displayed. However, at this point in the process of fault surface extraction, the quads are not yet linked together to form a surface mesh. I have only a collection of quads, what is sometimes called “quad soup.” In the example shown in Figure 2, this soup contained 436111 quads.

Before linking quads together to form meshes that represent fault surfaces, I perform one more test for consistency. Each of the four nodes referenced by any quad has associated estimates of fault strike and dip that define a fault normal vector \mathbf{n} . If we let \mathbf{x}_a , \mathbf{x}_b , \mathbf{x}_c and \mathbf{x}_d denote the spatial coordinates of these four quad nodes (in either clockwise or counterclockwise order), then the vector cross product $\mathbf{n}_q = (\mathbf{x}_a - \mathbf{x}_c) \times (\mathbf{x}_b - \mathbf{x}_d)$ should be (approximately, because the quad nodes need not be coplanar) normal to the quad. I keep only quads for which the angles between the vector \mathbf{n}_q and each of the normal vectors \mathbf{n} for its nodes are all less than 30 degrees. I remove from the soup any quad that fails this test. In the example shown in Figure 2, 337986 quads remained after this test.

3.2 Linking quads

The next step in extracting fault surfaces is to link quads together to form a mesh. Each quad in such a mesh may have up to four quad neighbors, where two quads are neighbors if they share an edge between two quad nodes. In the illustration in Figure 7 each quad has exactly two neighbors.

In the quad soup obtained using the process described above, it is possible that some edges between two quad nodes may be shared by more than two quads. Due to the multiple tests for consistency used in that process, this situation is rare; but where it occurs I choose to link none of the quads that share such an edge. This choice implies that two fault surfaces extracted in this way cannot intersect precisely, although they may be separated by only one grid sample.

Let us define the orientation of a quad such that its normal vector \mathbf{n}_q points toward a viewer that sees the quad nodes a , b , c and d labelled in counter-clockwise order. From the opposite side, those same nodes would appear to be labelled in clockwise order. We can then

define back and front sides of a quad such that the quad normal vector \mathbf{n}_q points into the quad’s back side and points out from its front side.

At this stage in the process of linking quads, there is no guarantee that quad neighbors are oriented consistently. The normal vectors of a quad and its quad neighbors may point in opposite directions. However, any difference in the orientations of quad neighbors can be easily detected and accounted for when determining which quads share an edge between two of their nodes.

After all links between quads and their neighbors have been found, I apply three more filters to eliminate quads and links that are inconsistent with any geologically feasible model of fault surfaces. These filters work much like the consistency checks applied to quads in the quad soup described above.

The first filter unlinks any quads that are folded on top of one another. Specifically, I unlink two quads if the dihedral angle between them (computed from their normal vectors) is less than 90 degrees. This filter ensures that the orientation of a fault surface does not vary too rapidly from one quad to the next.

The second filter unlinks two quads if either of them (1) has no other neighbors or (2) has only one neighbor on the opposite side. Such quads tend to appear as fins or bridges between two nearby fault surfaces. In unlinking such quads, I assume that fault surfaces must nowhere be too skinny, with a width or height of only one quad.

After the first two filters remove geologically infeasible links between quad neighbors, the third filter simply removes any quads that have no neighbors. In applying this filter, I require that any fault surface must have a size of more than one quad.

When applied to the quads extracted in Figure 2, these three filters caused 15976 of the 337986 quads to be removed from the quad soup. All of the 322010 quads that remained were linked to at least two quad neighbors.

3.3 Constructing oriented fault surfaces

After extracting quads and linking them together, the final step in extracting fault surfaces is to find collections of quads that are linked either directly as neighbors or recursively as neighbors of neighbors. These collections form quad meshes that represent fault surfaces.

I assume that fault surfaces are orientable, that they have topologically distinct front and back sides, unlike the surfaces extracted from medical images by Schultz et al. (2010). In other words, I assume that fault normal vectors can be chosen consistently for every quad in the surface, so that the front side of every quad coincides with the front side of the surface.

This orientability assumption may be neither valid nor necessary, but is convenient here because of another assumption that I make when estimating fault throws as

described in the following section of this paper. I compute fault throws as a vector field of displacements from the back side to the front side of a fault surface, and vice-versa. In that process I assume that fault throws vary smoothly within each fault surface, and this assumption is most easily enforced where all of the quads in a fault surface are oriented consistently. Therefore, when collecting quads to form fault surfaces, I flip the orientations of quads as necessary to be consistent with their neighbors.

The collection process begins with a loop over all quads in any order. I first construct a new fault surface containing any quad. I then add neighbors of this first quad to the surface, flipping their orientations as necessary to be consistent with that of the first quad. I then recursively add quad neighbors, if not already in the surface, again flipping their orientations as necessary. The first fault surface is complete when there exist no linked quad neighbors that are not already part of that surface.

The collection process then returns to the loop over quads. When I find a quad that is not yet part of a fault surface, I again construct a new fault surface with that quad and recursively add quad neighbors to it in the same way as for the first fault surface. When the loop over all quads is complete, every quad belongs to exactly one fault surface.

I determine whether or not a fault surface is orientable during the recursive collection of quads. If, while examining neighbors of a quad, I find a neighbor that is already part of the surface and that has an inconsistent orientation, then the surface is not orientable. Otherwise, if all quad neighbors have consistent orientations, the surface is orientable.

In the extraction of surfaces shown in Figure 2, I found 1922 surfaces, and all of them were orientable. This figure displays only the 20 largest fault surfaces, those with at least 2000 quads.

In another example I found that 0.03% of surfaces extracted were not orientable. I make such surfaces orientable by simply unlinking any quad neighbors that are already part of the surface and that have inconsistent orientations. In effect, this unlinking makes a surface orientable by cutting it in a rather arbitrary way that depends on the recursive order in which I add quads to the surface. Better methods for choosing the cut may be possible, and may be important where a large number of surfaces must be cut to make them orientable.

Because we typically view faults from the hanging wall side (above), and not from the footwall side (below), one final orientation of fault surfaces is useful for display. If necessary, I flip the orientations of all quads in each fault surface so that the average of the normal vectors for those quads points upward, not downward. After this final orientation, when viewing fault surfaces from the hanging wall side, we see the front side of the surface.

The filtering based on fault sizes used to obtain the 20 surfaces shown in Figure 2 is just one example of the sort of filtering that is possible after constructing fault surfaces. We could also filter these surfaces based on their average strikes or dips, (Pedersen et al., 2003; Pedersen, 2007, 2011), their fault likelihoods, or any combination of statistics derived from attributes computed for the quads that comprise the surfaces.

4 FAULT THROWS

Conceptually the problem of estimating fault throws from 3D seismic images is a simple one. We must correlate seismic reflections on one side of the fault with those on the other side, and compute vector displacements between corresponding reflections. In practice, this problem is difficult for several reasons.

4.1 Difficulties

First, our resolution of faults is limited by the resolution of seismic images, so that reflections on one side of a fault may extend somewhat into the other side. (See, for example, the image displayed in Figure 3a.) As suggested by Liang et al. (2010), our correlation of reflections across faults must in some way mimic the visual correlation of experienced seismic interpreters. That is, we must estimate fault throws from coherent seismic reflections extending well away from a fault, not only those immediately adjacent to it.

A second difficulty is that fault throws vary within a fault surface and within any local windows that we might use to correlate seismic reflections. To ease this difficulty, Aurnhammer and Tönnies (2005) used several geologic and geometric constraints in a generic algorithm to estimate fault displacements.

Still, inconsistency remains in estimating a fault throw from a window of image samples in which that throw may vary significantly (L. Liang, personal communication, 2011). This second difficulty is exacerbated by the fact that such windows must be at least as long as the longest fault throw vector to be estimated.

A third difficulty is related to the fact that we can best estimate fault throws from strong seismic reflections with high signal-to-noise ratios, but these may be far apart, with weaker and noisier reflections in between. Constraints (e.g., Aurnhammer and Tönnies, 2005) are therefore needed to ensure continuity of fault throws estimated between strong reflections.

A fourth difficulty lies in constraining fault throws to vary smoothly in both dip and strike directions within a fault surface. Imposing this constraint is difficult partly because a fault surface typically cannot be projected onto a plane and then represented as a single-valued function (such as distance) of coordinates within that plane. This means that we cannot simply estimate

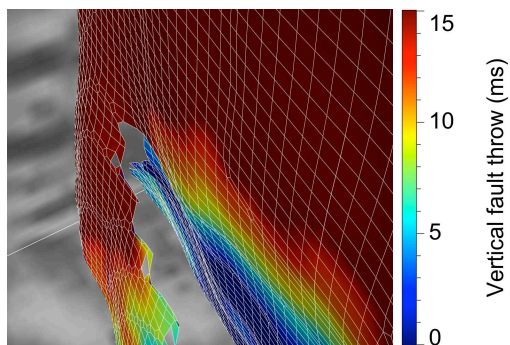


Figure 8. Fault throws computed for a fault surface in which one part of the surface lies in front of another part. For such surfaces, we cannot compute fault throws from footwall and hanging-wall images extracted alongside the fault.

fault throws from 3D seismic images by correlating one 2D image extracted from the footwall side of a fault surface with another 2D image extracted from the hanging-wall side.

An example is shown in Figure 8, where part of a fault surface lies in front of another part of that same surface. This situation occurs often in the 20 fault surfaces shown in Figure 2. Another example is the roughly conical fault displayed in Figure 1b. For such surfaces we cannot extract 2D images from the footwall and hanging-wall sides of a fault surface, so we cannot use 2D crosscorrelations of such images to estimate smoothly varying fault throws.

4.2 Dynamic warping

I address the difficulties summarized above with an extension of a classic method for estimating relative shifts between two acoustic signals in the problem of speech recognition (Sakoe and Chiba, 1978). This method is today widely known as *dynamic time warping*. For the problem of estimating fault throws, the most important aspect of this method is that it estimates a time-varying (dynamic) shift between two sampled functions of time, without any local windows. Another important aspect of this method is that the change in the shift with time can be easily constrained with no additional cost.

In a separate paper (Hale, 2012) I propose an extension of the dynamic time warping algorithm to the problem of dynamic image warping, in which we seek to constrain estimates of relative shifts between two images to vary smoothly in all directions. The extension is an alternating sequence of vertical (top-down, bottom-up) and lateral (left-right, right-left) smoothings of differences in image sample values, followed by the classic dynamic time warping algorithm. The vertical and lateral smoothings are non-linear, but simple and computationally efficient.

Unfortunately, as implied by Figure 8, we cannot reduce the problem of estimating fault throws to that

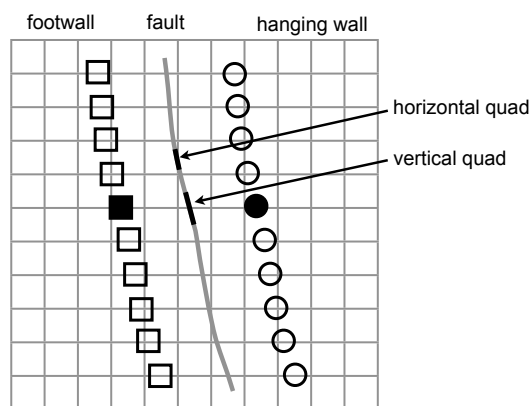


Figure 9. Fault throws are computed from differences between image sample values (squares) on the footwall side and values (circles) on the hanging wall side of a fault. These sample values are slightly offset from vertical quads, which intersect horizontal edges in the image-sampling grid. Fault strike is perpendicular to the plane of this figure.

of finding an optimal warping between footwall and hanging-wall images. We can, however, adapt the image warping solution to the problem of estimating fault throws, by performing a similar sequence of vertical and lateral smoothings within fault surfaces like those displayed in Figure 8.

Recall that each quad in this fault surface intersects exactly one edge in the sampling grid of a 3D seismic image, as shown in Figure 7. Some quads intersect vertical edges, but I assume that most quads, like those in Figure 7, intersect horizontal edges of the sampling grid. This assumption is valid even for fault dip angles greater than 45 degrees, measured from vertical, because seismic images are typically sampled more finely in vertical directions than in horizontal directions.

As illustrated in Figure 9, let us refer to quads intersecting vertical edges as horizontal quads, and quads intersecting horizontal edges as vertical quads, even though quads are rarely exactly horizontal or vertical. I make this distinction because I use only the vertical quads in the dynamic warping process used to estimate fault throws.

My first step in estimating fault throws is to compute for each vertical quad two sequences of squared differences between image sample values collected from both sides of a fault surface. This first step is similar to computing squared differences of sample values in image warping, except that here I obtain sample values by following fault dip vectors up and down the fault surface, as illustrated in Figure 9.

Each intersection of a fault with a horizontal edge of the sampling grid corresponds to one vertical quad in a fault surface, like that near the center of the fault surface in Figure 9. For each such vertical quad I first find the value of the nearest sample (the filled square) located

at the same time (or depth), with some small lateral offset, in the footwall side of the fault. The purpose of this small offset is to compensate for limited resolution in 3D seismic images of faults. In Figure 9 the offset is two samples.

I then compute and store with this vertical quad the squared differences between the one footwall sample value (the filled square) and all of the values (the circles) on the hanging wall side of the fault. These squared differences form a sequence that is indexed by vertical lag.

I use that sequence and those computed for all other vertical quads in the same fault surface as inputs to the dynamic warping algorithm to estimate vertical components of throw vectors from the footwall side to the hanging wall side of the fault. I then estimate the horizontal components of throw vectors by once again following the fault dip vectors up or down the fault.

I repeat this process to estimate fault throws from the hanging-wall side to the footwall side of the fault. By estimating throws in both directions, we can check pairs of throw vectors for consistency. If we follow the fault throw vector from a sample on the footwall side of the fault to a sample on the hanging-wall side, and then follow the throw vector found there back to the footwall side, we should return to the first sample at which we began.

A less rigorous test is to simply reject throw vectors on opposite sides of a fault where their vertical components have the same sign. In other words, we may assume that a fault exists only where throw vectors on opposite sides of the fault have vertical components with different signs.

This assumption is valid for all fault throws shown in this paper. The vertical components of all fault throws illustrated in figures are positive because the throws shown are those from the footwall sides to the hanging-wall sides of normal faults.

In summary, the dynamic warping process described above computes fault throws that minimize the sum of squared differences between image sample values on two sides of the faults, while constraining the rate at which the throws may change within the fault surface.

4.3 Unfaulted images

A good test of the fidelity of estimated fault throws is to use them to undo faulting apparent in the seismic images from which they were derived. Luo and Hale (2012) describe this unfaulting process in detail. Here I use this process only to illustrate the accuracy of fault surfaces and fault throws computed using the methods proposed in this paper.

Figure 2 provides one example in which fault surfaces have roughly planar shapes. Faulting and unfaulting are most significant in the inline sections, because faults with large throws have strike vectors that point

approximately in the crossline direction. Throws for these faults vary somewhat in the strike direction while generally increasing with depth.

Vertical exaggeration in these sections makes the faults appear to be more vertical than they really are; an approximate time-to-depth conversion (with 1 s equivalent to 1 km) indicates that the dips of most faults are about 45 degrees from vertical.

Visual comparison of the continuity of reflections before and after unfaulting suggests that estimated fault throw vectors are generally accurate. However, one location where estimated fault throw appears to have the wrong sign is at about 1.7 s and 4.5 km in the crossline section.

4.4 Conical faults

Figures 10 and 11 show faults extracted from the same seismic image for a shallower portion of the subsurface with more chaotic structure. Faults extracted from this portion have roughly conical shapes, in which the apex of each cone lies above its base. Figure 1b displays a close-up view of one of these conical faults.

These conical shapes became apparent to me only after extracting fault surfaces, partly because I had never seen faults with such shapes before, and so did not recognize their appearance in horizontal and vertical slices of the 3D seismic image.

This experience highlights an important benefit in using an automated process to extract information from 3D seismic images. The process used here could not exclude such shapes simply because they were unexpected.

After recognizing the conical shapes of these faults, they are easily seen in 3D seismic images, even without the fault throws shown in Figures 10 and 11. In vertical sections, these faults appear to be hyperbolic, because a vertical slice (conic section) of a cone is a hyperbola. When interactively moving a vertical slice through the 3D image, one can clearly see these hyperbolas rising and falling as the image slice moves through the cones.

Moreover, even in this relatively chaotic part of the seismic image, reflections are more continuous in the unfaulted image than in the original image. Some discontinuities that remain may be due to my including only the largest fault surfaces when computing fault throws. Of the 4608 surfaces extracted, I computed throws for only the 30 largest surfaces, those with at least 2000 quads.

5 CONCLUSION

I developed the methods proposed in this paper as parts of a three-step process to (1) compute fault images of likelihood, strike and dip, (2) extract fault surfaces, and (3) estimate fault throws. I once hoped to skip the first two steps, to simply compute fault throws everywhere

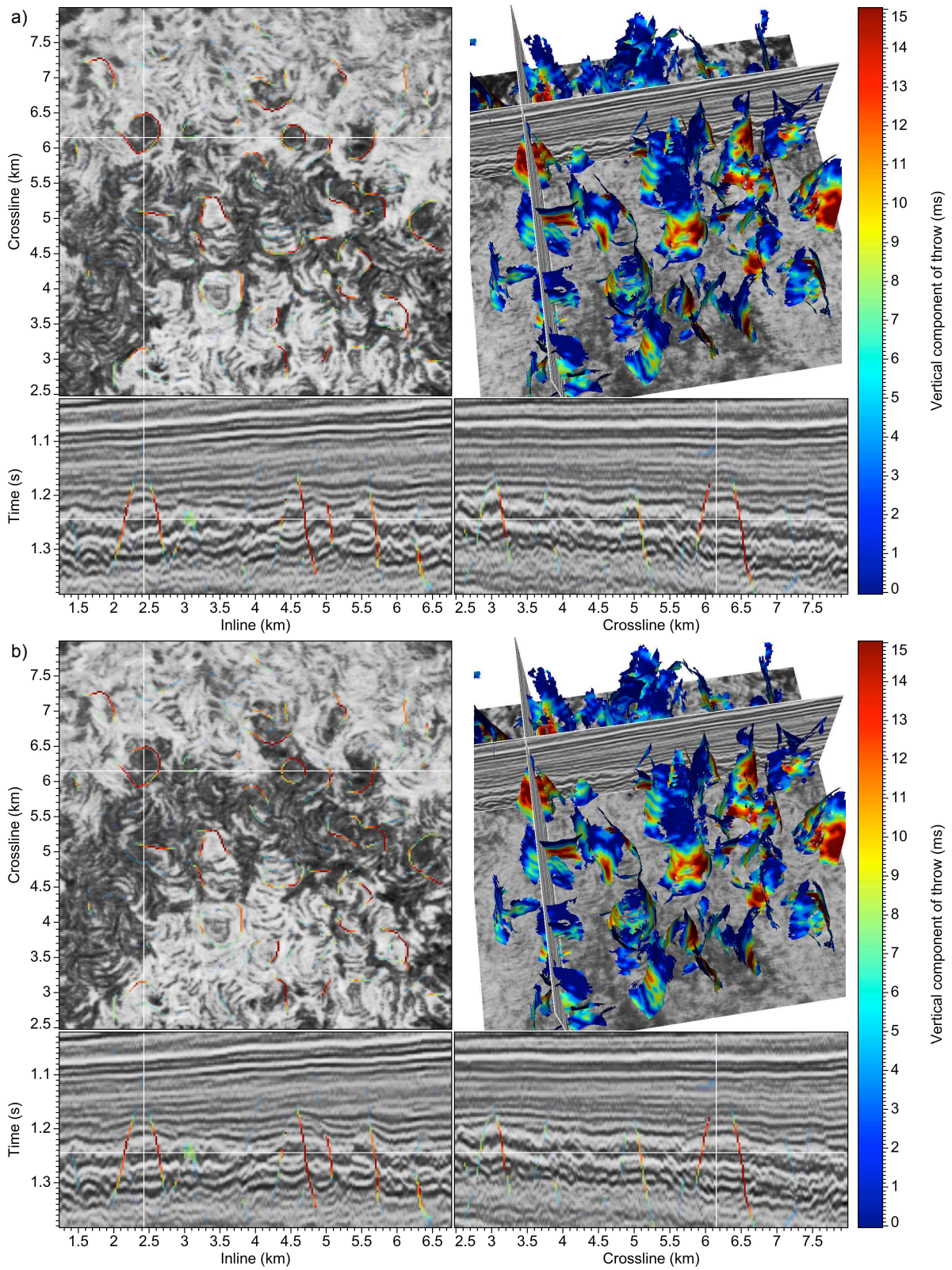


Figure 10. Fault surfaces and fault throws for a 3D seismic image before (a) and after (b) unfaulting. The shape for many of these faults is roughly conical, and the two vertical sections intersect near the center of one of these conical faults.

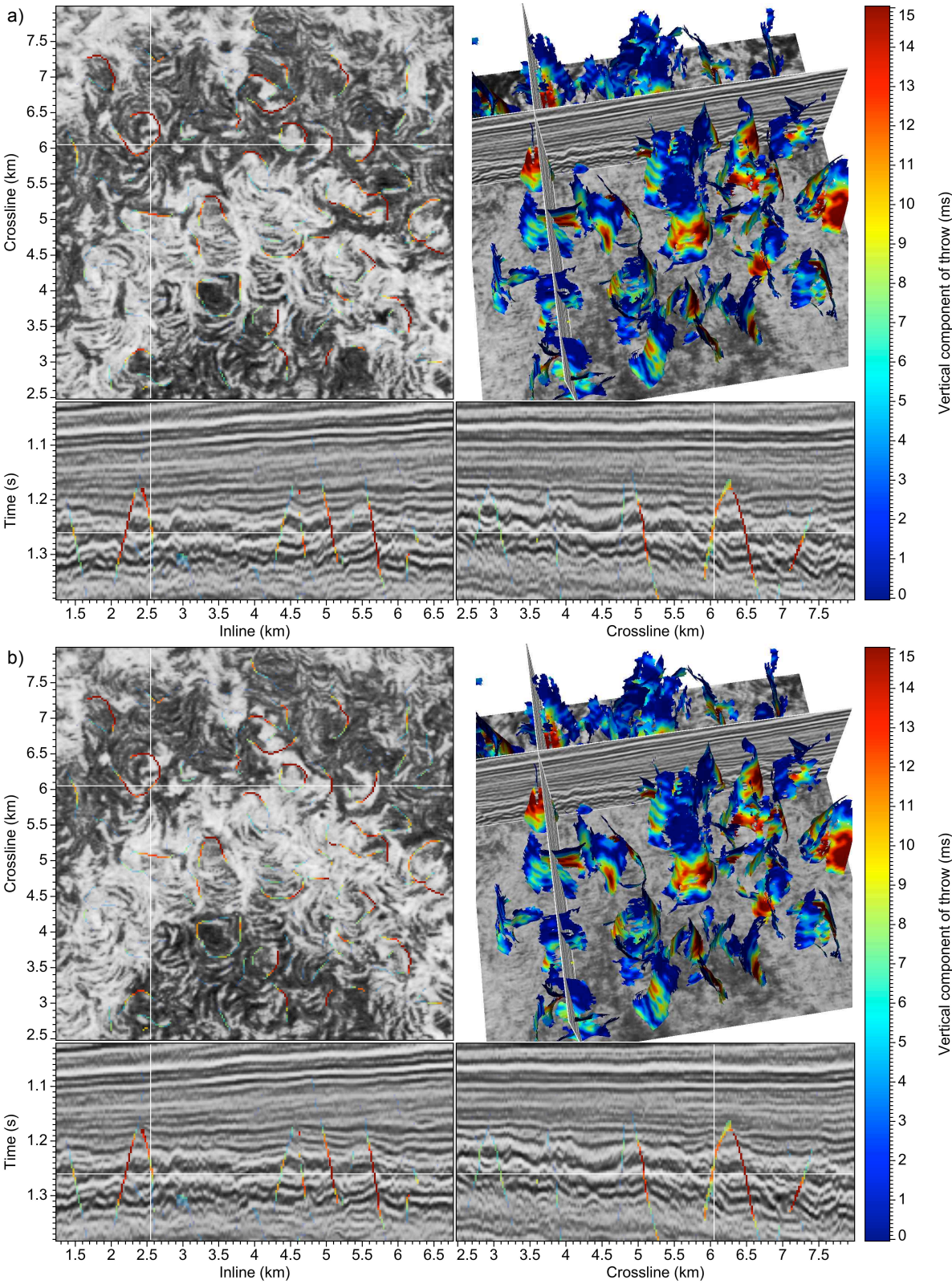


Figure 11. Fault surfaces and fault throws for a 3D seismic image before (a) and after (b) unfaulting. Conical faults appear as hyperbolas in vertical seismic sections.

and then let faults be defined as locations where fault throws are significant. However I was unable to find a computationally feasible implementation of this potentially simpler one-step process.

It is significant that the scan in the first step yields images of fault strikes and dips for which fault likelihood is maximized. These estimates of fault orientations are useful in several consistency tests performed in the second step used to extract fault surfaces.

I chose the quad-mesh representation for those fault surfaces in part to facilitate the third step of estimating fault throws. Because throw vectors connect samples on one side of a fault to those on the other side, it is especially convenient that quads in the fault surface lie between two adjacent samples of the seismic image at the same time or depth. In addition, the quad mesh also provides up-down and left-right connectivity needed to implement the dynamic warping algorithm used to estimate vertical throws.

Most of the computation time in this three-step process lies in the first step, which currently requires a scan over all possible fault orientations. I improve the computational efficiency of this scan by using fast recursive smoothing filters within each potential fault plane, but further improvements may be worthwhile. My current implementation of this scan for about 500 fault orientations requires about two hours to process a 3D image of 1000^3 samples on a 12-core workstation.

Perhaps the biggest current limitation in this process is in its handling of intersecting faults. As discussed above, to simplify the extraction of fault surfaces and estimation of fault throws, I have incorrectly assumed that faults do not intersect. Further work is required to extend these two steps to properly account for fault intersections.

ACKNOWLEDGMENTS

In much of the research described in this paper I benefited greatly from discussions with others, including Luming Liang, Marko Maučec, Bob Howard, Dean Witte, and Anastasia Mironova. The 3D seismic image used in this study was graciously provided by dGB Earth Sciences B.V. through OpendTect.

REFERENCES

- Admasu, F., 2008, A stochastic method for automated matching of horizons across a fault in 3D seismic data: PhD thesis, Otto-von-Guericke-University Magdeburg.
- Admasu, F., S. Back, and K. Toennies, 2006, Auto-tracking of faults on 3D seismic data: *Geophysics*, **71**, A49–A53.
- Aqrawi, A., and T. Boe, 2011, Improved fault segmentation using a dip-guided and modified 3D sobel filter: Presented at the 81st Annual International Meeting, SEG, Expanded Abstracts.
- Aurnhammer, M., and K. Tönnies, 2005, A genetic algorithm for automated horizon correlation across faults in seismic images: *IEEE Transactions on Evolutionary Computation*, **9**, 201–210.
- Bates, K., T. Cheret, F. Pauget, and S. Lacaze, 2009, Vertical displacement on faults extracted from seismic, offshore Nigeria case study: Presented at the EAGE 71st Conference and Exhibition, Expanded Abstracts.
- Borgos, H., T. Skov, T. Randen, and L. Sonneland, 2003, Automated geometry extraction from 3D seismic data: Presented at the 73th Annual International Meeting, SEG, Expanded Abstracts.
- Carrillat, A., H. Borgos, T. Randen, L. Sonneland, L. Kvamme, and K. Hansch, 2004, Fault system analysis with automatic fault displacement estimates — a case study: Presented at the EAGE 66th Conference and Exhibition, Expanded Abstracts.
- Cohen, I., N. Coult, and A. Vassiliou, 2006, Detection and extraction of fault surfaces in 3D seismic data: *Geophysics*, **71**, P21–P27.
- Gersztenkorn, A., and K. Marfurt, 1999, Eigenstructure-based coherence computations as an aid to 3-D structural and stratigraphic mapping: *Geophysics*, **64**, 1468–1479.
- Gibson, D., M. Spann, T. J., and T. Wright, 2005, Fault surface detection in 3-D seismic data: *IEEE Transactions on Geoscience and Remote Sensing*, **43**, 2094–2102.
- Hale, D., 2012, Dynamic warping of seismic images: CWP Report 723.
- Kadlec, B., 2011, Visulation of geologic features using data representations thereof: US Patent Application 2011/0,115,787.
- Kadlec, B., G. Dorn, H. Tufo, and D. Yuen, 2008, Interactive 3-D computation of fault surfaces using level sets: *Visual Geoscience*, **13**, 133–138.
- Liang, L., D. Hale, and M. Maučec, 2010, Estimating fault displacements in seismic images: 80th Annual International Meeting, SEG, Expanded Abstracts, 1357–1361.
- Luo, S., and D. Hale, 2012, Unfaulting and unfolding 3D seismic images: CWP Report 722.
- Marfurt, K., R. Kirlin, S. Farmer, and M. Bahorich, 1998, 3-D seismic attributes using a semblance-based coherency algorithm: *Geophysics*, **63**, P1150–P1165.
- Marfurt, K., V. Sudhaker, A. Gersztenkorn, K. Crawford, and S. Nissen, 1999, Coherence calculations in the presense of structural dip: *Geophysics*, **64**, P104–111.
- Neff, D., J. Grismore, and W. Lucas, 2000, Automated seismic fault detection and picking: US Patent 6,018,498.
- Pedersen, S., 2007, Image feature extraction: US Patent 7,203,342.

- , 2011, Image feature extraction: US Patent 8,055,026.
- Pedersen, S., T. Randen, L. Sonneland, and O. Steen, 2002, Automatic 3D fault interpretation by artificial ants: Presented at the 72nd Annual International Meeting, SEG, Expanded Abstracts.
- Pedersen, S., T. Skov, A. Hetlelid, P. Fayemendy, T. Randen, and L. Sonneland, 2003, New paradigm of fault interpretation: Presented at the 73rd Annual International Meeting, SEG, Expanded Abstracts.
- Randen, T., S. Pedersen, and L. Sonneland, 2001, Automatic extraction of fault surfaces from three-dimensional seismic data: Presented at the 71st Annual International Meeting, SEG, Expanded Abstracts.
- Rodgers, J., and W. Nicewander, 1988, Thirteen ways to look at the correlation coefficient: *The American Statistician*, **42**, 59–66.
- Sakoe, H., and S. Chiba, 1978, Dynamic programming algorithm optimization for spoken word recognition: *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **26**, 43–49.
- Schultz, T., H. Theisel, and H.-P. Seidel, 2010, Crease surfaces: from theory to extraction and application to diffusion tensor MRI: *IEEE Transactions on Visualization and Computer Graphics*, **16**, 109–119.
- Skov, T., M. Oygaren, H. Borgos, M. Nickel, and L. Sonneland, 2004, Analysis from 3D fault displacement extracted from seismic data: Presented at the EAGE 66th Conference and Exhibition, Expanded Abstracts.
- Taner, M., and F. Koehler, 1969, Velocity spectra — digital computer derivation and applications: *Geophysics*, **34**, 859–881.
- Van Bommel, P., and R. Pepper, 2011, Seismic signal processing method and apparatus for generating a cube of variance values: US Patent 8,055,026.
- van Vliet, L., and P. Verbeek, 1995, Estimators for orientation and anisotropy in digitized images: *Proceedings of the first annual conference of the Advanced School for Computing and Imaging*, 442–450.
- Weickert, J., 1999, Coherence-enhancing diffusion filtering: *International Journal of Computer Vision*, **31**, 111–127.